ON THE POWER OF CONTEXT-ENHANCED LEARNING IN LLMS

Xingyu Zhu*, **Abhishek Panigrahi***, **Sanjeev Arora**
Princeton Language and Intelligence, Princeton University
{xingyu.zhu, ap34}@princeton.edu

ABSTRACT

We formalize a new concept for LLMs, **context-enhanced learning**. It involves standard gradient-based learning on text except that the context is enhanced with additional data on which no auto-regressive gradients are computed. This setting is a gradient-based analog of usual in-context learning (ICL) and appears in some recent works. Using a multi-step reasoning task, we prove in a simplified setting that context-enhanced learning can be **exponentially more sample-efficient** than standard learning when the model is capable of ICL. At a mechanistic level, we find that the benefit of context-enhancement arises from a more accurate gradient learning signal. Our findings highlight the potential of context-enhanced learning to bridge gradient-based learning and ICL, offering new insights into their interplay.

1 INTRODUCTION

Pre-trained LLMs (Brown et al., 2020; Touvron et al., 2023; Team et al., 2023) show strong capability to learn new material at inference time, for instance via in-context-learning (ICL). There is also emerging evidence that gradient-based learning on a piece of text (say, math Q&A) can be enhanced if additional helpful text is placed in the context, even though no auto-regressive loss is computed on this helpful text Liao et al. (2024); Zou et al. (2024). Furthermore, such strategies have also been shown to benefit pre-training as prepending source URLs to documents can enhance the model's training efficiency and memorization capacity (Allen-Zhu & Li, 2024; Gao et al., 2025).

The current paper seeks to formally study this phenomenon, whereby LLMs' gradient-based learning is *enhanced* via placement of additional helpful material in the context, but without actual auto-regressive gradient updates on this material. We will call this form of learning *context-enhanced learning*. Because the material used for context-enhancement can evolve over the course of training, this approach naturally aligns with the idea of a *curriculum*.

Context-enhanced learning intuitively mirrors how humans learn: when solving problems, they refer to textbooks or demonstrations for guidance, yet they do not seek to memorize these resources *per se*. An analogous concept *Learning using Privileged Information* (or LUPI), has been well-studied in the context of kernel SVMs Vapnik & Vashist (2009) and classification models. Our work adapts this concept for LLMs, and surfaces the following questions:

Q1: Even though auto-regressive loss is computed on the same set of tokens, can context-enhanced learning be more powerful than usual auto-regressive learning with no additional in-context materials? If so, can we theoretically characterize and understand such improvement?

Q2: Do models need a certain capability level to benefit from context-enhanced learning? This is a natural question, since leveraging in-context information (e.g., ICL) likely requires a minimum capability level or model size Brown et al. (2020); Wei et al. (2022).

Paper Overview: Section 2.1 formally defines context-enhanced learning. To allow rigorous understanding of the power of context-enhanced learning, Section 2.2 introduces a multi-step reasoning task called *Multi-layer translation*. This is a synthetic setting involving $d+1$ languages $L_1, L_2, \ldots, L_{d+1}$ over finite alphabets. For each i, there is a simple *phrasebook* that describes how to translate from L_i to L_{i+1}, and the mapping from L_1 to L_{d+1} is a sequential application of the set of phrasebooks. The

*Contributed equally to the theoretical framework and problem formulation, XZ conducted most experiments.

goal is to learn how to translate text from L_1 into L_{d+1} without explicitly writing down intermediate steps. The learner is provided excerpts from these phrasebooks as helpful information in the context during training, but allowed no auto-regressive gradient updates on these tokens.

If we train with auto-regressive loss on translation output conditioning on the phrasebooks' excerpts and the input, a model with certain ICL capacity level may quickly learn to leverage the in-context phrasebooks and correctly conduct the translations. However, this learning could be brittle, meaning that the model becomes reliant on having the phrasebooks' excerpts in its context. This reliance can be weaned off by use of probabilistic *dropout* on phrasebooks tokens in context. Intuitively, this simple curriculum forces the model to not only read phrasebooks' excerpts, but also (over time) internalize the phrasebooks' contents. Over time, the model's ability to translate from L_1 to L_{d+1} will become robust to the dropout of phrasebooks' excerpts, and eventually, their complete removal.

Experiments show that this training strategy indeed works when the learner is a pre-trained LLM that is capable of ICL (but fails when LLM is incapable of ICL). Even when training with 20% dropout rate, the model can perfectly translate strings from L_1 to L_{d+1} without any phrasebooks' excerpts at test time. The rest of the paper is structured as follows:

• Section 3 details our experiments and the findings sketched above. Experiments show that an ICL-capable model follows an intuitive sequential processing of the phrasebooks provided in-context, whereby transformer layers approach stages of translation in an intuitive way; e.g., $L_3 \to L_4$ is done after $L_2 \to L_3$ (Section 3.2).

• In Section 4, we propose a theoretical framework using a surrogate/simplified model that represents an ideal LLM for the translation task (Section 4.1). This framework shows an exponential gap in sample complexity depending on whether the model is trained with or without in-context information on phrasebooks (Sections 4.2 and 4.3). Experiments reveal that the mechanism behind the increased sample efficiency of context-enhanced learning is an improved gradient signal, measured by gradient prediction accuracy (Section 4.4).

2 SETUP

2.1 CONTEXT-ENHANCED LEARNING

Let X be the space of all possible text strings and let $\mathcal{Y}$ be the space of all possible distributions over texts. Let g be a language task mapping inputs $x \in X_g \subset X$ to a distribution $Y \in \mathcal{Y}$. Let $f_\theta : X \to \mathcal{Y}$ be an auto-regressive language model. We characterize f_θ's capability on task g as:

Definition 2.1 (g-capable model, informal)**.** A language model f_θ is g-capable for a language task g if f_θ is close to g, as measured by a suitable metric on X_g.

Vanilla supervised fine-tuning (SFT) aims to create a g-capable model by minimizing auto-regressive loss ℓ_{auto} on a supervised dataset $D_g = \{(x_i, y_i)\}_{i=1}^N$, where the label y_i for each $x_i \in X_g$ is sampled from $g(x_i)$. *Context-enhanced learning* involves augmenting the supervision with additional curriculum-text that depends on the task g, input x, and training step t. We denote curriculum-text as $\text{CURR}_g(x, t)$, which could be anything (excerpts from textbooks, worked-out examples, etc.).

On sample (x, y) drawn from the supervised dataset, we use auto-regressive loss for model's prediction on y conditioned on $[\text{CURR}_g(x, t), x]$ to train our models. Note that *no loss is computed for curriculum-text tokens.* We denote this loss as $\ell_{\text{auto}}\left(f_\theta([\text{CURR}_g(x, t), x, y]), y\right)$.

2.2 MULTI-LEVEL TRANSLATION (**MLT**)

To study the power of context-enhanced learning, we introduce a translation task that is easy to learn with a straightforward curriculum in the context, but very difficult to learn with just input-output examples. The task is inspired by encryption methods such as the Feistel cipher Knudsen (1993).

Concretely, let $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_{d+1}$ be $d+1$ alphabets all of the same size with n characters. For every consecutive pair of alphabets $\boldsymbol{A}_i$ and $\boldsymbol{A}_{i+1}$, we fix a *phrasebook* $\pi_i : \boldsymbol{A}_i^2 \to \boldsymbol{A}_{i+1}^2$ as a bijective mapping from 2-tuples in $\boldsymbol{A}_i$ to 2-tuples in $\boldsymbol{A}_{i+1}$. Each phrasebook π_i can be represented by a binary stochastic matrix $\texttt{Matrix}(\pi_i)$ with rules represented as one-hot columns (see Definition G.4).

The input of the translation process is an even-length sequence $\boldsymbol{s}_1 \in \boldsymbol{A}_1^L$ where L is the sequence length. The translation process modifies $\boldsymbol{s}_1$ recursively. For every $i \in [d]$, $\boldsymbol{s}_i \in \boldsymbol{A}_i^L$ will be transformed to $\boldsymbol{s}_{i+1} \in \boldsymbol{A}_{i+1}^L$ using *phrasebook* π_i through the following 2 sub-processes:

• *Circular shift*: The characters in $\boldsymbol{s}_i \in \boldsymbol{A}_i^L$ are shifted by 1 character leftward (and wrapped around to the end if necessary) to give sequence $\tilde{\boldsymbol{s}}_i \in \boldsymbol{A}_i^L$. Formally, for each $j \in [1, L]$ we have

$$\tilde{\boldsymbol{s}}_{i,j} = \boldsymbol{s}_{i,(j+1)\%L}.$$

• *Translate*: Using the *phrasebook* $\pi_i : \boldsymbol{A}_i^2 \to \boldsymbol{A}_{i+1}^2$, we translate 2-tuples (bigrams) of consecutive characters in sequence $\tilde{\boldsymbol{s}}_i$ to create $\boldsymbol{s}_{i+1}$. That is, for every odd $j \in [1, L]$, we have

$$(\boldsymbol{s}_{i+1,j}, \boldsymbol{s}_{i+1,j+1}) = \pi_i(\tilde{\boldsymbol{s}}_{i,j}, \tilde{\boldsymbol{s}}_{i,j+1}).$$

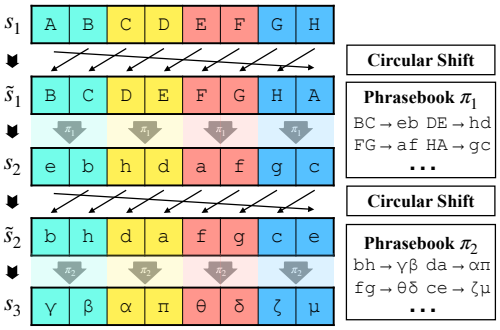


Figure 1: Illustration of an $\mathbf{MLT}(2, 8)$ instance with input sequence of 8 tokens. Input sequence $\boldsymbol{s}_1$ went through 2 translation steps to $\boldsymbol{s}_3$. Each output character depends on 4 input characters. e.g., character μ is derived from `c,e` in $\boldsymbol{s}_2$, which, in turn, are computed from `A,B,C,H` in $\boldsymbol{s}_1$.

We denote the above mapping from $\boldsymbol{s}_i$ to $\boldsymbol{s}_{i+1}$ as $\boldsymbol{s}_{i+1} = T_{\pi_i}(\boldsymbol{s}_i)$. $\boldsymbol{\Pi} = \{\pi_i\}_{i=1}^d$ denotes the set of phrasebooks of all levels, and we use $\mathbf{MLT}_{\boldsymbol{\Pi}} : \boldsymbol{s}_1 \mapsto \mathbf{MLT}_{\boldsymbol{\Pi}}(\boldsymbol{s}_1) := T_{\pi_d} \circ \cdots \circ T_{\pi_1}(\boldsymbol{s}_1)$ to denote the mapping from input $\boldsymbol{s}_1$ to output $\boldsymbol{s}_{d+1}$ using $\boldsymbol{\Pi}$. We use $\mathbf{MLT}(d, n)$ to refer to the family of translations involving d step and n characters in each alphabet.

We note two key properties of $\mathbf{MLT}(d, n)$: **1.** Once the phrasebooks are fixed, the translation task defines a bijection between input and output strings since *Circular shift* and *Translate* are invertible (see Lemma E.1). **2.** Each character in the output string depends on $2d$ characters from the input text string (see caption of Figure 1), making learning from input-output pairs very difficult (Theorem 4.4).

For each phrasebook π_i, we compose its textual representation $\text{STR}(\pi_i)$ to be of the form `... a b -> C D; e d -> B A; ...` which lists (insensitive of ordering) phrasebook rules between 2-tuples in the previous alphabet and the next alphabet. Moreover, we denote the concatenation $[\text{STR}(\pi_1), \ldots, \text{STR}(\pi_d)]$ as $\text{STR}(\boldsymbol{\Pi})$, and will be used to define curriculum-text.

2.3 Needed: Curriculum without Explicit CoT

To teach the model a particular translation task $\mathbf{MLT}_{\boldsymbol{\Pi}}$ from input-output pairs of the form $(\boldsymbol{s}_1, \mathbf{MLT}_{\boldsymbol{\Pi}}(\boldsymbol{s}_1))$, we can train it with relevant sections of the phrasebooks $\text{STR}(\boldsymbol{\Pi})$ in context as curriculum, but at test time it would not have access to the phrasebook so it is important not to teach it explicit chain-of-thought (CoT) containing in-context information. (Another consideration is data privacy, with the phrasebook being considered privileged information.) However, a dual use of CoT is to provide the model extra compute at inference time Goyal et al. (2023), which is needed here since the translation task has d stages. To facilitate such silent computation we teach the model to output a fixed number of `<THINK>` tokens, sometimes refered to as *silent CoT* or *internalized CoT*.

2.4 ICL-capablity for $\mathbf{MLT}(d, n)$

To learn from books, one needs to know how to read. The analogous question here is whether context-enhanced learning requires capability to sort-of "understand" the in-context material (**Q2**). In the context of **MLT**, we formalize such capability as being able to achieve low loss on the translation task when provided with the relevant phrasebook sections in context while allowing silent CoT.

Definition 2.2 ($\mathbf{MLT}(d, n)$-ICL-capability, informal)**.**
A language model f_θ is $\mathbf{MLT}(d, n)$-*ICL-capable* if for any set of phrasebooks $\boldsymbol{\Pi}$ in $\mathbf{MLT}(d, n)$, $f_\theta([\text{STR}(\boldsymbol{\Pi}), \boldsymbol{s}_1])$ is close to $\boldsymbol{s}_{d+1} = \mathbf{MLT}_{\boldsymbol{\Pi}}(\boldsymbol{s}_1)$ disregarding the `<THINK>` tokens, when measured by a discrepancy metric over all valid input strings $\boldsymbol{s}_1$.

3 EXPERIMENTS AND OBSERVATIONS

In this section, we fix a set of phrasebooks $\mathbf{\Pi}^*$ and study context-enhanced learning on $\mathbf{MLT}_{\mathbf{\Pi}^*}$. We use the **Llama 3.2-3B instruction-tuned model** (Dubey et al., 2024) as the base model and fix $d = 5$ with $n = 8$ or 10. Detailed configurations are available in Appendix B.5. We first outline the preparation of an $\mathbf{MLT}(d, n)$-ICL-capable model, then introduce context-enhanced learning curriculum with random rule dropping, demonstrate sample efficiency, and conclude with mechanistic insights.

3.1 EXPERIMENTAL SETUP AND RESULTS

(i) Preparing an $\mathbf{MLT}(d, n)$-ICL-Capable Model:
The Llama 3.2B model has not seen the translation task during its training, and so it is not $\mathbf{MLT}(d, n)$-ICL-capable. To make it ICL-capable for our purpose, we use SFT on other random translation tasks with random phrasebooks $\mathbf{\Pi}_1, \ldots, \mathbf{\Pi}_M$, following common CoT internalization pipeline (Deng et al., 2024; Hao et al., 2024). We use one training example per set of phrasebooks to prevent memorization of specific phrasebooks. At the end of training, given input $[\texttt{STR}(\mathbf{\Pi}), \boldsymbol{s}_1]$ for any $\boldsymbol{s}_1$ and $\mathbf{\Pi}$ the model can generate $\texttt{<THINK>}, \ldots, \mathbf{MLT}_{\mathbf{\Pi}}(\boldsymbol{s}_1)$ correctly. Details on the first stage of training are described in Appendix B.4.

(ii) Setting up context-enhanced learning for $\mathbf{MLT}_{\mathbf{\Pi}^*}$:
We use the $\mathbf{MLT}(d, n)$-ICL-capable model above as initialization and train for $\mathbf{MLT}_{\mathbf{\Pi}^*}$. Supervised dataset $D_{\mathbf{\Pi}^*}$ is curated with input-label pairs of the form $(\boldsymbol{s}_1, [\texttt{<THINK>}, \ldots, \mathbf{MLT}_{\mathbf{\Pi}^*}(\boldsymbol{s}_1)])$, where $\boldsymbol{s}_1$ is a random string sampled from $\boldsymbol{A}_1$, with length between 20 and 40. We define curriculum-text $\text{CURR}_{\mathbf{\Pi}^*}(\boldsymbol{s}_1, t)$ using excerpts from phrasebooks $\texttt{STR}(\mathbf{\Pi}^*)$ (selected based on $\boldsymbol{s}_1$) with random dropout of rules (parameterized by step t).

To check the sample efficiency benefit of context-enhanced learning (**Q1**), we construct supervised datasets $D_{\mathbf{\Pi}^*}$ with 10^4 to 10^6 unique samples and train the models for one epoch on each.[1] We report the next-token prediction accuracy on the final answer tokens (ignoring thought tokens) for held-out samples when conditioning on no curriculum-text (100% dropped-out) and compare against the supervised dataset size. To test the sample efficiency of context-enhanced learning and determine the necessary components for context-enhanced learning to work, we test and ablate on the following curriculum setups:

- **No Context** (vanilla SFT): Empty curriculum.
- **Fixed Dropout**: A simple strategy independent of step t; given $\boldsymbol{s}_1$, only curate rules in $\mathbf{\Pi}^*$ used in the translation of $\boldsymbol{s}_1$, then randomly drop 20% of the curated rules.
- **Annealing Dropout**: for $\boldsymbol{s}_1$, select the necessary rules from $\mathbf{\Pi}^*$ plus 25% unused rules. Apply random dropout on these rules, increasing linearly from 0% to 100% over the first 60% of training, then maintain 100%.
- **No Dropout** (ablation): Given $\boldsymbol{s}_1$, always provide all rules in $\mathbf{\Pi}^*$ used in the translation of $\boldsymbol{s}_1$ in curriculum-text.
- **Wrong Context** (ablation): Equivalent to **Annealing Dropout** but with incorrect rules.
- **No ICL** (ablation) To check the necessity of proper ICL capability (**Q2**), we ablate with **Annealing Dropout** but starting from non-$\mathbf{MLT}(d, n)$-ICL-Capable 3B base model.

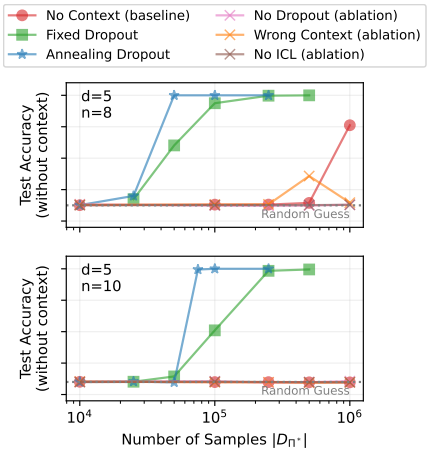


Figure 2: Context-enhanced learning of $\mathbf{MLT}_{\mathbf{\Pi}^*}$ when $n = 8$ and $n = 10$. **Annealing Dropout** learns the fastest followed by **Fixed Dropout**, requiring 10x less samples compared to vanilla SFT. Ablations show the necessity of correct context, proper dropout, and sufficient ICL capability as a good initialization.

Figure 2 demonstrates the significant sample efficiency of context-enhanced learning. Moreover, models trained with subsets of phrasebooks and just 20% dropout give perfect heldout test-time accuracy with 100% dropout rate. Thus they are able to effectively use phrasebook rules

[1] We fix 1 epoch for fair comparison on sample efficiency.

from subsets of the phrasebook that did not co-occur in the same training sample. Clearly, the model has learned the phrasebook atomically, and can combine the rules as needed at test time.

In an ablation (Appendix C), we show that context-enhanced learning only internalizes the rules whose dropout from curriculum-text leads to an increase in loss on training data. The experiment results can be summarized as follows: **(i)** Context-enhanced learning from an ICL-capable model greatly improves training sample efficiency. **(ii)** phrasebook rules are internalized atomically, and only when missing them incurs an increased loss.

3.2 Mechanistic Insights: Layer by Layer Mapping of the Translation Task

To understand the behavior of context-enhanced learning, we probe into the hidden representations and weights of models before and after context-enhanced learning.

Sequential Processing in ICL-capable model

First, we look into how phrasebooks in curriculum-text are used by an $\mathbf{MLT}(d, n)$-ICL-capable model for solving a translation task. We perform our experiments on a random set of phrasebooks $\mathbf{\Pi} = \{\pi_1, \ldots, \pi_d\}$. On any input $\boldsymbol{s}_1$, we feed in input sequence $[\textsc{str}(\mathbf{\Pi}), \boldsymbol{s}_1, \texttt{<THINK>}, \ldots, \boldsymbol{s}_{d+1}]$, and record the model's hidden representations, i.e. output of each transformer layer, for the tokens $\texttt{<THINK>}, \ldots, \boldsymbol{s}_{d+1}$.

Now, we perturb each phrasebook independently in curriculum-text and measure the change in the model's representations. That is, for each level $1 \leq i \leq d$, we substitute $\textsc{str}(\pi_i)$ with the textual representation of another random phrasebook, say $\textsc{str}(\hat{\pi}_i)$, and compute the norm difference between the model's representations before and after substitution. The first layer with significant difference in its output representation is identified as the layer where the model begins processing the phrasebook.

In Figure 3, we show that an ICL-capable model processes phrasebooks in curriculum-text sequentially, with earlier phrasebooks read by earlier layers. Formal details are in Appendix B.2 and additional experiments are in Appendix C.2.

Localized Storage after Context-Enhanced Learning

Here, we verify whether a similar sequential pattern for internalizing phrasebooks is used in a $\mathbf{MLT}_{\mathbf{\Pi}^*}$-capable model. We denote the ICL-capable model as f_θ and its post context-enhanced learning counterpart as f_{θ^*}. As f_{θ^*}'s capability no longer depends on textual representation of the phrasebooks, our analysis focuses on the model's parameters.

We assess the importance of each layer in model f_{θ^*} for each phrasebook by constructing "stitched" models and measuring their output behavior. For every $1 \leq L_{\text{start}} \leq L_{\text{end}} \leq 28$ (total layers in Llama3.2 3B), we replace layers L_{start} to L_{end} of the ICL model f_θ with corresponding layers from f_{θ^*} to create a stitched model. This strategy has been used in prior works on localizing information after SFT (Gong et al., 2022; Panigrahi et al., 2023; Wei et al., 2024).

For each level $1 \leq i \leq d$, we evaluate the stitched models on $\mathbf{MLT}_{\mathbf{\Pi}^*}$ using the following in-context information: $[\textsc{str}(\pi_1^*), \cdots, \textsc{str}(\pi_{i-1}^*), \textsc{str}(\pi_{i+1}^*), \cdots, \textsc{str}(\pi_d^*)]$. Note here the textual representation of π_i^* has been dropped. If a stitched model shows high accuracy on inputs that use π_i^* for translation without having π_i^*'s information in context, then the layers selected from f_{θ^*} to create the stitched model can be deemed responsible for storing information on π_i^* in their parameter space.

Figure 4 demonstrates that information of all phrasebooks can be localized to a few mutually disjoint layers in f_{θ^*}. Moreover, the end of the group of layers where π_i^* is localized in f_{θ^*} marks the start of the layers that begin processing level i phrasebook in the ICL-capable model f_θ (e.g. compare the role of layer 17 in Figures 3 and 4). Details and additional experiments are in Appendices B.3 and C.2.

This suggests that instead of storing phrasebooks as a single chunk in its parameters, the model re-learns each translation step locally to compensate for missing information when rules are dropped. Thus, we conjecture that context-enhanced learning leverages curriculum-text to improve training by localizing learning in the parameter space. We build a surrogate model in Section 4 to show how such localized learning can prove beneficial for faster training.

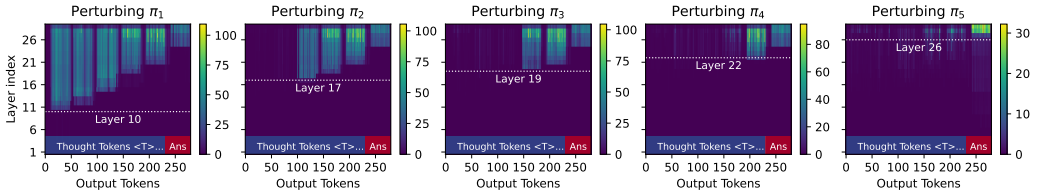


Figure 3: Evidence for sequential processing in a $\mathbf{MLT}(5, 8)$-ICL-capable model. Each entry is the l_2 norm between the latent representation pre and post-perturbing π_i. Perturbing later phrasebooks in the context changes output representations in the later layers.

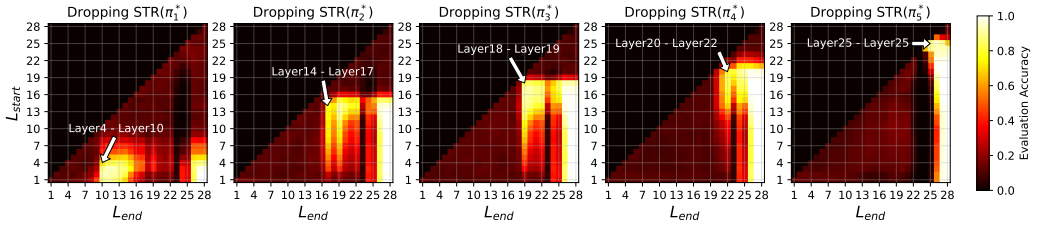


Figure 4: Evaluation of stitched models when dropping different phrasebooks from the context. Bright colors close to the diagonal reflects localized storage in a small subset of layers in f_{θ^*} to compensate the dropped phrasebook. For any level i, the end of the group of layers storing π_i^* matches the start of the layers for reading π_i (see Figure 3) before context-enhanced learning.

4 Mathematical Analysis

Having empirically demonstrated the sample efficiency benefits of context-enhanced learning, we now formalize them through a mathematical lens. We first define the sample complexity of an algorithm for learning the task.

Definition 4.1 (informal)**.** Sample complexity for an algorithm to learn $\mathbf{MLT}_{\mathbf{\Pi}}$ is defined as the minimum total length of all (possibly repeated) input sequences required by the algorithm to return an $\mathbf{MLT}_{\mathbf{\Pi}}$-capable model f_θ.

Analysis of gradient-based learning on a multilayer transformer (let alone a 28-layer model like Llama 3.2-3B) is an open mathematical question. We use our mechanistic findings (i.e., translation layers map onto transformer layers) to propose a surrogate model to think about how transformers learn $\mathbf{MLT}(d, n)$. In this surrogate model we demonstrate that learning a task $\mathbf{MLT}_{\mathbf{\Pi}}$ via vanilla SFT will require $n^{\Omega(d)}$ samples. Then, we prove that, with context-enhanced learning, the surrogate model can learn $\mathbf{MLT}_{\mathbf{\Pi}^*}$ with a sample complexity of $\mathcal{O}(\mathsf{poly}(n)d \log d)$.

4.1 Surrogate Model (Surr-MLT)

Formalization of the surrogate model, in short Surr-MLT, relies on observations from Section 3.2, which reveal that an ICL-capable model (Definition 2.2) performs the translation task step-by-step, with earlier phrasebook processed by lower layers. This aligns with how the model stores internalized knowledge in layers after context-enhanced learning. Our Surr-MLT represents an idealized and simplified transformer that has already been "pre-conditioned" to solve **MLT** in this sequential fashion. Using Surr-MLT, we will show the benefits of context-enhanced learning.

Without loss of generality we assume the alphabet sets as $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_{d+1} := \boldsymbol{A} = \{1, 2, \ldots, n\}$. Surr-MLT will represent a length-L sequence $\boldsymbol{s}_i = (s_{i,1}, \ldots, s_{i,L})$ as an embedding matrix $\boldsymbol{V}_i \in \mathbb{R}^{n^2 \times L/2}$ that uses $\{\boldsymbol{v}(s_{i,1}, s_{i,2}), \cdots, \boldsymbol{v}(s_{i,L-1}, s_{i,L})\}$ as columns. Here, for any 2-tuple (a, b), $\boldsymbol{v}(a, b) \in \mathbb{R}^{n^2}$ represents a one-hot vector with 1 at dimension $an + b$.

Surr-MLT operates on embedding matrices, transforming $\boldsymbol{V}_1 \to \boldsymbol{V}_2 \to \cdots \to \boldsymbol{V}_{d+1}$. Each layer i will be primarily defined by two matrices, $\boldsymbol{C}_i, \boldsymbol{W}_i \in \mathbb{R}^{n^2 \times n^2}$. $\boldsymbol{C}_i$ presents a (possibly partial or completely dropped) phrasebook that is provided in-context, and $\boldsymbol{W}_i$ is a trainable parameter storing phrasebook information during context-enhanced learning.

Definition 4.2. SURR-MLT with trainable parameters $\{\boldsymbol{W}_i\}_{i=1}^d$ and in-context representation $\{\boldsymbol{C}_i\}_{i=1}^d$, is represented by its operation on an input $\boldsymbol{s}_1$ as

$$\boldsymbol{V}_{d+1} = \text{SURR-MLT}_{\{\boldsymbol{W}_i\}_{i=1}^d}\left(\{\boldsymbol{C}_i\}_{i=1}^d, \boldsymbol{V}_1\right), \quad \text{where}$$
$$\boldsymbol{V}_{i+1} = \texttt{HardMax}(\boldsymbol{C}_i + \boldsymbol{W}_i)\,\texttt{Shift}(\boldsymbol{V}_i), \text{ for } i \geq 1,$$

and $\boldsymbol{V}_1$ is the embedding matrix for the input string $\boldsymbol{s}_1$.

`Shift` represents *Circular shift* operation and is defined as a Hadamard product on the embedding matrices (details in Definition G.2). `HardMax` represents hard-max function converting $\boldsymbol{C}_i + \boldsymbol{W}_i$ to a binary column stochastic matrix. The surrogate model can perfectly represent an ICL-capable model by fixing all training parameters as 0s and feeding in the phrasebooks using their stochastic matrix representations into the in-context representations. Similarly, it can represent an $\mathbf{MLT}_{\mathbf{\Pi}^*}$-capable model, by setting the in-context representations as 0s and contain phrasebooks as $\{\texttt{Matrix}(\pi_i^\star)\}_{i=1}^d$ in its trainable parameters $\{\boldsymbol{W}_i\}_{i=1}^d$.

SURR-MLT as an Ideal Transformer for MLT: The following theorem constructs a transformer that can simulate SURR-MLT. Thus, while our discussions and proofs focus on the surrogate model for simplicity, they remain fully applicable to the transformer architecture.[2]

Theorem 4.3 (cf Lemma H.4)**.** *There exists a transformer that can simulate* SURR-MLT *with* $2d$ *self-attention and* $2d$ *MLP layers with embedding dimension* $2n^2 + 2d + 4$.

4.2 SAMPLE COMPLEXITY FOR VANILLA SFT

For the surrogate model, vanilla SFT corresponds to always setting in-context representations $\{\boldsymbol{C}_i\}_{i=1}^d$ to 0s when training for $\mathbf{MLT}_{\mathbf{\Pi}^*}$. While we use the surrogate model for consistency in our discussion, the argument generalizes to any model learning $\mathbf{MLT}_{\mathbf{\Pi}^*}$ with vanilla SFT.

We build on Statistical Query (SQ) framework (Kearns, 1998), which measures the difficulty of learning tasks using algorithms that rely on expectation estimates of specific functions to approximate the true solution. Gradient-based methods, such as Stochastic Gradient Descent (SGD), fall under this framework as they compute gradients by expectations of loss functions and their derivatives.

The complexity of learning a task is quantified by the SQ dimension, which measures the number of candidate functions that are pairwise uncorrelated under the input distribution and difficult to distinguish with limited samples. A higher SQ dimension implies a richer hypothesis class, that requires more samples to identify the correct function. We show in the following theorem that the SQ dimension of $\mathbf{MLT}(d, n)$ grows exponentially with task parameters.

Theorem 4.4. *SQ dimension of* $\mathbf{MLT}(d, n)$ *under uniform input distribution is at least* $n^{\Omega(d)}$.

Informally, this suggests that any algorithm that tries to learn a $\mathbf{MLT}_{\mathbf{\Pi}^*}$-capable SURR-MLT without access to the phrasebooks by minimizing loss across samples can require at least $n^{\Omega(d)}$ sample complexity. A corollary is that for vanilla SFT with SGD, sample complexity to learn $\mathbf{MLT}_{\mathbf{\Pi}^*}$ can be at least $n^{\Omega(d)}$. This is adapted from Edelman et al. (2023), who analyse for sparse parity that has similar SQ dimension (Corollary F.6). The proof for Theorem 4.4 is given in Appendix F.5.

4.3 SAMPLE COMPLEXITY OF CONTEXT-ENHANCED LEARNING

An $\mathbf{MLT}(d, n)$-ICL-capable model gets a set of phrasebooks $\mathbf{\Pi}^*$ by setting $\{\texttt{Matrix}(\pi_i^*)\}_{i=1}^d$ as in-context representations $\{\boldsymbol{C}_i\}_{i=1}^d$. When a curriculum is followed such that a translation rule is dropped from a phrasebook, say π_i^*, SURR-MLT will set the corresponding column in its in-context representation $\boldsymbol{C}_i$ as 0s. Due to this, the expected loss will increase for the task $\mathbf{MLT}_{\mathbf{\Pi}^*}$. A heuristic search algorithm, that searches among n^2 possibilities for a dropped rule and stores its one-hot representation in the corresponding column in $\boldsymbol{W}_i$, can be used to minimize the loss. By randomly dropping rules, followed by a search and store process, the algorithm achieves polynomial sample complexity.

[2]Llama model in Section 3 is larger than the required size of SURR-MLT for our dataset scale.

Theorem 4.5 (Informal; cf Corollary G.17)**.** *For any task* $\boldsymbol{MLT}_{\boldsymbol{\Pi}^*}$*, there is a heuristic search algorithm, paired with a curriculum of iteratively dropping a random rule from phrasebooks, that can learn a* $\boldsymbol{MLT}_{\boldsymbol{\Pi}^*}$*-capable* SURR-MLT *with sample complexity* $\mathcal{O}(n^6 d \log d)$ *with high probability.*

The enumerative step in the heuristic search algorithm requires $\Theta(n^2)$ steps on average, as the algorithm needs to search over $\Theta(n^2)$ possibilities when a rule is dropped from a phrasebook. Instead, we show that gradient descent requires only a few steps per dropped rule. Dropping a rule sets the respective column in the in-context representation to 0, causing the gradient for the corresponding column in the trainable parameters to strongly align with the missing column. We formally present results for $d = 2$; due to exponentially growing number of terms to analyse with higher d, we keep the result for general d as a conjecture. In appendix fig. 11, we train a surrogate model of much higher depth, and show trainable parameters quickly learn the phrasebooks in $\boldsymbol{\Pi}^*$ with gradient descent.

Theorem 4.6 (Informal; cf Corollary G.23)**.** *When* $d = 2$*, there is a gradient descent based algorithm, paired with a curriculum of iteratively dropping a random rule from phrasebooks, that can return* $\mathbf{MLT}_{\boldsymbol{\Pi}^*}$*-capable* SURR-MLT *with sample complexity* $\mathcal{O}(n^4)$ *with high probability.*

4.4 INSIGHT FROM SURR-MLT: GRADIENT QUALITY

Inspired by gradient descent analysis, we now show the major difference between context-enhanced learning and vanilla SFT to be predictive information in gradients for the trainable parameters.

We define a measure called **gradient prediction accuracy**. For an ICL-capable SURR-MLT, we randomly drop one or more rules from the phrasebooks by zeroing out columns in in-context representations. We then measure whether gradients computed using a batch of examples for the trainable parameters align with the representations of the dropped rules. We report the average over gradients of trainable parameters in the first layer of SURR-MLT. A more formal description is in Appendix B.1. We evaluate for two cases: (1) rules dropped only from the first phrasebook and (2) rules dropped from multiple phrasebooks. In both cases, higher dropping rates significantly degrade accuracy, with larger batch sizes offering slower improvement. Dropping rules from multiple phrasebooks further degrades gradient prediction accuracy.

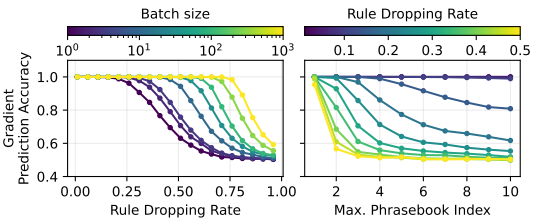


Figure 5: Gradient prediction accuracy when (left) varying dropping rates for rules in π_1 and batch sizes of gradient computation, and (right) varying dropping rates and max phrasebook index (k) for rules dropped from $\pi_1, \ldots, \pi_k$. Higher dropping rates lead to lower gradient prediction accuracy.

5 DISCUSSION, LIMITATIONS, AND FUTURE WORK

Some experimental works have implicitly used the notion of context-enhanced learning but the current paper formalized this notion for auto-regressive models and showed, using **MLT**, that this form of learning can be exponentially more sample-efficient than standard SFT. At the end of training it is hard to recover the in-context information seen during training from the model's output probabilities. We note that this finding appears to have implications about copyright law (e.g., whether or not LLM training amounts to "transformative use" of text Carlini et al. (2021; 2022); Karamolegkou et al. (2023)) whose further study is left for future work.

Our experiments focus on a synthetic **MLT** task for a few reasons: (1) to ensure that the task is absent from LLM pre-training, which allows precise quantification of benefits of context-enhanced learning, including not revealing the curriculum text at inference time. (2) the task is too difficult (at least for Llama 3.2 3B model) to learn via vanilla SFT, but is learnable via context-enhanced learning. Extending these findings to real-world complicated tasks (e.g., in math and coding) is left for future work.

Our convergence analysis for context-enhanced learning relies on a surrogate model, and extending it to an actual transformer remains an open challenge for theory of deep learning. Extending formalization of context-enhanced learning to explore LLM training in multi-agent settings–where models collaborate and learn from each other to discover novel concepts–would be an exciting avenue for future research.

ACKNOWLEDGMENT

We thank Yun Cheng, Simon Park, Tianyu Gao, Yihe Dong, Zixuan Wang, Haoyu Zhao, and Bingbin Liu for discussions, suggestions, and proof-reading at various stages of the paper. AP, SA acknowledge funding from NSF, PLI, DARPA, ONR, and OpenAI. XZ is additionally supported by a Gordon Y.S. Wu Fellowship in Engineering.

REFERENCES

Ekin Akyurek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. What learning algorithm is in-context learning? investigations with linear models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=0g0X4H8yN4I.

Ekin Akyürek, Bailin Wang, Yoon Kim, and Jacob Andreas. In-context language learning: Architectures and algorithms. *arXiv preprint arXiv:2401.12973*, 2024.

Zeyuan Allen-Zhu and Yuanzhi Li. Physics of language models: Part 1, Context-free grammar. *arXiv preprint arXiv:2305.13673*, 2023a.

Zeyuan Allen-Zhu and Yuanzhi Li. Physics of language models: Part 3.2, Knowledge manipulation. *arXiv preprint arXiv:2309.14402*, 2023b.

Zeyuan Allen-Zhu and Yuanzhi Li. Physics of language models: Part 3.3, knowledge capacity scaling laws. *arXiv preprint arXiv:2404.05405*, 2024.

Cem Anil, Yuhuai Wu, Anders Andreassen, Aitor Lewkowycz, Vedant Misra, Vinay Ramasesh, Ambrose Slone, Guy Gur-Ari, Ethan Dyer, and Behnam Neyshabur. Exploring length generalization in large language models. 35:38546–38556, 2022.

Simran Arora, Avanika Narayan, Mayee F Chen, Laurel Orr, Neel Guha, Kush Bhatia, Ines Chami, Frederic Sala, and Christopher Ré. Ask me anything: A simple strategy for prompting language models. *arXiv preprint arXiv:2210.02441*, 2022.

Mohammad Bavarian, Heewoo Jun, Nikolas Tezak, John Schulman, Christine McLeavey, Jerry Tworek, and Mark Chen. Efficient training of language models to fill in the middle. *arXiv preprint arXiv:2207.14255*, 2022.

Satwik Bhattamishra, Kabir Ahuja, and Navin Goyal. On the ability and limitations of transformers to recognize formal languages. pp. 7096–7116, 2020.

Avrim Blum, Merrick Furst, Jeffrey Jackson, Michael Kearns, Yishay Mansour, and Steven Rudich. Weakly learning dnf and characterizing statistical query learning using fourier analysis. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pp. 253–262, 1994.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pp. 2633–2650, 2021.

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. Quantifying memorization across neural language models. *arXiv preprint arXiv:2202.07646*, 2022.

Michele Ceccarelli and Antonio Maratea. Improving fuzzy clustering of biological data by metric learning with side information. *International Journal of Approximate Reasoning*, 47(1):45–57, 2008.

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. 35:18878–18891, 2022.

Jiaao Chen, Xiaoman Pan, Dian Yu, Kaiqiang Song, Xiaoyang Wang, Dong Yu, and Jianshu Chen. Skills-in-context prompting: Unlocking compositionality in large language models. *arXiv preprint arXiv:2308.00304*, 2023.

Jixu Chen, Xiaoming Liu, and Siwei Lyu. Boosting with side information. In *Asian Conference on Computer Vision*, pp. 563–577. Springer, 2012.

Lele Cheng, Xiangzeng Zhou, Liming Zhao, Dangwei Li, Hong Shang, Yun Zheng, Pan Pan, and Yinghui Xu. Weakly supervised learning with side information for noisy labeled images. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXX 16*, pp. 306–321. Springer, 2020.

Yuntian Deng, Yejin Choi, and Stuart Shieber. From explicit cot to implicit cot: Learning to internalize cot step by step. *arXiv preprint arXiv:2405.14838*, 2024.

Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. Chain-of-verification reduces hallucination in large language models. *arXiv preprint arXiv:2309.11495*, 2023.

Aniket Didolkar, Anirudh Goyal, Nan Rosemary Ke, Siyuan Guo, Michal Valko, Timothy Lillicrap, Danilo Rezende, Yoshua Bengio, Michael Mozer, and Sanjeev Arora. Metacognitive capabilities of llms: An exploration in mathematical problem solving. *arXiv preprint arXiv:2405.12205*, 2024.

Chris Donahue, Mina Lee, and Percy Liang. Enabling language models to fill in the blanks. *arXiv preprint arXiv:2005.05339*, 2020.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Benjamin L Edelman, Surbhi Goel, Sham Kakade, Eran Malach, and Cyril Zhang. Pareto frontiers in neural feature learning: Data, compute, width, and luck. *arXiv preprint arXiv:2309.03800*, 2023.

Ronen Eldan and Yuanzhi Li. TinyStories: How small can language models be and still speak coherent English? *arXiv preprint arXiv:2305.07759*, 2023.

Vivek F Farias and Andrew A Li. Learning preferences with side information. *Management Science*, 65(7):3131–3149, 2019.

Tianyu Gao, Alexander Wettig, Luxi He, Yihe Dong, Sadhika Malladi, and Danqi Chen. Metadata conditioning accelerates language model pre-training. *arXiv preprint arXiv:2501.01956*, 2025.

Zhuocheng Gong, Di He, Yelong Shen, Tie-Yan Liu, Weizhu Chen, Dongyan Zhao, Ji-Rong Wen, and Rui Yan. Finding the dominant winning ticket in pre-trained language models. In *Findings of the Association for Computational Linguistics: ACL 2022*, pp. 1459–1472, 2022.

Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. Think before you speak: Training language models with pause tokens. *arXiv preprint arXiv:2310.02226*, 2023.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*, 2024.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

Judy Hoffman, Saurabh Gupta, and Trevor Darrell. Learning with side information through modality hallucination. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 826–834, 2016.

Rico Jonschkowski, Sebastian Höfer, and Oliver Brock. Patterns for learning with side information. *arXiv preprint arXiv:1511.06429*, 2015.

Antonia Karamolegkou, Jiaang Li, Li Zhou, and Anders Søgaard. Copyright violations and large language models. *arXiv preprint arXiv:2310.13771*, 2023.

Michael Kearns. Efficient noise-tolerant learning from statistical queries. *Journal of the ACM (JACM)*, 45(6):983–1006, 1998.

Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of naacL-HLT*, volume 1. Minneapolis, Minnesota, 2019.

Lars R Knudsen. Practically secure feistel ciphers. In *International Workshop on Fast Software Encryption*, pp. 211–221. Springer, 1993.

Pirkko Kuusela and Daniel Ocone. Learning with side information: Part i. 02 2002.

Dongyuan Li, Jingyi You, Kotaro Funakoshi, and Manabu Okumura. A-tip: attribute-aware text infilling via pre-trained language model. In *Proceedings of the 29th International Conference on Computational Linguistics*, pp. 5857–5869, 2022.

Yuchen Li, Yuanzhi Li, and Andrej Risteski. How do transformers learn topic structure: Towards a mechanistic understanding. pp. 19689–19729. PMLR, 2023.

Huanxuan Liao, Shizhu He, Yupu Hao, Xiang Li, Yuanzhe Zhang, Kang Liu, and Jun Zhao. Skintern: Internalizing symbolic knowledge for distilling better cot capabilities into small language models. *arXiv preprint arXiv:2409.13183*, 2024.

Y Liu, M Ott, N Goyal, J Du, M Joshi, D Chen, O Levy, M Lewis, L Zettlemoyer, and V Stoyanov. Roberta: A robustly optimized bert pretraining approach. arxiv [preprint](2019). *arXiv preprint arXiv:1907.11692*, 1907.

David Lopez-Paz, Léon Bottou, Bernhard Schölkopf, and Vladimir Vapnik. Unifying distillation and privileged information. *arXiv preprint arXiv:1511.03643*, 2015.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=Bkg6RiCqY7`.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. *arXiv preprint arXiv:2104.08786*, 2021.

Ahmadreza Momeni, Kedar Tatwawadi, and U Stanford. Understanding lupi (learning using privileged information). *Ionosphere*, 201(7):6, 2018.

Rajeev Motwani. *Randomized Algorithms*. Cambridge University Press, 1995.

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. 2023.

Alex Nguyen and Gautam Reddy. Differential learning kinetics govern the transition from memorization to generalization during in-context learning. *arXiv preprint arXiv:2412.00104*, 2024.

Abhishek Panigrahi, Nikunj Saunshi, Haoyu Zhao, and Sanjeev Arora. Task-specific skill localization in fine-tuned language models. In *International Conference on Machine Learning*, pp. 27011–27033. PMLR, 2023.

Dmitry Pechyony and Vladimir Vapnik. On the theory of learnining with privileged information. *Advances in neural information processing systems*, 23, 2010.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. *arXiv preprint arXiv:2210.03350*, 2022.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

Gautam Reddy. The mechanistic basis of data dependence and abrupt learning in an in-context classification task. 2024.

Elyas Sabeti, Joshua Drews, Narathip Reamaroon, Elisa Warner, Michael W Sjoding, Jonathan Gryak, and Kayvan Najarian. Learning using partially available privileged information and label uncertainty: application in detection of acute respiratory distress syndrome. *IEEE journal of biomedical and health informatics*, 25(3):784–796, 2020.

Pier Giuseppe Sessa, Ilija Bogunovic, Andreas Krause, and Maryam Kamgarpour. Contextual games: Multi-agent learning with side information. *Advances in Neural Information Processing Systems*, 33:21912–21922, 2020.

Chenglei Si, Zhe Gan, Zhengyuan Yang, Shuohang Wang, Jianfeng Wang, Jordan Boyd-Graber, and Lijuan Wang. Prompting gpt-3 to be reliable. *arXiv preprint arXiv:2210.09150*, 2022.

Aaditya Singh, Stephanie Chan, Ted Moskovitz, Erin Grant, Andrew Saxe, and Felix Hill. The transient nature of emergent in-context learning in transformers. 36, 2024a.

Aaditya K Singh, Ted Moskovitz, Felix Hill, Stephanie CY Chan, and Andrew M Saxe. What needs to go right for an induction head? A mechanistic study of in-context learning circuits and their formation. *arXiv preprint arXiv:2404.07129*, 2024b.

DiJia Su, Sainbayar Sukhbaatar, Michael Rabbat, Yuandong Tian, and Qinqing Zheng. Dualformer: Controllable fast and slow thinking by learning with randomized reasoning traces. *arXiv preprint arXiv:2410.09918*, 2024.

Hongjin Su, Jungo Kasai, Chen Henry Wu, Weijia Shi, Tianlu Wang, Jiayi Xin, Rui Zhang, Mari Ostendorf, Luke Zettlemoyer, Noah A Smith, et al. Selective annotation makes language models better few-shot learners. *arXiv preprint arXiv:2209.01975*, 2022.

Gabriel Synnaeve, Thomas Schatz, and Emmanuel Dupoux. Phonetics embedding learning with side information. In *2014 IEEE Spoken Language Technology Workshop (SLT)*, pp. 106–111, 2014. doi: 10.1109/SLT.2014.7078558.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Vladimir Vapnik and Akshay Vashist. A new learning paradigm: Learning using privileged information. *Neural networks*, 22(5-6):544–557, 2009.

Vladimir Vapnik, Rauf Izmailov, et al. Learning using privileged information: similarity control and knowledge transfer. *J. Mach. Learn. Res.*, 16(1):2023–2049, 2015.

Boshi Wang, Xiang Yue, Yu Su, and Huan Sun. Grokked transformers are implicit reasoners: A mechanistic journey to the edge of generalization. *arXiv preprint arXiv:2405.15071*, 2024.

Boyi Wei, Kaixuan Huang, Yangsibo Huang, Tinghao Xie, Xiangyu Qi, Mengzhou Xia, Prateek Mittal, Mengdi Wang, and Peter Henderson. Assessing the brittleness of safety alignment via pruning and low-rank modifications. *arXiv preprint arXiv:2402.05162*, 2024.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022.

Zhiyong Wu, Yaoxiang Wang, Jiacheng Ye, and Lingpeng Kong. Self-adaptive in-context learning: An information compression perspective for in-context example selection and ordering. *arXiv preprint arXiv:2212.10375*, 2022.

Huazhe Xu, Yang Gao, Fisher Yu, and Trevor Darrell. End-to-end learning of driving models from large-scale video datasets. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2174–2182, 2017.

Linyi Yang, Shuibai Zhang, Zhuohao Yu, Guangsheng Bao, Yidong Wang, Jindong Wang, Ruochen Xu, Wei Ye, Xing Xie, Weizhu Chen, et al. Supervised knowledge makes large language models better in-context learners. *arXiv preprint arXiv:2312.15918*, 2023.

Sohee Yang, Elena Gribovskaya, Nora Kassner, Mor Geva, and Sebastian Riedel. Do large language models latently perform multi-hop reasoning? *arXiv preprint arXiv:2402.16837*, 2024.

Shunyu Yao, Binghui Peng, Christos Papadimitriou, and Karthik Narasimhan. Self-attention networks can process bounded hierarchical languages. pp. 3770–3785, 2021.

Dingli Yu, Simran Kaur, Arushi Gupta, Jonah Brown-Cohen, Anirudh Goyal, and Sanjeev Arora. Skill-mix: A flexible and expandable family of evaluations for ai models. *arXiv preprint arXiv:2310.17567*, 2023.

Ping Yu, Jing Xu, Jason Weston, and Ilia Kulikov. Distilling system 2 into system 1. *arXiv preprint arXiv:2407.06023*, 2024.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 32, 2019.

Rui Zhang, Feiping Nie, Muhan Guo, Xian Wei, and Xuelong Li. Joint learning of fuzzy k-means and nonnegative spectral clustering with side information. *IEEE transactions on Image Processing*, 28(5):2152–2162, 2018.

Haoyu Zhao, Abhishek Panigrahi, Rong Ge, and Sanjeev Arora. Do transformers parse while predicting the masked word? pp. 16513–16542, 2023.

Haoyu Zhao, Simran Kaur, Dingli Yu, Anirudh Goyal, and Sanjeev Arora. Can models learn skill composition from examples? In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Ziqian Zhong, Ziming Liu, Max Tegmark, and Jacob Andreas. The clock and the pizza: Two stories in mechanistic explanation of neural networks. 36, 2023.

Hattie Zhou, Arwen Bradley, Etai Littwin, Noam Razin, Omid Saremi, Josh Susskind, Samy Bengio, and Preetum Nakkiran. What algorithms can Transformers learn? A study in length generalization. *arXiv preprint arXiv:2310.16028*, 2023.

Jiaru Zou, Mengyu Zhou, Tao Li, Shi Han, and Dongmei Zhang. Promptintern: Saving inference costs by internalizing recurrent prompt during large language model fine-tuning. *arXiv preprint arXiv:2407.02211*, 2024.

Part I
Appendix

Table of Contents

A STRUCTURE OF THE APPENDIX

Here, we outline the structure for the appendix for easier readability. Due to space constraints, we had to defer a lot of details from the main paper. Appendix B.1 contains details on the experiments on gradient prediction accuracy from Section 4.4. Appendix B.4 shows the details on CoT internalization pipeline that we followed to get an $\mathbf{MLT}(d, n)$-ICL-capable model. Appendix B.5 gives more details on the context-enhanced learning experiments conducted in Section 3.

Appendix C shows mechanistic experiments, designed in Section 3.2, for additional experimental settings. Appendix C.3 present additional details and results for information recovery through querying experiments conducted in **??**. Appendix D presents extensive discussion of relevant related works.

Appendix E discusses few properties on the **MLT**. Appendix F presents the formal theorems on the computational hardness of learning the translation task and their proofs, which were informally outlined in Section 4.2 in the main paper. Appendix G then presents the formal statements and proofs for context-enhanced learning in the surrogate model, which were informally outlined in Section 4.1 in the main paper. Finally, Appendix H presents the theoretical construction of an ideal transformer that can simulate the surrogate model. We present extensive details on prompts, and examples of training sequences, that we used for context-enhanced learning in Appendix I.

B DEFERRED DEFINITIONS AND EXPERIMENT DETAILS FROM MAIN PAPER

B.1 GRADIENT PREDICTION ACCURACY

Our theoretical analysis in Theorem 4.6 was built on the fact that when a translation rule in a phrasebook is dropped by zeroing out a column in an in-context representation $\boldsymbol{C}_i$, the gradient for the corresponding column in $\boldsymbol{W}_i$ points to the direction of the dropped rule.

On the other hand, we show that when multiple rules are simultaneously dropped from the phrasebooks, the gradients for the trainable parameters become increasingly noisy. To quantify this degradation, we compute gradient for each column of the trainable parameters for an ICL-capable model, when the corresponding rule is dropped from phrasebooks by zeroing out the relevant column in the in-context representations. We then compute whether the computed gradient points to the right rule. By progressively increasing the number of simultaneously dropped rules, we measure the resulting degradation in the accuracy of the gradient's predictions.

More formally, denote RANDOM-DROP as an operation that takes in column dropping rates per layer $p_1, \cdots, p_d$, set of phrasebooks $\mathbf{\Pi}^*$, and returns in-context representations $\{\boldsymbol{C}_i\}_{i=1}^d$, such that $(\boldsymbol{C}_i)_{:,j} = (\texttt{Matrix}\,(\pi_i^*))_{:,j}$ (jth columns of $\boldsymbol{C}_i$ and $\texttt{Matrix}\,(\pi_i^*)$ are equal) with probability $1 - p_i$ and 0 otherwise. Then,

Definition B.1. For a set of column dropping rates per layer $p_1, \cdots, p_d$, set of phrasebooks $\mathbf{\Pi}^*$ and $\mathbf{MLT}(d, n)$-ICL-capable model, predictive accuracy of gradients is defined as

$$\mathbb{E}_{\{\boldsymbol{C}_i\}_{i=1}^d = \text{RANDOM-DROP}(p_1, \cdots, p_d, \mathbf{\Pi}^*)}$$
$$\mathbb{E}_{j \in [1, n^2] | (\boldsymbol{C}_1)_{:,j} = 0} \mathbb{I}\left[\texttt{HardMax}\left(-\nabla_{(\boldsymbol{W}_1)_{:,j}} \mathcal{L}\right) = (\texttt{Matrix}\,(\pi_i^*))_{:,j}\right]$$
$$\mathcal{L} = \mathbb{E}_{\boldsymbol{s}_1} \ell\left(\text{SURR-MLT}_{\{\boldsymbol{W}_i\}_{i=1}^d}\left(\{\boldsymbol{C}_i\}_{i=1}^d, \boldsymbol{V}_1\right), \mathbf{MLT}_{\mathbf{\Pi}^*}(\boldsymbol{s}_1)\right),$$

where ℓ, adapted from Section 2.1, computes cross-entropy loss on the predicted output embeddings of surrogate model using true output string and $\mathbb{I}$ denotes the indicator function.

The above definition computes gradients on the expected loss of the model. We primarily focus on predicting the trainable parameters of the first layer, i.e. $\boldsymbol{W}_1$, as that is the deepest layer in the surrogate model and intuitively should suffer the most with noise accumulation from dropped rules. On the other hand, we can further adapt the definition to compute the accuracy for batched gradients, where the gradients are computed using average loss on a randomly sampled batch of input sequences.

In Figure 5, we report the predictive accuracy of gradient for an $\mathbf{MLT}(d, n)$-ICL-capable model, and its behavior with varying batch size and the column dropping rates. We report for two cases, one where column dropping rates is non-zero only for the first phrasebook, and one where we increase the number of phrasebooks for which rules are independently and uniformly dropped. In both cases,

- Increased column dropping rates leads to noisier gradients and reduced prediction accuracy.
- Larger batch sizes improve gradient accuracy but cannot fully compensate for high dropout rates.
- Dropping rules from multiple phrasebooks significantly degrades gradient prediction accuracy.

B.2 Hidden Representations of a Model

A transformer f_θ with embedding dimension p and K layers takes any input sequence $\boldsymbol{x}$, say of length L, converts to an embedding matrix $\boldsymbol{H}_1 \in \mathbb{R}^{L\times p}$, and modifies the embeddings using a succession of K transformer layers; which we will denote by $f_\theta^{(1)}, f_\theta^{(2)}, \cdots, f_\theta^{(K)}$.

We refer to the hidden representations for the input $\boldsymbol{x}$, with embedding matrix $\boldsymbol{H}_1$, as the output of the model after every layer. We will denote them as $\boldsymbol{H}_{i+1} \in \mathbb{R}^{L\times p}$ for the output of layer $f_\theta^{(i)}$. That is,

$$\boldsymbol{H}_{i+1} = f_\theta^{(i)} \circ \cdots \circ f_\theta^{(2)} \circ f_\theta^{(1)}\left(\boldsymbol{H}_1\right), \quad \text{for all } i \geq 1.$$

ℓ_2-norm in change in hidden representation with perturbation in in-context information For an $\mathbf{MLT}(d, n)$-ICL-capable model, we supply in-context information for the textual description of a set of phrasebooks $\boldsymbol{\Pi}$ as $\textsc{str}(\boldsymbol{\Pi}) = [\textsc{str}(\pi_1), \cdots, \textsc{str}(\pi_d)]$. Our inputs to the transformer for an input string $\boldsymbol{s}_1$ will be of the form $[\textsc{str}\left(\boldsymbol{\Pi}\right), \boldsymbol{s}_1, \texttt{<THINK>}, \ldots, \boldsymbol{s}_{d+1}]$, where $\boldsymbol{s}_{d+1} = \mathbf{MLT}_{\boldsymbol{\Pi}}(\boldsymbol{s}_1)$. By the definition of hidden representations, $\boldsymbol{H}_2, \cdots, \boldsymbol{H}_{K+1}$ will denote the output of the transformer layers for this input string. However, we will be only interested in the hidden representations for the tokens involved in the tokens for $\texttt{<THINK>}, \ldots, \boldsymbol{s}_{d+1}$; and we will refer to the corresponding subsets of $\boldsymbol{H}_2, \cdots, \boldsymbol{H}_{K+1}$ that represent these specific tokens as $\boldsymbol{V}_2, \cdots, \boldsymbol{V}_{K+1}$.

Now, suppose we randomly take a phrasebook π_i in the $\boldsymbol{\Pi}$ and changed to a random phrasebook $\tilde{\pi}_i$. The corresponding textual description that will augment the context for an input string will then be $[\textsc{str}(\pi_1), \cdots, \textsc{str}(\pi_{i-1}), \textsc{str}(\tilde{\pi}_i), \textsc{str}(\pi_{i+1}), \cdots, \textsc{str}(\pi_d)]$. If $\tilde{\boldsymbol{V}}_2, \cdots, \tilde{\boldsymbol{V}}_{K+1}$ now denote the hidden representations that represent the tokens for $\texttt{<THINK>}, \ldots, \boldsymbol{s}_{d+1}$, then the ℓ_2-norm in the change of the hidden representation after layer j (for any $1 \leq j \leq K$) with the perturbation in the set of phrasebooks $\boldsymbol{\Pi}$ will be given by $\left\|\tilde{\boldsymbol{V}}_j - \boldsymbol{V}_j\right\|_2$.

B.3 Definition of a "stitched" model

We reuse notations from Appendix B.2. Suppose we have an $\mathbf{MLT}(d, n)$-ICL-capable model f_θ and an $\mathbf{MLT}_{\boldsymbol{\Pi}^*}$-capable model f_{θ^*}. Their corresponding transformer layers are denoted by $f_\theta^{(1)}, f_\theta^{(2)}, \cdots, f_\theta^{(K)}$ and $f_{\theta^*}^{(1)}, f_{\theta^*}^{(2)}, \cdots, f_{\theta^*}^{(K)}$. Each model takes in an input sequence and processes them with their K transformer layers.

Formally, we will write for the ICL-capable model. It takes in input sequence $\boldsymbol{x}$, and converts to an embedding matrix, say $\boldsymbol{H}_1$, and the output after the Klayers are given by:

$$\boldsymbol{H}_{K+1} = f_\theta^{(d)} \circ \cdots \circ f_\theta^{(2)} \circ f_\theta^{(1)}\left(\boldsymbol{H}_1\right).$$

Process of "stitching": The process of stitching takes in two parameters L_{start} and L_{end} and replaces all layers from L_{start} to L_{end} in f_θ with the corresponding layers in f_{θ^*} to give a "stitched" model, say $f_{\theta,\theta^*,L_{\text{start}},L_{\text{end}}}$. The output of the "stitched" model $f_{\theta,\theta^*,L_{\text{start}},L_{\text{end}}}$ on an input sequence $\boldsymbol{x}$ will be given by

$$\boldsymbol{H}_{K+1} = f_\theta^{(d)} \circ \cdots \circ f_\theta^{(L_{\text{end}}+1)} \circ \underbrace{f_{\theta^*}^{(L_{\text{end}})} \circ \cdots f_{\theta^*}^{(L_{\text{start}})}}_{\text{Layers are replaced by layers from } f_{\theta^*}} \circ f_\theta^{(L_{\text{start}}-1)} \circ f_\theta^{(2)} \circ f_\theta^{(1)}\left(\boldsymbol{H}_1\right).$$

B.4 PIPELINE ON COT INTERNALIZATION

We randomly sample M sets of phrasebooks $\mathbf{\Pi}_1, \ldots, \mathbf{\Pi}_M$ not equal to $\mathbf{\Pi}^*$. For each set of phrasebooks $\mathbf{\Pi}_i$, we randomly sample a single input sequence $\boldsymbol{s}_1$ and compute all the intermediate translation steps $\boldsymbol{s}_2, \ldots, \boldsymbol{s}_{d+1}$. We first train the model to do robust explicit CoT by auto-regressive training on sequences $[\textsc{str}(\mathbf{\Pi}_i), \boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_d, \boldsymbol{s}_{d+1}]$ with loss computed over $\boldsymbol{s}_2, \ldots, \boldsymbol{s}_{d+1}$.

Then we follow common CoT internalization strategies (Deng et al., 2024; Hao et al., 2024; Yu et al., 2024; Su et al., 2024) and gradually replace the intermediate sequences by <THINK> tokens in training. After all intermediate sequences have been replaced, the model has low loss on $\boldsymbol{s}_{d+1}$ with input $[\textsc{str}(\mathbf{\Pi}_1), \boldsymbol{s}_1, \texttt{<THINK>}, \ldots, \texttt{<THINK>}, \boldsymbol{s}_{d+1}]$, satisfying Definition 2.2. Since we only sample on sequence per set of phrasebooks, there is little memorization on particular phrasebooks.

Details on trainng hyperparameters: We use $M = 3 \times 10^5$ random sets of phrasebooks with length between 20 and 40, for getting $\mathbf{MLT}(5, 8)$-ICL-capable model, and $M = 10^6$ random sets of phrasebooks with length between 20 and 40, for getting $\mathbf{MLT}(5, 10)$-ICL-capable model. We use cosine learning rate schedule (Loshchilov & Hutter, 2016), with peak learning rate 10^{-4} and a 6% warmup phase, where learning rate is linearly increased from 0 to the peak. We use AdamW optimizer for optimization (Loshchilov & Hutter, 2019), with weight decay fixed at 10^{-4}. We use a batch size of 64 for training.

CoT internalization curriculum: For the first 10% fraction of training, we train the model with explicit CoT tokens that contain the intermediate steps in translation. Then between 10% to 60% fractions of training, CoT tokens are gradually replaced by <THINK> tokens, with the rate of replacement increasing linearly from 0% to 100%. We follow a deterministic first-to-last order for replacing CoT tokens; earlier CoT tokens are replaced first with <THINK> tokens. After that, the model is trained with the <THINK> CoT tokens till the end of training.

B.5 EXPERIMENT CONFIGURATION FOR CONTEXT-ENHANCED LEARNING

For context-enhanced learning, we create supervised datasets $D_{\mathbf{\Pi}^*}$ of different sizes; each containing between 10^4 to 10^6 samples. When performing **Annealing Dropout** or **Fixed Dropout**, at each step of training, we randomly perform dropout on the rules of all phrasebooks or apply dropout to the rules of a randomly sampled phrasebook to define curriculum-text. That is, if $\pi_1^*, \ldots, \pi_5^*$ represent the phrasebooks, then at each step of training, we either randomly drop rules uniformly from all of $\pi_1^*, \ldots, \pi_5^*$, or just drop from one of the phrasebooks randomly selected from $\pi_1^*, \ldots, \pi_5^*$, while keeping the rules of all other phrasebooks intact, to create curriculum-text.

Hyperparameters: Training hyperparameters are set equal to the optimization hyperparameters used in preparation of ICL-capable training phase (Appendix B.4), except we set weight decay to 0 in all experiments. We report the performance of the trained model after single epoch of training on each $D_{\mathbf{\Pi}^*}$ and plot against the size of the dataset in Figure 2.

C Additional Experiments Results

C.1 Selective Internalization of Context

In this experiment, we test whether the model can internalize rules that don't incur an increase in loss when dropped during training. To do so, we ablate on **Annealing Dropout**. We select a phrasebook and create 2 splits of rules in the phrasebook; one set of rules will be utilized by training samples for performing the translation task (which we call the training split), while other set of rules will appear in curriculum-text during training but never utilized for the translation task (which we call the heldout split).

At test time, we measure the performance of the trained model on 2 sets of evaluation examples, one that only use rules from the training split for their translation (equal to training distribution), and other that uses rules only from the heldout split for their translation (different from training distribution). Recall that the model is being measured without any phrasebooks information at evaluation.

We conduct the above experiment for each phrasebook, i.e. we create 5 sets of experiments where we only create heldout split for one specific level of phrasebook. In Figure 6, we show that the model fails to perform any translation that use the rules from the heldout split, in each experiment. This shows that the model only internalizes those rules that are important for the translation task for the training samples, and which incurs an increase in training loss when dropped.

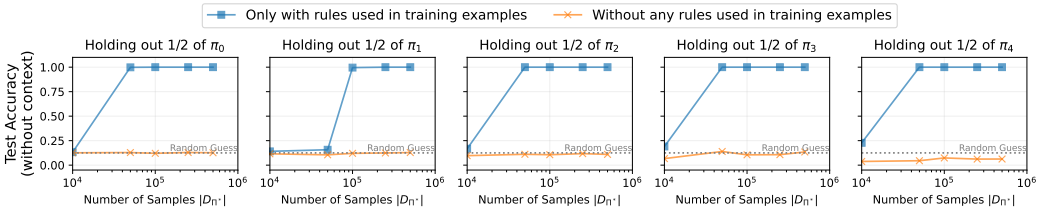
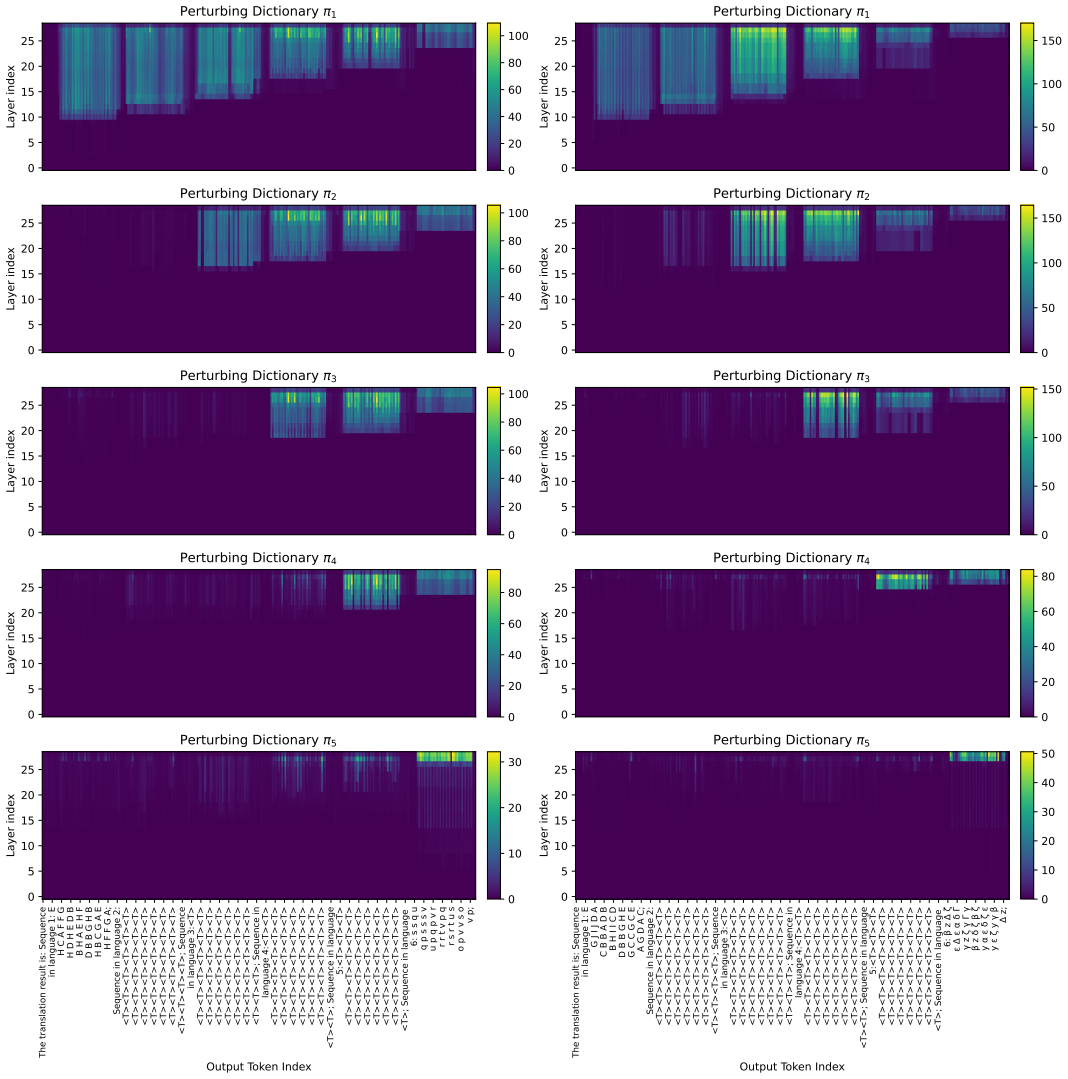


Figure 6: Ablations on **Annealing Dropout** for **MLT**$(5, 10)$ assess whether models internalize rules not used in training samples. (Left to right) $1 \leq i \leq 5$: Five independent experiments where a held-out split is created for phrasebook π_i^*, and training excludes samples that use the translation rules in the heldout split of π_i^*. Evaluation is conducted on two sets: one where samples do not use held-out rules from π_i^* and one where only held-out rules from π_i^* are used. The model's random performance on the latter indicates that it internalizes only rules used during training, particularly those whose removal increases loss.

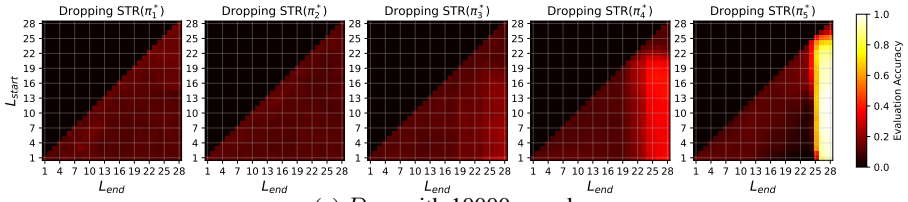
C.2 MECHANISTIC INSIGHTS

Here we provide additional figures corresponding to Figure 3 and Figure 4, but in more settings ($n = 10$ vs $n = 8$, **Annealing Dropout** vs **Fixed Dropout**).

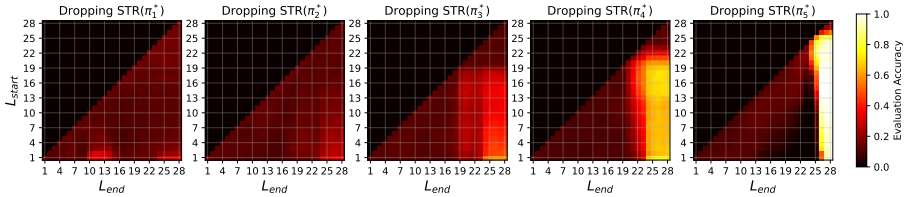


(a) **MLT**$(5, 8)$-ICL-capable Model (b) **MLT**$(5, 10)$-ICL-capable Model

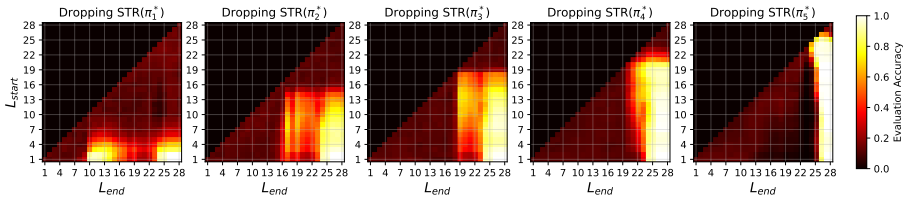
Figure 7: Evidence for sequential processing in **MLT**-ICL capable models ($n = 8, n = 10$). The left figure is identical to Figure 3. We observe the same behavior for **MLT**$(5, 10)$-ICL-capable model: perturbing later phrasebooks in the context changes output representations in the later layers. A clearer explanation on the context perturbation experiment is available in **??**



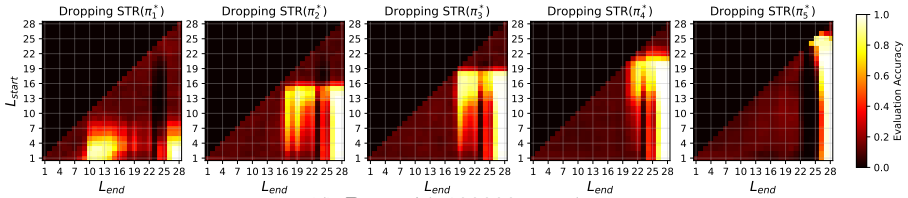
(a) $D_{\mathbf{\Pi}^*}$ with 10000 samples



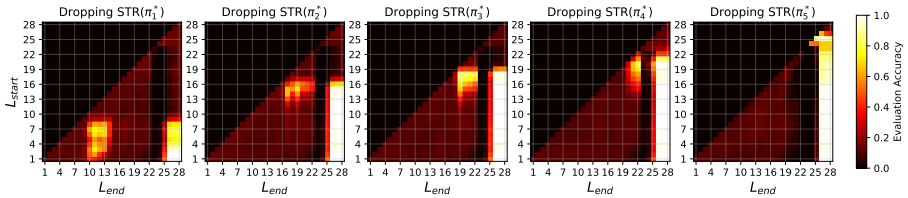
(b) $D_{\mathbf{\Pi}^*}$ with 25000 samples



(c) $D_{\mathbf{\Pi}^*}$ with 50000 samples

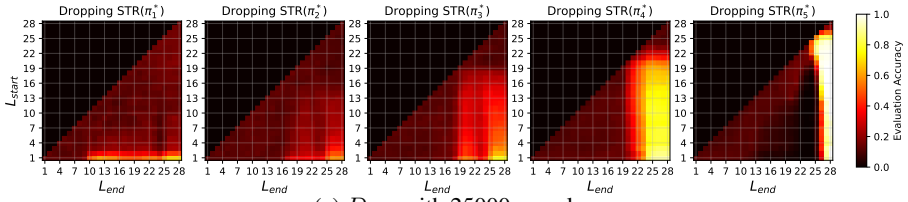


(d) $D_{\mathbf{\Pi}^*}$ with 100000 samples

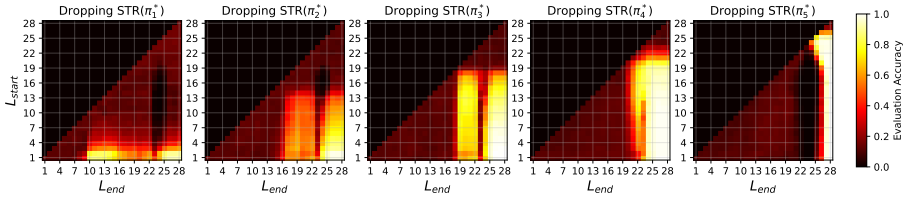


(e) $D_{\mathbf{\Pi}^*}$ with 250000 samples

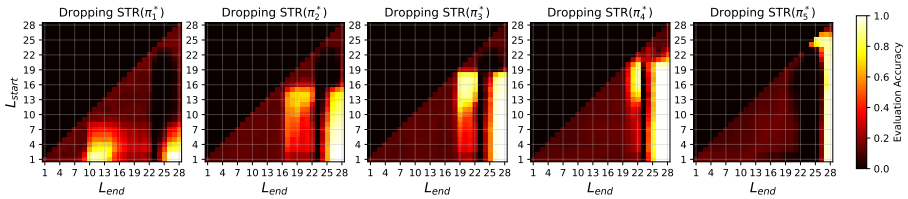
Figure 8: Repeated experiments from Figure 4 for models trained with context-enhanced learning at varying supervised dataset sizes (Plots for row (d) are identical to the plots in Figure 4). We observe that phrasebooks are progressively internalized with the number of training samples available during context-enhanced learning; later phrasebooks are internalized with fewer samples than the earlier ones. We observe similar localization patterns across layers from different phrasebooks across the models, however, we also observe that the localization patterns get increasingly sparser as training continues.



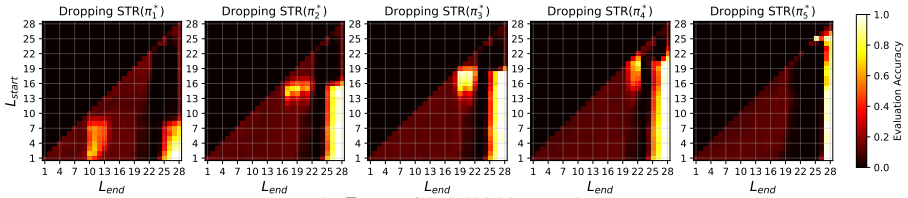
(a) $D_{\mathbf{\Pi}^*}$ with 25000 samples



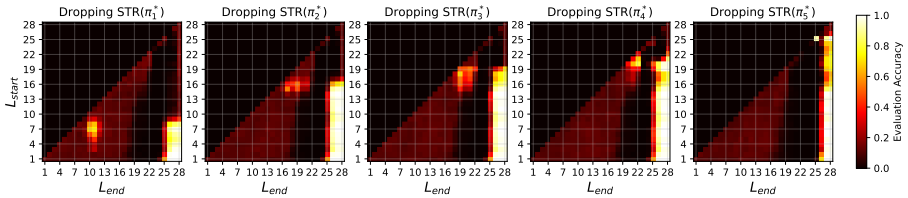
(b) $D_{\mathbf{\Pi}^*}$ with 50000 samples



(c) $D_{\mathbf{\Pi}^*}$ with 100000 samples



(d) $D_{\mathbf{\Pi}^*}$ with 250000 samples



(e) $D_{\mathbf{\Pi}^*}$ with 500000 samples

Figure 9: Repeated experiments from Figure 8 for models trained with **Fixed Dropout** instead of **Annealing Dropout**. Observations remain similar. This suggests that the position of phrasebook internalization is primarily decided by the **MLT**(d, n)-ICL-capable initialization, and less dependent on the specific dropout curriculum we use for context-enhanced learning.

C.3 (NON)VERBATIM MEMORIZATION OF PHRASEBOOK RULES

In this subsection, we provide a more detailed explanation for the evaluation on the feasibility of recovering phrasebook rules from models trained with context-enhanced learning with phrasebook excerpts in context.

Given a $\mathbf{MLT}_{\mathbf{\Pi}^*}$-capable model f_{θ^*} trained with context-enhanced learning where phrasebook rules from $\texttt{STR}(\mathbf{\Pi}^*)$ are provided within the context during training, we test if the model retains explicit memory of the textual format of rules. Namely, for each of the phrasebook rule of the form "`a b -> C D`" in $\texttt{STR}(\mathbf{\Pi}^*)$, whether it can complete the ground truth output tokens "`C D`" providing "`a b ->`" and additional context of the phrasebook $\texttt{STR}(\mathbf{\Pi}^*)$ before "`a b -> C D`". We conduct the test by computing the forward pass through f_{θ^*} with $\texttt{STR}(\mathbf{\Pi}^*)$ as the input (see exact input format in Figure 15).

Now we formally define the query strategies and metrics we used for creating **??**. Suppose there are h unique tokens and $\texttt{STR}(\mathbf{\Pi}^*)$ contains L tokens, let $\boldsymbol{S} \in \mathbb{R}^{h \times L}$ denote the corresponding output logit score matrix when we pass $\texttt{STR}(\mathbf{\Pi}^*)$ into f_{θ^*}. For a translation rule with ground truth output tokens of index $a_1, a_2 \in \boldsymbol{A}_{i+1} \subset [h]$, let $\boldsymbol{S}^{(k)}$ and $\boldsymbol{S}^{(k+1)} \in \mathbb{R}^h$ denote the logit vector corresponding to predicting these two entries.

If we are doing greedy decoding, then we will recover the correct phrasebook rule if and only if we greedily select both a_1 from $\boldsymbol{S}^{(k)}$ and a_2 from $\boldsymbol{S}^{(k+1)}$, so the probability of correctly recovering (a_1, a_2) conditioned on $\boldsymbol{S}$ is

$$\mathbb{P}\left[\text{Greedy-Recover}(a_1, a_2)\right] = \mathbb{1}\left[\arg\max_{i \in [h]} \boldsymbol{S}^{(k)} = a_1\right] \cdot \mathbb{1}\left[\arg\max_{i \in [h]} \boldsymbol{S}^{(k+1)} = a_2\right].$$

Meanwhile, if we apply random sampling with softmax temperature of 1, the probability of correctly recovering (a_1, a_2) conditioned on $\boldsymbol{S}$ is just then

$$\mathbb{P}\left[\text{Sampling-Recover}(a_1, a_2)\right] = \left(\frac{\exp\left(\boldsymbol{S}_{a_1}^{(k)}\right)}{\sum_{i \in [h]} \exp\left(\boldsymbol{S}_i^{(k)}\right)}\right) \cdot \left(\frac{\exp\left(\boldsymbol{S}_{a_1}^{(k+1)}\right)}{\sum_{i \in [h]} \exp\left(\boldsymbol{S}_i^{(k+1)}\right)}\right).$$

Recall from **??** that we have also introduced two stronger adversaries with additional *token filtering* when doing decoding: (1) setting probability of `<THINK>` tokens to zero (2) when querying for a rule in π_i with output alphabet $\boldsymbol{A}_{i+1}$, setting probability of all tokens outside $\boldsymbol{A}_{i+1}$ to zero.

Denoting the token index of `<THINK>` as $a_{\texttt{<THINK>}}$, then filter 1 corresponds to

$$\mathbb{P}\left[\text{Greedy-Filter1-Recover}(a_1, a_2)\right] = \mathbb{1}\left[\arg\max_{i \in [h] \setminus \{a_{\texttt{<THINK>}}\}} \boldsymbol{S}^{(k)} = a_1\right] \cdot \mathbb{1}\left[\arg\max_{i \in [h] \setminus \{a_{\texttt{<THINK>}}\}} \boldsymbol{S}^{(k+1)} = a_2\right],$$

$$\mathbb{P}\left[\text{Sampling-Filter1-Recover}(a_1, a_2)\right] = \left(\frac{\exp\left(\boldsymbol{S}_{a_1}^{(k)}\right)}{\sum_{i \in [h] \setminus \{a_{\texttt{<THINK>}}\}} \exp\left(\boldsymbol{S}_i^{(k+1)}\right)}\right) \cdot \left(\frac{\exp\left(\boldsymbol{S}_{a_1}^{(k)}\right)}{\sum_{i \in [h] \setminus \{a_{\texttt{<THINK>}}\}} \exp\left(\boldsymbol{S}_i^{(k+1)}\right)}\right).$$

Similarly, for filter 2, the probabilities are

$$\mathbb{P}\left[\text{Greedy-Filter2-Recover}(a_1, a_2)\right] = \mathbb{1}\left[\arg\max_{i \in \boldsymbol{A}_{i+1}} \boldsymbol{S}^{(k)} = a_1\right] \cdot \mathbb{1}\left[\arg\max_{i \in \boldsymbol{A}_{i+1}} \boldsymbol{S}^{(k+1)} = a_2\right],$$

$$\mathbb{P}\left[\text{Sampling-Filter2-Recover}(a_1, a_2)\right] = \left(\frac{\exp\left(\boldsymbol{S}_{a_1}^{(k)}\right)}{\sum_{i \in \boldsymbol{A}_{i+1}} \exp\left(\boldsymbol{S}_i^{(k+1)}\right)}\right) \cdot \left(\frac{\exp\left(\boldsymbol{S}_{a_1}^{(k)}\right)}{\sum_{i \in \boldsymbol{A}_{i+1}} \exp\left(\boldsymbol{S}_i^{(k+1)}\right)}\right).$$

Note that the second filter is a very strong adversarial assumption which assumes the user already has side information on the set of tokens contained in alphabet $\boldsymbol{A}_i$ and $\boldsymbol{A}_{i+1}$. To compute the final statistics, we sample 20 permutations of $\texttt{STR}(\mathbf{\Pi}^*)$ for forward passes and compute the mean of the above statistics over all atomic phrasebook rules appearing in the context. That is 1280 entries for each phrasebook in the case of $n = 8$ and 2000 entries for each phrasebook in the case of $n = 10$.

Table 1: Recovery Success Rate For $n = 8, d = 5$ Cases (Rounded to 2 decimals) Random Guess Baseline: 1.56%

Curriculum	# Samples	Query Method	Greedy Decoding		Sampling ($T = 1$)	
			$\pi_1 - \pi_4$	π_5	$\pi_1 - \pi_4$	π_5
Annealing Dropout	50000	Base	0.00%	0.00%	0.00%	0.01%
		Rule out <THINK>	0.06%	1.95%	0.00%	0.54%
		Only keeping $\boldsymbol{A}_i$	3.18%	1.95%	1.82%	1.49%
Annealing Dropout	100000	Base	0.00%	0.00%	0.00%	0.64%
		Rule out <THINK>	0.00%	0.08%	0.00%	1.25%
		Only keeping $\boldsymbol{A}_i$	1.37%	0.08%	1.69%	1.80%
Annealing Dropout	250000	Base	0.00%	0.23%	0.00%	1.25%
		Rule out <THINK>	0.00%	0.23%	0.00%	1.28%
		Only keeping $\boldsymbol{A}_i$	2.95%	0.23%	2.53%	1.38%
Fixed Dropout	50000	Base	0.00%	0.00%	0.00%	0.00%
		Rule out <THINK>	0.08%	0.78%	0.00%	0.09%
		Only keeping $\boldsymbol{A}_i$	0.82%	0.86%	1.48%	1.43%
Fixed Dropout	100000	Base	0.00%	0.00%	0.00%	0.21%
		Rule out <THINK>	0.43%	0.00%	0.03%	0.80%
		Only keeping $\boldsymbol{A}_i$	2.36%	0.00%	1.95%	1.32%
Fixed Dropout	250000	Base	0.00%	0.47%	0.02%	0.51%
		Rule out <THINK>	0.68%	1.09%	0.11%	0.73%
		Only keeping $\boldsymbol{A}_i$	2.23%	1.09%	2.19%	1.10%

Table 2: Recovery Success Rate For $n = 10, d = 5$ Cases (Rounded to 2 decimals) Random Guess Baseline: 1%

Curriculum	# Samples	Query Method	Greedy Decoding		Sampling ($T = 1$)	
			$\pi_1 - \pi_4$	π_5	$\pi_1 - \pi_4$	π_5
Annealing Dropout	100000	Base	0.00%	0.20%	0.00%	0.89%
		Rule out <THINK>	0.00%	0.20%	0.00%	0.90%
		Only keeping $\boldsymbol{A}_i$	1.66%	0.20%	1.28%	0.94%
Annealing Dropout	250000	Base	0.00%	1.80%	0.00%	1.12%
		Rule out <THINK>	0.00%	1.80%	0.00%	1.13%
		Only keeping $\boldsymbol{A}_i$	3.05%	1.80%	1.72%	1.23%
Fixed Dropout	250000	Base	0.00%	1.60%	0.00%	1.07%
		Rule out <THINK>	0.05%	1.60%	0.01%	1.07%
		Only keeping $\boldsymbol{A}_i$	2.46%	2.15%	1.99%	1.39%
Fixed Dropout	500000	Base	0.26%	2.05%	0.05%	1.13%
		Rule out <THINK>	0.33%	2.05%	0.07%	1.13%
		Only keeping $\boldsymbol{A}_i$	2.11%	2.05%	1.91%	1.34%

Here we report the query success rate for a variaty of models trained with context-enhanced learning using different curriculum on different datasets sizes. All models reaches nearly perfect test accuracy when no context is provided (see Figure 2), i.e., the in context phrasebook rules played significant role in the learning of the models.

For all runs, we can see that it is nearly impossible (with recovery probability $< 0.1\%$ in most cases) to recover the correct phrasebook rules for hidden steps $(\pi_1, \ldots, \pi_4)$ even when we provide the correct partial phrasebooks in context (see columns corresponding to Query Method "Base"). Ruling out <THINK> token when sampling also did not significantly increase the recovery rate. We note that the phrasebook knowledge **are not memorized in an completely undetectable manner**, as

the strongest token filtering gives non-random probability of outputting the correct target 2-tuple. However the probability is still very low ($< 3\%$), which can be considered as negligible to recover the full correct phrasebooks $\text{STR}(\mathbf{\Pi}^*)$.

D Related Works

Learning using Privileged Information (LUPI)

This was formally introduced by Vapnik & Vashist (2009) in kernel SVMs. A related concept is Learning with Side Information Kuusela & Ocone (2002); Jonschkowski et al. (2015); Zhang et al. (2018). Primarily, both concepts refer to training a model with additional information that could help training but may not be available at test time. This framework has been extended theoretically for classification tasks (e.g. Pechyony & Vapnik (2010); Momeni et al. (2018)) and has been used to explain benefits of knowledge distillation(Vapnik et al., 2015; Lopez-Paz et al., 2015). To name a few applications, this concept has been heavily studied for improving boosting for classification tasks (Chen et al., 2012), visual and video encoders Hoffman et al. (2016); Cheng et al. (2020); Xu et al. (2017), product preference predictions Farias & Li (2019), multi-agent games (Sessa et al., 2020), speech recognition (Synnaeve et al., 2014), and medical recognition (Ceccarelli & Maratea, 2008; Sabeti et al., 2020).

While traditionally applied to classification, extending LUPI to large language models (LLMs) introduces unique challenges due to their auto-regressive training. Due to the auto-regressive nature of training LLMs, such a concept raises questions on whether applying auto-regressive loss on the additional information is necessary to get the benefits of additional supervision information in context, and how the additional information changes the training behavior of LLMs. Hence, our work is a non trivial generalization of this framework to LLMs, that connects to their in-context learning strengths.

Differences with Masked Language Modeling (MLM) and Language Infilling

MLM models like BERT, RoBERTa, and T5 (Kenton & Toutanova, 2019; Liu et al., 1907; Raffel et al., 2020) train models by either masking or removing tokens from a sequence and compute loss on the predictions of the model on the missing tokens. This concept has been adapted for training auto-regressive models, with a task called language infilling task (Bavarian et al., 2022; Li et al., 2022; Donahue et al., 2020; Li et al., 2022). The primary difference from these works is that context-enhanced learning doesn't take loss on the removed context tokens when they are removed from curriculum-text.

In-context learning and memorization

Recent works have studied emergence of ICL and competition with in-weights learning (IWL) (equivalent to knowledge memorization) during pre-training of language models (Chan et al., 2022; Reddy, 2024; Singh et al., 2024a;b; Nguyen & Reddy, 2024). These studies show that data distribution properties affect the behavior of the model during training. Our work can be thought of as contemporary to these studies, where we show that strong ICL capabilities can be utilized for improving knowledge on a task by context-enhanced learning.

Benefits of in-context learning: In-context learning has been primarily studied in the context of few-shot prompting of large language models. Better supervision with in-context supervision can help in improved performance (e.g. some representative works (Arora et al., 2022; Si et al., 2022; Wu et al., 2022; Lu et al., 2021; Su et al., 2022)), OOD generalization and factuality (reduced hallucination) (Yang et al., 2023; Dhuliawala et al., 2023; Chen et al., 2023; Didolkar et al., 2024) for large language models in prompting. On the other hand, we show that improved supervision with in-context supervision can also help a model learn faster in SFT, while seemingly not leaking the in-context information in its output probabilities.

Compositional and OOD generalization

Measuring generalization for a transformer beyond training distribution has been a study of interest in many prior works. OOD generalization is measured by training a transformer on simpler examples

and measuring its performance on harder ones. Prominent studies include length generalization, informally defined as the ability of the model to reason longer than what it has been trained on (Zhou et al., 2023; Anil et al., 2022), and compositional generalization on concepts, defined as the ability of the model to reason on composition of concepts that it has seen during training (Zhao et al., 2024; Yu et al., 2023; Wang et al., 2024; Yang et al., 2024; Press et al., 2022; Allen-Zhu & Li, 2023b). Our experiments on **Fixed Dropout** in Section 3, where we train with 20% dropout on the phrasebooks information but measure performance with 100% dropout at test time, measures compositional OOD generalization behavior of the language model. The results show that the model internalizes the rules from the phrasebooks in an atomic way, and re-compose them together as necessary at test time.

MECHANISTIC BEHAVIOR OF TRANSFORMERS WITH SYNTHETIC DATASETS:

Our work builds on a growing body of research exploring the behavior of transformers trained on synthetic datasets. Prior studies have examined tasks such as modular addition (Nanda et al., 2023; Zhong et al., 2023), context-free grammars (Zhao et al., 2023; Allen-Zhu & Li, 2023a), regular and n-gram languages (Bhattamishra et al., 2020; Yao et al., 2021; Akyürek et al., 2024; Li et al., 2023), and synthetic article-style datasets (Allen-Zhu & Li, 2023b; 2024; Eldan & Li, 2023). While our work is structurally similar to these studies, it investigates the benefits of context-enhanced learning that has not been explored in previous works.

E PROPERTIES OF **MLT**

MLT(d, n) is defined by the phrasebooks $\pi_1, \cdots, \pi_d$ at each of its translation layers. We use $\mathcal{B}^\pi$ as all possible set of bijective maps that can be used for any phrasebook $\pi \in \{\pi_1, \cdots, \pi_d\}$. We will use variable π to refer to an arbitrary bijective map from the set $\mathcal{B}^\pi$. For simplicity of proof, we will refer to any alphabet set $\boldsymbol{A}$ of size n as $\{0, 1, \cdots, n-1\}$.

Here, we formally mention some of the properties of **MLT**(d, n).

Lemma E.1 (Invertibility of sequence translation)**.** *For any level $i \in [d]$, fixing $\{\pi_i, \pi_{i+1}, \ldots, \pi_d\}$ gives a bijection between $\boldsymbol{s}_i$ and $\boldsymbol{s}_{d+1}$.*

Proof. All of the four operations involved in the mapping from $\boldsymbol{s}_i$ to $\boldsymbol{s}_{i+1}$ are invertible. □

Structure of the mappings: The mappings π_i are selected as random bijective maps between 2-tuples of characters in $\boldsymbol{A}_i^2$ and 2-tuples of characters in $\boldsymbol{A}_{i+1}^2$. A combinatorial argument can then give the number of such possible phrasebooks to be $n^2!$.

Lemma E.2 (Number of mappings in each level)**.** *The number of possible bijective maps between 2-tuples of characters from alpbhabet sets $\boldsymbol{A}_i, \boldsymbol{A}_{i+1}$, each being of size n, is $n^2!$, i.e. $|\mathcal{B}^\pi| = n^2!$.*

Proof. Each alphabet set contains n^2 possible 2-tuples, meaning there are two sets of n^2 elements in $\boldsymbol{A}_i$ and $\boldsymbol{A}_{i+1}$ that must be mapped via a bijective function. Fixing the order of 2-tuples in $\boldsymbol{A}_{i+1}^2$ reduces the problem to ordering the 2-tuples in $\boldsymbol{A}_i^2$. The number of possible orderings is $n^2!$. □

Importance of *Circular shift*: The composition of the d random bijective maps can be demonstrated to result in another random bijective map. Consequently, without the *Circular shift*, each character in the output sequence $\boldsymbol{s}_{d+1}$ depends on only two characters from the input sequence $\boldsymbol{s}_1$ via a shared random map across 2-tuples.

Incorporating *Circular shift* on the other hand enables each character in the output sequence to depend on $2d$ characters from the input sequence. This is because the character positions are shifted to the right at each step, causing the input 2-tuples to also shift to the right at each stage. As we will elaborate later, incorporating *Circular shift* increases the required number of training samples to learn the set of phrasebooks from input and label pairs to $n^{\Omega(d)}$, whereas without *Circular shift*, this requirement is only $\mathcal{O}(n^2)$.

Representing the mappings on 2-tuples in-context: Each map can be defined using $\mathcal{O}(n^2)$ characters, as it can be simply defined by the n^2 relations each connecting 2 unique random 2-tuples from their corresponding alphabet sets. Thus, defining d maps in-context will require $\mathcal{O}(n^2 d)$ characters. In contrast, describing all possible random mappings that connect d characters in input sequence to a character in output sequence will require $\Omega(d\binom{n}{d})$ characters. Thus, **MLT** with d translation steps involving random mappings on 2-tuples and *Circular shift* helps define a mapping where each character in the output sequence can depend on d character in the input sequence, and the set of phrasebooks can be described using $\mathcal{O}(n^2 d)$ characters.

F Lower Bound: Hardness of Learning $\mathbf{MLT}(d, n)$ without Context

F.1 Brief introduction to SQ framework

Statistical query (SQ) bounds : The statistical query (SQ) framework measures the computational hardness of learning a task in the presence of noise. It measures the hardness of learning a task by the number of statistical queries needed by a learning algorithm to learn the true function. Statistical queries are defined by some polynomially-computable property Q of labeled instances and a tolerance parameter $\tau \in [0, 1]$ over $(x, y) \sim D$ where D is the data distribution. For a query, the algorithm receives a response from the oracle within τ error of the true value. The statistical dimension, or SQ-dim, is measured in terms of the number of functions in the hypothesis class that the learning algorithm need to distinguish and the number of queries necessary to do the same. Correlation between two functions is used to define the statistical query dimension.

Definition F.1. Correlation of two functions f_1, f_2 on a domain $\mathcal{X}$ with respect to a distribution $\mathcal{D}$ is given by

$$\text{Correlation}(f_1, f_2, \mathcal{D}) := \left| \Pr_{x \in \mathcal{D}}[f_1(x) = f_2(x)] - \Pr_{x \in \mathcal{D}}[f_1(x) \neq f_2(x)] \right|.$$

For functions $f_1, f_2 : \mathcal{X} \to \{0, 1\}$, the above definition is also equivalent to

$$\text{Correlation}(f_1, f_2, \mathcal{D}) := \left| 1 - 2\mathbb{E}_{x \in \mathcal{D}}[f_1(x) \oplus f_2(x)] \right|.$$

Remark F.2. Two functions $f_1, f_2 : \mathcal{X} \to \{0, 1\}$ are said to be uncorrelated w.r.t. $\mathcal{D}$ if

$$\Pr_{x \in \mathcal{D}}[f_1(x) = f_2(x)] = \Pr_{x \in \mathcal{D}}[f_1(x) \neq f_2(x)]$$
$$\text{(alternately) } \mathbb{E}_{x \in \mathcal{D}}[f_1(x) \oplus f_2(x)] = \frac{1}{2}.$$

On the other hand, if

$$Pr_{x \in \mathcal{D}}[f_1(x) = f_2(x)] = 1 \quad \text{(or 0)}$$
$$\text{(alternately) } \mathbb{E}_{x \in \mathcal{D}}[f_1(x) \oplus f_2(x)] = 0 \quad \text{(or 1)},$$

then $\text{Correlation}(f_1, f_2, \mathcal{D}) = 1$.

We take the following formal definitions of SQ-dim and its relation to computational hardness in the SQ framework from Blum et al. (1994).

Definition F.3 (Definition 2 in Blum et al. (1994))**.** For a function class $\mathcal{F}$ of boolean functions over $\{0, 1\}^n$ and $\mathcal{D}$ a distribution over $\{0, 1\}^n$, SQ-dim$(\mathcal{F}, \mathcal{D})$, the statistical query dimension of $\mathcal{F}$ with respect to *distribution*, is defined to be the largest natural number μ such that $\mathcal{F}$ contains μ functions $f_1, \cdots, f_\mu$ with the property that for all $i \neq j$ we have:

$$\text{Correlation}(f_i, f_j, \mathcal{D}) := \left| \Pr_{x \in \mathcal{D}}[f_i = f_j] - \Pr_{x \in \mathcal{D}}[f_i \neq f_j] \right| \leq \frac{1}{\mu^3}.$$

Theorem F.4 (Theorem 12 in Blum et al. (1994))**.** *Let $\mathcal{F}$ be a class of functions $\{0, 1\}^n$ and $\mathcal{D}$ a distribution such that SQ-dim$(\mathcal{F}, \mathcal{D}) \geq \mu \geq 16$. Then if all queries are made with a tolerance of atleast $\frac{1}{\mu^{1/3}}$, at least $\mu^{1/3}/2$ queries are required to learn $\mathcal{F}$ with error less than $1/2 - 1/\mu^3$ in the statistical query model.*

F.2 LOWER BOUND LEMMA

Here, we mention the 2 main theorems that study the SQ dimension of the **MLT** task of interest. The first theorem shows the SQ dimension bound when number of characters $n = 2$, which is then adapted to get the SQ dimension bound for general n. Proofs of both the theorems are given in Appendix F.4 and Appendix F.5 respectively.

Theorem F.10 (SQ dimension for $n = 2$)**.** *The family of translation task* $\boldsymbol{MLT}(d, 2)$ *on input distribution* $\mathcal{U}(\{0,1\}^{2d})$ *has statistical query dimension SQ-dim(*$\boldsymbol{MLT}(d, 2)$*) atleast* $2^{\Omega(d)}$.

Theorem F.18 (SQ dimension for general n)**.** *For the translation task* $\boldsymbol{MLT}(d, n)$ *that has depth* d *and* n *characters per level, the statistical query dimension SQ-dim(*$\boldsymbol{MLT}(d, n)$*) is atleast* $n^{\Omega(d)}$.

F.2.1 BOUNDS FOR SGD

The following corollary measures the sample complexity needed to learn $\mathbf{MLT}_{\mathbf{\Pi}^*}$ by SGD. It has been adapted from proposition 3 in Edelman et al. (2023), who study the sample complexity for learning d-sparse parity task on n length sequences, whose SQ dimension is $\binom{n}{d} = \Theta(n^d)$. We simply state the corollary without specifying the proof.

Loss function and SGD updates: Consider training of a model f_θ with r parameters that is trained with mean squared error, i.e. $\ell_{\text{auto}}\left(f_\theta([\text{CURR}_g(x,t), x, y]), y\right) = \left\|f_\theta([\text{CURR}_g(x,t), x, y]) - y\right\|^2$. The empirical loss on a batch S of batch size B from the supervised dataset $D_{\mathbf{\Pi}^*}$ will be denoted by $L_S(f_\theta, \mathbf{MLT}_{\mathbf{\Pi}^*})$; the population loss will be denoted by $L_{\mathcal{U}(\{0,1\}^L)}(f_\theta, \mathbf{MLT}_{\mathbf{\Pi}^*})$. SGD updates are of the form:

$$\theta_{t+1} = \theta_t - \eta_t(\nabla_\theta L_{S_t}(f_{\theta_t}, \mathbf{MLT}_{\mathbf{\Pi}^*}) + R(\theta_t) + \zeta_t).$$

for some sample S_t, step size η_t, regularizer $R(\cdot)$, and adversarial noise $\zeta_t \in [-\tau, \tau]^r$. For simplicity, we assume the gradient $\nabla_\theta L_{S_t}(\theta_t)$ is bounded on all parameters θ in parameter space of the model.

Fake trajectory: Suppose $\mathbf{0}$ denote the constant function that maps all inputs to 0. Then, with the contemporary losses $L_S(f_\theta, \mathbf{0})$ and $L_{\mathcal{U}(\{0,1\}^L)}(f_\theta, \mathbf{0})$ that computes difference from this constant function, consider the following trajectory $\tilde{\theta}_1, \cdots, \tilde{\theta}_t, \cdots$ starting from the same initiation θ_0:

$$\tilde{\theta}_{t+1} = \tilde{\theta}_t - \eta_t(\nabla_\theta L_{S_t}(f_{\tilde{\theta}_t}, \mathbf{0}) + R(\tilde{\theta}_t)).$$

Assumption F.5. For all t, suppose $\left\|\nabla_\theta L_{\mathcal{U}(\{0,1\}^L)}(f_{\tilde{\theta}_t}, \mathbf{0}) - \nabla_\theta L_{S_t}(f_{\tilde{\theta}_t}, \mathbf{0})\right\|_2 \leq \tau/2$.

Corollary F.6 (Sample complexity bounds for SGD)**.** *Fix an initialization* θ_0 *such that* f_{θ_0} *is statistically independent (correlation 0) from all possible* $\boldsymbol{MLT}_{\mathbf{\Pi}^*}$*. Further, set* $\tau = \mathcal{O}(n^{-d})$*. Under this assumption, if* $LTB \leq n^{\Omega(d)}/r$*, then there exists at least one task* $\boldsymbol{MLT}_{\mathbf{\Pi}^*} \in \boldsymbol{MLT}(n, d)$ *for which the functions obtained after the first* T *SGD updates,* $f_{\theta_1}, \ldots, f_{\theta_T}$*, remain statistically independent (correlation 0) from* $\boldsymbol{MLT}_{\mathbf{\Pi}^*}$ *despite training on it.*

LTB is the product of input length, total training steps, and batch size, which represents sample complexity used by the SGD algorithm.

F.3 USEFUL LEMMAS FOR $n = 2$

Here, we mention some useful lemmas for charaterizing the bijective maps on $\{0,1\}^2 \to \{0,1\}^2$ that we will regularly use for the proof of Theorem F.10. The first lemma will show that each bijective map can be represented by three operations on input characters, **copy**, **xor**, and **not** operations. The second lemma measures correlation on the outputs of two randomly sampled bijective maps. The proofs are given in Appendix F.6.

We define the following necessary operations to describe the random bijective maps on $\{0,1\}^2 \to \{0,1\}^2$:

1. **copy**: Given a tuple (x_1, x_2), and a position argument $i \in \{1, 2\}$, the operation returns value of x_i as output. We will use $\mathbf{copy}_1$ and $\mathbf{copy}_2$ to indicate the **copy** operation on positions 1 and 2 respectively.

2. **not**: Given a variable x_i, this operation returns flipped value of x_i. That is, if $x_i = 1$, then it returns 0 and vice-versa.
3. **xor**: Given a tuple (x_1, x_2), this operation returns $x_1 \oplus x_2$.

Lemma F.19 (Formulation of bijective maps for $n = 2$)**.** *Any bijective map $\pi : \{0,1\}^2 \to \{0,1\}^2$ can be expressed using **copy**, **not**, and **xor** operations. Furthermore, from 24 possible maps for π,*

1. *There are 6 maps $\mathbf{\Delta}_1, \mathbf{\Delta}_2, \cdots, \mathbf{\Delta}_6$ for which characters in the output tuple can be defined by **copy** and **xor** operations on the characters in the input tuple.*
2. ***Mirror maps:** For each map $\pi \in \{\mathbf{\Delta}_1, \mathbf{\Delta}_2, \cdots, \mathbf{\Delta}_6\}$, there exist mirror maps $\pi_{(1)}, \pi_{(2)}, \pi_{(3)}$ whose output on each input tuple can be defined by selective **not** operations on either or both characters of the output tuple of π. We call $\{\pi, \pi_{(1)}, \pi_{(2)}, \pi_{(3)}\}$ as a **mirror map set** of π, in short, Mirrorset(π).*

Remark F.7. Mirror map set definition is general and isn't restricted to maps $\pi \in \{\mathbf{\Delta}_1, \mathbf{\Delta}_2, \cdots, \mathbf{\Delta}_6\}$. Because mirror maps are defined in terms of **not** operation, a mirror map set Mirrorset(π) for $\pi \in \{\mathbf{\Delta}_1, \mathbf{\Delta}_2, \cdots, \mathbf{\Delta}_6\}$ is also equivalent to Mirrorset$(\pi_{(i)})$ for $i \in \{1, 2, 3\}$.

Remark F.8. Lemma F.19 can be re-stated as follows: the 24 possible bijective maps $\{0,1\}^2 \to \{0,1\}^2$ can be grouped into 6 family of maps, each containing 4 maps and represented by a map from $\{\mathbf{\Delta}_1, \mathbf{\Delta}_2, \cdots, \mathbf{\Delta}_6\}$. Each family is defined by **mirror map set** of their corresponding representative map.

Lemma F.20 (Correlation of bijective maps for $n = 2$)**.** *For two randomly selected maps $\pi^\alpha, \pi^\beta : \{0,1\}^2 \to \{0,1\}^2$, the following hold true.*

1. ***Correlation at output characters:** Fix an $i, j \in \{0,1\}$. With probability $\frac{1}{3}$ w.r.t. the random selection of the maps, the ith character in the output of π^α has non-zero correlation to jth character in the output of π^β, i.e.*

$$\begin{aligned} &\textit{Correlation}(\pi^\alpha(\cdot)_i, \pi^\beta(\cdot)_j, \mathcal{U}(\{0,1\}^2)) \\ &:= \left| \Pr_{x \sim \mathcal{U}(\{0,1\}^2)}[\pi^\alpha(x)_i \neq \pi^\beta(x)_j] - \Pr_{x \sim \mathcal{U}(\{0,1\}^2)}[\pi^\alpha(x)_i = \pi^\beta(x)_j] \right| = 1. \end{aligned}$$

2. *With probability $\frac{1}{3}$ w.r.t. the random selection of the maps, both the characters in the outputs of π^α, π^β have non-zero correlation, i.e. either one of the cases hold true*

$$\begin{aligned} &\textit{Correlation}(\pi^\alpha(\cdot)_1, \pi^\beta(\cdot)_1, \mathcal{U}(\{0,1\}^2)) = 1. \\ &\textit{Correlation}(\pi^\alpha(\cdot)_2, \pi^\beta(\cdot)_2, \mathcal{U}(\{0,1\}^2)) = 1. \end{aligned}$$

or

$$\begin{aligned} &\textit{Correlation}(\pi^\alpha(\cdot)_1, \pi^\beta(\cdot)_2, \mathcal{U}(\{0,1\}^2)) = 1. \\ &\textit{Correlation}(\pi^\alpha(\cdot)_2, \pi^\beta(\cdot)_1, \mathcal{U}(\{0,1\}^2)) = 1. \end{aligned}$$

Other cases are not possible,
i.e. for any $i \in \{1,2\}$, both Correlation$(\pi^\alpha(\cdot)_i, \pi^\beta(\cdot)_1, \mathcal{U}(\{0,1\}^2)) > 0$ and Correlation$(\pi^\alpha(\cdot)_i, \pi^\beta(\cdot)_2, \mathcal{U}(\{0,1\}^2)) = 1$ can't hold true.

Remark F.9. Implication of Lemma F.20 is as follows: With probability at most $\frac{1}{3}$, output of two random maps can stay correlated. This would suggest that the probability of the output of two chains of random maps staying correlated should decay exponentially with the length of the chains.

F.4 Proof for Statistical dimension lower bound for $n = 2$

We repeat the lemma of interest for presentation.

Theorem F.10 (SQ dimension for $n = 2$)**.** *The family of translation task* ***MLT***$(d, 2)$ *on input distribution* $\mathcal{U}(\{0,1\}^{2d})$ *has statistical query dimension SQ-dim(****MLT***$(d, 2))$ *atleast* $2^{\Omega(d)}$.

Proof. We narrow our argument to the first character output of the task. The proof goes through 2 major steps.

1. First, we show that two random instances of **MLT**$(d, 2)$, defined by 2 chains of random maps $\{\pi_1^\alpha, \cdots, \pi_d^\alpha, \}$ and $\{\pi_1^\beta, \cdots, \pi_d^\beta, \}$ as their phrasebooks at each level, will be uncorrelated with probability at least $1 - (1/3)(7/9)^{d-1}$ w.r.t. the selection of the random code-books. The lemma is formally given in Lemma F.11.
2. Then, we can apply a chernoff bound to show that we can have $2^{\Omega(d)}$ instances of **MLT**$(d, 2)$ that will have zero pairwise correlation of their output.

By the definition of SQ-dim from Definition F.3, the above observations suggest that SQ-dim(**MLT**$(d, 2)) \geq 2^{\Omega(d)}$. □

F.4.1 AUXILIARY LEMMAS

Here, we will prove the following primary lemma, that is used to prove Theorem F.10.

Lemma F.11. *With probability at least* $1 - \frac{1}{3}\left(\frac{7}{9}\right)^{d-1}$ *w.r.t. random map selection, the following holds true for* 2 *sets of random maps* $\{\pi_1^\alpha, \cdots, \pi_d^\alpha\}$ *and* $\{\pi_1^\beta, \cdots, \pi_d^\beta\}$*:*

$$\textit{Correlation}(\textbf{\textit{MLT}}(d,2)_{\mathbf{\Pi}=\{\pi_i^\alpha\}_{i=1}^d}(\cdot)_1, \textbf{\textit{MLT}}(d,2)_{\mathbf{\Pi}=\{\pi_i^\beta\}_{i=1}^d}(\cdot)_1, \mathcal{U}(\{0,1\}^{2d})) = 0.$$

Proof. We first dive into the dependencies between characters in input and output sequences in the translation process. In Lemma F.13, we show that at any step i of **MLT**$(d, 2)$ with phrasebooks $\{\pi_i\}_{i=1}^d$, the following relation holds true on an input $\boldsymbol{s}_1 \in \{0,1\}^{2d}$:

$$(\boldsymbol{s}_{i+1,1}, \boldsymbol{s}_{i+1,2}) = \pi_i((\boldsymbol{s}_{i+1,2}, \boldsymbol{s}_{i+1,3})).$$

We will use superscripts α and β to differentiate the intermediate outputs of **MLT** when the phrasebooks are set as $\{\pi_1^\alpha, \cdots, \pi_d^\alpha\}$ and $\{\pi_1^\beta, \cdots, \pi_d^\beta\}$ respectively. We will use an induction strategy to find the correlation of $\boldsymbol{s}_{d+1,1}^\alpha$ and $\boldsymbol{s}_{d+1,1}^\beta$. To do so, we will require the following variables:

1. $p_{\ell,ij}$: For one pair of $i, j \in \{2, 3\}$, this represents the probability at a level $2 \leq \ell \leq 1 + d$ w.r.t. the randomness of $\{\pi_1^\alpha, \cdots, \pi_\ell^\alpha\}$ and $\{\pi_1^\beta, \cdots, \pi_\ell^\beta\}$, that $\boldsymbol{s}_{\ell,i}^\alpha$ and $\boldsymbol{s}_{\ell,j}^\beta$ are perfectly correlated. Additionally, we define $p_{\ell,11}$ that represents the probability at a level $1 \leq \ell \leq 1 + d$ w.r.t. the randomness of $\{\pi_1^\alpha, \cdots, \pi_\ell^\alpha\}$ and $\{\pi_1^\beta, \cdots, \pi_\ell^\beta\}$, that $\boldsymbol{s}_{\ell,1}^\alpha$ and $\boldsymbol{s}_{\ell,1}^\beta$ are perfectly correlated.
2. $p_{\ell,\text{both}}$: This represents the probability at a level $2 \leq \ell \leq 1 + d$ w.r.t. the randomness of $\{\pi_1^\alpha, \cdots, \pi_\ell^\alpha\}$ and $\{\pi_1^\beta, \cdots, \pi_\ell^\beta\}$, that either $\boldsymbol{s}_{\ell,2}^\alpha$, $\boldsymbol{s}_{\ell,2}^\beta$ are correlated and $\boldsymbol{s}_{\ell,3}^\alpha$, $\boldsymbol{s}_{\ell,3}^\beta$ are correlated, or $\boldsymbol{s}_{\ell,2}^\alpha$, $\boldsymbol{s}_{\ell,3}^\beta$ are correlated and $\boldsymbol{s}_{\ell,3}^\alpha$, $\boldsymbol{s}_{\ell,2}^\beta$ are perfectly correlated.

There are three relations that we need to keep in mind, before we proceed with the induction proof.

1. First, the correlations are circular in nature, i.e., for $j \in \{2, 3\}$ and any integer k, $p_{\ell,jj} = p_{\ell,j(j+2k)\%L}$ (Lemma F.15). We will use this relation to connect $p_{\ell,11}$ with $p_{\ell,33}$, i.e.

$$p_{\ell,11} = p_{\ell,33}. \tag{1}$$

2. Second, $p_{\ell,\text{both}} \leq \max_{ij} p_{\ell,ij}$. Intuitively, this is because getting correlations at both positions must be at most as probable as getting correlations at one of the positions.
$$p_{\ell,\text{both}} \leq \max_{ij} p_{\ell,ij}. \tag{2}$$

3. Third, cross-correlation probabilities given by $p_{\ell,23}$ and $p_{\ell,32}$ must be $\leq \frac{1}{2}(p_{\ell,22} + p_{\ell,33})$ (Lemma F.17).
$$p_{\ell,ij} \leq \frac{1}{2}(p_{\ell,ii} + p_{\ell,jj}), \text{ for any } i,j \in \{2,3\}, i \neq j. \tag{3}$$

Base condition: We will use the values of the variables at $\ell = 2$ as our base condition. The input for first layer of translation is same to both $\mathbf{MLT}(d,2)_{\{\pi_i=\pi_i^\alpha\}_{i=1}^d}$, $\mathbf{MLT}(d,2)_{\{\pi_i=\pi_i^\beta\}_{i=1}^d}$. Then, for an input $\boldsymbol{s}_1$,
$$(\boldsymbol{s}_{2,1}^\alpha, \boldsymbol{s}_{2,2}^\alpha) = \pi_1^\alpha((\boldsymbol{s}_{1,2}, \boldsymbol{s}_{1,3}))$$
$$(\boldsymbol{s}_{2,1}^\beta, \boldsymbol{s}_{2,2}^\beta) = \pi_1^\alpha((\boldsymbol{s}_{1,2}, \boldsymbol{s}_{1,3})).$$
We can then use Lemma F.20 to show that
$$p_{2,ij} = \frac{1}{3}, \text{for all } i,j \in \{1,2\},$$
$$p_{2,\text{both}} = \frac{1}{3}.$$

Connecting the variables at ℓ and $\ell+1$: We connect $p_{\ell+1,22}$ to $p_{\ell,23}$, $p_{\ell,32}$, $p_{\ell,33}$ and $p_{\ell,\text{both}}$. This will undergo a case by case analysis.

1. First, if under maps $\pi_1^\alpha, \cdots, \pi_{\ell-1}^\alpha$ and $\pi_1^\beta, \cdots, \pi_{\ell-1}^\beta$, if
$$\begin{aligned}&\text{(either) } \boldsymbol{s}_{\ell,2}^\alpha \text{ and } \boldsymbol{s}_{\ell,2}^\beta \text{ are perfectly correlated , and } \boldsymbol{s}_{\ell,3}^\alpha \text{ and } \boldsymbol{s}_{\ell,3}^\beta \text{ are perfectly correlated}\\ &\quad\text{(or) } \boldsymbol{s}_{\ell,2}^\alpha \text{ and } \boldsymbol{s}_{\ell,3}^\beta \text{ are perfectly correlated , and } \boldsymbol{s}_{\ell,3}^\alpha \text{ and } \boldsymbol{s}_{\ell,2}^\beta \text{ are perfectly correlated ,}\end{aligned} \tag{4}$$
holds true, we can use Lemma F.12 to show that with probability $\frac{1}{3}$ w.r.t. the selection of π_ℓ^α and π_ℓ^β, $\boldsymbol{s}_{\ell+1,2}^\alpha$ and $\boldsymbol{s}_{\ell+1,2}^\beta$ will be correlated. Conditions necessary for Lemma F.12, i.e. uniform distribution of the characters in the input sequence, is shown to hold true in Lemma F.14. The probability w.r.t. the selection of maps $\pi_1^\alpha, \cdots, \pi_{\ell-1}^\alpha$ and $\pi_1^\beta, \cdots, \pi_{\ell-1}^\beta$ that Equation (4) holds true is given by the variable $p_{\ell,\text{both}}$.

2. On the other hand, if there exists a pair $i,j \in \{2,3\}$, if under maps $\pi_1^\alpha, \cdots, \pi_{\ell-1}^\alpha$ and $\pi_1^\beta, \cdots, \pi_{\ell-1}^\beta$, $\boldsymbol{s}_{\ell,i}^\alpha$ and $\boldsymbol{s}_{\ell,j}^\beta$ are perfectly correlated, then with probability $\frac{1}{9}$ w.r.t. the selection of π_ℓ^α and π_ℓ^β, $\boldsymbol{s}_{\ell+1,2}^\alpha$ and $\boldsymbol{s}_{\ell+1,2}^\beta$ will be correlated. We again refer to Lemma F.12 for this statement. Probability that this condition happens under maps $\pi_1^\alpha, \cdots, \pi_{\ell-1}^\alpha$ and $\pi_1^\beta, \cdots, \pi_{\ell-1}^\beta$ is given by the variable $p_{\ell,ij}$.

3. If none of the above situation occurs, then for all pairs $i,j \in \{2,3\}$, $\boldsymbol{s}_{\ell,i}^\alpha$ and $\boldsymbol{s}_{\ell,j}^\beta$ are uncorrelated, and so, by Lemma F.20, correlation between $\boldsymbol{s}_{\ell+1,2}^\alpha$ and $\boldsymbol{s}_{\ell+1,2}^\beta$ will stay 0 for any choice of π_ℓ^α and π_ℓ^β.

Thus, combining the 3 cases, we must have
$$p_{\ell+1,22} \leq \frac{1}{3}p_{\ell,\text{both}} + \frac{1}{9}\sum_{i,j\in\{2,3\}} p_{\ell,ij} \tag{5}$$
$$\leq \frac{1}{3}p_{\ell,\text{both}} + \frac{4}{9}\max_{i,j\in\{2,3\}} p_{\ell,ij}$$
$$\leq \frac{1}{3}\max_{i,j\in\{2,3\}} p_{\ell,ij} + \frac{4}{9}\max_{i,j\in\{2,3\}} p_{\ell,ij} = \frac{7}{9}\max_{i,j\in\{2,3\}} p_{\ell,ij} \tag{6}$$

Reasoning for each step is as follows:

1. Equation (5) follows from conditional probability computations using the 3 cases that we discussed before.

2. Equation (6) uses Equation (2) to connect $p_{\ell,\text{both}}$ to $p_{\ell,ij}$.

We can give the same inequality for $p_{\ell+1,11}$. As $p_{\ell+1,11} = p_{\ell+1,33}$ from Equation (1) and $p_{\ell,23}$ and $p_{\ell,32}$ must be atmost the average of $p_{\ell,22}$ and $p_{\ell,33}$ (Equation (3)), we can write

$$\max_{i,j\in\{2,3\}} p_{\ell+1,ij} \leq \frac{7}{9} \max_{i,j\in\{2,3\}} p_{\ell,ij}.$$

This implies that the probability of correlation at any bit under the two random sets of maps decays at the rate of $\frac{7}{9}$. Solving the recurrence will give:

$$\max_{i,j\in\{2,3\}} p_{d+1,ij} \leq \left(\frac{7}{9}\right)^{\ell-1} p_{2,ij} = \left(\frac{7}{9}\right)^{d-1} \frac{1}{3}.$$

As $p_{d,33}$ should be equal to $p_{d+1,11}$, this implies that with probability at least $1-p_{d,33} = 1-\left(\frac{7}{9}\right)^{d-1}\frac{1}{3}$, the following must hold true:

$$\text{Correlation}(\mathbf{MLT}(d,2)_{\mathbf{\Pi}=\{\pi_i^\alpha\}_{i=1}^d}(\cdot)_1, \mathbf{MLT}(d,2)_{\mathbf{\Pi}=\{\pi_i^\beta\}_{i=1}^d}(\cdot)_1, \mathcal{U}(\{0,1\}^{2d})) = 0.$$

□

Lemma F.12. *Suppose $f, g : \{0,1\}^k \to \{0,1\}^2$ denote 2 functions for some arbitrary k, satisfying*

$$\mathbb{E}_{x\sim\mathcal{U}(\{0,1\}^k)} f(x)_j = 1/2, \quad \mathbb{E}_{x\sim\mathcal{U}(\{0,1\}^k)} g(x)_j = 1/2$$
$$\mathbb{E}_{x\sim\mathcal{U}(\{0,1\}^k)} f(x)_1 \oplus f(x)_2 = 1/2, \quad \mathbb{E}_{x\sim\mathcal{U}(\{0,1\}^k)} g(x)_1 \oplus g(x)_2 = 1/2$$

for all $j \in \{1,2\}$. Then, the following holds true:

1. ***Both bits have correlations between f and g:** If*

   (either) Correlation$(f(\cdot)_1, g(\cdot)_1, \mathcal{U}(\{0,1\}^k)) = 1$ and Correlation$(f(\cdot)_2, g(\cdot)_2, \mathcal{U}(\{0,1\}^k)) = 1$
   (or) Correlation$(f(\cdot)_1, g(\cdot)_2, \mathcal{U}(\{0,1\}^k)) = 1$ and Correlation$(f(\cdot)_2, g(\cdot)_1, \mathcal{U}(\{0,1\}^k)) = 1$,

   then for two randomly picked maps π^α, π^β,

   (a) ***Correlation of an output bit of $\pi^\alpha(f(\cdot))$ and $\pi^\beta(g(\cdot))$:** Fix an $i, j \in \{1,2\}$. With probability $1/3$ w.r.t. the random selections of π^α, π^β,*

   $$\text{Correlation}(\pi^\alpha(f(\cdot))_i, \pi^\alpha(g(\cdot))_j, \mathcal{U}(\{0,1\}^k)) = 1$$

   (b) ***Correlation of both bits of $\pi^\alpha(f(\cdot))$ and $\pi^\beta(g(\cdot))$:** With probability $1/3$ w.r.t. the random selections of π^α, π^β, one of the following two conditions hold true.*

   $$\begin{aligned}&\text{(either) Correlation}(\pi^\alpha(f(\cdot))_1, \pi^\alpha(g(\cdot))_1, \mathcal{U}(\{0,1\}^k)) = 1 \text{ and}\\ &\text{Correlation}(\pi^\alpha(f(\cdot))_2, \pi^\alpha(g(\cdot))_2, \mathcal{U}(\{0,1\}^k)) = 1\\ &\text{(or) Correlation}(\pi^\alpha(f(\cdot))_1, \pi^\alpha(g(\cdot))_2, \mathcal{U}(\{0,1\}^k)) = 1 \text{ and}\\ &\text{Correlation}(\pi^\alpha(f(\cdot))_2, \pi^\alpha(g(\cdot))_1, \mathcal{U}(\{0,1\}^k)) = 1\end{aligned}$$

2. ***Only a pair of bits have correlations between f and g:** If there exists only one pair $i, j \in \{0,1\}$ such that*

   $$\text{Correlation}(f(\cdot)_i, g(\cdot)_j, \mathcal{U}(\{0,1\}^k) = 1,$$

   and for all other i', j', Correlation$(f(\cdot)_{i'}, g(\cdot)_{j'}, \mathcal{U}(\{0,1\}^k)) = 0$, then

(a) Fix any $i', j' \in \{1, 2\}$. With probability $\frac{1}{9}$ w.r.t. the random selections of π^α, π^β,

$$\textit{Correlation}(\pi^\alpha(f(\cdot))_{i'}, \pi^\beta(g(\cdot))_{j'}, \mathcal{U}(\{0,1\}^k)) = 1.$$

If the above condition holds true, for all other $\bar{i}, \bar{j}$ pairs,

$$\textit{Correlation}(\pi^\alpha(f(\cdot))_{\bar{i}}, \pi^\beta(g(\cdot))_{\bar{j}}, \mathcal{U}(\{0,1\}^k)) = 0.$$

3. ***No correlations between f and g:*** *If for all pairs $i, j \in \{0, 1\}$, Correlation$(f(\cdot)_i, g(\cdot)_j, \mathcal{U}(\{0,1\}^k)) = 0$, then for all pairs $i', j' \in \{0, 1\}$,*

$$\textit{Correlation}(\pi^\alpha(f(\cdot))_{i'}, \pi^\beta(g(\cdot))_{j'}, \mathcal{U}(\{0,1\}^k)) = 0.$$

Proof. We will prove each case separately.

1. **Both bits have correlations between f and g:** We will consider the case when Correlation$(f(\cdot)_1, g(\cdot)_1, \mathcal{U}(\{0,1\}^k)) = 1$ and Correlation$(f(\cdot)_2, g(\cdot)_2, \mathcal{U}(\{0,1\}^k)) = 1$, proof for the other case is similar. Then, there are 4 cases possible.

   Subcase 1: $f(x)_1 = g(x)_1, f(x)_2 = g(x)_2$ for all $x \in \{0,1\}^k$
   Subcase 2: $f(x)_1 = \textbf{not}(g(x)_1), f(x)_2 = g(x)_2$ for all $x \in \{0,1\}^k$
   Subcase 3: $f(x)_1 = g(x)_1, f(x)_2 = \textbf{not}(g(x)_2)$ for all $x \in \{0,1\}^k$
   Subcase 4: $f(x)_1 = \textbf{not}(g(x)_1), f(x)_2 = \textbf{not}(g(x)_2)$ for all $x \in \{0,1\}^k$

   Because $f(\cdot)$ and $g(\cdot)$ are inputs to the maps π^α and π^β respectively and $f(\cdot)$ and $g(\cdot)$ are connected by selective **not** operations in their outputs in each of the possible 4 subcases, the output of $\pi^\beta(g(\cdot))$ can be replaced by $\tilde{\pi}^\beta(f(\cdot))$ for a mirror map $\tilde{\pi}^\beta$ of π^β. We showcase this formally for one subcase, say subcase 2; argument for other subcases are similar.

   Subcase 2 is true: Then, for any two maps π^α and π^β, suppose $\tilde{\pi}^\beta$ denotes a mirror map of π^β such that

   $$\tilde{\pi}^\beta(x)_1 = \textbf{not}(\pi^\alpha(x)_1),\ \tilde{\pi}^\beta(x)_2 = \pi^\alpha(x)_2, \quad \text{for all } x \in \{0,1\}^2.$$

   Such a map exists by Lemma F.19. Then, for a pair $i, j \in \{1, 2\}$,

   $$\begin{aligned}
   &\text{Correlation}(\pi^\alpha(f(\cdot))_i, \pi^\beta(g(\cdot))_j, \mathcal{U}(\{0,1\}^k)) \\
   &= \text{Correlation}(\pi^\alpha((f(\cdot)_1, f(\cdot)_2))_i, \pi^\beta((g(\cdot)_1, g(\cdot)_2))_j, \mathcal{U}(\{0,1\}^k)) && (7)\\
   &= \text{Correlation}(\pi^\alpha((f(\cdot)_1, f(\cdot)_2))_i, \pi^\beta((\textbf{not}(f(\cdot)_1), f(\cdot)_2))_j, \mathcal{U}(\{0,1\}^k)) && (8)\\
   &= \text{Correlation}(\pi^\alpha((f(\cdot)_1, f(\cdot)_2))_i, \tilde{\pi}^\beta((f(\cdot)_1, f(\cdot)_2))_j, \mathcal{U}(\{0,1\}^k)) && (9)\\
   &= \text{Correlation}(\pi^\alpha(\cdot)_i, \tilde{\pi}^\beta(\cdot)_j, \mathcal{U}(\{0,1\}^2)). && (10)
   \end{aligned}$$

   Reasoning for each step is as follows.

   - Because the outputs of f and g are a tuple on any input, Equation (7) simply replaces $f(x)$ (similarly $g(x)$) as $(f(x)_1, f(x)_2)$ for any input x.
   - Equation (8) uses the relation between the outputs of f and g when subcase 2 is true.
   - Equation (9) simply replaces π^β with $\tilde{\pi}^\beta$ due to their relation as mirror maps.
   - Finally, because of the assumption on the output of f, i.e. $\mathbb{E}_{x \sim \mathcal{U}(\{0,1\}^k)} f(x)_j = 1/2, \mathbb{E}_{x \sim \mathcal{U}(\{0,1\}^k)} f(x)_1 \oplus f(x)_2 = 1/2$, the distribution of outputs of f can be shown to be identical to $\mathcal{U}(\{0,1\}^2)$.

   Hence, Correlation$(\pi^\alpha(f(\cdot))_i, \pi^\beta(g(\cdot))_j, \mathcal{U}(\{0,1\}^k))$ boils down to Correlation$(\pi^\alpha(\cdot)_i, \tilde{\pi}^\beta(\cdot)_j, \mathcal{U}(\{0,1\}^2))$. We can then use Lemma F.20 to show the probabilities of each of the desired conditions.

2. **Only a pair of bits have correlations between f and g:** For typographical simplicity, consider the case when

$$\text{Correlation}(f(\cdot)_1, g(\cdot)_1, \mathcal{U}(\{0,1\}^k)) = 1,$$

and for all pairs i', j' with atleast one of them being not equal to 1, $\text{Correlation}(f(\cdot)_{i'}, g(\cdot)_{j'}, \mathcal{U}(\{0,1\}^k)) = 0$. The argument when the condition holds for other i, j pairs can be similarly handled. Then, there are 2 cases possible.

$$\text{Subcase 1: } f(x)_1 = g(x)_1, \text{ for all } x \in \{0,1\}^k$$
$$\text{Subcase 2: } f(x)_1 = \textbf{not}(g(x)_1), \text{ for all } x \in \{0,1\}^k,$$

while in each of these pairs, $(f(\cdot)_2, g(\cdot)_2)$, $(f(\cdot)_2, g(\cdot)_1)$, and $(f(\cdot)_1, g(\cdot)_2)$, the output bits of f and g are completely independent of each other. We only consider subcase 1; subcase 2 can be handled using mirror maps similar to the proof for case 1 (in particular with Equation (9)).

For simplicity, we will argue for $i', j' = 1, 1$; the argument about other i', j' pairs are similar. The proof will follow by two steps,

- **Step (a):** We first argue about the probability with which $\text{Correlation}(\pi^\alpha(f(\cdot))_1, \pi^\beta(g(\cdot))_1, \mathcal{U}(\{0,1\}^k))$ is equal to 1,
- **Step (b):** We then argue that if the condition in (a) holds true, then $\text{Correlation}(\pi^\alpha(f(\cdot))_{i'}, \pi^\beta(g(\cdot))_{j'}, \mathcal{U}(\{0,1\}^k))$ must be 0 for any other pair i', j' where atleast one of them is not equal to 1.

Step (a): For simplicity, we will assume that $\pi^\alpha, \pi^\beta \in \{\boldsymbol{\Delta}_1, \boldsymbol{\Delta}_2, \cdots, \boldsymbol{\Delta}_6\}$ defined in Table 4; other cases can be handled as Lemma F.21.

$$\begin{aligned}
&\text{Correlation}(\pi^\alpha(f(\cdot))_1, \pi^\beta(g(\cdot))_1, \mathcal{U}(\{0,1\}^k)) \\
&= \text{Correlation}(\pi^\alpha((f(\cdot)_1, f(\cdot)_2))_1, \pi^\beta((g(\cdot)_1, g(\cdot)_2))_1, \mathcal{U}(\{0,1\}^k)) && (11)\\
&= \text{Correlation}(\pi^\alpha((f(\cdot)_1, f(\cdot)_2))_1, \pi^\beta((f(\cdot)_1, g(\cdot)_2))_1, \mathcal{U}(\{0,1\}^k)) && (12)\\
&= \Big| \Pr_{x\sim\mathcal{U}(\{0,1\}^k)} \left[\pi^\alpha((f(x)_1, f(x)_2))_1 = \pi^\beta((f(x)_1, g(x)_2))_1\right] - && (13)\\
&\qquad \Pr_{x\sim\mathcal{U}(\{0,1\}^k} \left[\pi^\alpha((f(x)_1, f(x)_2))_1 \neq \pi^\beta((f(x)_1, g(x)_2))_1\right] \Big| && (14)\\
&= \left| \Pr_{x\sim\mathcal{U}(\{0,1\})} \left[\Pr_{y,y'\sim\mathcal{U}(\{0,1\}^2)} \left[\pi^\alpha((x,y))_1 = \pi^\beta((x,y'))_1\right] - \Pr_{y,y'\sim\mathcal{U}(\{0,1\}^2)} \left[\pi^\alpha((x,y))_1 = \pi^\beta((x,y'))_1\right] \right] \right| && (15)
\end{aligned}$$

The reasoning for each step is as follows:

- Because the outputs of f and g are a tuple on any input, Equation (11) simply replaces $f(x)$ (similarly $g(x)$) as $(f(x)_1, f(x)_2)$ for any input x.
- Equation (12) uses the relation between the outputs of f and g when subcase 1 is true.
- Equation (9) simply writes the definition of correlation.
- Finally, because of the assumption on the output of f, i.e. $\mathbb{E}_{x\sim\mathcal{U}(\{0,1\}^k)} f(x)_j = 1/2, \mathbb{E}_{x\sim\mathcal{U}(\{0,1\}^k)} f(x)_1 \oplus f(x)_2 = 1/2$, the distribution of outputs of f can be shown to be identical to $\mathcal{U}(\{0,1\}^2)$.

From the definitions of $\pi^\alpha, \pi^\beta \in \{\boldsymbol{\Delta}_1, \boldsymbol{\Delta}_2, \cdots, \boldsymbol{\Delta}_6\}$ in Table 4, the first character in the outputs of π^α and π^β can be expressed using 3 operators on the input characters, which are $\textbf{copy}_1$, $\textbf{copy}_2$, and **xor**. These operators are independently selected for π^α and π^β, as they are randomly picked from these 6 possibilities. Formally, for any tuple of variables (x, y) and (x, y'),

$$\begin{aligned}
&\textbf{copy}_1(x,y) = x, \quad \textbf{copy}_2(x,y) = y, \quad \textbf{xor}(x,y) = x \oplus y,\\
&\textbf{copy}_1(x,y') = x, \quad \textbf{copy}_2(x,y') = y', \quad \textbf{xor}(x,y') = x \oplus y'.
\end{aligned}$$

However, because x, y, y' are independent variables, only operations $\mathbf{copy}_1(x, y)$ and $\mathbf{copy}_1(x, y')$ for $(x, y, y') \sim \mathcal{U}(\{0,1\}^3)$ will be correlated. This can be verified by writing the definition of correlation and using $\mathbf{copy}_1$ for the first character output of π^α and π^β:

$$\left| \Pr_{x\sim\mathcal{U}(\{0,1\})} \left[\Pr_{y,y'\sim\mathcal{U}(\{0,1\}^2)} \left[\pi^\alpha((x,y))_1 = \pi^\beta((x,y'))_1\right] - \Pr_{y,y'\sim\mathcal{U}(\{0,1\}^2)} \left[\pi^\alpha((x,y))_1 = \pi^\beta((x,y'))_1\right] \right] \right|$$
$$= \left| \Pr_{x\sim\mathcal{U}(\{0,1\})} \left[\Pr_{y,y'\sim\mathcal{U}(\{0,1\}^2)} \left[\mathbf{copy}_1(x,y) = \mathbf{copy}_1(x,y')\right] - \Pr_{y,y'\sim\mathcal{U}(\{0,1\}^2)} \left[\mathbf{copy}_1(x,y) \neq \mathbf{copy}_1(x,y')\right] \right] \right| = 1,$$

as the first term is 1 and the second term is 0. However, if you pick any other pair of operations, the correlation will be 0. We demonstrate by using $\mathbf{copy}_1$ and **xor** for π^α and π^β respectively.

$$\left| \Pr_{x\sim\mathcal{U}(\{0,1\})} \left[\Pr_{y,y'\sim\mathcal{U}(\{0,1\}^2)} \left[\pi^\alpha((x,y))_1 = \pi^\beta((x,y'))_1\right] - \Pr_{y,y'\sim\mathcal{U}(\{0,1\}^2)} \left[\pi^\alpha((x,y))_1 = \pi^\beta((x,y'))_1\right] \right] \right|$$
$$= \left| \Pr_{x\sim\mathcal{U}(\{0,1\})} \left[\Pr_{y,y'\sim\mathcal{U}(\{0,1\}^2)} \left[\mathbf{copy}_1(x,y) = \mathbf{xor}(x,y')\right] - \Pr_{y,y'\sim\mathcal{U}(\{0,1\}^2)} \left[\mathbf{copy}_1(x,y) \neq \mathbf{xor}(x,y')\right] \right] \right|$$
$$= \left| \Pr_{x\sim\mathcal{U}(\{0,1\})} \left[\Pr_{y,y'\sim\mathcal{U}(\{0,1\}^2)} \left[x = x \oplus y'\right] - \Pr_{y,y'\sim\mathcal{U}(\{0,1\}^2)} \left[x \neq x \oplus y'\right] \right] \right| = 0,$$

as both terms are equal to $\frac{1}{2}$ in the final step. By a counting argument, one can reason that the probability of $corr(\pi^\alpha(f(\cdot))_1, \pi^\beta(g(\cdot))_1, \mathcal{U}(\{0,1\}^k))$ being 1 is equal to the probability of $\mathbf{copy}_1$ being selected to define the first character of π^α and π^β, which will be equal to $\frac{1}{9}$.

Step (b): Now, say we have selected a pair π^α and π^β such that $\text{Correlation}(\pi^\alpha(f(\cdot))_1, \pi^\beta(g(\cdot))_1, \mathcal{U}(\{0,1\}^k)) = 1$. From the proof of step (a), this is only possible when $\mathbf{copy}_1$ was selected to define the first characters in the outputs of π^α and π^β. Any other operation pairs for defining the first characters in the outputs of π^α and π^β would have meant correlation to be 0.

We can use this same argument to show that for any other pair i', j' where atleast one of them is not equal to 1. $\text{Correlation}(\pi^\alpha(f(\cdot))_{i'}, \pi^\beta(g(\cdot))_{j'}, \mathcal{U}(\{0,1\}^k))$ must be 0. This is because once $\mathbf{copy}_1$ has been used to define the operation for the first character, it can't be used to define the operation for the second character (please look at the truth tables for $\{\boldsymbol{\Delta}_1, \cdots, \boldsymbol{\Delta}_6\}$ in Table 4. Following a similar argument, we can then show that any correlation between output characters $\pi^\alpha(f(\cdot))_{i'}, \pi^\beta(g(\cdot))_{j'}$ will be 0, as $\mathbf{copy}_1$ can't be used for at least one of these characters.

3. The third case is when none of the above conditions hold true. That is, for any pair $i, j \in \{1, 2\}$,

$$\text{Correlation}(f(\cdot)_i, g(\cdot)_j, \mathcal{U}(\{0,1\}^k)) = 0.$$

In such case, following similar arguments as case 1 and 2, one can show that for any $i, j \in \{1, 2\}$, for any choice of π^α and π^β:

$$\begin{aligned}
&\text{Correlation}(\pi^\alpha(f(\cdot))_i, \pi^\beta(g(\cdot))_j, \mathcal{U}(\{0,1\}^k)) \\
&= \Bigg| \Pr_{x,y\sim\mathcal{U}(\{0,1\}^2)} \Pr_{x',y'\sim\mathcal{U}(\{0,1\}^2)} \left[\pi^\alpha((x,y))_i = \pi^\beta((x',y'))_j\right] - \\
&\qquad \Pr_{x,y\sim\mathcal{U}(\{0,1\}^2)} \Pr_{x',y'\sim\mathcal{U}(\{0,1\}^2)} \left[\pi^\alpha((x,y))_i \neq \pi^\beta((x',y'))_j\right] \Bigg| \\
&= 0,
\end{aligned}$$

due to independence of inputs to π^α and π^β.

□

Lemma F.13. *At any step i of $\boldsymbol{MLT}(d, 2)$ with phrasebooks $\{\pi_i\}_{i=1}^d$, the following relation holds true for the intermediate outputs $\{\boldsymbol{s}_i\}_{i=2}^{d+1}$ on an input $\boldsymbol{s}_1 \in \{0,1\}^{2d}$:*

$$(\boldsymbol{s}_{i+1,1}, \boldsymbol{s}_{i+1,2}) = \pi_i((\boldsymbol{s}_{i,2}, \boldsymbol{s}_{i,3})).$$

Proof. $\mathbf{MLT}(d, 2)$ has 2 primary steps: *Circular shift* and *Translate*. By *Circular shift*, first, we first get sequence $\tilde{\boldsymbol{s}}_i$, where for $j \in [L]$ we have $\tilde{\boldsymbol{s}}_{i,j} = \boldsymbol{s}_{i,(j+1)\%L}$. After *Translate* step,

$$(\boldsymbol{s}_{i+1,1}, \boldsymbol{s}_{i+1,2}) = \pi_i((\tilde{\boldsymbol{s}}_{i,1}, \tilde{\boldsymbol{s}}_{i,2})) = \pi_i((\boldsymbol{s}_{i,2}, \boldsymbol{s}_{i,3}))$$

□

Lemma F.14. *At any step i of $\mathbf{MLT}(d, 2)$ with phrasebooks $\{\pi_i\}_{i=1}^d$, the following conditions hold true for the intermediate outputs $\{\boldsymbol{s}_i\}_{i=1}^{d+1}$:*

$$\mathbb{E}_{\boldsymbol{s}_1 \sim \mathcal{U}(\{0,1\}^L)} \boldsymbol{s}_{i,j} = \frac{1}{2}, \textit{ for all } 1 \le j \le L$$
$$\mathbb{E}_{\boldsymbol{s}_1 \sim \mathcal{U}(\{0,1\}^L)} \boldsymbol{s}_{i,j} \oplus \boldsymbol{s}_{i,j'} = \frac{1}{2}, \textit{ for all } j \ne j'.$$

The above conditions are equivalent to showing that $\boldsymbol{s}_{i,j}$ behaves like a uniformly random boolean variable, independent of any other character $\boldsymbol{s}_{i,j'}$ for all coordinates j .

Proof. The proof will follow by induction on the output of the translation task at each step. We will show the result for coordinate $j = 1$ in the $\boldsymbol{s}_i$ at each layer ℓ; similar argument holds for other coordinates j.

Base condition: At layer $i = 1$, the $\boldsymbol{s}_1$ represents the input sequence from $\mathcal{U}(\{0,1\}^L)$. By definition of uniform distribution, the conditions hold true for the input.

Induction step: Argument for general $i > 1$: Suppose the conditions are true for all layers $1 \le \ell < i$. Then for layer i, we will provide an argument for the condition to hold true for $j = 1$ and $j = 2$, arguments for other js will extend similarly. By Lemma F.13,

$$(\boldsymbol{s}_{i,1}, \boldsymbol{s}_{i,2}) = \pi_i((\boldsymbol{s}_{i-1,2}, \boldsymbol{s}_{i-1,3})).$$

From Lemma F.19, we have each output character can be defined in terms of **copy**, **xor**, and **not** operations on input characters. Then, the relations for $\boldsymbol{s}_{i,1}, \boldsymbol{s}_{i,2}$ can be computed as follows:

- If a map π^α is selected from Mirrorset(π) for some $\pi \in \{\boldsymbol{\Delta}_1, \cdots, \boldsymbol{\Delta}_6\}$, then one can show that the conditions hold true for $\pi_i = \pi^\alpha$ if the conditions hold true for $\pi_i = \pi$. This follows because the output of π^α follows from the output of π by selective **not** operations to the output of π. As **not** operation won't change the expected values of a variable which behaves like a random boolean variable, the argument follows.
- Now, we show that for $\pi_i = \pi$ for some $\pi \in \{\boldsymbol{\Delta}_1, \cdots, \boldsymbol{\Delta}_6\}$, the conditions hold true. For these maps, the output characters are defined by **copy** and **xor** operations. We use the definitions of these maps from Table 4 and show the expected values $\boldsymbol{s}_{i,1}$ in terms of expected values of $\boldsymbol{s}_{i-1,1}$ and $\boldsymbol{s}_{i-1,2}$, and the expected values $\boldsymbol{s}_{i,1} \oplus \boldsymbol{s}_{i,2}$ in terms of expected values of $\boldsymbol{s}_{i-1,1}$ and $\boldsymbol{s}_{i-1,2}$ in Table 3.

Both of the above arguments then can be combined to show that

$$\mathbb{E}_{\boldsymbol{s}_1 \sim \mathcal{U}(\{0,1\}^L)} \boldsymbol{s}_{i,j} = \frac{1}{2}, \text{ for } j \in \{1, 2\}$$
$$\mathbb{E}_{\boldsymbol{s}_1 \sim \mathcal{U}(\{0,1\}^L)} \boldsymbol{s}_{i,1} \oplus \boldsymbol{s}_{i,2} = \frac{1}{2}.$$

We can extend the argument for expectation of each individual character to other positions $j > 2$. We can also extend the argument for joint expectation of two consecutive characters $\boldsymbol{s}_{i,2k+1} \oplus \boldsymbol{s}_{i,2k+2}$ for any general k. This is similar to claiming that after *Split* operation on the sequence $\boldsymbol{s}_i$, in each of the resulting 2-tuples which are of the form $(\boldsymbol{s}_{i,2k+1}, \boldsymbol{s}_{i,2k+2})$, the individual characters are independent of each other.

Operator	Expected value	Expected $\oplus$ value with operator		
		copy$_1$	**copy**$_2$	**xor**
copy$_1$	$\mathbb{E}_{\boldsymbol{s}_1} \boldsymbol{s}_{i-1,2} = 1/2$	-	$\mathbb{E}_{\boldsymbol{s}_1} \boldsymbol{s}_{i-1,2} \oplus \boldsymbol{s}_{i-1,3} = 1/2$	$\mathbb{E}_{\boldsymbol{s}_1} \boldsymbol{s}_{i-1,3} = 1/2$
copy$_2$	$\mathbb{E}_{\boldsymbol{s}_1} \boldsymbol{s}_{i-1,3} = 1/2$	$\mathbb{E}_{\boldsymbol{s}_1} \boldsymbol{s}_{i-1,2} \oplus \boldsymbol{s}_{i-1,3} = 1/2$	-	$\mathbb{E}_{\boldsymbol{s}_1} \boldsymbol{s}_{i-1,2} = 1/2$
xor	$\mathbb{E}_{\boldsymbol{s}_1} \boldsymbol{s}_{i-1,2} \oplus \boldsymbol{s}_{i-1,3} = 1/2$	$\mathbb{E}_{\boldsymbol{s}_1} \boldsymbol{s}_{i-1,3} = 1/2$	$\mathbb{E}_{\boldsymbol{s}_1} \boldsymbol{s}_{i-1,2} = 1/2$	-

Table 3: $\mathbb{E}\boldsymbol{s}_{i,1}$ (similarly $\mathbb{E}\boldsymbol{s}_{i,2}$) and $\mathbb{E}\boldsymbol{s}_{i,1} \oplus \boldsymbol{s}_{i,2}$ under different π_i maps, defined by **copy** and **xor** operations on $\boldsymbol{s}_{i-1,2}$ and $\boldsymbol{s}_{i-1,3}$.

The final argument will be to show that characters across the resulting tuples are going to be independent of each other as well. Consider any two tuples $(\boldsymbol{s}_{i,2k+1}, \boldsymbol{s}_{i,2k+2})$ and $(\boldsymbol{s}_{i,2k'+1}, \boldsymbol{s}_{i,2k'+2})$, with $k \neq k'$. By adapting Lemma F.13, one can show that

$$(\boldsymbol{s}_{i,2k+1}, \boldsymbol{s}_{i,2k+2}) = \pi_i((\boldsymbol{s}_{i-1,2k+2}, \boldsymbol{s}_{i,2k+3}))$$
$$(\boldsymbol{s}_{i,2k'+1}, \boldsymbol{s}_{i,2k'+2}) = \pi_i((\boldsymbol{s}_{i-1,2k'+2}, \boldsymbol{s}_{i,2k'+3}))$$

The primary thing to note here is that both the tuples depend on two distinct tuples in layer $i-1$. By induction assumption, characters across these two tuples are independent of each other. As π_i is a bijective map, this will also suggest that characters across the tuples in the resulting output must also be independent of each other. This will give the final argument.

□

Lemma F.15. *Under the assumption that sequence lengths L are even, for any $1 \leq i \leq d$, where $\boldsymbol{s}^{\alpha}_{i,j}, \boldsymbol{s}^{\beta}_{i,j}$ denote the output after $i-1$st translation step for a random sequence $\boldsymbol{s}_1 \sim \mathcal{U}(\{0,1\}^L)$ at any position $1 \leq j \leq L$ under the two sets of random maps $\{\pi^{\alpha}_{\ell}\}_{\ell=1}^{i-1}$ and $\{\pi^{\beta}_{\ell}\}_{\ell=1}^{i-1}$, the following holds true for all positions j:*

Correlation$(\boldsymbol{s}^{\alpha}_{i,j}, \boldsymbol{s}^{\beta}_{i,j}, \mathcal{U}(\{0,1\}^L)) =$ *Correlation*$(\boldsymbol{s}^{\alpha}_{i,(j+2k)\%L}, \boldsymbol{s}^{\beta}_{i,(j+2k)\%L}, \mathcal{U}(\{0,1\}^L))$, *for all integer k.*

Proof. Fix an integer k. By Definition F.1, the necessary condition would be to show

$$\left|1 - 2\mathbb{E}_{\boldsymbol{s}_1 \sim \{0,1\}^L} \boldsymbol{s}^{\alpha}_{i,j} \oplus \boldsymbol{s}^{\beta}_{i,j}\right| = \left|1 - 2\mathbb{E}_{\boldsymbol{s}_1 \sim \{0,1\}^L} \boldsymbol{s}^{\alpha}_{i,(j+2k)\%L} \oplus \boldsymbol{s}^{\beta}_{i,(j+2k)\%L}\right|.$$

We will look at the behavior of the function $h(\boldsymbol{s}_1) = \boldsymbol{s}^{\alpha}_{i,j} \oplus \boldsymbol{s}^{\beta}_{i,j}$. Denote $\mathcal{C}_k$ as a circular operation that takes a sequence $\boldsymbol{s}_1$ and returns a shifted sequence $^{\circlearrowleft}\boldsymbol{s}_1$, i.e. if $^{\circlearrowleft}\boldsymbol{s}_1 = \mathcal{C}_k(\boldsymbol{s}_1)$ then for all j, $^{\circlearrowleft}\boldsymbol{s}_{1,j} = \boldsymbol{s}_{1,(j+2k)\%L}$. Note that, we can also define an inverse function $\mathcal{C}_k^{-1}$ and $\boldsymbol{s}_1 = \mathcal{C}_k^{-1}(\mathcal{C}_k(\boldsymbol{s}_1))$.

From the definition of $\mathbf{MLT}(2,d)_{\mathbf{\Pi}}$ for any code-book $\mathbf{\Pi}$, we have for any input $\boldsymbol{s}_1$:

$$\boldsymbol{s}_i = T_{\pi_{i-1}} \circ \cdots \circ T_{\pi_1}(\boldsymbol{s}_1),$$

where for any level i, T_{π_i} denotes the translation step at level i that includes *Circular shift*, *Split*, *Translate* with π_i, and *Merge*.

In Lemma F.16, we show that for any translation task $\mathbf{MLT}(2,d)_{\mathbf{\Pi}}$ and any input $\boldsymbol{s}_1$,

$$T_{\pi_{i-1}} \circ \cdots \circ T_{\pi_1}(\boldsymbol{s}_1) = \mathcal{C}_k^{-1}\left(T_{\pi_{i-1}} \circ \cdots \circ T_{\pi_1} \circ \mathcal{C}_k(\boldsymbol{s}_1)\right).$$

On input $^{\circlearrowleft}\boldsymbol{s}_1 = \mathcal{C}_k(\boldsymbol{s}_1)$, if $^{\circlearrowleft}\boldsymbol{s}^{\alpha}_i$ and $^{\circlearrowleft}\boldsymbol{s}^{\beta}_i$ represent the output of translations using $\{\pi^{\alpha}_{\ell}\}_{\ell=1}^{i-1}$ and $\{\pi^{\beta}_{\ell}\}_{\ell=1}^{i-1}$ respectively, then the above statement says that

$$\boldsymbol{s}^{\alpha}_i = \mathcal{C}_k^{-1}\left(^{\circlearrowleft}\boldsymbol{s}^{\alpha}_i\right), \quad \boldsymbol{s}^{\beta}_i = \mathcal{C}_k^{-1}\left(^{\circlearrowleft}\boldsymbol{s}^{\beta}_i\right)$$
$$\text{(or) } \mathcal{C}_k\left(\boldsymbol{s}^{\alpha}_i\right) = {}^{\circlearrowleft}\boldsymbol{s}^{\alpha}_i, \quad \mathcal{C}_k\left(\boldsymbol{s}^{\beta}_i\right) = {}^{\circlearrowleft}\boldsymbol{s}^{\beta}_i$$

Thus, for any position j, we will have $\mathcal{C}_k\left(\boldsymbol{s}_i^{\alpha}\right)_j = {}^{\circlearrowleft}\boldsymbol{s}_{i,j}^{\alpha}$ (and similarly for ${}^{\circlearrowleft}\boldsymbol{s}_{i,j}^{\beta}$). We can then compute the value of h on input ${}^{\circlearrowleft}\boldsymbol{s}_1$ as follows:

$$\begin{aligned} h({}^{\circlearrowleft}\boldsymbol{s}_1) &= {}^{\circlearrowleft}\boldsymbol{s}_{i,j}^{\alpha} \oplus {}^{\circlearrowleft}\boldsymbol{s}_{i,j}^{\beta} \\ &= \mathcal{C}_k\left(\boldsymbol{s}_i^{\alpha}\right)_j \oplus \mathcal{C}_k\left(\boldsymbol{s}_i^{\beta}\right)_j \\ &= \boldsymbol{s}_{i,(j+2k)\%L}^{\alpha} \oplus \boldsymbol{s}_{i,(j+2k)\%L}^{\beta} \end{aligned}$$

The last step follows from using the definition of $\mathcal{C}_k$. On the other hand,

$$\mathbb{E}_{{}^{\circlearrowleft}\boldsymbol{s}_1 \sim \mathcal{U}(\{0,1\}^L)} h({}^{\circlearrowleft}\boldsymbol{s}_1) = \mathbb{E}_{\boldsymbol{s}_1 \sim \mathcal{U}(\{0,1\}^L)} h(\boldsymbol{s}_1),$$

as the uniform distribution can be shown to not change under circular function $\mathcal{C}_k$. This will imply:

$$\begin{aligned} &\mathbb{E}_{\boldsymbol{s}_1 \sim \mathcal{U}(\{0,1\}^L)} \boldsymbol{s}_{i,(j+2k)\%L}^{\alpha} \oplus \boldsymbol{s}_{i,(j+2k)\%L}^{\beta} = \mathbb{E}_{\boldsymbol{s}_1 \sim \{0,1\}^L} \boldsymbol{s}_{i,j}^{\alpha} \oplus \boldsymbol{s}_{i,j}^{\beta} \\ &\implies \left|1 - 2\mathbb{E}_{\boldsymbol{s}_1 \sim \{0,1\}^L} \boldsymbol{s}_{i,j}^{\alpha} \oplus \boldsymbol{s}_{i,j}^{\beta}\right| = \left|1 - 2\mathbb{E}_{\boldsymbol{s}_1 \sim \{0,1\}^L} \boldsymbol{s}_{i,(j+2k)\%L}^{\alpha} \oplus \boldsymbol{s}_{i,(j+2k)\%L}^{\beta}\right| \\ &\implies \text{Correlation}(\boldsymbol{s}_{i,j}^{\alpha}, \boldsymbol{s}_{i,j}^{\beta}, \mathcal{U}(\{0,1\}^L)) = \text{Correlation}(\boldsymbol{s}_{i,(j+2k)\%L}^{\alpha}, \boldsymbol{s}_{i,(j+2k)\%L}^{\beta}, \mathcal{U}(\{0,1\}^L)). \end{aligned}$$

□

Lemma F.16. *For any translation task* $\boldsymbol{MLT}(2,d)_{\boldsymbol{\Pi}}$*, at any level* $i \leq d$ *and any input* $\boldsymbol{s}_1$*,*

$$T_{\pi_i} \circ \cdots \circ T_{\pi_1}\left(\boldsymbol{s}_1\right) = \mathcal{C}_k^{-1}\left(T_{\pi_i} \circ \cdots \circ T_{\pi_1} \circ \mathcal{C}_k\left(\boldsymbol{s}_1\right)\right).$$

Proof. Denote $\mathcal{C}_k$ as a circular operation that takes a sequence $\boldsymbol{s}_1$ and returns a shifted sequence ${}^{\circlearrowleft}\boldsymbol{s}_1$, i.e. if ${}^{\circlearrowleft}\boldsymbol{s}_1 = \mathcal{C}_k(\boldsymbol{s}_1)$ then for all j, ${}^{\circlearrowleft}\boldsymbol{s}_{1,j} = \boldsymbol{s}_{1,(j+2k)\%L}$. Note that, we can also define an inverse function $\mathcal{C}_k^{-1}$ and $\boldsymbol{s}_1 = \mathcal{C}_k^{-1}(\mathcal{C}_k(\boldsymbol{s}_1))$.

Recall that for any level i, T_{π_i} denotes the translation step at level i that includes *Circular shift*, *Split*, *Translate* with π_i, and *Merge*. The argument will again follow by an induction step.

Base condition: $i = 1$**:** We need to show that

$$T_{\pi_1}\left(\boldsymbol{s}_1\right) = \mathcal{C}_k^{-1}\left(T_{\pi_1} \circ \mathcal{C}_k\left(\boldsymbol{s}_1\right)\right).$$

To do so, we will look at the behavior of the first 2 characters. Argument for others can be extended. If ${}^{\circlearrowleft}\boldsymbol{s}_1 = \mathcal{C}_k\left(\boldsymbol{s}_1\right)$, $\boldsymbol{s}_2 = T_{\pi_1}(\boldsymbol{s}_1)$, ${}^{\circlearrowleft}\boldsymbol{s}_2 = T_{\pi_1}({}^{\circlearrowleft}\boldsymbol{s}_1)$, then by Lemma F.13,

$$\begin{aligned} (\boldsymbol{s}_{2,1}, \boldsymbol{s}_{2,2}) &= \pi_1((\boldsymbol{s}_{1,2}, \boldsymbol{s}_{1,3})) \\ ({}^{\circlearrowleft}\boldsymbol{s}_{2,1}, {}^{\circlearrowleft}\boldsymbol{s}_{2,2}) &= \pi_1(({}^{\circlearrowleft}\boldsymbol{s}_{1,2}, {}^{\circlearrowleft}\boldsymbol{s}_{1,3})). \end{aligned}$$

But by definition of $\mathcal{C}_k$,

$$({}^{\circlearrowleft}\boldsymbol{s}_{1,2}, {}^{\circlearrowleft}\boldsymbol{s}_{1,3}) = (\boldsymbol{s}_{1,(2+2k)\%L}, \boldsymbol{s}_{1,(3+2k)\%L}).$$

On the other hand, Lemma F.13 can be adapted to give

$$\left(\boldsymbol{s}_{2,(1+2k)\%L}, \boldsymbol{s}_{2,(2+2k)\%L}\right) = \pi_1(\boldsymbol{s}_{1,(2+2k)\%L}, \boldsymbol{s}_{1,(3+2k)\%L}).$$

Thus, we can show by combining the above 3 steps that

$$\begin{aligned} ({}^{\circlearrowleft}\boldsymbol{s}_{2,1}, {}^{\circlearrowleft}\boldsymbol{s}_{2,2}) &= \pi_1(({}^{\circlearrowleft}\boldsymbol{s}_{1,2}, {}^{\circlearrowleft}\boldsymbol{s}_{1,3})) \\ &= \pi_1(\boldsymbol{s}_{1,(2+2k)\%L}, \boldsymbol{s}_{1,(3+2k)\%L}) \\ &= \left(\boldsymbol{s}_{2,(1+2k)\%L}, \boldsymbol{s}_{2,(2+2k)\%L}\right). \end{aligned}$$

We can extend the above argument to show that for any position j,

$${}^{\circlearrowleft}\boldsymbol{s}_{2,j} = \boldsymbol{s}_{2,(j+2k)\%L},$$

which by definition of $\mathcal{C}_k$, implies

$$^{\circlearrowleft}\boldsymbol{s}_2 = \mathcal{C}_k(\boldsymbol{s}_2) \quad \text{or } \boldsymbol{s}_2 = \mathcal{C}_k^{-1}\left(^{\circlearrowleft}\boldsymbol{s}_2\right),$$

which can be further simplified (using the notations: $^{\circlearrowleft}\boldsymbol{s}_1 = \mathcal{C}_k(\boldsymbol{s}_1)$, $\boldsymbol{s}_2 = T_{\pi_1}(\boldsymbol{s}_1)$, $^{\circlearrowleft}\boldsymbol{s}_2 = T_{\pi_1}(^{\circlearrowleft}\boldsymbol{s}_1)$)

$$\begin{aligned}\boldsymbol{s}_2 &= \mathcal{C}_k^{-1}\left(^{\circlearrowleft}\boldsymbol{s}_2\right)\\ &= \mathcal{C}_k^{-1}\left(T_{\pi_1}\left(^{\circlearrowleft}\boldsymbol{s}_1\right)\right)\\ &= \mathcal{C}_k^{-1}\left(T_{\pi_1}\left(\mathcal{C}_k\left(\boldsymbol{s}_1\right)\right)\right) := \mathcal{C}_k^{-1}\left(T_{\pi_1}\circ\mathcal{C}_k\left(\boldsymbol{s}_1\right)\right).\end{aligned}$$

General argument for i: Suppose the induction condition holds true for all layers $\ell < i$. Then,

$$T_{\pi_i}\circ\cdots\circ T_{\pi_1}\left(\boldsymbol{s}_1\right) = T_{\pi_i}\left(\mathcal{C}_k^{-1}\left(T_{\pi_{i-1}}\cdots\circ T_{\pi_1}\circ\mathcal{C}_k\left(\boldsymbol{s}_1\right)\right)\right).$$

We can then follow the same argument as the base condition, and show that

$$T_{\pi_i}\left(\mathcal{C}_k^{-1}\left(T_{\pi_{i-1}}\cdots\circ T_{\pi_1}\circ\mathcal{C}_k\left(\boldsymbol{s}_1\right)\right)\right) = \mathcal{C}_k^{-1}\left(T_{\pi_i}\cdots\circ T_{\pi_1}\circ\mathcal{C}_k\left(\boldsymbol{s}_1\right)\right).$$

□

Lemma F.17. *For any $1 \le i \le d$, if $\boldsymbol{s}_{i,j}^{\alpha}, \boldsymbol{s}_{i,j}^{\beta}$ denote the output after $i-1$st translation step for a random sequence $\boldsymbol{s}_1 \sim \mathcal{U}(\{0,1\}^L)$ at any position $1 \le j \le L$ under the two sets of random maps $\boldsymbol{\Pi}_{:i}^{\alpha} := \{\pi_\ell^{\alpha}\}_{\ell=1}^{i-1}$ and $\boldsymbol{\Pi}_{:i}^{\beta} := \{\pi_\ell^{\beta}\}_{\ell=1}^{i-1}$, the following holds true for all positions j:*

$$\begin{aligned}&\Pr_{\boldsymbol{\Pi}_{:i}^{\alpha},\boldsymbol{\Pi}_{:i}^{\beta}}\left[\textit{Correlation}\left(\boldsymbol{s}_{i,j}^{\alpha},\boldsymbol{s}_{i,j+1}^{\beta},\mathcal{U}(\{0,1\}^L)\right)=1\right]\\ &\le \frac{1}{2}\left(\Pr_{\boldsymbol{\Pi}_{:i}^{\alpha},\boldsymbol{\Pi}_{:i}^{\beta}}\left[\textit{Correlation}\left(\boldsymbol{s}_{i,j}^{\alpha},\boldsymbol{s}_{i,j}^{\beta},\mathcal{U}(\{0,1\}^L)\right)=1\right]+\Pr_{\boldsymbol{\Pi}_{:i}^{\alpha},\boldsymbol{\Pi}_{:i}^{\beta}}\left[\textit{Correlation}\left(\boldsymbol{s}_{i,j+1}^{\alpha},\boldsymbol{s}_{i,j+1}^{\beta},\mathcal{U}(\{0,1\}^L)\right)=1\right]\right).\end{aligned}$$

Proof. We will prove the required result with a counting argument. We will create a family of code-books: let FAMILY(j) and FAMILY$(j+1)$ denote two sets of code-books, such that for every code-book $\boldsymbol{\Pi}_{:i}^{\alpha}$ in FAMILY(j), there exists at least one code-book $\boldsymbol{\Pi}_{:i}^{\beta}$ in FAMILY$(j+1)$ such that

$$\text{Correlation}\left(\boldsymbol{s}_{i,j}^{\alpha},\boldsymbol{s}_{i,j+1}^{\beta},\mathcal{U}(\{0,1\}^L)\right)=1,$$

where $\boldsymbol{s}_{i,j}^{\alpha},\boldsymbol{s}_{i,j+1}^{\beta}$ are outputs on a random sequence $\boldsymbol{s}_1 \sim \mathcal{U}(\{0,1\}^L)$ corresponding to using $\boldsymbol{\Pi}_{:i}^{\alpha}$ and $\boldsymbol{\Pi}_{:i}^{\beta}$ respectively.

We then apply a grouping algorithm GROUP to group correlated code-books together in each family. That is, in FAMILY(j), we create groups of code-books $\{S_1, S_2, \cdots\}$ such that for any two code-books $\boldsymbol{\Pi}_{:i}^{\alpha}$ and $\boldsymbol{\Pi}_{:i}^{\alpha'}$ that belong to a group S,

$$\text{Correlation}\left(\boldsymbol{s}_{i,j}^{\alpha},\boldsymbol{s}_{i,j}^{\alpha'},\mathcal{U}(\{0,1\}^L)\right)=1,$$

where $\boldsymbol{s}_{i,j}^{\alpha},\boldsymbol{s}_{i,j+1}^{\alpha'}$ are outputs on a random sequence $\boldsymbol{s}_1 \sim \mathcal{U}(\{0,1\}^L)$ corresponding to using $\boldsymbol{\Pi}_{:i}^{\alpha'}$ and $\boldsymbol{\Pi}_{:i}^{\alpha'}$ respectively. We call the resulting output of this operation as GROUP(FAMILY(j)). Similarly, we compute GROUP(FAMILY$(j+1)$).

One can then show the following two characteristics of GROUP(FAMILY(j)) and GROUP(FAMILY$(j+1)$):

1. For every set $S_1 \in$ GROUP(FAMILY(j)) there will exist one set $S_2 \in$ GROUP(FAMILY$(j+1)$), such that for all code-books $\boldsymbol{\Pi}_{:i}^{\alpha} \in$ FAMILY(j) and $\boldsymbol{\Pi}_{:i}^{\beta} \in$ FAMILY$(j+1)$, correlation will be 1 for output at positions j and $j+1$ respectively.
2. For every set $S_1 \in$ GROUP(FAMILY(j)) there can't exist two sets $S_2, S_2^* \in$ GROUP(FAMILY$(j+1)$), such that the maps in S_1 are correlated to maps from both S_2, S_2^* for output at positions j and $j+1$ respectively.

Let CORRELATION-MAP denote the map between GROUP(FAMILY(j)) and GROUP(FAMILY($j+1$)), which connects sets $S_1 \in$ GROUP(FAMILY(j)) to a set $S_2 \in$ GROUP(FAMILY($j+1$)) whose code-books have correlations for output at positions j and $j+1$ respectively.

The result will then follow from a counting argument. The number of possible pairs (can be equal maps) that can give correlations for output at position j are given by: $\sum_{S_1 \in \text{GROUP}(j)} |S_1|^2$. Similarly, the number of possible pairs that can give correlations for output at position $j+1$ are given by: $\sum_{S_2 \in \text{GROUP}(j+1)} |S_2|^2$. On the other hand, the number of possible pairs that can give correlations for output at position j and $j+1$ respectively are given by: $\sum_{S_1 \in \text{GROUP}(j); S_2 = \text{CORRELATION-MAP}(S_1)} |S_1||S_2|$. We then apply a AM-GM inequality to show that the average of the number of pairs for which correlation is 1 at outputs at either position j or $j+1$ is higher than the number of pairs for which correlation is 1 for output at positions j and $j+1$.

□

F.5 PROOF FOR STATISTICAL QUERY LOWER BOUND FOR GENERAL n

We present the main theorem statement again for readability.

Theorem F.18 (SQ dimension for general n)**.** *For the translation task* ***MLT***(d, n) *that has depth d and n characters per level, the statistical query dimension SQ-dim(****MLT***(d, n)*) is atleast $n^{\Omega(d)}$.*

Proof. We adapt the SQ-dimension proof for **MLT**$(d, 2)$ to show the SQ-dimension proof for **MLT**(d, n). We will design a family of code-books $\mathbf{\Pi} = \{\pi_i : \{0, 1, \cdots, n-1\}^2 \to \{0, 1, \cdots, n-1\}^2\}_{i=1}^{d}$, where each phrasebook builds on top of a translation task in **MLT**$(\log_2 n, 2)$ as follows:

For a character $a \in \{0, 1, \cdots, n-1\}$, suppose BIT$(a) \in \{0,1\}^{\log_2 n}$ indicates its binary representation, and NUMERIC represent map from binary representation to representation in $\{0, 1, \cdots, n-1\}$. Then, we design each map π in the code-book using a random translation task $\nu \in$ **MLT**$(\log_2 n, 2)$. For any tuple $(a, b) \in \{0, 1, \cdots, n-1\}^2$, output of π is given as

$$
\begin{aligned}
&\pi(a, b) = (o_1, o_2), \text{ where} \\
&o_1 = \text{NUMERIC}\left(\nu\left(\{\tilde{\boldsymbol{a}}_i \oplus \tilde{\boldsymbol{b}}_i\}_{i=1}^{\log_2 n}\right)\right) \\
&o_2 = \text{NUMERIC}\left(\nu\left(\tilde{\boldsymbol{b}}\right)\right) \\
&\tilde{\boldsymbol{a}} = \text{BIT}(a) \\
&\tilde{\boldsymbol{b}} = \text{BIT}(b)
\end{aligned}
$$

Primarily, the phrasebooks are defined as follows:

1. On a tuple of characters (a, b), we first compute their binary representations $(\text{BIT}(a), \text{BIT}(b))$. We then compute two intermediate outputs, one where a **xor** operation is applied on $\text{BIT}(a), \text{BIT}(b)$ at each bit, and another where $\text{BIT}(b)$ is simply copied. This operation is equivalent to applying a deterministic map on tuples of 2 characters ($\mathbf{\Delta}_6$ from Table 4), where tuples are created by pairing binary bits of a and b.
2. We then apply a random **MLT** of depth $\log_2 n$ on each of the intermediate outputs. This applies a random bijective map on the sequence of bits in the intermediate outputs, using a translation task in **MLT**$(\log_2 n, 2)$ (Lemma E.1).
3. The final tuple of characters is returned by applying a NUMERIC operation on the binary representations.

Thus, we have narrowed our focus on a special group of tasks from **MLT**(d, n) that applies **MLT**$(\log_2 n, 2)$ on the binary representations of the characters at each level. By Theorem F.10, we can create $2^{\Omega(\log_2 n)}$ maps that are pairwise uncorrelated for each level of translation task. We

can then give a similar proof as Theorem F.10 to show that we can create $\left(2^{\Omega(\log_2 n)}\right)^{\Omega(d)} = n^{\Omega(d)}$ translation tasks, using the above restriction, that are pairwise uncorrelated on any bit in the binary representation of any character in the output.

□

F.6 PROOFS OF USEFUL LEMMAS

Here we give the proofs for the useful lemmas necessary to prove Theorem F.10. We repeat the lemma statements for easier readability.

Lemma F.19 (Formulation of bijective maps for $n = 2$)**.** *Any bijective map $\pi : \{0,1\}^2 \to \{0,1\}^2$ can be expressed using **copy**, **not**, and **xor** operations. Furthermore, from 24 possible maps for π,*

1. *There are 6 maps $\mathbf{\Delta}_1, \mathbf{\Delta}_2, \cdots, \mathbf{\Delta}_6$ for which characters in the output tuple can be defined by **copy** and **xor** operations on the characters in the input tuple.*
2. ***Mirror maps:** For each map $\pi \in \{\mathbf{\Delta}_1, \mathbf{\Delta}_2, \cdots, \mathbf{\Delta}_6\}$, there exist mirror maps $\pi_{(1)}, \pi_{(2)}, \pi_{(3)}$ whose output on each input tuple can be defined by selective **not** operations on either or both characters of the output tuple of π. We call $\{\pi, \pi_{(1)}, \pi_{(2)}, \pi_{(3)}\}$ as a **mirror map set** of π, in short, Mirrorset(π).*

Map name (π)	Output for corresponding input tuple $\pi(a,b)$				General formulation on output for map π	
	$(0,0)$	$(0,1)$	$(1,0)$	$(1,1)$	$\pi(a,b)_1$	$\pi(a,b)_2$
$\mathbf{\Delta}_1$	$(0,0)$	$(0,1)$	$(1,0)$	$(1,1)$	$\mathbf{copy}_1(a,b)$	$\mathbf{copy}_2(a,b)$
$\mathbf{\Delta}_2$	$(0,0)$	$(0,1)$	$(0,1)$	$(1,0)$	$\mathbf{copy}_1(a,b)$	$\mathbf{xor}(a,b)$
$\mathbf{\Delta}_3$	$(0,0)$	$(1,0)$	$(0,1)$	$(1,1)$	$\mathbf{copy}_2(a,b)$	$\mathbf{copy}_1(a,b)$
$\mathbf{\Delta}_4$	$(0,0)$	$(1,1)$	$(0,1)$	$(1,0)$	$\mathbf{copy}_2(a,b)$	$\mathbf{xor}(a,b)$
$\mathbf{\Delta}_5$	$(0,0)$	$(1,0)$	$(1,1)$	$(0,1)$	$\mathbf{xor}(a,b)$	$\mathbf{copy}_1(a,b)$
$\mathbf{\Delta}_6$	$(0,0)$	$(1,1)$	$(1,0)$	$(0,1)$	$\mathbf{xor}(a,b)$	$\mathbf{copy}_2(a,b)$

Table 4: The table captures the definition of 6 bijective maps on 2-tuples $\{0,1\}^2 \to \{0,1\}^2$, whose output characters can be defined in terms of **copy** and **xor** operations on the input characters. Any other bijective map can be shown to belong to Mirrorset of one of these bijective maps.

Proof. There are 4 possible tuples $(0,0), (0,1), (1,0), (1,1)$. By Lemma E.2, the number of possible bijective maps are $4! = 24$. We will show that the output tuple of each map can be represented by the 3 operations.

Operation not creates mirror maps: For each map π, there exists 3 alternative maps $\pi_{(1)}, \pi_{(2)}, \pi_{(3)}$ such that if $\pi(a,b)_i$ represents the ith character in the output tuple for input tuple (a,b):

- $\pi_{(1)}$ selectively applies the **not** operation to the first character in the output tuple of π on any input tuple, i.e.
$$\pi_{(1)}(a,b) = (\mathbf{not}(\pi(a,b)_1), \pi(a,b))$$
for all tuples $(a,b) \in \{0,1\}^2$.
- $\pi_{(2)}$ selectively applies the **not** operation to the second character in the output tuple of π on any input tuple, i.e.
$$\pi_{(2)}(a,b) = (\pi(a,b)_1, \mathbf{not}(\pi(a,b)))$$
for all tuples $(a,b) \in \{0,1\}^2$.
- $\pi_{(3)}$ selectively applies the **not** operation to both characters in the output tuple of π on any input tuple, i.e.
$$\pi_{(3)}(a,b) = (\mathbf{not}(\pi(a,b)_1), \mathbf{not}(\pi(a,b)))$$
for all tuples $(a,b) \in \{0,1\}^2$.

Thus, for each map π, there exist 3 other alternative maps that simply modify the output of map π with the **not** operation.

After removing the mirror maps: We now show that there 6 possible maps that apply either a **xor** or a **copy** on the input characters to get the output characters. We name them $\mathbf{\Delta}_1, \mathbf{\Delta}_2, \cdots, \mathbf{\Delta}_6$. We give the output of each map on the 4 tuples in Table 4 and show that the each character in the output tuple can be represented using **copy** and **xor** operations.

□

Lemma F.20 (Correlation of bijective maps for $n = 2$)**.** *For two randomly selected maps $\pi^\alpha, \pi^\beta : \{0,1\}^2 \to \{0,1\}^2$, the following hold true.*

1. ***Correlation at output characters:*** *Fix an $i, j \in \{0,1\}$. With probability $\frac{1}{3}$ w.r.t. the random selection of the maps, the ith character in the output of π^α has non-zero correlation to jth character in the output of π^β, i.e.*

$$\begin{aligned}&\textit{Correlation}(\pi^\alpha(\cdot)_i, \pi^\beta(\cdot)_j, \mathcal{U}(\{0,1\}^2))\\ :=&\left|\Pr_{x\sim\mathcal{U}(\{0,1\}^2)}[\pi^\alpha(x)_i \neq \pi^\beta(x)_j] - \Pr_{x\sim\mathcal{U}(\{0,1\}^2)}[\pi^\alpha(x)_i = \pi^\beta(x)_j]\right| = 1.\end{aligned}$$

2. *With probability $\frac{1}{3}$ w.r.t. the random selection of the maps, both the characters in the outputs of π^α, π^β have non-zero correlation, i.e. either one of the cases hold true*

$$\textit{Correlation}(\pi^\alpha(\cdot)_1, \pi^\beta(\cdot)_1, \mathcal{U}(\{0,1\}^2)) = 1.$$
$$\textit{Correlation}(\pi^\alpha(\cdot)_2, \pi^\beta(\cdot)_2, \mathcal{U}(\{0,1\}^2)) = 1.$$

or

$$\textit{Correlation}(\pi^\alpha(\cdot)_1, \pi^\beta(\cdot)_2, \mathcal{U}(\{0,1\}^2)) = 1.$$
$$\textit{Correlation}(\pi^\alpha(\cdot)_2, \pi^\beta(\cdot)_1, \mathcal{U}(\{0,1\}^2)) = 1.$$

Other cases are not possible,
i.e. for any $i \in \{1,2\}$, both Correlation$(\pi^\alpha(\cdot)_i, \pi^\beta(\cdot)_1, \mathcal{U}(\{0,1\}^2)) > 0$ and Correlation$(\pi^\alpha(\cdot)_i, \pi^\beta(\cdot)_2, \mathcal{U}(\{0,1\}^2)) = 1$ can't hold true.

Proof. We prove the lemma for case 1, cases 2 and 3 can be similarly proved. From Lemma E.2, π^α and π^β can be randomly selected from a set of 24 possible candidates. On the other hand, Lemma F.19 shows that there are 6 maps $\{\mathbf{\Delta}_1, \cdots, \mathbf{\Delta}_6\}$ whose output can be defined in terms of **copy** and **xor** operations of characters in the input tuple. For each π in this set, there are mirror maps $\pi_{(1)}, \pi_{(2)}, \pi_{(3)}$ whose output are defined by selective **not** operations on the output of π, and the set of 4 maps is represented by Mirrorset(π).

The proof will follow from 2 steps: first, we argue about correlations when π^α and π^β are selected from $\{\mathbf{\Delta}_1, \cdots, \mathbf{\Delta}_6\}$, and then we argue about the general case when π^α and π^β are selected from the general set of bijective maps.

- In Lemma F.22, we show that for two maps that are randomly selected from $\{\mathbf{\Delta}_1, \cdots, \mathbf{\Delta}_6\}$, the correlation is 1 with probability $1/3$.
- The remaining possibility is when π^α and π^β belong to Mirrorset(π_i) and Mirrorset(π_j) for some $\pi_i, \pi_j \in \{\mathbf{\Delta}_1, \cdots, \mathbf{\Delta}_6\}$. We show in Lemma F.21, correlation of π^α and π^β will be equal to correlation of π_i, π_j.

Thus, we can combine all the observations to show that for two random maps π^α, π^β that belong to Mirrorset(π_i) and Mirrorset(π_j) for some $\pi_i, \pi_j \in \{\mathbf{\Delta}_1, \cdots, \mathbf{\Delta}_6\}$,

$$\begin{aligned}&\text{Correlation}(\pi^\alpha(\cdot)_1, \pi^\beta(\cdot)_1, \mathcal{U}(\{0,1\}^2))\\ =\ &\text{Correlation}(\pi_i(\cdot)_1, \pi_j(\cdot)_1, \mathcal{U}(\{0,1\}^2)) = \begin{cases}1, & \text{w.p. } 1/3 \text{ w.r.t. randomness in } \pi_i, \pi_j\\ 0, & \text{otherwise.}\end{cases}\end{aligned}$$

□

Lemma F.21. *The following holds true for any maps π^α and π^β with $\pi^\alpha, \pi^\beta \in \{\boldsymbol{\Delta}_1, \cdots, \boldsymbol{\Delta}_6\}$ and for all $\tilde{\pi}^\alpha \in$ Mirrorset(π^α), $\tilde{\pi}^\beta \in$ Mirrorset(π^β):*

$$\textit{Correlation}(\pi^\alpha(\cdot)_1, \pi^\beta(\cdot)_1, \mathcal{U}(\{0,1\}^2)) = \textit{Correlation}(\tilde{\pi}^\alpha(\cdot)_1, \tilde{\pi}^\beta(\cdot)_1, \mathcal{U}(\{0,1\}^2)),$$

Proof. We prove as follows:

$$\begin{aligned}
\text{Correlation}(\pi^\alpha(\cdot)_1, \pi^\beta(\cdot)_1, \mathcal{U}(\{0,1\}^2)) &= \left| \Pr_{x \sim \mathcal{U}(\{0,1\}^2)}[\pi^\alpha(x)_1 \neq \pi^\beta(x)_1] - \Pr_{x \sim \mathcal{U}(\{0,1\}^2)}[\pi^\alpha(x)_1 = \pi^\beta(x)_1] \right| \\
&= \left| 2 \Pr_{x \sim \mathcal{U}(\{0,1\}^2)}[\pi^\alpha(x)_1 \neq \pi^\beta(x)_1] - 1 \right| \qquad (16) \\
&= \begin{cases} \left| 2 \Pr_{x \sim \mathcal{U}(\{0,1\}^2)}[\pi^\alpha(x)_1 \neq \pi^\beta(x)_1] - 1 \right|, \text{if condition ``c1'' is true} \\ \left| 1 - 2 \Pr_{x \sim \mathcal{U}(\{0,1\}^2)}[\pi^\alpha(x)_1 = \pi^\beta(x)_1] \right|, \text{if condition ``c2'' is true} \end{cases} \qquad (17) \\
&= \left| \Pr_{x \sim \mathcal{U}(\{0,1\}^2)}[\tilde{\pi}^\alpha(x)_1 \neq \tilde{\pi}^\beta(x)_1] - \Pr_{x \sim \mathcal{U}(\{0,1\}^2)}[\tilde{\pi}^\alpha(x)_1 = \tilde{\pi}^\beta(x)_1] \right| \\
&= \text{Correlation}(\tilde{\pi}^\alpha(\cdot)_1, \tilde{\pi}^\beta(\cdot)_1, \mathcal{U}(\{0,1\}^2)), \qquad (18)
\end{aligned}$$

where the second and the penultimate steps follow from the law of total probability. Here, condition "c1" holds when either case is true,

$$\begin{aligned}
\tilde{\pi}^\alpha(x)_1 = \textbf{not}(\pi^\alpha(x)_1), \quad \tilde{\pi}^\beta(x)_1 = \textbf{not}(\pi^\alpha(x)_1), \quad \text{for all } x \in \{0,1\}^2 \\
\tilde{\pi}^\alpha(x)_1 = \pi^\alpha(x)_1, \quad \tilde{\pi}^\beta(x)_1 = \pi^\alpha(x)_1, \quad \text{for all } x \in \{0,1\}^2.
\end{aligned}$$

and condition "c2" holds when either case is true,

$$\begin{aligned}
\tilde{\pi}^\alpha(x)_1 = \pi^\alpha(x)_1, \quad \tilde{\pi}^\beta(x)_1 = \textbf{not}(\pi^\alpha(x)_1), \quad \text{for all } x \in \{0,1\}^2 \\
\tilde{\pi}^\alpha(x)_1 = \textbf{not}(\pi^\alpha(x)_1), \quad \tilde{\pi}^\beta(x)_1 = \pi^\alpha(x)_1, \quad \text{for all } x \in \{0,1\}^2.
\end{aligned}$$

One of condition "c1" or condition "c2" is true because $\tilde{\pi}^\alpha$ and π^α (similarly, $\tilde{\pi}^\beta$ and π^β) are mirror maps.

□

Lemma F.22. *The following holds true for two randomly selected maps π^α and π^β with $\pi^\alpha, \pi^\beta \in \{\boldsymbol{\Delta}_1, \cdots, \boldsymbol{\Delta}_6\}$:*

- *With probability $1/3$ w.r.t. random selection,*

$$\textit{Correlation}(\pi^\alpha(\cdot)_1, \pi^\beta(\cdot)_1, \mathcal{U}(\{0,1\}^2)) > 0 \quad (= 1).$$

- *With probability $1/3$ w.r.t. random selection,*

$$\textit{Correlation}(\pi^\alpha(\cdot)_2, \pi^\beta(\cdot)_2, \mathcal{U}(\{0,1\}^2)) > 0 \quad (= 1).$$

Proof. We prove for case 1, proof for case 2 is analogous.

Among $\{\boldsymbol{\Delta}_1, \cdots, \boldsymbol{\Delta}_6\}$, we can create three family of maps: $F_{\textbf{copy}_1} : \{\boldsymbol{\Delta}_1, \boldsymbol{\Delta}_2\}$, $F_{\textbf{copy}_2} : \{\boldsymbol{\Delta}_3, \boldsymbol{\Delta}_4\}$, $F_{\textbf{xor}} : \{\boldsymbol{\Delta}_5, \boldsymbol{\Delta}_6\}$ that are identical in operation ($\textbf{copy}_1, \textbf{copy}_2, \textbf{xor}$ respectively) at the first character in output tuple. This would imply, if the π^α and π^β both belong to one of these families, Correlation$(\pi^\alpha(\cdot)_1, \pi^\beta(\cdot)_1, \mathcal{U}(\{0,1\}^2))$ will be 1.

On the other hand, one can show that for any operation $f_1, f_2 \in \{\textbf{copy}_1, \textbf{copy}_2, \textbf{xor}\}$ with $f_1 \neq f_2$ will have Correlation$(f_1, f_2, \mathcal{U}(\{0,1\}^2)) = 0$. That would then suggest that if π^α and π^β belong to different families among $F_{\textbf{copy}_1}, F_{\textbf{copy}_2}, F_{\textbf{xor}}$, then Correlation$(\pi^\alpha(\cdot)_1, \pi^\beta(\cdot)_1, \mathcal{U}(\{0,1\}^2))$ will be 0.

By a simple counting argument, with probability $\frac{1}{3}$, two maps π^α and π^β randomly selected from $\{\boldsymbol{\Delta}_1, \cdots, \boldsymbol{\Delta}_6\}$ will have non-zero correlation. □

G UPPER BOUND: CONTEXT-ENHANCED LEARNING OF $\mathbf{MLT}(n, d)$ WITH SIMPLE SURROGATE MODEL

G.1 SETUP OF SURROGATE MODEL WITH IN-CONTEXT CAPABILITY

Given $d+1$ alphabets $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_{d+1}$ of size n and d bijective phrasebooks $\pi_i : \boldsymbol{A}_i^2 \to \boldsymbol{A}_{i+1}^2$ The input of the translation process is an even-length sequence in the first alphabet, which we denote as $\boldsymbol{s}_1 \in \boldsymbol{A}_1^L$ where L is the sequence length. The translation process modifies the input string recursively from $\boldsymbol{s}_i$ to $\boldsymbol{s}_{i+1}$ through the following 4 sub-processes:

1. *Circular shift*: The characters in $\boldsymbol{s}_i \in \boldsymbol{A}_i^L$ are shifted by 1 character leftward (and wrapped around to the end if necessary) to give sequence $\tilde{\boldsymbol{s}}_i \in \boldsymbol{A}_i^L$. For $j \in [L]$ we have
$$\tilde{\boldsymbol{s}}_{i,j} = \boldsymbol{s}_{i,(j+1)\%L}.$$
2. *Split*: The shifted sequence $\tilde{\boldsymbol{s}}_i$ splits into consecutive 2-tuples $(\tilde{\boldsymbol{s}}_i^{(1)}, \tilde{\boldsymbol{s}}_i^{(2)}, \cdots, \tilde{\boldsymbol{s}}_i^{(L/2)})$ where for any $k \in [L/2]$,
$$\tilde{\boldsymbol{s}}_i^{(k)} = (\tilde{\boldsymbol{s}}_{i,2k-1}, \tilde{\boldsymbol{s}}_{i,2k}) \in \boldsymbol{A}_i^2.$$
3. *Translate*: Using the *dictionary* $\pi_i : \boldsymbol{A}_i^2 \to \boldsymbol{A}_{i+1}^2$, each 2-tuple $\tilde{\boldsymbol{s}}_i^{(k)} \in \boldsymbol{A}_i^2$ is translated to a tuple in $\boldsymbol{A}_{i+1}^2$, giving
$$(\pi_i(\tilde{\boldsymbol{s}}_i^{(1)}), \pi_i(\tilde{\boldsymbol{s}}_i^{(2)}), \cdots, \pi_i(\tilde{\boldsymbol{s}}_i^{(L/2)})).$$
4. *Merge*: The translated tuples are merged to give the sequence $\boldsymbol{s}_{i+1} \in \boldsymbol{A}_{i+1}^L$. Where for $j \in [L]$ we have
$$s_{i+1,j} = \pi_i(\tilde{\boldsymbol{s}}_i^{(\lceil j/2 \rceil)})_{j\%2}.$$

Now let us revisit the surrogate model introduced in Section 4.1. Without loss of generality let $\boldsymbol{A}_1 = \boldsymbol{A}_2 = \cdots = \boldsymbol{A}_{d+1} := \boldsymbol{A} = \{1, 2, \ldots, n\}$. For any single character $a \in \boldsymbol{A}$, let its vector representation be a one-hot vector $\boldsymbol{e}_a \in \mathbb{R}^n$ such that $(\boldsymbol{e}_a)_a = 1$. For any 2-tuple $(a, b) \in \boldsymbol{A}^2$, let its vector representation be a n^2-dimensional vector $\boldsymbol{v}(a, b) \triangleq \boldsymbol{e}_a \otimes \boldsymbol{e}_b$. Note that $\boldsymbol{v}(a, b)$ is also a one-hot vector where

$$\boldsymbol{v}(a, b)_i = \begin{cases} 1 & \text{if } i = an + b \\ 0 & \text{elsewhere} \end{cases}.$$

We use the notation $\bar{\boldsymbol{e}}_a$ (long one-hot) to denote a one-hot vector in $\mathbb{R}^{n^2}$ with a-th position being 1 to avoid confusion.

Definition G.1 (Matrix Representation of Sequence)**.**
For a length-L input sequence $\boldsymbol{s}_i = (s_{i,1}, \ldots, s_{i,L})$, let its matrix representation be $\mathbf{Mat}(\boldsymbol{s}_i) \triangleq \boldsymbol{V}_i \in \mathbb{R}^{n^2 \times L/2}$ that

$$\boldsymbol{V}_i = \begin{bmatrix} | & | & \cdots & | \\ \boldsymbol{v}(s_{i,1}, s_{i,2}) & \boldsymbol{v}(s_{i,3}, s_{i,4}) & \cdots & \boldsymbol{v}(s_{i,L-1}, s_{i,L}) \\ | & | & \cdots & | \end{bmatrix}$$

For each $j \in [L/2]$, we use $\boldsymbol{V}_i^{(j)}$ to denote the j-th column of V_i. We also denote the above conversion from a sequence $\boldsymbol{s}_i$ to its matrix form as $\boldsymbol{V}_i = \mathbf{Mat}(\boldsymbol{s}_i)$ and assume that $\boldsymbol{V}_1$ serves as the input to the surrogate model.

Note that the matricization operation is invertible by construction: for each column $\boldsymbol{V}_i^{(j)}$, let $x = \arg\max \boldsymbol{V}_i^{(j)}$, we may read off the two characters in the original alphabet by computing $\mathbf{Mat}^{-1}(\boldsymbol{V}_i^{(j)}) = (\lceil x/n \rceil, x\%n)$.

At each level of translation, we assume the surrogate model will perform the following operations to $\boldsymbol{V}_i$:

(a) Circular shift from V_i to $\tilde{V}_i$

Definition G.2 (Circular Shifting Operator `Shift`)**.**
Given a matrix representation $\boldsymbol{V} \in \mathbb{R}^{n^2 \times L/2}$ of a sequence $\boldsymbol{s}$, the circular shifting operator `Shift` acts on $\boldsymbol{V}$ as $\texttt{Shift}(\boldsymbol{V}) := \tilde{\boldsymbol{V}} \in \mathbb{R}^{n^2 \times L/2}$ where for all $j \in [L/2]$,

$$\tilde{\boldsymbol{V}}^{(j)} = \boldsymbol{Q}\boldsymbol{V}^{(j)} \odot \boldsymbol{Q}^\top \boldsymbol{V}^{((j+1)\%L)}$$

where $\boldsymbol{Q} = (I_n \otimes \mathbf{1}_n)(\mathbf{1}_n \otimes I_n)^\top$, $\mathbf{1}_n \in \mathbb{R}^{n \times 1}$ is the all-ones vector, and $\odot$ is the Hadamard product.

Lemma G.3 (Equivalence of `Shift` and circular shift)**.**
For any sequence $\boldsymbol{s} \in \boldsymbol{A}^L$, let $\tilde{\boldsymbol{s}}$ be the circular shifted $\boldsymbol{s}$, then

$$\textbf{Mat}(\tilde{\boldsymbol{s}}) = \texttt{Shift}(\textbf{Mat}(\boldsymbol{s}))\,.$$

Proof of Lemma G.3. In this proof we will show that $\textbf{\textit{Mat}}(\tilde{\boldsymbol{s}})$ and $\texttt{Shift}(\textbf{\textit{Mat}}(\boldsymbol{s}))$ agrees on every column.

Fix a column $j \in [n/2]$, without loss of generality let the input sequence $\boldsymbol{s}$ be $(a, b, c, d) \in \boldsymbol{A}$ starting from the $(2j-1)$-th position to the $(2j+2)$-th position (wrapped around when necessary). By construction each output column of $\tilde{\boldsymbol{V}}^{(j)}$ is dependent on at most $\boldsymbol{V}^{(j)}$ and $\boldsymbol{V}^{(j+1)}$ which corresponds to 4 characters in the sequence $\boldsymbol{s}$.

By definition of $\boldsymbol{V}$ we then have $\boldsymbol{V}^{(j)} = \boldsymbol{v}(a, b) = \boldsymbol{e}_a \otimes \boldsymbol{e}_b$ and $\boldsymbol{V}^{(j+1)\%L} = \boldsymbol{v}(c, d) = \boldsymbol{e}_c \otimes \boldsymbol{e}_d$. It follows that

$$\begin{aligned}
\tilde{\boldsymbol{V}}^{(j)} &= \boldsymbol{Q}\boldsymbol{V}^{(j)} \odot \boldsymbol{Q}^\top \boldsymbol{V}^{((j+1)\%L)} \\
&= \left((I_n \otimes \mathbf{1}_n)\left(\mathbf{1}_n^\top \otimes I_n^\top\right)(\boldsymbol{e}_a \otimes \boldsymbol{e}_b)\right) \odot \left((\mathbf{1}_n \otimes I_n)\left(I_n^\top \otimes \mathbf{1}_n^\top\right)(\boldsymbol{e}_c \otimes \boldsymbol{e}_d)\right) \\
&= \left((I_n \otimes \mathbf{1}_n)\left(\mathbf{1}_n^\top \boldsymbol{e}_a \otimes I_n^\top \boldsymbol{e}_b\right)\right) \odot \left((\mathbf{1}_n \otimes I_n)\left(I_n^\top \boldsymbol{e}_c \otimes \mathbf{1}_n^\top \boldsymbol{e}_d\right)\right) \\
&= \left((I_n \otimes \mathbf{1}_n)(1 \otimes \boldsymbol{e}_b)\right) \odot \left((\mathbf{1}_n \otimes I_n)(\boldsymbol{e}_c \otimes 1)\right) \quad (\boldsymbol{e}_a, \boldsymbol{e}_d \text{ are one-hot, } \mathbf{1}_n^\top \boldsymbol{e}_a = \mathbf{1}_n^\top \boldsymbol{e}_d = 1) \\
&= \left((I_n \otimes \mathbf{1}_n)(\boldsymbol{e}_b \otimes 1)\right) \odot \left((\mathbf{1}_n \otimes I_n)(1 \otimes \boldsymbol{e}_c)\right) \\
&= (\boldsymbol{e}_b \otimes \mathbf{1}_n) \odot (\mathbf{1}_n \otimes \boldsymbol{e}_c) \\
&= \boldsymbol{e}_b \otimes \boldsymbol{e}_c. \qquad \text{(by definition of Kronecker product)}
\end{aligned}$$

Note that $\boldsymbol{e}_b \otimes \boldsymbol{e}_c$ is just $\textbf{Mat}(\tilde{\boldsymbol{s}})^{(j)}$ since the $(2j-1)$-th and the $2j$-th character of the shifted sequence $\tilde{\boldsymbol{s}}$ is now (b, c).

This concludes the proof. □

(b) Translation from $\tilde{V}_i$ to V_{i+1}

With `Shift` effectively completing *Merge*, *Circular shift*, and *Split*, what remains is the translation leveraging π_i. Since $\boldsymbol{A}_i = \boldsymbol{A}_{i+1} = \boldsymbol{A} = [n]$, there is a natural bijection between the space of binary column-stochastic matrix and the space of all possible (not necessarily bijective) mappings between 2-tuples from $\boldsymbol{A}$. Concretely

Definition G.4 (Column-Stochastic Matrix Representation of phrasebook)**.**
Given phrasebook $\pi : \boldsymbol{A}^2 \to \boldsymbol{A}^2$, its matrix representation is defined to be $\texttt{Matrix}^{(\pi)} \in \mathbb{R}^{n^2 \times n^2}$ that for $i, j \in [n^2]$,

$$\texttt{Matrix}^{(\pi)}_{j,i} = \begin{cases} 1 & \text{if } \pi(\lceil i/n \rceil, i\%n) = (\lceil j/n \rceil, j\%n) \\ 0 & \text{elsewhere} \end{cases}.$$

Let $\boldsymbol{V}_{i+1} = \texttt{Matrix}\tilde{\boldsymbol{V}}_i$ where $\texttt{Matrix}$ is a binary column-stochastic matrix, we can show that, a complete translation process for one level can be formally expressed as follows:

Lemma G.5 (Equivalence of $\texttt{Matrix}$ and *Translate*)**.**
For any sequence $\boldsymbol{s}_i \in \boldsymbol{A}^L$, let π_i be a bijective phrasebook $\boldsymbol{A}^2 \to \boldsymbol{A}^2$ defined in Section 2.2, then

$$\boldsymbol{V}_{i+1} = \texttt{Matrix}^{(\pi_i)}\, \texttt{Shift}\left(\mathbf{Mat}\left(\boldsymbol{s}_i\right)\right) = \mathbf{Mat}\left(T_{\pi_i}(\boldsymbol{s}_i)\right) = \mathbf{Mat}(\boldsymbol{s}_{i+1}).$$

Proof of Lemma G.5. Fix any column $j \in [L/2]$, let (a, b) be the $(2j-1)$-th and the $2j$-th character of the shifted sequence $\tilde{\boldsymbol{s}}_i$. Let $(c, d) = \pi_i(a, b)$, by the translation construction we know the $(2j-1)$-th and the $2j$-th character of $\boldsymbol{s}_{i+1}$ is just c and d. Hence $\mathbf{Mat}(\boldsymbol{s}_{i+1})^{(j)} = v(c, d)$.

From Lemma G.3 we know that $\texttt{Shift}(\mathbf{Mat}(\boldsymbol{s}_i))^{(j)} = v(a, b)$. By construction of $\texttt{Matrix}^{(\pi_i)}$ above, we know that $\texttt{Matrix}^{(\pi_i)}\, \texttt{Shift}(\mathbf{Mat}(\boldsymbol{s}_i))^{(j)}$ is a one-hot vector at the $(cn+d)$-th position, i.e. $\boldsymbol{V}_{i+1}^{(j)} = \texttt{Matrix}^{(\pi_i)}\, \texttt{Shift}(\mathbf{Mat}(\boldsymbol{s}_i))^{(j)} = \boldsymbol{e}_c \otimes \boldsymbol{e}_c = \boldsymbol{v}(c, d)$. Since $\mathbf{Mat}(\boldsymbol{s}_{i+1})^{(j)} = \boldsymbol{V}_{i+1}^{(j)}$ for all j, we have $\boldsymbol{V}_{i+1} = \mathbf{Mat}(\boldsymbol{s}_{i+1})$. □

Parameterization of the Translation Matrix $\texttt{Matrix}$.

With the two key operations in place, now we can introduce the surrogate model in its full detail. Since the multi-level translation task is a naturally sequential operation, we model the surrogate operations as a multi-layer network as well (which also matches with the solution found by Llama-based models when trained on real data as in Section 3.2).

For each level, the surrogate model needs to present both in-context learning capability at the initialization and in-weight capability toward the end of context-enhanced learning on a certain set of phrasebooks $\boldsymbol{\Pi}^*$. The in-weight capability requires certain parameter to store $\boldsymbol{\Pi}^*$ on its own that is independent of the context.

To capture both capabilities at the same time, we parameterize the translation matrix $\texttt{Matrix}_i$ as a combination of in-context information $\boldsymbol{C}_i \in \mathbb{R}^{n^2 \times n^2}$ and in-weight memory $\boldsymbol{W}_i \in \mathbb{R}^{n^2 \times n^2}$. Concretely, we define

$$\texttt{Matrix}_i = \texttt{HardMax}\left(\boldsymbol{C}_i + \boldsymbol{W}_i\right)$$

where $\texttt{HardMax}$ is the column-wise hard-max function converting $\boldsymbol{C}_i + \boldsymbol{W}_i$ to a binary column stochastic matrix. Column k in $\texttt{Matrix}_i$, which we denote as $\texttt{Matrix}_i^{(k)}$, is equal to the one-hot vector at $\arg\max(\boldsymbol{C}_i^{(k)} + \boldsymbol{W}_i^{(k)})$.

Surrogate Model with In-Context Capability

Now we can formally introduce the surrogate model.

Definition G.6 (Surrogate Model for **MLT**)**.**
The surrogate model for $\mathbf{MLT}(d, n)$ can be represented by the recursive expression

$$\boldsymbol{V}_{i+1} = \texttt{HardMax}(\boldsymbol{C}_i + \boldsymbol{W}_i)\, \texttt{Shift}(\boldsymbol{V}_i) \tag{19}$$

The learnable parameters for the surrogate model are the weight matrices $\{\boldsymbol{W}_i\}_{i=1}^d := \{\boldsymbol{W}_1, \boldsymbol{W}_2, \ldots, \boldsymbol{W}_d\}$. We denote the surrogate model parameterized by $\{\boldsymbol{W}_i\}_{i=1}^d$ as $\textsc{Surr-MLT}_{\{\boldsymbol{W}_i\}_{i=1}^d}(\cdot)$ which maps the input and the in-context information to the output as

$$\begin{aligned}
\boldsymbol{V}_{d+1} &= \textsc{Surr-MLT}_{\{\boldsymbol{W}_i\}_{i=1}^d}(\boldsymbol{C}_1, \boldsymbol{C}_2, \ldots, \boldsymbol{C}_d, \boldsymbol{V}_1) \\
&\triangleq \texttt{HardMax}(\boldsymbol{C}_d + \boldsymbol{W}_d) \\
&\quad \texttt{Shift}\left(\texttt{HardMax}(\boldsymbol{C}_{d-1} + \boldsymbol{W}_{d-1})\, \texttt{Shift}\left(\cdots \texttt{HardMax}(\boldsymbol{C}_1 + \boldsymbol{W}_1)\, \texttt{Shift}(\boldsymbol{V}_1)\cdots\right)\right).
\end{aligned} \tag{20}$$

In this surrogate model, the in-context descriptive text DESC is just $\{\boldsymbol{C}_1, \ldots, \boldsymbol{C}_d\}$, where each matrix is directly being passed into the corresponding layer. When providing information about a set of phrasebooks $\boldsymbol{\Pi} = \{\pi_i\}_{i=1}^d$, we have $\boldsymbol{C}_i = \texttt{Matrix}^{(\pi_i)}$ where $\texttt{Matrix}^{(\pi_i)}$ is the column-stochastic matrix representation of π_i as defined in Definition G.4.

When partial phrasebook information is provided (corresponding to dropping certain `ab->CD` entries in the language model context), we zero-out the corresponding column in $\boldsymbol{C}_i$. When no in-context information is provided for level i, we just have C_i be the all-zero matrix containing no information (assuming zero as prior).

Now let us check what does Definition 2.2 (ICL-capable) and Definition 2.1 (specific task-capable) mean in the context of the surrogate model. To make things more rigorous we introduce two stronger notions of capabilities:

Definition G.7 (Strongly $\mathbf{MLT}(n, d)$-ICL-capable surrogate model)**.**
We say a surrogate model $\textsc{Surr-MLT}_{\{\boldsymbol{W}_i\}_{i=1}^d}(\cdot)$ is strongly $\mathbf{MLT}(n, d)$-ICL-capable if for any set of phrasebooks $\boldsymbol{\Pi} = \{\pi_i\}_{i=1}^d$ in $\mathbf{MLT}(n, d)$, for any input sequence $\boldsymbol{s}_1 \in \boldsymbol{A}^L$ where L is even, we have

$$\textsc{Surr-MLT}_{\{\boldsymbol{W}_i\}_{i=1}^d}(\texttt{Matrix}^{(\pi_1)}, \ldots, \texttt{Matrix}^{(\pi_d)}, \mathbf{Mat}(\boldsymbol{s}_1)) = \mathbf{Mat}(\mathbf{MLT}_{\boldsymbol{\Pi}}(\boldsymbol{s}_1)).$$

Definition G.8 (Strongly $\mathbf{MLT}_{\boldsymbol{\Pi}^*}$-capable surrogate model)**.**
For a fixed set of phrasebooks $\boldsymbol{\Pi}^* = \{\pi_i^*\}_{i=1}^d$ in $\mathbf{MLT}(n, d)$, we say a surrogate model $\textsc{Surr-MLT}_{\{\boldsymbol{W}_i\}_{i=1}^d}(\cdot)$ is strongly $\mathbf{MLT}_{\boldsymbol{\Pi}^*}$-capable if for any input sequence $\boldsymbol{s}_1 \in \boldsymbol{A}^L$ where L is even, we have

$$\textsc{Surr-MLT}_{\{\boldsymbol{W}_i\}_{i=1}^d}(\mathbf{0}, \ldots, \mathbf{0}, \mathbf{Mat}(\boldsymbol{s}_1)) = \mathbf{Mat}(\mathbf{MLT}_{\boldsymbol{\Pi}^*}(\boldsymbol{s}_1)).$$

Now we can show the following properties of the surrogate model $\textsc{Surr-MLT}_{\{\boldsymbol{W}_i\}_{i=1}^d}(\cdot)$:

Lemma G.9. *When $\|\boldsymbol{W}_i\|_0 < \frac{1}{2}$ for all $i \in [d]$, $\textsc{Surr-MLT}_{\{\boldsymbol{W}_i\}_{i=1}^d}$ is strongly $\mathbf{MLT}(d, n)$-ICL-capable.*

Proof. Fix any set of phrasebooks $\boldsymbol{\Pi} = \{\pi_i\}_{i=1}^d$ and its corresponding matrix representations $\{\texttt{Matrix}^{(\pi_i)}\}_{i=1}^d$. Since $\|\boldsymbol{W}_i\|_0 < \frac{1}{2}$, for any column k, no entries in $\boldsymbol{W}_i^{(k)}$ can flip the argmax of $\texttt{Matrix}^{(\pi_i)(k)} + \boldsymbol{W}_i^{(k)}$ away from being $\arg\max \texttt{Matrix}^{(\pi_i)(k)}$. Therefore we have $\texttt{Matrix}_i = \texttt{HardMax}(\texttt{Matrix}^{(\pi_i)} + \boldsymbol{W}_i) = \texttt{Matrix}^{(\pi_i)}$ for all layers $i \in [d]$.

By Lemma G.5, for all $i \in [d]$ we have $\texttt{Matrix}_i = \texttt{Matrix}^{(\pi_i)}$ recovering T_{π_i}. Hence for any input sequence $\boldsymbol{s}_1 \in \boldsymbol{A}^L$, we have $\textsc{Surr-MLT}_{\{\boldsymbol{W}_i\}_{i=1}^d}(\texttt{Matrix}^{(\pi_1)}, \ldots, \texttt{Matrix}^{(\pi_d)}, \mathbf{Mat}(\boldsymbol{s}_1)) = \mathbf{Mat}(\mathbf{MLT}_{\boldsymbol{\Pi}}(\boldsymbol{s}_1))$. □

Lemma G.10. *Fix a target set of phrasebooks $\boldsymbol{\Pi}^* = \{\pi_i^*\}_{i=1}^d$, when $\texttt{HardMax}(\boldsymbol{W}_i) = \texttt{Matrix}^{(\pi_i^*)}$ for all $i \in [d]$, $\textsc{Surr-MLT}_{\{\boldsymbol{W}_i\}_{i=1}^d}(\cdot)$ is strongly $\mathbf{MLT}_{\boldsymbol{\Pi}^*}$-capable.*

Proof. By Lemma G.5, for all $i \in [d]$ we have $\texttt{Matrix}_i = \texttt{HardMax}(\boldsymbol{W}_i + \mathbf{0}) = \texttt{Matrix}^{(\pi_i^*)}$ recovering $T_{\pi_i^*}$. Hence for any input sequence $\boldsymbol{s}_1 \in \boldsymbol{A}^L$, we have $\textsc{Surr-MLT}_{\{\boldsymbol{W}_i\}_{i=1}^d}(\mathbf{0}, \ldots, \mathbf{0}, \mathbf{Mat}(\boldsymbol{s}_1)) = \mathbf{Mat}(\mathbf{MLT}_{\boldsymbol{\Pi}^*}(\boldsymbol{s}_1))$. □

Lemma G.9 suggests that when the weight matrices have small initializations, the model has perfect ICL capability. Meanwhile Lemma G.10 suggests that when the weights $\boldsymbol{W}_i$ recover $\texttt{Matrix}^{(\pi_i^*)}$ in

the column-wise hard-max sense, then the surrogate model can perform $\mathbf{MLT}_{\mathbf{\Pi}^*}$ when no context is being provided ($\boldsymbol{C}_i = \mathbf{0}$).

G.2 Learning $\mathbf{\Pi}^*$ in $\mathbf{MLT}(d, n)$ with Heuristics Search

In this section, we provide a brute-force algorithm that can learn any target set of phrasebooks $\mathbf{\Pi}^*$ in $\mathbf{MLT}(d, n)$ using a single "short" sequence whose length is not exponentially dependent on d.

Before proceeding to the details, let us first investigate more on the nature of **MLT** and the surrogate model. For simplicity of notations, given $\mathbf{\Pi}^* = \{\pi_1^*, \dots, \pi_d^*\}$, we denote the general translation operator $\texttt{Matrix}$ as $\boldsymbol{P}$, and denote the translation operator $\texttt{Matrix}^{(\pi_1^*)}$ as $\boldsymbol{W}_i^*$. Also, with slight abuse of notations we use $\mathbf{MLT}_{\mathbf{\Pi}^*}(\boldsymbol{V}_1)$ to denote $\mathbf{Mat}\left(\mathbf{MLT}_{\mathbf{\Pi}^*}(\mathbf{Mat}^{-1}(\boldsymbol{V}_1))\right)$.

First, we characterize the input sequence that is good for providing learning signals.

Definition G.11 ($\mathbf{\Pi}^*$-coverable input)**.**
Fix a target set of phrasebooks $\mathbf{\Pi}^* = \{\pi_1^*, \dots, \pi_d^*\}$ in $\mathbf{MLT}(d, n)$ and an input matrix $\boldsymbol{V}_1 \in \mathbb{R}^{n^2 \times L}$, let $\tilde{\boldsymbol{V}}_1^*, \boldsymbol{V}_2^*, \tilde{\boldsymbol{V}}_2^*, \dots, \tilde{\boldsymbol{V}}_d^*, \boldsymbol{V}_{d+1}^* \in \mathbb{R}^{n^2 \times L}$ be the intermediate outputs when applying $\mathbf{MLT}_{\mathbf{\Pi}^*}$ on $\boldsymbol{V}_1$. We say $\boldsymbol{V}_1$ is $\mathbf{\Pi}^*$-coverable if for all levels $i \in [d]$, $\tilde{\boldsymbol{V}}_i^*$ is of rank-n^2.

Note that as a matrix with only one-hot columns, $\tilde{\boldsymbol{V}}_i^*$ being rank-n^2 suggests that for all $k \in [n^2]$, there exists some column $j \in [L]$ such that $\tilde{\boldsymbol{V}}_i^{*(j)} = \boldsymbol{e}_k$. In the context of the translation process, it means that the correct translation process of a $\mathbf{\Pi}^*$-coverable input $\boldsymbol{V}_1$ would require all entries of all phrasebooks in $\mathbf{\Pi}^*$.

Next we will show that if we use the context to condition all translation operators $\boldsymbol{P}_l$ of the surrogate model to be $\boldsymbol{P}^{(\pi_l^*)}$ for all but one level $l \in [d] \setminus \{i\}$ (i as the unconditioned level), then for the surrogate model to correctly perform $\mathbf{MLT}_{\mathbf{\Pi}^*}$, the operator for the unconditioned level i must also be equal to $\boldsymbol{P}^{(\pi_i^*)}$.

Lemma G.12 (Uniqueness of a single $\boldsymbol{P}_i$ when conditioning all other levels)**.**

Fix a target set of phrasebooks $\mathbf{\Pi}^* = \{\pi_1^*, \dots, \pi_d^*\}$ in $\mathbf{MLT}(d, n)$ and a $\mathbf{\Pi}^*$-coverable input $\boldsymbol{V}_1$. For any level $i \in [d]$, consider a surrogate model $\textsc{Surr-MLT}_{\{\boldsymbol{W}_i\}_{i=1}^d}(\boldsymbol{C}_1, \boldsymbol{C}_2, \dots, \boldsymbol{C}_d, \cdot)$ with certain context $\boldsymbol{C}_1, \boldsymbol{C}_2, \dots, \boldsymbol{C}_d$ such that $\boldsymbol{P}_l = \boldsymbol{W}_l^*$ for all $l \in [d]$ except $l = i$. Then $\textsc{Surr-MLT}_{\{\boldsymbol{W}_i\}_{i=1}^d}(\boldsymbol{C}_1, \boldsymbol{C}_2, \dots, \boldsymbol{C}_d, \boldsymbol{V}_1) = \mathbf{MLT}_{\mathbf{\Pi}^*}(\boldsymbol{V}_1)$ if and only if

$$\boldsymbol{P}_i = \texttt{HardMax}\left(\boldsymbol{C}_i + \boldsymbol{W}_i\right) = \boldsymbol{W}_i^*.$$

Proof. This lemma is a direct consequence of the bijective property of the translation process shown in Lemma E.1. Let $\tilde{\boldsymbol{V}}_1^*, \boldsymbol{V}_2^*, \tilde{\boldsymbol{V}}_2^*, \dots, \tilde{\boldsymbol{V}}_d^*, \boldsymbol{V}_{d+1}^* \in \mathbb{R}^{n^2 \times L}$ be the intermediate outputs when applying $\mathbf{MLT}_{\mathbf{\Pi}^*}$ on $\boldsymbol{V}_1$, and let $\tilde{\boldsymbol{V}}_1, \boldsymbol{V}_2, \tilde{\boldsymbol{V}}_2, \dots, \tilde{\boldsymbol{V}}_d, \boldsymbol{V}_{d+1} \in \mathbb{R}^{n^2 \times L}$ be the intermediate outputs when applying $\textsc{Surr-MLT}_{\{\boldsymbol{W}_i\}_{i=1}^d}(\boldsymbol{C}_1, \boldsymbol{C}_2, \dots, \boldsymbol{C}_d, \cdot)$ as described. Since we assume $\boldsymbol{P}_l = \boldsymbol{P}^{(\pi_l^*)}$ for all $l < i$, we have $\boldsymbol{V}_i = \boldsymbol{V}_i^*$ and therefore $\tilde{\boldsymbol{V}}_i = \tilde{\boldsymbol{V}}_i^*$.

On the other end, since $\textsc{Surr-MLT}_{\{\boldsymbol{W}_i\}_{i=1}^d}(\boldsymbol{C}_1, \boldsymbol{C}_2, \dots, \boldsymbol{C}_d, \boldsymbol{V}_1) = \mathbf{MLT}_{\mathbf{\Pi}^*}(\boldsymbol{V}_1) = \boldsymbol{V}_{d+1}^*$ and $\boldsymbol{P}_l = \boldsymbol{P}^{(\pi_l^*)}$ for all $l > i$, by the invertible property of the translation process (Lemma E.1) we must have $\tilde{\boldsymbol{V}}_{i+1} = \tilde{\boldsymbol{V}}_{i+1}^*$ and thus $\boldsymbol{V}_{i+1} = \boldsymbol{V}_{i+1}^*$. Combining both ends we know that

$$\boldsymbol{P}_i \tilde{\boldsymbol{V}}_i = \boldsymbol{V}_{i+1} = \boldsymbol{V}_{i+1}^* = \boldsymbol{W}_i^* \tilde{\boldsymbol{V}}_i^* = \boldsymbol{W}_i^* \tilde{\boldsymbol{V}}_i. \tag{21}$$

Since $\tilde{\boldsymbol{V}}_i = \tilde{\boldsymbol{V}}_i^*$ is rank n^2 by the $\mathbf{\Pi}^*$-coverable assumption and $\boldsymbol{P}_i \in \mathbb{R}^{n^2 \times n^2}$, it must be so that $\boldsymbol{P}_i = \boldsymbol{W}_i^*$. □

If we further condition on the held-out level $\boldsymbol{P}_i$ such that we only leave one column of $\boldsymbol{P}_i^{(k)}$ not necessarily equal to $\boldsymbol{W}_i^{*(k)}$, we have the following corollary

Corollary G.13 (Uniqueness of a single $\boldsymbol{P}_i$ column when conditioning everything else)**.**

Fix a target set of phrasebooks $\boldsymbol{\Pi}^* = \{\pi_1^*, \ldots, \pi_d^*\}$ in $\mathbf{MLT}(d, n)$ and a $\boldsymbol{\Pi}^*$-coverable input $\boldsymbol{V}_1$. For any level $i \in [d]$ and translation entry $k \in [n^2]$, consider a surrogate model $\text{SURR-MLT}_{\{\boldsymbol{W}_i\}_{i=1}^d}(\boldsymbol{C}_1, \boldsymbol{C}_2, \ldots, \boldsymbol{C}_d, \cdot)$ with certain context $\boldsymbol{C}_1, \boldsymbol{C}_2, \ldots, \boldsymbol{C}_d$ such that $\boldsymbol{P}_l^{(j)} = \boldsymbol{W}_l^{*(j)}$ for all $(l, j) \in [d] \times [n^2] \backslash \{(i, k)\}$. Then $\text{SURR-MLT}_{\{\boldsymbol{W}_i\}_{i=1}^d}(\boldsymbol{C}_1, \boldsymbol{C}_2, \ldots, \boldsymbol{C}_d, \boldsymbol{V}_1) = \mathbf{MLT}_{\boldsymbol{\Pi}^*}(\boldsymbol{V}_1)$ if and only if

$$\boldsymbol{P}_i^{(k)} = \texttt{HardMax}\left(\boldsymbol{C}_i^{(j)} + \boldsymbol{W}_i^{(j)}\right) = \boldsymbol{W}_i^{*(k)}.$$

This suggests that if we condition everything else except for one column of the translation operator, then to match the final output on a $\boldsymbol{\Pi}^*$-coverable sequence, the model must recover the held-out column to be the same as the ground truth in the set of phrasebooks.

Now we can introduce the search algorithm, which simply enumerate over all translation columns $\boldsymbol{W}_i^{(j)}$ as learning target, generate a contextual information that only leaves that column unconditioned, and search over all possible one-hot vectors for $\boldsymbol{W}_i^{(j)}$ until the output matches with $\mathbf{MLT}_{\boldsymbol{\Pi}^*}$. Once the output matches, by Corollary G.13 we know $\texttt{HardMax}(\boldsymbol{W}_i^{(j)})$ recovers $\boldsymbol{W}_i^{*(j)}$ and we move on to the next learning target. The algorithm can be formalized as follows:

Algorithm 1 Context-Enhanced Searching Algorithm for $\mathbf{MLT}(d, n)$

Input:
input $\boldsymbol{V}_1 \in \mathbb{R}^{n^2 \times L}$, label $\boldsymbol{V}_{d+1}^* \in \mathbb{R}^{n^2 \times L}$, descriptive text $\boldsymbol{W}_1^*, \ldots, \boldsymbol{W}_d^* \in \mathbb{R}^{n^2 \times n^2}$

Initialize $\boldsymbol{W}_1, \ldots, W_d \leftarrow \boldsymbol{0}$ # Start with zero initialization
for $i = 1$ **to** d **do**
  for $k = 1$ **to** n^2 **do**
    Initialize $\boldsymbol{C}_{i(k)} \leftarrow \boldsymbol{W}_i^* \left(\boldsymbol{I}_{n^2} - \text{diag}(\bar{\boldsymbol{e}}_k)\right)$ # Create masked context matrix
    # Search Loop
    for $a = 1$ **to** n^2 **do**
      $\boldsymbol{W}_i^{(k)} \leftarrow \bar{\boldsymbol{e}}_a$ # Search over one-hot columns
      $\boldsymbol{V}_{d+1} \leftarrow \text{SURR-MLT}_{\{\boldsymbol{W}_i\}_{i=1}^d}(\boldsymbol{W}_1^* \ldots, \boldsymbol{W}_{i-1}^*, \boldsymbol{C}_{i(k)}, \boldsymbol{W}_{i+1}^* \ldots, \boldsymbol{W}_d^*, \boldsymbol{V}_1)$
      if $\boldsymbol{V}_{d+1} = \boldsymbol{V}_{d+1}^*$ **then**
        break # Break when found the right column
      end if
    end for
  end for
end for
Return $\boldsymbol{W}_1, \ldots, \boldsymbol{W}_d$.

Theorem G.14 (Learning $\boldsymbol{\Pi}^*$ with context-enhanced search with $\boldsymbol{\Pi}^*$-coverable input)**.** *For any target set of phrasebooks $\boldsymbol{\Pi}^* = \{\pi_1^*, \ldots, \pi_d^*\}$ in $\mathbf{MLT}(d, n)$, given an $\boldsymbol{\Pi}^*$-coverable input $\boldsymbol{V}_1$ and the corresponding ground truth label $\boldsymbol{V}_{d+1}^* = \mathbf{MLT}_{\boldsymbol{\Pi}^*}(\boldsymbol{V}_1)$, Algorithm 1 terminates with $\boldsymbol{W}_i = \boldsymbol{W}_i^*$ for all $i \in [d]$ with $O(n^4 d)$ forward passes through the surrogate model.*

Proof. The statement can be proven by a simple induction.

Let the inductive hypothesis be such that when the enumeration goes to k-th column of the i-th layer, if $\texttt{HardMax}(\boldsymbol{W}_l^{(j)} + \boldsymbol{W}_l^{*(j)}) = \boldsymbol{W}_l^{*(j)}$ for all $(l, j) \in [d] \times [n^2]$ and $\boldsymbol{W}_l^{(j)} = \boldsymbol{W}_l^{*(j)}$ for all (l, j) such that $l < i$ or $l = i \wedge j < k$, then the search loop (**??** 8: line 13) breaks with $\boldsymbol{W}_i^{(k)} = \boldsymbol{W}_i^{*(k)}$ while $\texttt{HardMax}(\boldsymbol{W}_l^{(j)} + \boldsymbol{W}_l^{*(j)}) = \boldsymbol{W}_l^{*(j)}$ for all $(l, j) \in [d] \times [n^2]$ is preserved.

The base case is satisfied as with zero initialization, we have $\texttt{HardMax}(\boldsymbol{W}_l^{(j)} + \boldsymbol{W}_l^{*(j)}) = \texttt{HardMax}(\boldsymbol{0} + \boldsymbol{W}_l^{*(j)}) = \boldsymbol{W}_l^{*(j)}$ for all $(l, j) \in [d] \times [n^2]$ and there are no requirements for $\boldsymbol{W}_i^{(k)} = \boldsymbol{W}_i^{*(k)}$ yet.

For the induction step, we note that with the condition of $\texttt{HardMax}(\boldsymbol{W}_l^{(j)} + \boldsymbol{W}_l^{*(j)}) = \boldsymbol{W}_l^{*(j)}$ for all $(l, j) \in [d] \times [n^2]$, $\boldsymbol{W}_1^* \ldots, \boldsymbol{W}_{i-1}^*, \boldsymbol{C}_{i(k)}, \boldsymbol{W}_{i+1}^* \ldots, \boldsymbol{W}_d^*$ will correctly condition all columns of $\boldsymbol{P}$'s except for the $\boldsymbol{P}_i^{(k)}$ since

$$\boldsymbol{C}_{i(k)}^{(k)} = \boldsymbol{W}_i^* \left(\boldsymbol{I}_{n^2} - \text{diag}(\bar{\boldsymbol{e}}_k)\right)^{(k)} = \boldsymbol{0}. \tag{22}$$

Thus by Corollary G.13, we know that the search loop will terminate when it finds $\boldsymbol{W}_i^{(k)} = \boldsymbol{W}_i^{*(k)}$. The newly added column provides the correct inductive hypothesis on $\boldsymbol{W}_l^{(j)} = \boldsymbol{W}_l^{*(j)}$ for the next enumeration step.

By induction to $i = d$ and $k = n^2$, we will be able to recover $\boldsymbol{W}_i = \boldsymbol{W}_i^*$ for all $i \in [d]$. □

Given that we can learn $\boldsymbol{W}_i$ effectively with $\boldsymbol{\Pi}^*$-coverable input, how should we construct such inputs? It turned out that with high probability, short random strings suffices.

Lemma G.15 (Distribution of intermediate sequences)**.** *Fix a target set of phrasebooks $\boldsymbol{\Pi}^* = \{\pi_1^*, \ldots, \pi_d^*\}$ in $\mathbf{MLT}(d, n)$. Let $\mathbf{V}_1 \in \mathbb{R}^{n^2 \times L}$ be a random input matrix to $\mathbf{MLT}(d, n)$ such that each column $\mathbf{V}_1^{(j)}$ is i.i.d. sampled from $\mathcal{U}(\{\bar{\boldsymbol{e}}_k\}_{k=1}^{n^2})$ (the uniform distribution over one-hot vectors $\{\bar{\boldsymbol{e}}_k\}_{k=1}^{n^2}$), then the columns of intermediate random sequences $\tilde{\mathbf{V}}_1^*, \mathbf{V}_2^*, \tilde{\mathbf{V}}_2^*, \ldots, \tilde{\mathbf{V}}_d^*, \mathbf{V}_{d+1}^* \in \mathbb{R}^{n^2 \times L}$ obtained by passing the input $\mathbf{V}_1$ through $\mathbf{MLT}_{\boldsymbol{\Pi}^*}$ also follow the same i.i.d. uniform distribution.*

Proof. We will prove the claim by induction on depth i. Let the inductive hypothesis be that columns in $\mathbf{V}_i$ independently follow an uniform distribution over the one-hot vectors $\{\bar{\boldsymbol{e}}_k\}_{k=1}^{n^2}$. Note that the base case is just the assumption.

Now we prove for the inductive step. For any $j \in [L]$, we can write $\mathbf{V}_i^{(j)} = \boldsymbol{e}_{\mathrm{a}_{(2j-1)}} \otimes \boldsymbol{e}_{\mathrm{a}_{(2j)}}$ where $\mathrm{a}_{(i)}$'s follows i.i.d. $\mathcal{U}([n])$. Intuitively this means each 2-tuple in the random input sequence is formed from two i.i.d. uniformly random characters, which is straightforward by construction. Now by Lemma G.3 we have $\tilde{\mathbf{V}}_i^{*(j)} = \boldsymbol{e}_{\mathrm{a}_{(2j)}} \otimes \boldsymbol{e}_{\mathrm{a}_{(2j+1)}}$ also following $\mathcal{U}(\{\bar{\boldsymbol{e}}_k\}_{k=1}^{n^2})$. Note that there is total independency of $\tilde{\mathbf{V}}_i^{*(j)}$ with respect to the set of any other columns of $\tilde{\mathbf{V}}_i$ since $\mathrm{a}_{(2j)}$ and $\mathrm{a}_{(2j+1)}$ are independent from the generative process of any other columns in $\tilde{\mathbf{V}}_i$.

With columns in $\tilde{\mathbf{V}}_i$ i.i.d. following $\mathcal{U}(\{\bar{\boldsymbol{e}}_k\}_{k=1}^{n^2})$, permuting the indices via $\boldsymbol{P}^{(\pi_i^*)}$ does not change the distribution by symmetry of the uniform distribution. Therefore we have columns of $\mathbf{V}_{i+1} = \boldsymbol{P}^{(\pi_i^*)}\tilde{\mathbf{V}}_i$ also i.i.d. distributed as $\mathcal{U}(\{\bar{\boldsymbol{e}}_k\}_{k=1}^{n^2})$ and this complete the inductive step. By induction on i we have the desired statement proved. □

Since each column in $\tilde{\mathbf{V}}_i$ i.i.d. follows $\mathcal{U}(\{\bar{\boldsymbol{e}}_k\}_{k=1}^{n^2})$, sampling $\tilde{\mathbf{V}}_i$ to be rank n^2 becomes identical to the classic coupon collection problem (see Lemma G.27 from Motwani (1995)). Thus we have the following bound:

Lemma G.16 (Short $\boldsymbol{\Pi}^*$-coverable random sequence)**.**
A random sequence $\mathbf{V}_1$ of length $L \geq 2n^2 \log \frac{nd}{\delta}$ is $\boldsymbol{\Pi}^*$-coverable with probability at least $1 - \delta$.

Proof. Let the event A_i denote that $\tilde{\mathbf{V}}_i$ is not rank n^2. By Lemma G.15, each column of $\tilde{\mathbf{V}}_i$ is i.i.d. distributed following $\mathcal{U}(\{\bar{\boldsymbol{e}}_k\}_{k=1}^{n^2})$. Thus making $\tilde{\mathbf{V}}_i$ being rank n^2 is equivalent to a coupon collecting

problem (Motwani, 1995) with set size n^2. By Lemma G.27 we know that with $L = 2n^2 \log \frac{nd}{\delta}$, $\mathbb{P}[A_i] \leq \frac{\delta}{d}$. Thus by a simple union bound the probability that $\tilde{\mathbf{V}}_i$ being not $\mathbf{\Pi}^*$-coverable is

$$\mathbb{P}\left[\bigcup_{i=1}^{d} A_i\right] \leq \sum_{i=1}^{d} \mathbb{P}[A_i] \leq d\frac{\delta}{d} = \delta. \tag{23}$$

□

Now we can apply the above result and extend Theorem G.14.

Corollary G.17 (Learning $\mathbf{\Pi}^*$ with random input using heuristics search)**.** *For any target set of phrasebooks $\mathbf{\Pi}^* = \{\pi_1^*, \ldots, \pi_d^*\}$ in $\mathbf{MLT}(d, n)$, with probability at least $1-\delta$ over a uniformly random input $\boldsymbol{V}_1$ of length $L = 2n^2 \log \frac{nd}{\delta}$, Algorithm 1 provided with ground truth label $\boldsymbol{V}_{d+1}^* = \mathbf{MLT}_{\mathbf{\Pi}^*}(\boldsymbol{V}_1)$ terminates with $\boldsymbol{W}_i = \boldsymbol{W}_i^*$ for all $i \in [d]$ with $O(n^4 d)$ forward passes through the surrogate model.*

G.3 Learning $\mathbf{\Pi}^*$ in $\mathbf{MLT}(2, n)$ with Surrogate Gradient Descent

In this section we take the analysis one step beyond the heuristics searching regime. We will show that any set of phrasebooks $\mathbf{\Pi}^* = \{\pi_1^*, \pi_2^*\}$ can be sample-efficiently learned by a gradient-descent based algorithm. In this particular case, the surrogate model is parameterized by

$$\boldsymbol{V}_3 = \textsc{Surr-MLT}_{\{\boldsymbol{W}_i\}_{i=1}^d}(\boldsymbol{C}_1, \boldsymbol{C}_2, \boldsymbol{V}_1) \triangleq \texttt{HardMax}(\boldsymbol{C}_2 + \boldsymbol{W}_2)\,\texttt{Shift}\left(\texttt{HardMax}(\boldsymbol{C}_1 + \boldsymbol{W}_1)\,\texttt{Shift}\left(\boldsymbol{V}_1\right)\right). \tag{24}$$

We start with any weight initializations $\boldsymbol{W}_1^{(0)}, \boldsymbol{W}_2^{(0)} \in \mathbb{R}^{n^2 \times n^2}$ satisfying $\|\boldsymbol{W}_1^{(0)}\|_1 < \frac{1}{2}, \|\boldsymbol{W}_2^{(0)}\|_1 < \frac{1}{2}$, by Lemma G.9 the initialization is strongly $\mathbf{MLT}(n, d)$-ICL-capable.

For simplicity, we denote the ground truth permutation matrix induced by π_1^* as $\boldsymbol{W}_1^* \triangleq \boldsymbol{P}^{(\pi_1^*)}$ and similarly the ground truth permutation matrix induced by π_2^* as $\boldsymbol{W}_2^* \triangleq \boldsymbol{P}^{(\pi_2^*)}$. From Lemma G.10 we know that the learning is successful if we have $\texttt{HardMax}(\boldsymbol{W}_1) = \boldsymbol{W}_1^*$ and $\texttt{HardMax}(\boldsymbol{W}_2) = \boldsymbol{W}_2^*$ [3] (i.e. the maximum index of each weight column agrees with that of the ground truth).

We employ a layer-wise gradient descent algorithm for the learning process. The algorithm takes in a single fixed sequence $\boldsymbol{s}_1$ with matrix representation $\boldsymbol{V}_1$ and its corresponding ground truth label

$$\boldsymbol{V}_3^* \triangleq \mathbf{Mat}(\mathbf{MLT}_{\mathbf{\Pi}^*}(\boldsymbol{s}_1)) = \textsc{Surr-MLT}_{(\boldsymbol{W}_1^{(0)}, \boldsymbol{W}_2^{(0)})}(\boldsymbol{W}_1^*, \boldsymbol{W}_2^*, \boldsymbol{V}_1). \tag{25}$$

Given the input and label, we employ the following gradient descent based algorithm to update the weights:

The training happens in a layer-wise and column-wise fashion: We first freeze $\boldsymbol{W}_2$ and set $\boldsymbol{W}_1$ as the trainable parameter. For each entry $k \in [n^2]$, we create a context matrix $\boldsymbol{C}_{1(k)} \triangleq \boldsymbol{W}_1^*(\boldsymbol{I}_{n^2} - \text{diag}(\boldsymbol{e}_k))$ which essentially creates a copy of $\boldsymbol{W}_1^*$ except of setting the k-th column to be zero. Then we take a forward pass through the surrogate model with the one-column dropped-out context and get output $\boldsymbol{V}_{3(1,k)} \triangleq \textsc{Surr-MLT}_{(\boldsymbol{W}_1, \boldsymbol{W}_2)}(\boldsymbol{C}_{1(k)}, \boldsymbol{W}_2^*, \boldsymbol{V}_1)$. Here the subscript $\cdot_{(1,k)}$ denotes the final output when the i-th column of the first context matrix is being dropped.

The weight update follows a surrogate gradient update scheme where we use the MSE loss: $\mathcal{L} = \|\boldsymbol{V}_{3(1,k)} - \boldsymbol{V}_3^*\|_2^2$. Since it is difficult to take gradient through the hardmax function, we instead compute the gradient of the loss with respect to the translation matrix $\boldsymbol{P}_1 = \texttt{HardMax}\left(\boldsymbol{W}_1 + \boldsymbol{C}_{1(k)}\right)$ and apply the update $\boldsymbol{W}_1^{(k)} \leftarrow \boldsymbol{W}_1^{(k)} - \frac{\partial \mathcal{L}}{\partial \boldsymbol{P}_1^{(k)}}$. We apply such gradient update *twice* for each dropped column k.

[3] Note that this is not the unique solution to attain strongly $\mathbf{MLT}_{\mathbf{\Pi}^*}$-capable model

For the second layer, we freeze the first layer $\boldsymbol{W}_1$ and apply one-column dropouts to the $\boldsymbol{C}_2$. Similarly we apply the surrogate gradient update $\boldsymbol{W}_2 \leftarrow \boldsymbol{W}_2 - \frac{\partial \mathcal{L}}{\partial \boldsymbol{P}_2}$ but we only need one gradient step per column.

Algorithm 2 Context-Enhanced Layerwise Gradient Descent

Input: input $\boldsymbol{V}_1 \in \mathbb{R}^{n^2 \times L}$, label $\boldsymbol{V}_3^* \in \mathbb{R}^{n^2 \times L}$, descriptive text $\boldsymbol{W}_1^*, \boldsymbol{W}_2^* \in \mathbb{R}^{n^2 \times n^2}$, init $\boldsymbol{W}_1^{(0)}, \boldsymbol{W}_2^{(0)} \in \mathbb{R}^{n^2 \times n^2}$

# Train the first layer
for $k = 1$ **to** n^2 **do**
  $\boldsymbol{C}_{1(k)} \triangleq \boldsymbol{W}_1^* (\boldsymbol{I}_{n^2} - \mathrm{diag}(\boldsymbol{e}_k))$ # Create context matrix with k-th column dropped.
  for $t = 1$ **to** 2 **do**
    $\boldsymbol{V}_{3(1,k)} \leftarrow \textsc{Surr-MLT}_{(\boldsymbol{W}_1, \boldsymbol{W}_2)}(\boldsymbol{C}_{1(k)}, \boldsymbol{W}_2^*, \boldsymbol{V}_1)$ # Forward pass
    $\mathcal{L} \leftarrow \|\boldsymbol{V}_{3(1,k)} - \boldsymbol{V}_3^*\|_2^2$
    $\boldsymbol{W}_1^{(k)} \leftarrow \boldsymbol{W}_1^{(k)} - \frac{\partial l}{\partial \boldsymbol{P}_1^{(k)}}$ # Surrogate gradient update
  end for
end for

# Train the second layer
for $k = 1$ **to** n^2 **do**
  $\boldsymbol{C}_{2(k)} \triangleq \boldsymbol{W}_1^* (\boldsymbol{I}_{n^2} - \mathrm{diag}(\boldsymbol{e}_k))$ # Create context matrix with k-th column dropped.
  $\boldsymbol{V}_{3(2,k)} \leftarrow \textsc{Surr-MLT}_{(\boldsymbol{W}_1, \boldsymbol{W}_2)}(W_1^*, \boldsymbol{C}_{2(k)}, \boldsymbol{V}_1)$ # Forward pass
  $\mathcal{L} \leftarrow \|\boldsymbol{V}_{3(2,k)} - \boldsymbol{V}_3^*\|_2^2$
  $\boldsymbol{W}_2^{(k)} \leftarrow \boldsymbol{W}_2^{(k)} - \frac{\partial l}{\partial \boldsymbol{P}_2^{(k)}}$ # Surrogate gradient update
end for
Return $\boldsymbol{W}_1, \boldsymbol{W}_2$

We claim the surrogate gradient descent update can correctly recover $\boldsymbol{P}_1^*$ and $\boldsymbol{P}_2^*$ similar to the heuristics search case.

Theorem G.22 (Learning $\boldsymbol{\Pi}^*$ with context-enhanced surrogate GD with $\boldsymbol{\Pi}^*$-coverable input)**.**

For any initialization $\boldsymbol{W}_{1(0)}, \boldsymbol{W}_{2(0)} \in \mathbb{R}^{n^2 \times n^2}$ such that $\|\boldsymbol{W}_{1(0)}\|_0 \leq \frac{1}{2}$ and $\|\boldsymbol{W}_{2(0)}\|_0 \leq \frac{1}{2}$, for any target set of phrasebooks $\boldsymbol{\Pi}^* = \{\pi_1^*, \pi_2^*\}$ in $\mathbf{MLT}(2, n)$, given an $\boldsymbol{\Pi}^*$-coverable input $\boldsymbol{V}_1$ and the corresponding ground truth label $\boldsymbol{V}_3^* = \mathbf{MLT}_{\boldsymbol{\Pi}^*}(\boldsymbol{V}_1)$, Algorithm 1 terminates with $\texttt{HardMax}(\boldsymbol{W}_1) = \boldsymbol{W}_1^*$ *and* $\texttt{HardMax}(\boldsymbol{W}_2) = \boldsymbol{W}_2^*$.

To prove for Theorem G.22, we will carefully analyze the learning of the first layer and second layer respectively, and provide a similar induction argument as in the proof for the heuristics search case.

G.3.1 Learning the First Layer

To study the learning process we first need to compute the closed-form gradient $\frac{\partial \mathcal{L}}{\partial \boldsymbol{P}_1^{(k)}}$, which requires the following lemma:

Lemma G.18 (Gradient with respect to incorrect column in $\boldsymbol{P}_1$)**.**

When only the k-th column of translation matrix $\boldsymbol{P}_1^{(k)}$ is not equal to $\boldsymbol{P}_1^{*(k)}$, let $\boldsymbol{P}_1^{(k)} = \boldsymbol{e}_a \otimes \boldsymbol{e}_b$ and $\boldsymbol{P}_1^{*(k)} = \boldsymbol{e}_{a^*} \otimes \boldsymbol{e}_{b^*}$ for some $a, b, a^*, b^* \in [n]$. If $\boldsymbol{P}_1^{(k)}$ is used in the forward pass, there exists

$\alpha \in \mathbb{Z}_+$ and $\beta \in \mathbb{N}$ such that the gradient of $\mathcal{L}$ with respect to $\boldsymbol{P}_1^{(k)}$ is of the form

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{P}_1^{(k)}} = \begin{cases} (2\alpha+2\beta)\,(\mathbf{1}_n \otimes \boldsymbol{e}_b) - (2\alpha+2\beta)\,(\mathbf{1}_n \otimes \boldsymbol{e}_{b^*}) + 2\beta\,(\boldsymbol{e}_{a^*} \otimes \mathbf{1}_n) & \text{if } a^* = a, b^* \neq b \\ (2\alpha+2\beta)\,(\boldsymbol{e}_a \otimes \mathbf{1}_n) - (2\alpha+2\beta)\,(\boldsymbol{e}_{a^*} \otimes \mathbf{1}_n) + 2\beta\,(\mathbf{1}_n \otimes \boldsymbol{e}_{b^*}) & \text{if } a^* \neq a, b^* = b \\ (2\alpha+2\beta)\,(\boldsymbol{e}_a \otimes \mathbf{1}_n) + (2\alpha+2\beta)\,(\mathbf{1}_n \otimes \boldsymbol{e}_b) - 2\alpha\,(\boldsymbol{e}_{a^*} \otimes \mathbf{1}_n) - 2\alpha\,(\mathbf{1}_n \otimes \boldsymbol{e}_{b^*}) & \text{if } a^* \neq a, b^* \neq b. \end{cases}$$

Proof of Lemma G.18. In this proof we use $\bar{\boldsymbol{e}}_a$ (long one-hot) to denote the one-hot vector in $\mathbb{R}^{n^2}$ with 1 on the a-th index and use $\boldsymbol{e}_a$ (short one-hot) to denote the one-hot vector in $\mathbb{R}^n$ with 1 on the a-th index. We use $\tilde{\boldsymbol{V}}_1, \boldsymbol{V}_2, \tilde{\boldsymbol{V}}_2$ and $\boldsymbol{V}_3$ to denote the intermediate sequences attained with translation matrix $\boldsymbol{P}_1^{(k)}$ and $\tilde{\boldsymbol{V}}_1^*, \boldsymbol{V}_2^*, \tilde{\boldsymbol{V}}_2^*$ and $\boldsymbol{V}_3^*$ to denote the counterfactual intermediate sequences should the forward pass is done with the ground truth translations $\boldsymbol{P}_1^{*(k)} = \boldsymbol{W}_1^*$.

Now we can proceed to the gradient calculations. First note that with $\mathcal{L}(\boldsymbol{P}_1) = \|\boldsymbol{V}_3 - \boldsymbol{V}_3^*\|_2^2$, by chain rule we have

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{P}_1} = \sum_{j=1}^{L} \left(\frac{\partial \boldsymbol{V}_3^{(j)}}{\partial \boldsymbol{P}_1}\right)^\top \left(\frac{\partial \|\boldsymbol{V}_3^{(j)} - \boldsymbol{V}_3^{*(j)}\|_2^2}{\partial \boldsymbol{V}_3^{(j)}}\right) = 2\sum_{j=1}^{L} \left(\frac{\partial \boldsymbol{V}_3^{(j)}}{\partial \boldsymbol{P}_1}\right)^\top \left(\boldsymbol{V}_3^{(j)} - \boldsymbol{V}_3^{*(j)}\right). \tag{26}$$

Specifically for each column $l \in [n^2]$ of $\boldsymbol{P}_1$ we have

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{P}_1^{(l)}} = 2\sum_{j=1}^{L} \left(\frac{\partial \boldsymbol{V}_3^{(j)}}{\partial \boldsymbol{P}_1^{(l)}}\right)^\top \left(\boldsymbol{V}_3^{(j)} - \boldsymbol{V}_3^{*(j)}\right). \tag{27}$$

Let us fix a particular column $j \in [L]$ in the output $\boldsymbol{V}_3$ and compute the gradients. With $\boldsymbol{M}^{(j)}$ as the j-th column of the matrix $\boldsymbol{M}$, the computation graph for the forward pass is of the form:

$$\begin{array}{ccccccccc} \boldsymbol{V}_1^{(j)} & \to & \tilde{\boldsymbol{V}}_1^{(j)} & \xrightarrow{\boldsymbol{P}_1} & \boldsymbol{V}_2^{(j)} & \to & \tilde{\boldsymbol{V}}_2^{(j)} & \xrightarrow{\boldsymbol{P}_2} & \boldsymbol{V}_3^{(j)} \\ & \nearrow & & & & \nearrow & & & \\ \boldsymbol{V}_1^{(j+1)} & \to & \tilde{\boldsymbol{V}}_1^{(j+1)} & \xrightarrow{\boldsymbol{P}_1} & \boldsymbol{V}_2^{(j+1)} & & & & \\ & \nearrow & & & & & & & \\ \boldsymbol{V}_1^{(j+2)} & & & & & & & & \end{array} \tag{28}$$

We can see that $\boldsymbol{V}_3^{(j)}$ is only affected by $\boldsymbol{P}_1$ through $\boldsymbol{V}_2^{(j)}$ and $\boldsymbol{V}_2^{(j+1)}$.

Let us first compute $\partial \boldsymbol{V}_3^{(j)} / \partial \boldsymbol{P}_1$. Assume $\boldsymbol{V}_2^{(j)} = \bar{\boldsymbol{e}}_p$ and $\boldsymbol{V}_2^{(j+1)} = \bar{\boldsymbol{e}}_q$ for some $p, q \in [n^2]$, $\boldsymbol{V}_3^{(j)}$ is computed as

$$\boldsymbol{V}_3^{(j)} = \boldsymbol{W}_2^* \tilde{\boldsymbol{V}}_2^{(j)} = \boldsymbol{W}_2^* \left(\boldsymbol{Q}\boldsymbol{V}_2^{(j)} \odot \boldsymbol{Q}^\top \boldsymbol{V}_2^{(j+1)}\right) = \boldsymbol{W}_2^* \left(\boldsymbol{Q}\boldsymbol{P}_1^{(p)} \odot \boldsymbol{Q}^\top \boldsymbol{P}_1^{(q)}\right). \tag{29}$$

We may express the Hadamard product in the following two ways:

$$\begin{aligned} \boldsymbol{V}_3^{(j)} &= \boldsymbol{W}_2^* \left(\boldsymbol{Q}\boldsymbol{P}_1^{(p)} \odot \boldsymbol{Q}^\top \boldsymbol{P}_1^{(q)}\right) = \boldsymbol{W}_2^* \mathrm{diag}\left(\boldsymbol{Q}\boldsymbol{P}_1^{(p)}\right) \boldsymbol{Q}^\top \boldsymbol{P}_1^{(q)}; \\ &= \boldsymbol{W}_2^* \left(\boldsymbol{Q}^\top \boldsymbol{P}_1^{(q)} \odot \boldsymbol{Q}\boldsymbol{P}_1^{(p)}\right) = \boldsymbol{W}_2^* \mathrm{diag}\left(\boldsymbol{Q}^\top \boldsymbol{P}_1^{(q)}\right) \boldsymbol{Q}\boldsymbol{P}_1^{(p)}. \end{aligned} \tag{30}$$

Therefore when $p \neq q$ we have

$$\frac{\partial \boldsymbol{V}_3^{(j)}}{\partial \boldsymbol{P}_1^{(p)}} = \boldsymbol{W}_2^* \mathrm{diag}\left(\boldsymbol{Q}^\top \boldsymbol{P}_1^{(q)}\right) \boldsymbol{Q}; \quad \frac{\partial \boldsymbol{V}_3^{(j)}}{\partial \boldsymbol{P}_1^{(q)}} = \boldsymbol{W}_2^* \mathrm{diag}\left(\boldsymbol{Q}\boldsymbol{P}_1^{(p)}\right) \boldsymbol{Q}^\top; \quad \forall l \notin \{p, q\} : \frac{\partial \boldsymbol{V}_3^{(j)}}{\partial \boldsymbol{P}_1^{(l)}} = \mathbf{0}. \tag{31}$$

When $p = q$, the expression would be

$$\frac{\partial \boldsymbol{V}_3^{(j)}}{\partial \boldsymbol{P}_1^{(p)}} = \boldsymbol{W}_2^* \mathrm{diag}\left(\boldsymbol{Q}^\top \boldsymbol{P}_1^{(p)}\right) \boldsymbol{Q} + \boldsymbol{W}_2^* \mathrm{diag}\left(\boldsymbol{Q}\boldsymbol{P}_1^{(p)}\right) \boldsymbol{Q}^\top; \quad \forall l \neq p : \frac{\partial \boldsymbol{V}_3^{(j)}}{\partial \boldsymbol{P}_1^{(l)}} = \mathbf{0}. \tag{32}$$

Now we move on to compute $(\boldsymbol{V}_3^{(j)} - \boldsymbol{V}_3^{*(j)})$. There are in total four cases to consider: depending on whether $\boldsymbol{P}_1^{(k)}$ is being used when computing $\tilde{\boldsymbol{V}}_2^{(j)}$ and $\tilde{\boldsymbol{V}}_2^{(j+1)}$. Let us go over these cases one-by-one.

1. **Case 1:** $\tilde{\boldsymbol{V}}_1^{(j)} \neq \bar{\boldsymbol{e}}_k$ and $\tilde{\boldsymbol{V}}_1^{(j+1)} \neq \bar{\boldsymbol{e}}_k$.

   Assume $\tilde{\boldsymbol{V}}_1^{(j)} = \bar{\boldsymbol{e}}_p$ and $\tilde{\boldsymbol{V}}_1^{(j+1)} = \bar{\boldsymbol{e}}_q$ for some $p, q \neq k$. Since $\boldsymbol{V}_2^{(j)} = \boldsymbol{P}_1 \tilde{\boldsymbol{V}}_1^{(j)}$ while $\boldsymbol{P}_1$ equals $\boldsymbol{P}_1^*$ for all columns not equal to k by assumption, we have $\boldsymbol{V}_2^{(j)} = \boldsymbol{P}_1 \bar{\boldsymbol{e}}_p = \boldsymbol{P}_1^{(p)} = \boldsymbol{P}_1^{*(p)} = \boldsymbol{P}_1^* \bar{\boldsymbol{e}}_p = \boldsymbol{V}_2^{*(j)}$. With an identical argument we have $\boldsymbol{V}_2^{(j+1)} = \boldsymbol{V}_2^{*(j+1)}$. Thus in this case $\boldsymbol{V}_3^{(j)} = \boldsymbol{V}_3^{*(j)}$ and hence

   $$\left(\frac{\partial \boldsymbol{V}_3^{(j)}}{\partial \boldsymbol{P}_1}\right)^\top \left(\boldsymbol{V}_3^{(j)} - \boldsymbol{V}_3^{*(j)}\right) = \boldsymbol{0}. \tag{33}$$

2. **Case 2:** $\tilde{\boldsymbol{V}}_1^{(j)} = \bar{\boldsymbol{e}}_k$ and $\tilde{\boldsymbol{V}}_1^{(j+1)} \neq \bar{\boldsymbol{e}}_k$.

   Assume $\tilde{\boldsymbol{V}}_1^{(j+1)} = \bar{\boldsymbol{e}}_q$ for some $q \neq k$, from case 1 we know $\boldsymbol{V}_2^{(j+1)} = \boldsymbol{V}_2^{*(j+1)}$. However since $\tilde{\boldsymbol{V}}_1^{(j)} = \bar{\boldsymbol{e}}_k$ we have $\boldsymbol{V}_2^{(j)} = \boldsymbol{P}_1 \bar{\boldsymbol{e}}_k = \boldsymbol{P}_1^{(k)}$, which is not equal to $\boldsymbol{V}_2^{*(j)} = \boldsymbol{P}_1^{*(k)}$. Therefore

   $$\begin{aligned}
   \boldsymbol{V}_3^{(j)} - \boldsymbol{V}_3^{*(j)} &= \boldsymbol{W}_2^* \left(\boldsymbol{Q}\boldsymbol{P}_1^{(k)} \odot \boldsymbol{Q}^\top \boldsymbol{P}_1^{(q)}\right) - \boldsymbol{W}_2^* \left(\boldsymbol{Q}\boldsymbol{P}_1^{*(k)} \odot \boldsymbol{Q}^\top \boldsymbol{P}_1^{(q)}\right) \\
   &= \boldsymbol{W}_2^* \left(\boldsymbol{Q}^\top \boldsymbol{P}_1^{(q)} \odot \boldsymbol{Q}\left(\boldsymbol{P}_1^{(k)} - \boldsymbol{P}_1^{*(k)}\right)\right) \\
   &= \boldsymbol{W}_2^* \mathrm{diag}\left(\boldsymbol{Q}^\top \boldsymbol{P}_1^{(q)}\right) \boldsymbol{Q}\left(\boldsymbol{P}_1^{(k)} - \boldsymbol{P}_1^{*(k)}\right).
   \end{aligned} \tag{34}$$

   For the gradient with respect to $\boldsymbol{P}_1^{(k)}$, combining Equation (34) with Equation (31) (substituting $p = k$) we have

   $$\begin{aligned}
   \left(\frac{\partial \boldsymbol{V}_3^{(j)}}{\partial \boldsymbol{P}_1^{(k)}}\right)^\top \left(\boldsymbol{V}_3^{(j)} - \boldsymbol{V}_3^{*(j)}\right) &= \left(\boldsymbol{W}_2^* \mathrm{diag}\left(\boldsymbol{Q}^\top \boldsymbol{P}_1^{(q)}\right) \boldsymbol{Q}\right)^\top \left(\boldsymbol{W}_2^* \mathrm{diag}\left(\boldsymbol{Q}^\top \boldsymbol{P}_1^{(q)}\right) \boldsymbol{Q}\left(\boldsymbol{P}_1^{(k)} - \boldsymbol{P}_1^{*(k)}\right)\right) \\
   &= \boldsymbol{Q}^\top \mathrm{diag}\left(\boldsymbol{Q}^\top \boldsymbol{P}_1^{(q)}\right) \boldsymbol{W}_2^{*\top} \boldsymbol{W}_2^* \mathrm{diag}\left(\boldsymbol{Q}^\top \boldsymbol{P}_1^{(q)}\right) \boldsymbol{Q}\left(\boldsymbol{P}_1^{(k)} - \boldsymbol{P}_1^{*(k)}\right) \\
   &= \boldsymbol{Q}^\top \mathrm{diag}\left(\boldsymbol{Q}^\top \boldsymbol{P}_1^{(q)}\right) \mathrm{diag}\left(\boldsymbol{Q}^\top \boldsymbol{P}_1^{(q)}\right) \boldsymbol{Q}\left(\boldsymbol{P}_1^{(k)} - \boldsymbol{P}_1^{*(k)}\right) \\
   &= \left((\boldsymbol{1}_n \boldsymbol{1}_n^\top) \otimes I_n\right)\left(\boldsymbol{P}_1^{(k)} - \boldsymbol{P}_1^{*(k)}\right) \qquad \text{(by Lemma G.25)}
   \end{aligned} \tag{35}$$

   where we dropped $\boldsymbol{W}_2^{*\top}\boldsymbol{W}_2^*$ in the third step since $\boldsymbol{W}_2^*$ is a permutation matrix and $\boldsymbol{W}_2^{*\top}\boldsymbol{W}_2^* = I_n$.

   Now plugging in $\boldsymbol{P}_1^{(k)} = \boldsymbol{e}_a \otimes \boldsymbol{e}_b$ and $\boldsymbol{P}_1^{*(k)} = \boldsymbol{e}_{a^*} \otimes \boldsymbol{e}_{b^*}$ we have

   $$\begin{aligned}
   \left(\frac{\partial \boldsymbol{V}_3^{(j)}}{\partial \boldsymbol{P}_1^{(k)}}\right)^\top \left(\boldsymbol{V}_3^{(j)} - \boldsymbol{V}_3^{*(j)}\right) &= \left((\boldsymbol{1}_n \boldsymbol{1}_n^\top) \otimes I_n\right)(\boldsymbol{e}_a \otimes \boldsymbol{e}_b) - \left((\boldsymbol{1}_n \boldsymbol{1}_n^\top) \otimes I_n\right)(\boldsymbol{e}_{a^*} \otimes \boldsymbol{e}_{b^*}) \\
   &= \left((\boldsymbol{1}_n \boldsymbol{1}_n^\top \boldsymbol{e}_a) \otimes I_n \boldsymbol{e}_b\right) - \left((\boldsymbol{1}_n \boldsymbol{1}_n^\top \boldsymbol{e}_{a^*}) \otimes I_n \boldsymbol{e}_{b^*}\right) \\
   &= \boldsymbol{1}_n \otimes \boldsymbol{e}_b - \boldsymbol{1}_n \otimes \boldsymbol{e}_{b^*} \\
   &= \boldsymbol{1}_n \otimes (\boldsymbol{e}_b - \boldsymbol{e}_{b^*}).
   \end{aligned} \tag{36}$$

3. **Case 3:** $\tilde{\boldsymbol{V}}_1^{(j)} \neq \bar{\boldsymbol{e}}_k$ and $\tilde{\boldsymbol{V}}_1^{(j+1)} = \bar{\boldsymbol{e}}_k$.

   This is a symmetric case with respect to case 2. Assume $\tilde{\boldsymbol{V}}_1^{(j)} = \bar{\boldsymbol{e}}_p$ for some $p \neq k$, from case 1 we know $\boldsymbol{V}_2^{(j)} = \boldsymbol{V}_2^{*(j)}$. However since $\tilde{\boldsymbol{V}}_1^{(j)} = \bar{\boldsymbol{e}}_k$ we have $\boldsymbol{V}_2^{(j+1)} = \boldsymbol{P}_1 \bar{\boldsymbol{e}}_k =$

$\boldsymbol{P}_1^{(k)}$, which is not equal to $\boldsymbol{V}_2^{*(j+1)} = \boldsymbol{P}_1^{*(k)}$. Therefore similar to case 2 we have

$$\boldsymbol{V}_3^{(j)} - \boldsymbol{V}_3^{*(j)} = \boldsymbol{W}_2^* \text{diag}\left(\boldsymbol{Q}\boldsymbol{P}_1^{(p)}\right)\boldsymbol{Q}^\top\left(\boldsymbol{P}_1^{(k)} - \boldsymbol{P}_1^{*(k)}\right). \tag{37}$$

Combining Equation (37) with Equation (32) gives

$$\begin{aligned}
\left(\frac{\partial \boldsymbol{V}_3^{(j)}}{\partial \boldsymbol{P}_1^{(k)}}\right)^\top\left(\boldsymbol{V}_3^{(j)} - \boldsymbol{V}_3^{*(j)}\right) &= \left(\boldsymbol{W}_2^* \text{diag}\left(\boldsymbol{Q}\boldsymbol{P}_1^{(p)}\right)\boldsymbol{Q}^\top\right)^\top\left(\boldsymbol{W}_2^* \text{diag}\left(\boldsymbol{Q}\boldsymbol{P}_1^{(p)}\right)\boldsymbol{Q}^\top\left(\boldsymbol{P}_1^{(k)} - \boldsymbol{P}_1^{*(k)}\right)\right) \\
&= \boldsymbol{Q}\text{diag}\left(\boldsymbol{Q}\boldsymbol{P}_1^{(p)}\right)\boldsymbol{W}_2^{*\top}\boldsymbol{W}_2^* \text{diag}\left(\boldsymbol{Q}\boldsymbol{P}_1^{(p)}\right)\boldsymbol{Q}^\top\left(\boldsymbol{P}_1^{(k)} - \boldsymbol{P}_1^{*(k)}\right) \\
&= \boldsymbol{Q}\text{diag}\left(\boldsymbol{Q}\boldsymbol{P}_1^{(p)}\right)\text{diag}\left(\boldsymbol{Q}\boldsymbol{P}_1^{(p)}\right)\boldsymbol{Q}^\top\left(\boldsymbol{P}_1^{(k)} - \boldsymbol{P}_1^{*(k)}\right) \\
&= \left(I_n \otimes (\mathbf{1}_n\mathbf{1}_n^\top)\right)\left(\boldsymbol{P}_1^{(k)} - \boldsymbol{P}_1^{*(k)}\right) \qquad \text{(by Lemma G.26)}
\end{aligned} \tag{38}$$

Now plugging in $\boldsymbol{P}_1^{(k)} = \boldsymbol{e}_a \otimes \boldsymbol{e}_b$ and $\boldsymbol{P}_1^{*(k)} = \boldsymbol{e}_{a^*} \otimes \boldsymbol{e}_{b^*}$ we have

$$\begin{aligned}
\left(\frac{\partial \boldsymbol{V}_3^{(j)}}{\partial \boldsymbol{P}_1^{(k)}}\right)^\top\left(\boldsymbol{V}_3^{(j)} - \boldsymbol{V}_3^{*(j)}\right) &= \left(I_n \otimes (\mathbf{1}_n\mathbf{1}_n^\top)\right)(\boldsymbol{e}_a \otimes \boldsymbol{e}_b) - \left(I_n \otimes (\mathbf{1}_n\mathbf{1}_n^\top)\right)(\boldsymbol{e}_{a^*} \otimes \boldsymbol{e}_{b^*}) \\
&= \left(I_n\boldsymbol{e}_a \otimes (\mathbf{1}_n\mathbf{1}_n^\top\boldsymbol{e}_b)\right) - \left(I_n\boldsymbol{e}_{a^*} \otimes (\mathbf{1}_n\mathbf{1}_n^\top\boldsymbol{e}_{b^*})\right) \\
&= \boldsymbol{e}_a \otimes \mathbf{1}_n - \boldsymbol{e}_{a^*} \otimes \mathbf{1}_n \\
&= (\boldsymbol{e}_a - \boldsymbol{e}_{a^*}) \otimes \mathbf{1}_n.
\end{aligned} \tag{39}$$

4. **Case 4:** $\tilde{\boldsymbol{V}}_1^{(j)} = \tilde{\boldsymbol{V}}_1^{(j+1)} = \bar{\boldsymbol{e}}_k$.

   This is the most complicated case since the loss is contributed by two different paths. We can first decompose the negative residual as

$$\begin{aligned}
\boldsymbol{V}_3^{(j)} - \boldsymbol{V}_3^{*(j)} &= \boldsymbol{W}_2^*\left(\boldsymbol{Q}\boldsymbol{P}_1^{(k)} \odot \boldsymbol{Q}^\top\boldsymbol{P}_1^{(k)}\right) - \boldsymbol{W}_2^*\left(\boldsymbol{Q}\boldsymbol{P}_1^{*(k)} \odot \boldsymbol{Q}^\top\boldsymbol{P}_1^{*(k)}\right) \\
&= \boldsymbol{W}_2^*\left(\boldsymbol{Q}\boldsymbol{P}_1^{(k)} \odot \boldsymbol{Q}^\top\boldsymbol{P}_1^{(k)}\right) - \boldsymbol{W}_2^*\left(\boldsymbol{Q}\boldsymbol{P}_1^{(k)} \odot \boldsymbol{Q}^\top\boldsymbol{P}_1^{*(k)}\right) \\
&\quad + \boldsymbol{W}_2^*\left(\boldsymbol{Q}\boldsymbol{P}_1^{(k)} \odot \boldsymbol{Q}^\top\boldsymbol{P}_1^{*(k)}\right) - \boldsymbol{W}_2^*\left(\boldsymbol{Q}\boldsymbol{P}_1^{*(k)} \odot \boldsymbol{Q}^\top\boldsymbol{P}_1^{*(k)}\right) \\
&= \boldsymbol{W}_2^* \text{diag}\left(\boldsymbol{Q}\boldsymbol{P}_1^{(k)}\right)\boldsymbol{Q}^\top\left(\boldsymbol{P}_1^{(k)} - \boldsymbol{P}_1^{*(k)}\right) + \boldsymbol{W}_2^* \text{diag}\left(\boldsymbol{Q}^\top\boldsymbol{P}_1^{*(p)}\right)\boldsymbol{Q}\left(\boldsymbol{P}_1^{(k)} - \boldsymbol{P}_1^{*(k)}\right).
\end{aligned} \tag{40}$$

   Combining with Equation (32), we have

$$\begin{aligned}
&\left(\frac{\partial \boldsymbol{V}_3^{(j)}}{\partial \boldsymbol{P}_1}\right)^\top\left(\boldsymbol{V}_3^{(j)} - \boldsymbol{V}_3^{*(j)}\right) \\
&= \left(\boldsymbol{W}_2^* \text{diag}\left(\boldsymbol{Q}^\top\boldsymbol{P}_1^{(k)}\right)\boldsymbol{Q} + \boldsymbol{W}_2^* \text{diag}\left(\boldsymbol{Q}\boldsymbol{P}_1^{(k)}\right)\boldsymbol{Q}^\top\right)^\top \\
&\qquad \left(\boldsymbol{W}_2^* \text{diag}\left(\boldsymbol{Q}\boldsymbol{P}_1^{(k)}\right)\boldsymbol{Q}^\top\left(\boldsymbol{P}_1^{(k)} - \boldsymbol{P}_1^{*(k)}\right) + \boldsymbol{W}_2^* \text{diag}\left(\boldsymbol{Q}^\top\boldsymbol{P}_1^{*(k)}\right)\boldsymbol{Q}\left(\boldsymbol{P}_1^{(k)} - \boldsymbol{P}_1^{*(k)}\right)\right) \\
&= \boldsymbol{Q}^\top \text{diag}\left(\boldsymbol{Q}^\top\boldsymbol{P}_1^{(k)}\right)\text{diag}\left(\boldsymbol{Q}\boldsymbol{P}_1^{(k)}\right)\boldsymbol{Q}^\top\left(\boldsymbol{P}_1^{(k)} - \boldsymbol{P}_1^{*(k)}\right) && \text{(a)} \\
&\quad + \boldsymbol{Q}\text{diag}\left(\boldsymbol{Q}\boldsymbol{P}_1^{(k)}\right)\text{diag}\left(\boldsymbol{Q}\boldsymbol{P}_1^{(k)}\right)\boldsymbol{Q}^\top\left(\boldsymbol{P}_1^{(k)} - \boldsymbol{P}_1^{*(k)}\right) && \text{(b)} \\
&\quad + \boldsymbol{Q}^\top \text{diag}\left(\boldsymbol{Q}^\top\boldsymbol{P}_1^{(k)}\right)\text{diag}\left(\boldsymbol{Q}^\top\boldsymbol{P}_1^{*(k)}\right)\boldsymbol{Q}\left(\boldsymbol{P}_1^{(k)} - \boldsymbol{P}_1^{*(k)}\right) && \text{(c)} \\
&\quad + \boldsymbol{Q}\text{diag}\left(\boldsymbol{Q}\boldsymbol{P}_1^{(k)}\right)\text{diag}\left(\boldsymbol{Q}^\top\boldsymbol{P}_1^{*(k)}\right)\boldsymbol{Q}\left(\boldsymbol{P}_1^{(k)} - \boldsymbol{P}_1^{*(k)}\right). && \text{(d)}
\end{aligned}$$

   Now let us analyze the four cross terms term-by-term.

(a) With $\boldsymbol{P}_1^{(k)} = \boldsymbol{e}_a \otimes \boldsymbol{e}_b$, by Lemma G.24 we have $\boldsymbol{Q}^\top \boldsymbol{P}_1^{(k)} = \mathbf{1}_n \otimes \boldsymbol{e}_a$ and $\boldsymbol{Q}\boldsymbol{P}_1^{(k)} = \boldsymbol{e}_b \otimes \mathbf{1}_n$. Therefore $\boldsymbol{Q}^\top \boldsymbol{P}_1^{(k)} \odot \boldsymbol{Q}\boldsymbol{P}_1^{(k)} = \boldsymbol{e}_b \otimes \boldsymbol{e}_a$ and hence

$$\operatorname{diag}\left(\boldsymbol{Q}^\top \boldsymbol{P}_1^{(k)}\right) \operatorname{diag}\left(\boldsymbol{Q}\boldsymbol{P}_1^{(k)}\right) = \operatorname{diag}\left(\boldsymbol{e}_b \otimes \boldsymbol{e}_a\right) = \operatorname{diag}\left(\boldsymbol{e}_b\right) \otimes \operatorname{diag}\left(\boldsymbol{e}_a\right) \tag{41}$$

It follows that

$$\begin{aligned}
&\boldsymbol{Q}^\top \operatorname{diag}\left(\boldsymbol{Q}^\top \boldsymbol{P}_1^{(k)}\right) \operatorname{diag}\left(\boldsymbol{Q}\boldsymbol{P}_1^{(k)}\right) \boldsymbol{Q}^\top \\
&= \left(\mathbf{1}_n \otimes I_n\right)\left(I_n^\top \otimes \mathbf{1}_n^\top\right)\left(\operatorname{diag}\left(\boldsymbol{e}_b\right) \otimes \operatorname{diag}\left(\boldsymbol{e}_a\right)\right)\left(\mathbf{1}_n \otimes I_n\right)\left(I_n^\top \otimes \mathbf{1}_n^\top\right) \\
&= \left(\mathbf{1}_n \otimes I_n\right)\left(I_n^\top \otimes \mathbf{1}_n^\top\right)\left(\boldsymbol{e}_b \otimes \operatorname{diag}\left(\boldsymbol{e}_a\right)\right)\left(I_n^\top \otimes \mathbf{1}_n^\top\right) \\
&= \left(\mathbf{1}_n \otimes I_n\right)\left(I_n^\top \otimes \mathbf{1}_n^\top\right)\left(\boldsymbol{e}_b \otimes \boldsymbol{e}_a^\top\right) \\
&= \left(\mathbf{1}_n \otimes I_n\right)\left(I_n^\top \otimes \mathbf{1}_n^\top\right)\left(\boldsymbol{e}_b \boldsymbol{e}_a^\top \otimes 1\right) \\
&= \left(\mathbf{1}_n \otimes I_n\right)\left(\boldsymbol{e}_b \boldsymbol{e}_a^\top \otimes \mathbf{1}_n^\top\right)
\end{aligned} \tag{42}$$

Plugging into (a) we have

$$\begin{aligned}
\text{(a)} &= \boldsymbol{Q}^\top \operatorname{diag}\left(\boldsymbol{Q}^\top \boldsymbol{P}_1^{(k)}\right) \operatorname{diag}\left(\boldsymbol{Q}\boldsymbol{P}_1^{(k)}\right) \boldsymbol{Q}^\top \left(\boldsymbol{P}_1^{(k)} - \boldsymbol{P}_1^{*(k)}\right) \\
&= \left(\mathbf{1}_n \otimes I_n\right)\left(\boldsymbol{e}_b \boldsymbol{e}_a^\top \otimes \mathbf{1}_n^\top\right)\left(\boldsymbol{e}_a \otimes \boldsymbol{e}_b\right) - \left(\mathbf{1}_n \otimes I_n\right)\left(\boldsymbol{e}_b \boldsymbol{e}_a^\top \otimes \mathbf{1}_n^\top\right)\left(\boldsymbol{e}_{a^*} \otimes \boldsymbol{e}_{b^*}\right) \\
&= \left(\mathbf{1}_n \otimes I_n\right)\left(\boldsymbol{e}_b \boldsymbol{e}_a^\top \boldsymbol{e}_a \otimes 1\right) - \left(\mathbf{1}_n \otimes I_n\right)\left(\boldsymbol{e}_b \boldsymbol{e}_a^\top \boldsymbol{e}_{a^*} \otimes 1\right) \\
&= \left(\mathbf{1}_n \otimes I_n\right)\left(1 \otimes \boldsymbol{e}_b \boldsymbol{e}_a^\top \boldsymbol{e}_a\right) - \left(\mathbf{1}_n \otimes I_n\right)\left(1 \otimes \boldsymbol{e}_b \boldsymbol{e}_a^\top \boldsymbol{e}_{a^*}\right) \\
&= \begin{cases} \mathbf{1}_n \otimes \boldsymbol{e}_b & \text{when } a \neq a^* \\ \mathbf{0} & \text{otherwise} \end{cases}
\end{aligned} \tag{43}$$

(b) We have seen the same term as in case 3, by Lemma G.26 we have

$$\text{(b)} = \boldsymbol{Q}\operatorname{diag}\left(\boldsymbol{Q}\boldsymbol{P}_1^{(k)}\right) \operatorname{diag}\left(\boldsymbol{Q}\boldsymbol{P}_1^{(k)}\right) \boldsymbol{Q}^\top \left(\boldsymbol{P}_1^{(k)} - \boldsymbol{P}_1^{*(k)}\right) = \left(\boldsymbol{e}_a - \boldsymbol{e}_{a^*}\right) \otimes \mathbf{1}_n. \tag{44}$$

(c) With $\boldsymbol{P}_1^{(k)} = \boldsymbol{e}_a \otimes \boldsymbol{e}_b$ and $\boldsymbol{P}_1^{*(k)} = \boldsymbol{e}_{a^*} \otimes \boldsymbol{e}_{b^*}$, we have

$$\operatorname{diag}\left(\boldsymbol{Q}^\top \boldsymbol{P}_1^{(k)}\right) \operatorname{diag}\left(\boldsymbol{Q}^\top \boldsymbol{P}_1^{*(k)}\right) = \operatorname{diag}\left(\mathbf{1}_n \otimes \boldsymbol{e}_a\right) \operatorname{diag}\left(\mathbf{1}_n \otimes \boldsymbol{e}_{a^*}\right) = \begin{cases} \operatorname{diag}\left(\mathbf{1}_n \otimes \boldsymbol{e}_a\right) & \text{if } a = a^* \\ \mathbf{0} & \text{otherwise} \end{cases} \tag{45}$$

Thus by Lemma G.25 we have

$$\boldsymbol{Q}^\top \operatorname{diag}\left(\boldsymbol{Q}^\top \boldsymbol{P}_1^{(k)}\right) \operatorname{diag}\left(\boldsymbol{Q}^\top \boldsymbol{P}_1^{*(k)}\right) \boldsymbol{Q} = \begin{cases} \left(\mathbf{1}_n \mathbf{1}_n^\top\right) \otimes I_n & \text{if } a = a^* \\ \mathbf{0} & \text{otherwise} \end{cases} \tag{46}$$

When $a = a^*$, we then have

$$\begin{aligned}
&\boldsymbol{Q}^\top \operatorname{diag}\left(\boldsymbol{Q}^\top \boldsymbol{P}_1^{(k)}\right) \operatorname{diag}\left(\boldsymbol{Q}^\top \boldsymbol{P}_1^{*(k)}\right) \boldsymbol{Q} \left(\boldsymbol{P}_1^{(k)} - \boldsymbol{P}_1^{*(k)}\right) \\
&= \left(\left(\mathbf{1}_n \mathbf{1}_n^\top\right) \otimes I_n\right)\left(\boldsymbol{e}_a \otimes \boldsymbol{e}_b\right) - \left(\left(\mathbf{1}_n \mathbf{1}_n^\top\right) \otimes I_n\right)\left(\boldsymbol{e}_{a^*} \otimes \boldsymbol{e}_{b^*}\right) \\
&= \mathbf{1}_n \otimes \left(\boldsymbol{e}_b - \boldsymbol{e}_{b^*}\right).
\end{aligned} \tag{47}$$

Thus in summary

$$\text{(c)} = \boldsymbol{Q}^\top \operatorname{diag}\left(\boldsymbol{Q}^\top \boldsymbol{P}_1^{(k)}\right) \operatorname{diag}\left(\boldsymbol{Q}^\top \boldsymbol{P}_1^{*(k)}\right) \boldsymbol{Q} \left(\boldsymbol{P}_1^{(k)} - \boldsymbol{P}_1^{*(k)}\right) = \begin{cases} \mathbf{1}_n \otimes \left(\boldsymbol{e}_b - \boldsymbol{e}_{b^*}\right) & \text{if } a = a^* \\ \mathbf{0} & \text{otherwise.} \end{cases} \tag{48}$$

(d) Now for the last term, note that

$$\begin{aligned}
&\boldsymbol{Q}\text{diag}\left(\boldsymbol{Q}\boldsymbol{P}_1^{(k)}\right)\text{diag}\left(\boldsymbol{Q}^\top\boldsymbol{P}_1^{*(k)}\right)\boldsymbol{Q}\\
&=\boldsymbol{Q}\text{diag}\left(\boldsymbol{e}_b\otimes\mathbf{1}_n\right)\text{diag}\left(\mathbf{1}_n\otimes\boldsymbol{e}_{a^*}\right)\boldsymbol{Q}\\
&=\boldsymbol{Q}\text{diag}\left(\boldsymbol{e}_b\otimes\boldsymbol{e}_{a^*}\right)\boldsymbol{Q}\\
&=\left(I_n\otimes\mathbf{1}_n\right)\left(\mathbf{1}_n^\top\otimes I_n^\top\right)\left(\text{diag}\left(\boldsymbol{e}_b\right)\otimes\text{diag}\left(\boldsymbol{e}_{a^*}\right)\right)\left(I_n\otimes\mathbf{1}_n\right)\left(\mathbf{1}_n^\top\otimes I_n^\top\right)\\
&=\left(I_n\otimes\mathbf{1}_n\right)\left(\mathbf{1}_n^\top\otimes I_n^\top\right)\left(\text{diag}\left(\boldsymbol{e}_b\right)\otimes\boldsymbol{e}_{a^*}\right)\left(\mathbf{1}_n^\top\otimes I_n^\top\right)\\
&=\left(I_n\otimes\mathbf{1}_n\right)\left(\boldsymbol{e}_b^\top\otimes\boldsymbol{e}_{a^*}\right)\left(\mathbf{1}_n^\top\otimes I_n^\top\right)\\
&=\left(I_n\otimes\mathbf{1}_n\right)\left(\boldsymbol{e}_{a^*}\boldsymbol{e}_b^\top\otimes 1\right)\left(\mathbf{1}_n^\top\otimes I_n^\top\right)\\
&=\left(\boldsymbol{e}_{a^*}\boldsymbol{e}_b^\top\otimes\mathbf{1}_n\right)\left(\mathbf{1}_n^\top\otimes I_n^\top\right).
\end{aligned}\tag{49}$$

Plugging in $(\boldsymbol{P}_1^{(k)}-\boldsymbol{P}_1^{*(k)})$, we have that

$$\begin{aligned}
&\boldsymbol{Q}\text{diag}\left(\boldsymbol{Q}\boldsymbol{P}_1^{(k)}\right)\text{diag}\left(\boldsymbol{Q}^\top\boldsymbol{P}_1^{*(k)}\right)\boldsymbol{Q}\left(\boldsymbol{P}_1^{(k)}-\boldsymbol{P}_1^{*(k)}\right)\\
=&\left(\boldsymbol{e}_{a^*}\boldsymbol{e}_b^\top\otimes\mathbf{1}_n\right)\left(\mathbf{1}_n^\top\otimes I_n^\top\right)\left(\boldsymbol{e}_a\otimes\boldsymbol{e}_b\right)-\left(\boldsymbol{e}_{a^*}\boldsymbol{e}_b^\top\otimes\mathbf{1}_n\right)\left(\mathbf{1}_n^\top\otimes I_n^\top\right)\left(\boldsymbol{e}_{a^*}\otimes\boldsymbol{e}_{b^*}\right)\\
=&\left(\boldsymbol{e}_{a^*}\boldsymbol{e}_b^\top\otimes\mathbf{1}_n\right)\left(1\otimes\boldsymbol{e}_b\right)-\left(\boldsymbol{e}_{a^*}\boldsymbol{e}_b^\top\otimes\mathbf{1}_n\right)\left(1\otimes\boldsymbol{e}_{b^*}\right)\\
=&\left(\boldsymbol{e}_{a^*}\boldsymbol{e}_b^\top\otimes\mathbf{1}_n\right)\left(\boldsymbol{e}_b\otimes 1\right)-\left(\boldsymbol{e}_{a^*}\boldsymbol{e}_b^\top\otimes\mathbf{1}_n\right)\left(\boldsymbol{e}_{b^*}\otimes 1\right)\\
=&\left(\boldsymbol{e}_{a^*}\boldsymbol{e}_b^\top\boldsymbol{e}_b\otimes\mathbf{1}_n\right)-\left(\boldsymbol{e}_{a^*}\boldsymbol{e}_b^\top\boldsymbol{e}_{b^*}\otimes\mathbf{1}_n\right)\\
=&\begin{cases}\boldsymbol{e}_{a^*}\otimes\mathbf{1}_n & \text{if } b\neq b^*\\ \mathbf{0} & \text{otherwise.}\end{cases}
\end{aligned}\tag{50}$$

Summing the four terms together, we then have

$$\left(\frac{\partial\boldsymbol{V}_3^{(j)}}{\partial\boldsymbol{P}_1}\right)^\top\left(\boldsymbol{V}_3^{(j)}-\boldsymbol{V}_3^{*(j)}\right)=\begin{cases}0 & \text{when } a^*=a, b^*=b\\ \mathbf{1}_n\otimes\left(\boldsymbol{e}_b-\boldsymbol{e}_{b^*}\right)+\boldsymbol{e}_{a^*}\otimes\mathbf{1}_n & \text{when } a^*=a, b^*\neq b\\ \mathbf{1}_n\otimes\boldsymbol{e}_{b^*}+\left(\boldsymbol{e}_a-\boldsymbol{e}_{a^*}\right)\otimes\mathbf{1}_n & \text{when } a^*\neq a, b^*=b\\ \mathbf{1}_n\otimes\boldsymbol{e}_b+\boldsymbol{e}_a\otimes\mathbf{1}_n & \text{when } a^*\neq a, b^*\neq b.\end{cases}\tag{51}$$

Now we are ready to provide the gradient expression for the loss over the entire sequence. Observe that for every consecutive sequence of m columns $\{\boldsymbol{V}_1^{*(j)},\boldsymbol{V}_1^{*(j+1)},\ldots,\boldsymbol{V}_1^{*(j+m-1)}\}$ that all equals to $\bar{\boldsymbol{e}}_k$, it will result in one incorrect column $\boldsymbol{V}_3^{(j-1)}$ in case 2, one incorrect column $\boldsymbol{V}_3^{(j+m-1)}$ in case 3, and $m-1$ incorrect columns $(\boldsymbol{V}_3^{(j)},\ldots,\boldsymbol{V}_3^{(j+m-2)})$ in case 4.

For illustration, one can refer to the computation graph in Figure 10. In the graph, green entries agrees with the counterfactual values with correct $\boldsymbol{P}_1^{*(k)}$, Red and pink entries are incorrect entries where red entries are consequence solely dependent on $\boldsymbol{P}_1^{(k)}$ (case 4) and pink entries depend on other correct columns (case 2,3).

Assume that in total there are α columns in $\boldsymbol{V}_3$ under case 2, α columns in $\boldsymbol{V}_3$ under case 3, and β columns in $\boldsymbol{V}_3$ under case 4, then by Equation (27) the total gradient can be expressed as follows:

- When $a=a^*, b\neq b^*$:

$$\begin{aligned}
\frac{\partial\mathcal{L}}{\partial\boldsymbol{P}_1^{(k)}}&=2\alpha\left(\mathbf{1}_n\otimes\left(\boldsymbol{e}_b-\boldsymbol{e}_{b^*}\right)+2\alpha\left(\boldsymbol{e}_a-\boldsymbol{e}_{a^*}\right)\otimes\mathbf{1}_n\right)+2\beta\left(\mathbf{1}_n\otimes\left(\boldsymbol{e}_b-\boldsymbol{e}_{b^*}\right)+\boldsymbol{e}_{a^*}\otimes\mathbf{1}_n\right)\\
&=\left(2\alpha+2\beta\right)\left(\mathbf{1}_n\otimes\boldsymbol{e}_b\right)-\left(2\alpha+2\beta\right)\left(\mathbf{1}_n\otimes\boldsymbol{e}_{b^*}\right)+2\beta\left(\boldsymbol{e}_{a^*}\otimes\mathbf{1}_n\right).
\end{aligned}\tag{52}$$

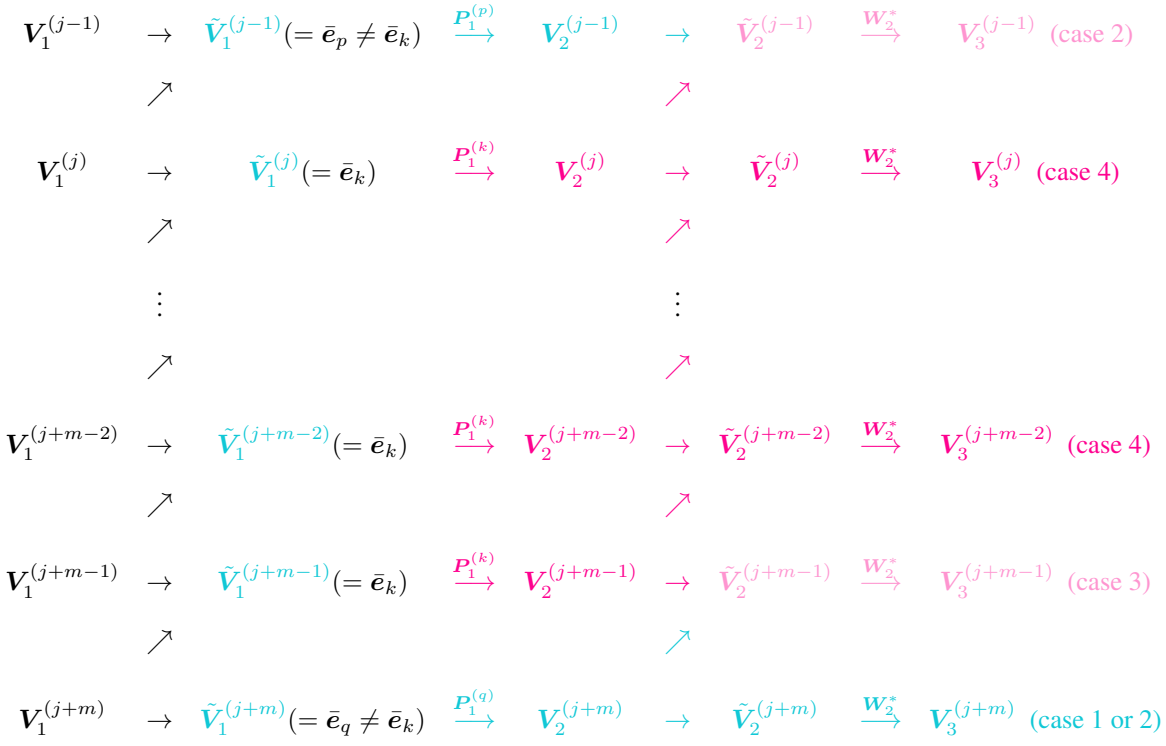


Figure 10: Error propagation of $\boldsymbol{P}_1^{(k)}$

- When $a \neq a^*, b = b^*$:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \boldsymbol{P}_1^{(k)}} &= 2\alpha\left(\mathbf{1}_n \otimes (\boldsymbol{e}_b - \boldsymbol{e}_{b^*})\right) + 2\alpha\left((\boldsymbol{e}_a - \boldsymbol{e}_{a^*}) \otimes \mathbf{1}_n\right) + 2\beta\left(\mathbf{1}_n \otimes \boldsymbol{e}_{b^*} + (\boldsymbol{e}_a - \boldsymbol{e}_{a^*}) \otimes \mathbf{1}_n\right) \\ &= (2\alpha + 2\beta)\left(\boldsymbol{e}_a \otimes \mathbf{1}_n\right) - (2\alpha + 2\beta)\left(\boldsymbol{e}_{a^*} \otimes \mathbf{1}_n\right) + 2\beta\left(\mathbf{1}_n \otimes \boldsymbol{e}_{b^*}\right).\end{aligned} \tag{53}$$

- When $a \neq a^*, b \neq b^*$:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \boldsymbol{P}_1^{(k)}} &= 2\alpha\left(\mathbf{1}_n \otimes (\boldsymbol{e}_b - \boldsymbol{e}_{b^*})\right) + 2\alpha\left((\boldsymbol{e}_a - \boldsymbol{e}_{a^*}) \otimes \mathbf{1}_n\right) + 2\beta\left(\mathbf{1}_n \otimes \boldsymbol{e}_b + \boldsymbol{e}_a \otimes \mathbf{1}_n\right) \\ &= (2\alpha + 2\beta)\left(\boldsymbol{e}_a \otimes \mathbf{1}_n\right) + (2\alpha + 2\beta)\left(\mathbf{1}_n \otimes \boldsymbol{e}_b\right) - 2\alpha\left(\boldsymbol{e}_{a^*} \otimes \mathbf{1}_n\right) - 2\alpha\left(\mathbf{1}_n \otimes \boldsymbol{e}_{b^*}\right).\end{aligned} \tag{54}$$

This gives the desired expression of gradient. □

Now with the gradient expression, we are ready to prove for the learning of a single missing column.

Lemma G.19 (Learning Column of $\boldsymbol{W}_1$)**.**
Fix an input sequence $\boldsymbol{V}_1 \in \mathbb{R}^{n^2 \times L}$ and any column index $k \in [n^2]$, if $\texttt{HardMax}\left(\boldsymbol{C}_2 + \boldsymbol{W}_2\right) = \boldsymbol{W}_2^*$ and $\texttt{HardMax}\left(\boldsymbol{C}_{1(k)} + \boldsymbol{W}_1\right)$ equals to $\boldsymbol{W}_1^*$ everywhere except for the k-th column and if there exists a non-empty subset of indices $\mathcal{J} \subset [L]$ such that $\boldsymbol{V}_1^{*(j)} = \bar{\boldsymbol{e}}_k$ for all $j \in \mathcal{J}$, for any initialization $\boldsymbol{W}_{1(0)}^{(k)} \in \mathbb{R}^{n^2}$ such that $\|\boldsymbol{W}_{1(0)}^{(k)}\|_0 \leq \frac{1}{2}$. Taking two surrogate gradient updates on $\boldsymbol{W}_{1(0)}^{(k)}$ as described in Algorithm 2 gives $\boldsymbol{W}_{1(2)}^{(k)}$ such that $\texttt{HardMax}(\boldsymbol{W}_{1(2)}^{(k)}) = \boldsymbol{W}_1^{*(k)}$.

Proof of Lemma G.19.

Without loss of generality, assume at the initialization $\boldsymbol{P}_{1(0)}^{(k)} = \boldsymbol{e}_{a_{(0)}} \otimes \boldsymbol{e}_{b_{(0)}}$ and $\boldsymbol{P}_1^{*(k)} = \boldsymbol{e}_{a^*} \otimes \boldsymbol{e}_{b^*}$ for some $a_{(0)}, b_{(0)}, a^*, b^* \in [n]$. We first note that the conditions specified in the lemma meets the assumptions required by Lemma G.18, namely there is only one incorrect column in $\boldsymbol{W}_1$ missing and that column is being used in the forward pass at least one time (since $\mathcal{J}$ is non-empty).

To prove $\texttt{HardMax}(\boldsymbol{W}_{1(2)}^{(k)}) = \boldsymbol{W}_1^{*(k)}$, it is sufficient to show that $\arg\max(\boldsymbol{W}_{1(2)}^{(k)}) = a^*n + b^*$.

Now we will leverage the gradient expressions in Lemma G.18. We will dive into three different cases:

- **When** $a_{(0)} \neq a^*$ **and** $b_{(0)} \neq b^*$.

  By Lemma G.18 we have for some $\alpha \in \mathbb{Z}_+$ and $\beta \in \mathbb{N}$ that

  $$\frac{\partial \mathcal{L}}{\partial \boldsymbol{P}_{1(0)}^{(k)}} = (2\alpha + 2\beta)\left(\boldsymbol{e}_{a_{(0)}} \otimes \boldsymbol{1}_n\right) + (2\alpha + 2\beta)\left(\boldsymbol{1}_n \otimes \boldsymbol{e}_{b_{(0)}}\right) - 2\alpha\left(\boldsymbol{e}_{a^*} \otimes \boldsymbol{1}_n\right) - 2\alpha\left(\boldsymbol{1}_n \otimes \boldsymbol{e}_{b^*}\right). \tag{55}$$

  It is not hard to verify that the unique smallest entry is of index $a^*n + b^*$ with value -4α. This entry is contributed by the intersection of $-2\alpha\left(\boldsymbol{e}_{a^*} \otimes \boldsymbol{1}_n\right) - 2\alpha\left(\boldsymbol{1}_n \otimes \boldsymbol{e}_{b^*}\right)$, the remaining smaller entries are of value -2α contributed by non-intersecting entries in the same expression above.

  Thus we know $\arg\min \partial\mathcal{L}/\partial \boldsymbol{P}_{1(0)}^{(k)} = a^*n + b^*$ with a margin of at least -2 (since $\alpha \geq 1$).

  Now we can apply the gradient step to the weight initialization. Since $\|\boldsymbol{W}_{1(0)}^{(k)}\|_0 \leq \frac{1}{2}$, the margin of $a^*n + b^*$ dominates the largest margin in the initialization (which is 1), we have

  $$\arg\max\left(\boldsymbol{W}_{1(1)}^{(k)}\right) = \arg\max\left(\boldsymbol{W}_{1(0)}^{(k)} - \frac{\partial \mathcal{L}}{\partial \boldsymbol{P}_{1(0)}^{(k)}}\right) = a^*n + b^*. \tag{56}$$

  Therefore $\boldsymbol{P}_{1(1)}^{(k)} = \boldsymbol{P}_1^{*(k)}$ with the first step. We will reach zero loss after the first step and hence the second step is static, so we have shown $\arg\max(\boldsymbol{W}_{1(2)}^{(k)}) = a^*n + b^*$ as desired.

- **When** $a_{(0)} = a^*$ **and** $b_{(0)} \neq b^*$.

  By Lemma G.18 we have for some $\alpha \in \mathbb{Z}_+$ and $\beta \in \mathbb{N}$ that

  $$\frac{\partial \mathcal{L}}{\partial \boldsymbol{P}_{1(0)}^{(k)}} = (2\alpha + 2\beta)\left(\boldsymbol{1}_n \otimes \boldsymbol{e}_{b_{(0)}}\right) - (2\alpha + 2\beta)\left(\boldsymbol{1}_n \otimes \boldsymbol{e}_{b^*}\right) + 2\beta\left(\boldsymbol{e}_{a^*} \otimes \boldsymbol{1}_n\right). \tag{57}$$

  In this case we no longer have an unique smallest entry, the set of negative entries is of index $nx + b^*$ where $x \in [n]$. The values are -2α for the case of $x = a^*$ and $-2(\alpha + \beta)$ for other x's. These negative entries are contributed by the $-(2\alpha + 2\beta)\left(\boldsymbol{1}_n \otimes \boldsymbol{e}_{b^*}\right)$, and all other entries are at least 0.

  Thus we also have a negative margin of at least -2 since $\alpha \geq 1$. Therefore after taking the first gradient step, we know that there exists some $x \in [n]$ such that

  $$\arg\max\left(\boldsymbol{W}_{1(1)}^{(k)}\right) = \arg\max\left(\boldsymbol{W}_{1(0)}^{(k)} - \frac{\partial \mathcal{L}}{\partial \boldsymbol{P}_{1(0)}^{(k)}}\right) = nx + b^*. \tag{58}$$

  Let $P_{1(1)}^{(k)} = \arg\max(\boldsymbol{W}_{1(1)}^{(k)}) = \boldsymbol{e}_{a_{(1)}} \otimes \boldsymbol{e}_{b_{(1)}}$, then now we are in the case of $b_{(1)} = b^*$ since the negative margin of -2 dominates any entry-wise difference in the initialization as $\|\boldsymbol{W}_{1(0)}^{(k)}\|_0 \leq \frac{1}{2}$. If $\beta = 0$ and it happens that $a_{(1)} = a^*$, then we have zero loss after the first

step and we are done as the second step will be static. If $b_{(1)} \neq b^*$, then by Lemma G.18 the second step gradient is of the form

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{P}_{1(1)}^{(k)}} = (2\alpha + 2\beta)\left(\boldsymbol{e}_{a_{(1)}} \otimes \mathbf{1}_n\right) - (2\alpha + 2\beta)\left(\boldsymbol{e}_{a^*} \otimes \mathbf{1}_n\right) + 2\beta\left(\mathbf{1}_n \otimes \boldsymbol{e}_{b^*}\right). \tag{59}$$

This is a bit tricky to analyze directly since we no longer have the small entry-wise difference from the initialization in the weights, but one may note that the sum of the two update steps is of the form

$$\begin{aligned}
&\frac{\partial \mathcal{L}}{\partial \boldsymbol{P}_{1(1)}^{(k)}} + \frac{\partial \mathcal{L}}{\partial \boldsymbol{P}_{1(0)}^{(k)}} \\
&= (2\alpha + 2\beta)\left(\boldsymbol{e}_{a_{(1)}} \otimes \mathbf{1}_n\right) - (2\alpha + 2\beta)\left(\boldsymbol{e}_{a^*} \otimes \mathbf{1}_n\right) + 2\beta\left(\mathbf{1}_n \otimes \boldsymbol{e}_{b^*}\right) \\
&\quad + (2\alpha + 2\beta)\left(\mathbf{1}_n \otimes \boldsymbol{e}_{b_{(0)}}\right) - (2\alpha + 2\beta)\left(\mathbf{1}_n \otimes \boldsymbol{e}_{b^*}\right) + 2\beta\left(\boldsymbol{e}_{a^*} \otimes \mathbf{1}_n\right) \\
&= (2\alpha + 2\beta)\left(\boldsymbol{e}_{a_{(1)}} \otimes \mathbf{1}_n\right) + (2\alpha + 2\beta)\left(\mathbf{1}_n \otimes \boldsymbol{e}_{b_{(0)}}\right) - 2\alpha\left(\boldsymbol{e}_{a^*} \otimes \mathbf{1}_n\right) - 2\alpha\left(\mathbf{1}_n \otimes \boldsymbol{e}_{b^*}\right).
\end{aligned} \tag{60}$$

This is identical to the single step gradient as in Equation (55) in the first case, so follow the identical argument we have

$$\arg\max\left(\boldsymbol{W}_{1(2)}^{(k)}\right) = \arg\max\left(\boldsymbol{W}_{1(0)}^{(k)} - \left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{P}_{1(0)}^{(k)}} + \frac{\partial \mathcal{L}}{\partial \boldsymbol{P}_{1(1)}^{(k)}}\right)\right) = a^* n + b^* \tag{61}$$

as desired.

- **When** $a_{(0)} \neq a^*$ **and** $b_{(0)} = b^*$

  Note that all expressions are symmetric with respect to a and b with an additional swap of the Kronecker products, so we may follow the exact same argument as in the case of $a_{(0)} = a^*$ and $b_{(0)} \neq b^*$ and arrive at the same conclusion.

Thus for any initializations, after two surrogate gradient steps on $\boldsymbol{W}_1^{(k)}$, we have $\texttt{HardMax}(\boldsymbol{W}_{1(2)}^{(k)}) = \boldsymbol{W}_1^{*(k)}$. □

G.3.2 Learning the Second Layer

Now for the second layer, we will similarly first derive the gradient (which is much simpler) and show that with only one gradient step, one can learn the correct entry.

Lemma G.20 (Gradient with respect to incorrect column in $\boldsymbol{P}_2$)**.**

When only the k-th column of translation matrix $\boldsymbol{P}_2^{(k)}$ is not equal to $\boldsymbol{P}_2^{*(k)}$. If $\boldsymbol{P}_2^{(k)}$ is used in the forward pass, there exists $\alpha \in \mathbb{Z}_+$ such that the gradient of $\mathcal{L}$ with respect to $\boldsymbol{P}_2^{(k)}$ is of the form

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{P}_2^{(k)}} = 2\alpha\left(\boldsymbol{P}_2^{(k)} - \boldsymbol{P}_2^{*(k)}\right).$$

Proof. We follow the same set of notations as used in the proof for Lemma G.18. In particular, we use $\tilde{\boldsymbol{V}}_1, \boldsymbol{V}_2, \tilde{\boldsymbol{V}}_2$ and $\boldsymbol{V}_3$ to denote the intermediate sequences attained with translation matrix $\boldsymbol{P}_2^{(k)}$ and $\tilde{\boldsymbol{V}}_1^*, \boldsymbol{V}_2^*, \tilde{\boldsymbol{V}}_2^*$ and $\boldsymbol{V}_3^*$ to denote the counterfactual intermediate sequences should the forward pass is done with the ground truth translations $\boldsymbol{P}_2^{*(k)} = \boldsymbol{W}_1^*$. Since we assume that $\boldsymbol{P}_1 = \boldsymbol{P}_1^*$, we have $\tilde{\boldsymbol{V}}_2 = \tilde{\boldsymbol{V}}_2^*$. Therefore for all column j such that $\boldsymbol{V}_3^{(j)} \neq \boldsymbol{V}_3^{*(j)}$, it must be so that $\tilde{\boldsymbol{V}}_2^{*(j)} = \bar{\boldsymbol{e}}_k$, and the residual can be written as

$$\boldsymbol{V}_3^{(j)} - \boldsymbol{V}_3^{*(j)} = \boldsymbol{P}_2 \tilde{\boldsymbol{V}}_2^{*(j)} - \boldsymbol{P}_2^* \tilde{\boldsymbol{V}}_2^{*(j)} = \boldsymbol{P}_2 \bar{\boldsymbol{e}}_k - \boldsymbol{P}_2^* \bar{\boldsymbol{e}}_k = \boldsymbol{P}_2^{(k)} - \boldsymbol{P}_2^{*(k)}. \tag{62}$$

Assume that $\boldsymbol{P}_2^{(k)}$ has been used α times in the forward pass, follow the chain rule we then have

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{P}_2^{(k)}} = 2\sum_{j=1}^{L}\left(\frac{\partial \boldsymbol{V}_3^{(j)}}{\partial \boldsymbol{P}_2^{(k)}}\right)^\top \left(\boldsymbol{V}_3^{(j)} - \boldsymbol{V}_3^{*(j)}\right) = 2\alpha\left(\boldsymbol{P}_2^{(k)} - \boldsymbol{P}_2^{*(k)}\right). \tag{63}$$

□

Lemma G.21 (Learning $\boldsymbol{W}_2$)**.**
Fix an input sequence $\boldsymbol{V}_1 \in \mathbb{R}^{n^2\times L}$ and any column index $k \in [n^2]$, if $\texttt{HardMax}(\boldsymbol{C}_1 + \boldsymbol{W}_1) = \boldsymbol{W}_1^*$ and $\texttt{HardMax}\left(\boldsymbol{C}_{2(k)} + \boldsymbol{W}_2\right)$ equals to $\boldsymbol{W}_2^*$ everywhere except for the k-th column and if there exists a non-empty subset of indices $\mathcal{J} \subset [L]$ such that $\boldsymbol{V}_2^{*(j)} = \bar{\boldsymbol{e}}_k$ for all $j \in \mathcal{J}$, for any initialization $\boldsymbol{W}_{2(0)}^{(k)} \in \mathbb{R}^{n^2}$ such that $\|\boldsymbol{W}_{2(0)}^{(k)}\|_0 \le \frac{1}{2}$. Taking one surrogate gradient updates on $\boldsymbol{W}_{2(0)}^{(k)}$ as described in Algorithm 2 gives $\boldsymbol{W}_{2(1)}^{(k)}$ such that $\texttt{HardMax}(\boldsymbol{W}_{2(1)}^{(k)}) = \boldsymbol{W}_2^{*(k)}$.

Proof. Without loss of generality, assume at the initialization $\boldsymbol{P}_{2(0)}^{(k)} = \boldsymbol{e}_{a_{(0)}} \otimes \boldsymbol{e}_{b_{(0)}}$ and $\boldsymbol{P}_2^{*(k)} = \boldsymbol{e}_{a^*} \otimes \boldsymbol{e}_{b^*}$ for some $a_{(0)}, b_{(0)}, a^*, b^* \in [n]$. We first note that the conditions specified in the lemma meets the assumptions required by Lemma G.20, namely there is only one incorrect column in $\boldsymbol{W}_2$ missing and that column is being used in the forward pass at least one time (since $\mathcal{J}$ is non-empty). To prove $\texttt{HardMax}(\boldsymbol{W}_{2(1)}^{(k)}) = \boldsymbol{W}_2^{*(k)}$, it is sufficient to show that $\arg\max(\boldsymbol{W}_{2(1)}^{(k)}) = a^*n + b^*$. By Lemma G.20, we have

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{P}_2^{(k)}} = 2\sum_{j=1}^{L}\left(\frac{\partial \boldsymbol{V}_3^{(j)}}{\partial \boldsymbol{P}_2^{(k)}}\right)^\top \left(\boldsymbol{V}_3^{(j)} - \boldsymbol{V}_3^{*(j)}\right) = 2\alpha\left(\boldsymbol{P}_2^{(k)} - \boldsymbol{P}_2^{*(k)}\right) = 2\alpha \boldsymbol{e}_{a_{(0)}} \otimes \boldsymbol{e}_{b_{(0)}} - 2\alpha \boldsymbol{e}_{a^*} \otimes \boldsymbol{e}_{b^*}. \tag{64}$$

Since $\alpha \ge 1$, the negative margin of the $a^*n + b^*$-th entry dominates the initial difference in the initialization which is bounded by $\|\boldsymbol{W}_{2(0)}^{(k)}\|_0 \le \frac{1}{2}$. Thus we have

$$\arg\max\left(\boldsymbol{W}_{2(1)}^{(k)}\right) = \arg\max\left(\boldsymbol{W}_{2(0)}^{(k)} - \frac{\partial \mathcal{L}}{\partial \boldsymbol{P}_{2(0)}^{(k)}}\right) = a^*n + b^* \tag{65}$$

as desired. □

G.3.3 Proof for Theorem G.22

Now we are ready to prove for the main theorem restated below:

Theorem G.22 (Learning $\boldsymbol{\Pi}^*$ with context-enhanced surrogate GD with $\boldsymbol{\Pi}^*$-coverable input)**.**

For any initialization $\boldsymbol{W}_{1(0)}, \boldsymbol{W}_{2(0)} \in \mathbb{R}^{n^2\times n^2}$ such that $\|\boldsymbol{W}_{1(0)}\|_0 \le \frac{1}{2}$ and $\|\boldsymbol{W}_{2(0)}\|_0 \le \frac{1}{2}$, for any target set of phrasebooks $\boldsymbol{\Pi}^* = \{\pi_1^*, \pi_2^*\}$ in $\mathbf{MLT}(2, n)$, given an $\boldsymbol{\Pi}^*$-coverable input $\boldsymbol{V}_1$ and the corresponding ground truth label $\boldsymbol{V}_3^* = \mathbf{MLT}_{\boldsymbol{\Pi}^*}(\boldsymbol{V}_1)$, Algorithm 1 terminates with $\texttt{HardMax}(\boldsymbol{W}_1) = \boldsymbol{W}_1^*$ and $\texttt{HardMax}(\boldsymbol{W}_2) = \boldsymbol{W}_2^*$.

Proof. The statement can be proven by a similar induction as in the proof for Theorem G.14.

Let the induction hypothesis be such that when the enumeration goes to k-th column of the i-th layer, if $\texttt{HardMax}(\boldsymbol{W}_l^{(j)} + \boldsymbol{W}_l^{*(j)}) = \boldsymbol{W}_l^{*(j)}$ for all $(l, j) \in [2] \times [n^2]$ and $\texttt{HardMax}\left(\boldsymbol{W}_l^{(j)}\right) = \boldsymbol{W}_l^{*(j)}$ for all (l, j) such that $l < i$ or $l = i \wedge j < k$, then the gradient update on the k-th column of the i-th

layer ends with $\boldsymbol{W}_i^{(k)} = \boldsymbol{W}_i^{*(k)}$ while $\texttt{HardMax}(\boldsymbol{W}_l^{(j)} + \boldsymbol{W}_l^{*(j)}) = \boldsymbol{W}_l^{*(j)}$ for all $(l, j) \in [2] \times [n^2]$ is preserved.

The base case is satisfied as with initialization of $\|\boldsymbol{W}_{1(0)}\|_0 \le \frac{1}{2}$ and $\|\boldsymbol{W}_{2(0)}\|_0 \le \frac{1}{2}$, by Lemma G.9, we have $\texttt{HardMax}(\boldsymbol{W}_l^{(j)} + \boldsymbol{W}_l^{*(j)}) = \texttt{HardMax}(\boldsymbol{0} + \boldsymbol{W}_l^{*(j)}) = \boldsymbol{W}_l^{*(j)}$ for all $(l, j) \in [d] \times [n^2]$ and there are no requirements for $\boldsymbol{W}_i^{(k)} = \boldsymbol{W}_i^{*(k)}$ yet.

For the induction step, we note that with the inductive hypothesis of $\texttt{HardMax}(\boldsymbol{W}_l^{(j)} + \boldsymbol{W}_l^{*(j)}) = \boldsymbol{W}_l^{*(j)}$ for all $(l, j) \in [2] \times [n^2]$, $(\boldsymbol{C}_{1(k)}, \boldsymbol{W}_2^*)$ (when $i = 1$) or $(\boldsymbol{W}_1^*, \boldsymbol{C}_{2(k)})$ (when $i = 2$) will correctly condition all columns of $\boldsymbol{P}$'s except for the $\boldsymbol{P}_i^{(k)}$ since

$$\boldsymbol{C}_{i(k)}^{(k)} = \boldsymbol{W}_i^* \left(\boldsymbol{I}_{n^2} - \text{diag}(\bar{\boldsymbol{e}}_k)\right)^{(k)} = \boldsymbol{0}. \tag{66}$$

Thus by Lemma G.19 (when $i = 1$) or Lemma G.21 (when $i = 2$), we know that after updating $\boldsymbol{W}_i^{(k)}$, we have $\texttt{HardMax}(\boldsymbol{W}_i^{(k)}) = \boldsymbol{W}_i^{*(k)}$. The newly added column provides the correct inductive hypothesis on $\texttt{HardMax}(\boldsymbol{W}_l^{(j)}) = \boldsymbol{W}_l^{*(j)}$ for the next enumeration step.

By induction to $i = 2$ and $k = n^2$, we will be able to recover $\texttt{HardMax}(\boldsymbol{W}_i) = \boldsymbol{W}_i^*$ for all $i \in [d]$. □

Similarly we may use the coupon collecting argument to generalize the input to uniformly random strings as follows:

Corollary G.23 (Learning $\boldsymbol{\Pi}^*$ in $\mathbf{MLT}(2, n)$ with context-enhanced surrogate GD with random input)**.**
For any initialization $\boldsymbol{W}_{1(0)}, \boldsymbol{W}_{2(0)} \in \mathbb{R}^{n^2 \times n^2}$ such that $\|\boldsymbol{W}_{1(0)}\|_0 \le \frac{1}{2}$ and $\|\boldsymbol{W}_{2(0)}\|_0 \le \frac{1}{2}$, for any target set of phrasebooks $\boldsymbol{\Pi}^* = \{\pi_1^*, \pi_2^*\}$ in $\mathbf{MLT}(2, n)$, with probability at least $1 - \delta$ over a uniformly random input $\boldsymbol{V}_1$ of length $L = 2n^2 \log \frac{2n}{\delta}$, Algorithm 1 provided with the ground truth label $\boldsymbol{V}_3^* = \mathbf{MLT}_{\boldsymbol{\Pi}^*}(\boldsymbol{V}_1)$ terminates with $\texttt{HardMax}(\boldsymbol{W}_1) = \boldsymbol{W}_1^*$ and $\texttt{HardMax}(\boldsymbol{W}_2) = \boldsymbol{W}_2^*$.

G.4 Auxiliary Lemmas for Learning Surrogate Models

Lemma G.24. *For any long one-hot vector $\boldsymbol{v} = \boldsymbol{e}_a \otimes \boldsymbol{e}_b$, with $\boldsymbol{Q} = (I_n \otimes \boldsymbol{1}_n)(\boldsymbol{1}_n \otimes I_n)^\top$ we have*

$$\boldsymbol{Q}\boldsymbol{v} = \boldsymbol{e}_b \otimes \boldsymbol{1}_n; \qquad \boldsymbol{Q}^\top \boldsymbol{v} = \boldsymbol{1}_n \otimes \boldsymbol{e}_a. \tag{67}$$

Proof. Note that with $\boldsymbol{v} = \boldsymbol{e}_a \otimes \boldsymbol{e}_b$, we have

$$\begin{aligned}
\boldsymbol{Q}\boldsymbol{v} &= (I_n \otimes \boldsymbol{1}_n)\left(\boldsymbol{1}_n^\top \otimes I_n^\top\right)(\boldsymbol{e}_a \otimes \boldsymbol{e}_b) = (I_n \otimes \boldsymbol{1}_n)(1 \otimes \boldsymbol{e}_b) = (I_n \otimes \boldsymbol{1}_n)(\boldsymbol{e}_b \otimes 1) = \boldsymbol{e}_b \otimes \boldsymbol{1}_n; \\
\boldsymbol{Q}^\top \boldsymbol{v} &= (\boldsymbol{1}_n \otimes I_n)\left(I_n^\top \otimes \boldsymbol{1}_n^\top\right)(\boldsymbol{e}_a \otimes \boldsymbol{e}_b) = (\boldsymbol{1}_n \otimes I_n)(\boldsymbol{e}_a \otimes 1) = (\boldsymbol{1}_n \otimes I_n)(1 \otimes \boldsymbol{e}_a) = \boldsymbol{1}_n \otimes \boldsymbol{e}_a.
\end{aligned} \tag{68}$$

□

Lemma G.25. *For any one-hot vector $\boldsymbol{v} = \boldsymbol{e}_a \otimes \boldsymbol{e}_b \in \mathbb{R}^{n^2}$,*

$$\boldsymbol{Q}^\top \textit{diag}\left(\boldsymbol{Q}^\top \boldsymbol{v}\right) \textit{diag}\left(\boldsymbol{Q}^\top \boldsymbol{v}\right) \boldsymbol{Q} = (\boldsymbol{1}_n \boldsymbol{1}_n^\top) \otimes I_n.$$

Proof. By Lemma G.24, $\boldsymbol{Q}^\top \boldsymbol{v} = \boldsymbol{1}_n \otimes \boldsymbol{e}_a$. Therefore $\text{diag}\left(\boldsymbol{Q}^\top \boldsymbol{v}\right) = \text{diag}\left(\boldsymbol{1}_n\right) \otimes \text{diag}\left(\boldsymbol{e}_a\right) = I_n \otimes \text{diag}\left(\boldsymbol{e}_a\right)$. Thus

$$\text{diag}\left(\boldsymbol{Q}^\top \boldsymbol{v}\right) \boldsymbol{Q} = \left(I_n \otimes \text{diag}\left(\boldsymbol{e}_a\right)\right)\left(I_n \otimes \boldsymbol{1}_n\right)\left(\boldsymbol{1}_n^\top \otimes I_n^\top\right) = \left(I_n \otimes \boldsymbol{e}_a\right)\left(\boldsymbol{1}_n^\top \otimes I_n^\top\right) \tag{69}$$

and therefore we have

$$\begin{aligned}\boldsymbol{Q}^\top \text{diag}\left(\boldsymbol{Q}^\top \boldsymbol{v}\right) \text{diag}\left(\boldsymbol{Q}^\top \boldsymbol{v}\right) \boldsymbol{Q} &= (\mathbf{1}_n \otimes I_n)\left(I_n \otimes \boldsymbol{e}_a^\top\right)(I_n \otimes \boldsymbol{e}_a)(\mathbf{1}_n^\top \otimes I_n^\top) \\ &= (\mathbf{1}_n \otimes I_n)(I_n \otimes 1)(\mathbf{1}_n^\top \otimes I_n^\top) \qquad \left(\text{since } \boldsymbol{e}_a^\top \boldsymbol{e}_a = 1\right) \\ &= (\mathbf{1}_n \otimes I_n)(1 \otimes I_n)(\mathbf{1}_n^\top \otimes I_n^\top) \\ &= (\mathbf{1}_n \otimes I_n)(\mathbf{1}_n^\top \otimes I_n^\top) \\ &= (\mathbf{1}_n \mathbf{1}_n^\top) \otimes I_n \end{aligned} \tag{70}$$

□

Lemma G.26. *For any one-hot vector $\boldsymbol{v} = \boldsymbol{e}_a \otimes \boldsymbol{e}_b \in \mathbb{R}^{n^2}$,*

$$\boldsymbol{Q}\,\text{diag}\left(\boldsymbol{Q}\boldsymbol{v}\right)\,\text{diag}\left(\boldsymbol{Q}\boldsymbol{v}\right)\boldsymbol{Q}^\top = I_n \otimes (\mathbf{1}_n \mathbf{1}_n^\top).$$

Proof. This proof is very similar to the proof for Lemma G.25. By Lemma G.24, $\boldsymbol{Q}^\top \boldsymbol{v} = \mathbf{1}_n \otimes \boldsymbol{e}_a$. Therefore $\text{diag}\left(\boldsymbol{Q}\boldsymbol{v}\right) = \text{diag}\left(\boldsymbol{e}_b\right) \otimes \text{diag}\left(\mathbf{1}_n\right) = \text{diag}\left(\boldsymbol{e}_b\right) \otimes I_n$. Thus

$$\text{diag}\left(\boldsymbol{Q}\boldsymbol{v}\right)\boldsymbol{Q}^\top = \left(\text{diag}\left(\boldsymbol{e}_b\right) \otimes I_n\right)\left(\mathbf{1}_n \otimes I_n\right)\left(I_n^\top \otimes \mathbf{1}_n^\top\right) = \left(\boldsymbol{e}_b \otimes I_n\right)\left(I_n^\top \otimes \mathbf{1}_n^\top\right) \tag{71}$$

and therefore we have

$$\begin{aligned}\boldsymbol{Q}\text{diag}\left(\boldsymbol{Q}\boldsymbol{v}\right)\text{diag}\left(\boldsymbol{Q}\boldsymbol{v}\right)\boldsymbol{Q}^\top &= (I_n \otimes \mathbf{1}_n)\left(\boldsymbol{e}_b^\top \otimes I_n\right)(\boldsymbol{e}_b \otimes I_n)(I_n^\top \otimes \mathbf{1}_n^\top) \\ &= (I_n \otimes \mathbf{1}_n)(1 \otimes I_n)(I_n^\top \otimes \mathbf{1}_n^\top) \qquad \left(\text{since } \boldsymbol{e}_b^\top \boldsymbol{e}_b = 1\right) \\ &= (I_n \otimes \mathbf{1}_n)(I_n \otimes 1)(I_n^\top \otimes \mathbf{1}_n^\top) \\ &= (I_n \otimes \mathbf{1}_n)(I_n^\top \otimes \mathbf{1}_n^\top) \\ &= I_n \otimes (\mathbf{1}_n \mathbf{1}_n^\top). \end{aligned} \tag{72}$$

□

Lemma G.27 (Tail Bound for Coupon Collector Problem (Motwani, 1995))**.**
For a set S of size n, with probability at least $1-\delta$ one can cover all unique elements of S in $n \log \frac{n}{\delta}$ independent uniformly random sampling trials from S.

G.5 Learning $\boldsymbol{\Pi}^*$ in $\mathbf{MLT}_{\boldsymbol{\Pi}^*}$ with Gradient Descent (Empirical Evidence)

In this section we provide more details on empirically optimizing the simple surrogate model SURR-MLT$_{\{\boldsymbol{W}_i\}_{i=1}^d}$, which was only briefly discussed in the main text by the end of Section 4.1. We will first introduce the approximations we made to the surrogate model to make gradient-based optimization easy and stable, then we will present empirical results on the model learning target sets of phrasebooks $\mathbf{MLT}_{\boldsymbol{\Pi}^*}$ in $\mathbf{MLT}(5, 10)$, $\mathbf{MLT}(10, 10)$, and even $\mathbf{MLT}(20, 10)$.

G.5.1 Approximated Latent Model for GD

Recall that with input sequence represented by $\boldsymbol{V}_1 \in \mathbb{R}^{n^2 \times L}$, the surrogate model for a depth-d translation is being recursively defined by the translation + shifting operations

$$\boldsymbol{V}_{i+1} = \texttt{HardMax}(\boldsymbol{C}_i + \boldsymbol{W}_i)\,\texttt{Shift}(\boldsymbol{V}_i) \tag{73}$$

until we reach $\boldsymbol{V}_{d+1}$. While this model captures the essence of transition from ICL capability to memorization of specific set of phrasebooks, `HardMax` is making it not directly differentiable and hard to optimize. To address this issue, we approximate it with an column-wise softmax function with very low temperature ($T = 1/25$). The recursive definition in the approximated model is then

$$\tilde{\boldsymbol{V}}_{i+1} = \texttt{SoftMax}(25(\boldsymbol{C}_i + \boldsymbol{W}_i))\,\texttt{Shift}(\tilde{\boldsymbol{V}}_i). \tag{74}$$

We denote the recursive surrogate model with the softmax substitution as $\tilde{\text{SURR-MLT}}_{\{\boldsymbol{W}_i\}_{i=1}^d}(\boldsymbol{C}_1, \boldsymbol{C}_2, \ldots, \boldsymbol{C}_d, \boldsymbol{V}_1)$ where

$$\begin{aligned}\tilde{\boldsymbol{V}}_{d+1} &= \tilde{\text{SURR-MLT}}_{\{\boldsymbol{W}_i\}_{i=1}^d}(\boldsymbol{C}_1, \boldsymbol{C}_2, \ldots, \boldsymbol{C}_d, \boldsymbol{V}_1) \\ &\triangleq \texttt{SoftMax}(25\boldsymbol{C}_d + 25\boldsymbol{W}_d) \\ &\quad \texttt{Shift}\left(\texttt{SoftMax}(25\boldsymbol{C}_{d-1} + 25\boldsymbol{W}_{d-1})\,\texttt{Shift}\left(\cdots \texttt{SoftMax}(25\boldsymbol{C}_1 + 25\boldsymbol{W}_1)\,\texttt{Shift}(\tilde{\boldsymbol{V}}_1)\cdots\right)\right).\end{aligned} \tag{75}$$

We define the objective function as the column-wise cross-entropy loss between the final output and the input. Namely for input $\boldsymbol{V}_1$ with prediction $\tilde{\boldsymbol{V}}_{d+1}$ and ground truth label $\boldsymbol{V}^*_{d+1}$, the loss is computed as

$$\mathcal{L} = \sum_{k=1}^{L} \text{CrossEntropy}(\tilde{\boldsymbol{V}}^{(k)}_{d+1}, \boldsymbol{V}^{*(k)}_{d+1}). \tag{76}$$

We follow the same masking (dropout) curriculum as described in Appendix G.2 and Appendix G.3, that at each step we zero-out a single column from a single context matrix $\boldsymbol{C}_i$. We experiment on two gradient update schemes:

- Layer-wise Training: at each step, if we are masking a column on $\boldsymbol{C}_i$, we only compute the gradient with respect to $\boldsymbol{W}_i$ and update it. This training is more akin to the theoretical analysis described in Appendix G.3.
- Full Parameter Training: at any step, we compute the gradient with respect to each of the weight matrices and update all parameters. This is more akin to the real gradient-based training as we do not have the heuristics for localized update.

To allow for fast and stable training, we adopt a very large learning rate of $\eta = 100$ and apply parameter clipping between $[0, 1]$ after each update. The complete algorithm is described as follows:

Algorithm 3 Layerwise Gradient Descent with Context-Enhanced Learning For Optimizing SURR-MLT

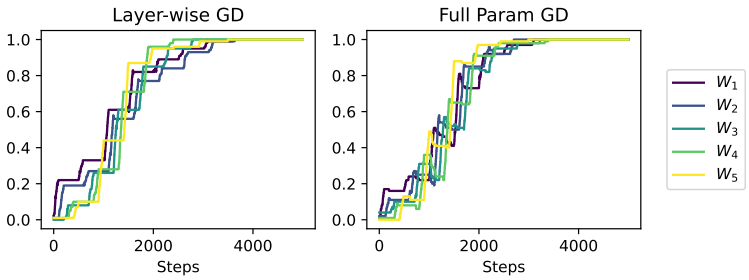
Input:
input $\boldsymbol{V}_1 \in \mathbb{R}^{n^2 \times L}$, label $\boldsymbol{V}^*_{d+1} \in \mathbb{R}^{n^2 \times L}$, descriptive text $\boldsymbol{W}^*_1, \ldots, \boldsymbol{W}^*_d \in \mathbb{R}^{n^2 \times n^2}$, learning rate η, total steps T

Initialize $\boldsymbol{W}_1, \ldots, W_d \leftarrow \boldsymbol{0}$ # Start with zero initialization
for $t = 1$ **to** T **do**
  $i \leftarrow \lfloor (t-1)/n^2 \rfloor \% d + 1$ # Get the layer to be masked
  $k \leftarrow ((t-1) \% n^2) + 1$ # Get the column index to be masked
  Initialize $\boldsymbol{C}_{i(k)} \leftarrow \boldsymbol{W}^*_i\,(\boldsymbol{I}_{n^2} - \text{diag}(\bar{\boldsymbol{e}}_k))$ # Create masked context matrix
  $\tilde{\boldsymbol{V}}_{d+1} \leftarrow \tilde{\text{SURR-MLT}}_{\{\boldsymbol{W}_i\}_{i=1}^d}(\boldsymbol{W}^*_1 \ldots, \boldsymbol{W}^*_{i-1}, \boldsymbol{C}_{i(k)}, \boldsymbol{W}^*_{i+1} \ldots, \boldsymbol{W}^*_d, \boldsymbol{V}_1)$
  $\mathcal{L} \leftarrow \text{CrossEntropy}(\tilde{\boldsymbol{V}}_{d+1}, \boldsymbol{V}^*_{d+1})$
  $\boldsymbol{W}_i \leftarrow \boldsymbol{W}_i - \eta \nabla_{\boldsymbol{W}_i} \mathcal{L}$ # Update the weight for the layer with mask
end for
Return $\boldsymbol{W}_1, \ldots, \boldsymbol{W}_d$.

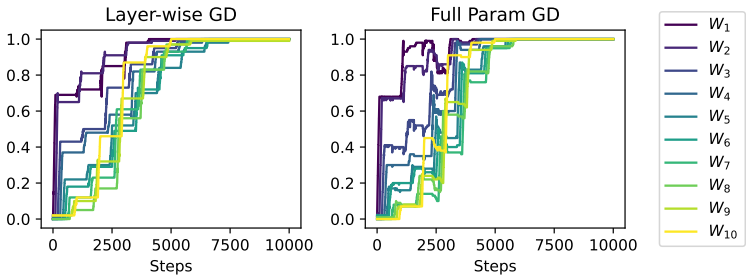
Algorithm 4 Full Parameter Gradient Descent with Context-Enhanced Learning For Optimizing SURR-MLT

Input:
input $\boldsymbol{V}_1 \in \mathbb{R}^{n^2 \times L}$, label $\boldsymbol{V}_{d+1}^* \in \mathbb{R}^{n^2 \times L}$, descriptive text $\boldsymbol{W}_1^*, \ldots, \boldsymbol{W}_d^* \in \mathbb{R}^{n^2 \times n^2}$, learning rate η, total steps T

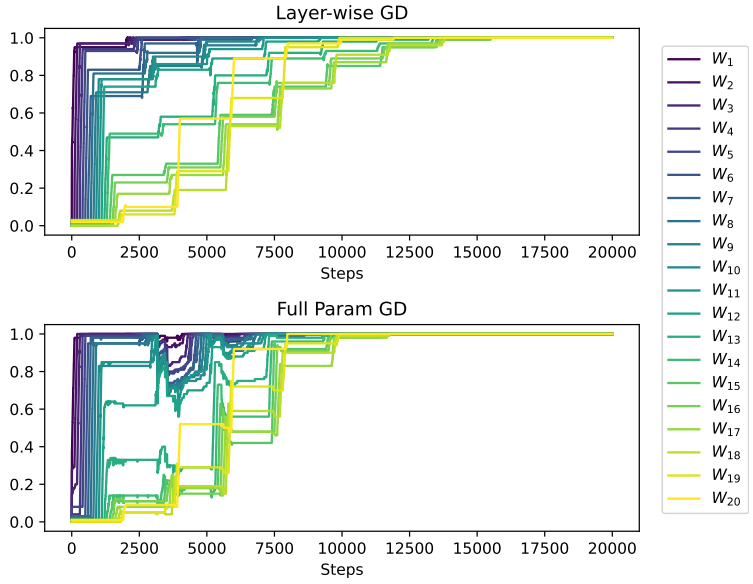
Initialize $\boldsymbol{W}_1, \ldots, W_d \leftarrow \boldsymbol{0}$ # Start with zero initialization
for $t = 1$ **to** T **do**
  $i \leftarrow \lfloor (t-1)/n^2 \rfloor \% d + 1$ # Get the layer to be masked
  $k \leftarrow ((t-1) \% n^2) + 1$ # Get the column index to be masked
  Initialize $\boldsymbol{C}_{i(k)} \leftarrow \boldsymbol{W}_i^* \left(\boldsymbol{I}_{n^2} - \text{diag}(\bar{\boldsymbol{e}}_k)\right)$ # Create masked context matrix
  $\tilde{\boldsymbol{V}}_{d+1} \leftarrow \text{SURR-MLT}_{\{\boldsymbol{W}_i\}_{i=1}^d}(\boldsymbol{W}_1^* \ldots, \boldsymbol{W}_{i-1}^*, \boldsymbol{C}_{i(k)}, \boldsymbol{W}_{i+1}^* \ldots, \boldsymbol{W}_d^*, \boldsymbol{V}_1)$
  $\mathcal{L} \leftarrow \text{CrossEntropy}(\tilde{\boldsymbol{V}}_{d+1}, \boldsymbol{V}_{d+1}^*)$
  for $l = 1$ **to** d **do**
    $\boldsymbol{W}_l \leftarrow \boldsymbol{W}_l - \eta \nabla_{\boldsymbol{W}_l} \mathcal{L}$ # Update the weight for all layers
  end for
end for
Return $\boldsymbol{W}_1, \ldots, \boldsymbol{W}_d$.



(a) **MLT**$(5, 10)$



(b) **MLT**$(10, 10)$



(c) **MLT**$(20, 10)$

Figure 11: We perform Layer-wise and Full Parameter Training with Gradient Descent (defined in Appendix G.5.1) on the SURR-MLT (Definition G.6) designed for $\mathbf{MLT}_{\mathbf{\Pi}^*}$ in $\mathbf{MLT}(5, 10), \mathbf{MLT}(10, 10), \mathbf{MLT}(20, 10)$ respectively (alternately, depth $d = 5, 10, 20$ respectively, while number of characters is fixed at $n = 10$). Here, we report the portion of columns from the trainable parameters, which after $\texttt{HardMax}$ application $\{\texttt{HardMax}(\boldsymbol{W}_i)\}_{i=1}^{d}$, align with the corresponding stochastic matrices of the phrasebooks $\{\texttt{Matrix}(\pi_i^*)\}_{i=1}^{d}$.We observe that under both algorithms, the trainable parameters quickly learn the relevant stochastic matrices.

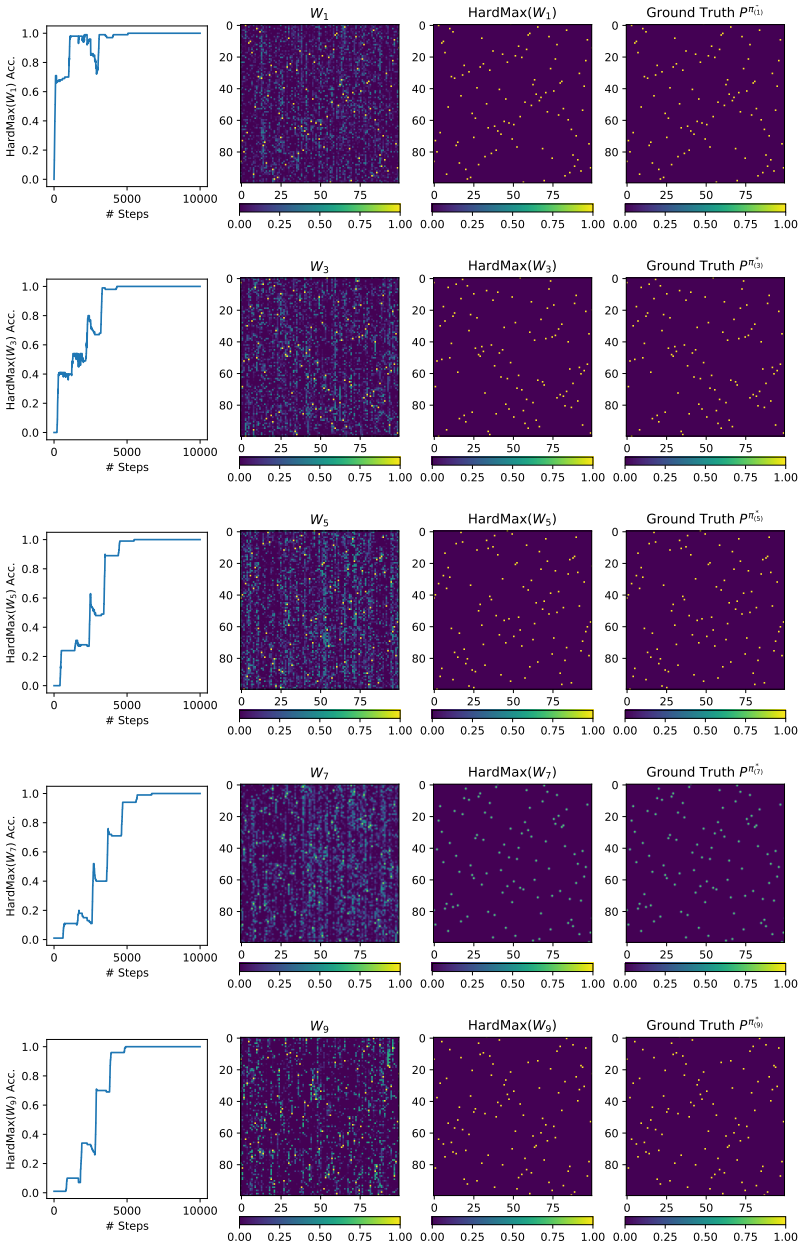


Figure 12: Detailed analysis on the Full parameter training behavior of the trainable parameters in SURR-MLT from Figure 11 for $\mathbf{MLT}_{\mathbf{\Pi}^*}$ in $\mathbf{MLT}(10, 10)$ (i.e. depth $d = 10$ and number of characters $n = 10$). We report the behavior of all odd-index parameters $\boldsymbol{W}_1, \boldsymbol{W}_3, \cdots, \boldsymbol{W}_9$. (left to right) first, we show the number of columns of the trainable parameter, which after $\texttt{HardMax}(\boldsymbol{W}_i)$ align with the corresponding columns of $\texttt{Matrix}(\pi_i^*)$. Second, third and fourth visualize the matrices, $\boldsymbol{W}_i$, $\texttt{HardMax}(\boldsymbol{W}_i)$, and $\texttt{Matrix}(\pi_i^*)$ respectively. All matrices learn to match $\texttt{Matrix}(\pi_i^*)$ at the end of training with $\texttt{HardMax}$ operation.

H CONSTRUCTION OF A TRANSFORMER THAT CAN SIMULATE THE LATENT MODEL

H.1 USEFUL DEFINITIONS AND LEMMAS

Definition H.1 (Relative self-attention with 1 head)**.** For a set of matrices $\{\boldsymbol{W}_{query}, \boldsymbol{W}_{key}, \boldsymbol{W}_{value}\}$ with each matrix $\in \mathbb{R}^{k\times k}$ for some $k > 0$ and a set of $(t+1)$ biases $\{b_i\}_{t\le i\le 0}$, the self-attention computation on an input sequence $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_L$ with each $\boldsymbol{x}_{p_2} \in \mathbb{R}^k$ is given by the output sequence $\boldsymbol{y}_1, \cdots, \boldsymbol{y}_L$, where for all $p_1 \in [1, L]$

$$\boldsymbol{y}_{p_1} = \sum_{p_2=1}^{L} a_{p_1,p_2}\boldsymbol{o}_j, \text{ where } a_{p_1p_2} = \frac{e^{\boldsymbol{q}_{p_1}^\top \boldsymbol{k}_{p_2}+b_{p_2-p_1}}}{\sum_{p_2'\le p_1} e^{\boldsymbol{q}_{p_1}^\top \boldsymbol{k}_{p_2'}+b_{p_2'-p_1}}} \text{ if } p_2 \le p_1, 0 \text{ otherwise}$$
$$\boldsymbol{q}_{p_2} = \boldsymbol{W}_{query}\boldsymbol{x}_{p_2}, \quad \boldsymbol{k}_{p_2} = \boldsymbol{W}_{key}\boldsymbol{x}_{p_2}, \quad \boldsymbol{o}_{p_2} = \boldsymbol{W}_{value}\boldsymbol{x}_{p_2}, \quad \text{for all } p_2 \in [1, L].$$

For a relative self-attention with H heads, we will simply add the output of the H heads as the final output.

Definition H.2 (MLP)**.** For a set of matrices $\boldsymbol{W}_{outer} \in \mathbb{R}^{k\times H}, \boldsymbol{W}_{inner} \in \mathbb{R}^{H\times k}$ for some $k, H > 0$ and an activation function σ, the output of the MLP layer on an input sequence $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_L$ with each $\boldsymbol{x}_i \in \mathbb{R}^k$ is given by the output sequence $\boldsymbol{y}_1, \cdots, \boldsymbol{y}_L$, where for all i

$$\boldsymbol{y}_i = \boldsymbol{W}_{outer}\sigma(\boldsymbol{W}_{inner}\boldsymbol{x}_i).$$

Lemma H.3. *For GELU (Hendrycks & Gimpel, 2016) activation function, which takes $x \in \mathbb{R}$ as input and returns $x\Phi(x)$ as output, with $\Phi(x)$ representing the standard Gaussian cumulative distribution function, for any two variables $x, y \in \mathbb{R}$, the following holds true:*

$$\sqrt{\pi/2}\,(\textit{GELU}(x+y) - \textit{GELU}(x) - \textit{GELU}(y)) = xy + \mathcal{O}(x^3y^3).$$

The above lemma has been taken from (Akyurek et al., 2023).

H.2 TRANSFORMER CONSTRUCTION

Recall that a translation task in $\mathbf{MLT}(d, n)$ involves two primary operations at each step : *Circular shift* and *Translate*. We will refer to the surrogate model to use notations for different operations in the translation task. Recall from Equation (19), the surrogate model for $\mathbf{MLT}(d, n)$, denoted by $\text{SURR-MLT}_{\{\boldsymbol{W}_i\}_{i=1}^d}(\cdot)$ with trainable parameters $\{\boldsymbol{W}_i\}_{i=1}^d = \{\boldsymbol{W}_1, \cdots, \boldsymbol{W}_d\}$, can be represented by the following recursive expression

$$\boldsymbol{V}_{i+1} = \texttt{HardMax}(\boldsymbol{C}_i + \boldsymbol{W}_i)\,\texttt{Shift}(\boldsymbol{V}_i) := \texttt{HardMax}(\boldsymbol{C}_i + \boldsymbol{W}_i)\tilde{\boldsymbol{V}}_i, \tag{77}$$

for $1 \le i \le d$. Here `Shift` represents *Circular shift* operation, and $\texttt{HardMax}(\boldsymbol{C}_i + \boldsymbol{W}_i)$ represents *Translate* operation, where the operation can be done either using the relevant in-context information $\boldsymbol{C}_i$ or the in-weights memory paramters $\boldsymbol{W}_i$ when in-context information isn't provided in form of $\boldsymbol{C}_i$. `HardMax` is the column-wise hard-max function converting $\boldsymbol{C}_i + \boldsymbol{W}_i$ to a binary column stochastic matrix.

Lemma H.4. *For the family of translation tasks $\mathbf{MLT}(d, n)$, there exists a transformer model with embedding size $2n^2 + 2d + 4$, $2d$ relative self-attention layers (containing either 1 or 3 heads), and $2d$ MLP layers that can simulate the surrogate model, $\text{SURR-MLT}_{\{\boldsymbol{W}_i\}_{i=1}^d}(\cdot)$.*

For each input sequence $\boldsymbol{s}_1$ of length L, the input sequence of embeddings to the transformer will be of length $n^2d + L/2 + d$, where the last d embeddings are padding tokens (<P>) and given as 0s, and the length of output sequence of the transformer model will be $n^2d + d + L/2$, where the last $L/2$ output embeddings will be used for loss computation. The middle d embeddings in output are represented by <THINK> tokens. [a]

[a] In our experiments, we used output sequence length as $n^2d + d + dL/2$, where $(d+1)L/2$ tokens in the output sequence were represented by <THINK> tokens. Instead, we only use d <THINK> tokens in our output construction for simplicity, the proof can be easily modified to align with the experiments.

Outline of the construction: Our argument will be for any general $\mathbf{MLT}(d, n)$. To create the transformer, we will create similar transformer modules that handle $\texttt{HardMax}(\boldsymbol{C}_i+\boldsymbol{W}_i)\,\texttt{Shift}(\boldsymbol{V}_i)$ for each i. We will refer to the constructed modules as MLT-MODULE and will mention the specific step i as an argument when we attempt to use the module to perform translation step $\texttt{HardMax}(\boldsymbol{C}_i + \boldsymbol{W}_i)\,\texttt{Shift}(\boldsymbol{V}_i)$. W.l.o.g., we will assume we are building a MLT-MODULE to perform translation step $\texttt{HardMax}(\boldsymbol{C}_i+\boldsymbol{W}_i)\,\texttt{Shift}(\boldsymbol{V}_i)$. MLT-MODULE will contain two modules, CIRCULAR SHIFT-MODULE and TRANSLATE-MODULE, which simulate *Circular shift* and *Translate* operations, and have been outlined in Algorithms 5 and 6.

Next, we explain the structure of the embeddings in the transformer architecture. Our embeddings will be built on the context $\{\boldsymbol{C}_1, \cdots, \boldsymbol{C}_d\}$ and the matrix representation $\boldsymbol{V}_1$ from sequences $(\boldsymbol{s}_{1,1}, \cdots, \boldsymbol{s}_{1,L})$ and subsequent intermediate representation of the translation task. Furthermore, our embeddings will contain additional information like indices of the context matrices when utilizing them in-context, segment indicators that represent whether embeddings belong to context matrices, or the input sequence tokens, and start and end indicators that indicate the start and the end embeddings representing the input sequence. This information can be extracted from the input sequence using a few input processing layers, though we do not delve into the specifics.

H.3 STRUCTURE OF INPUT EMBEDDINGS TO MLT-MODULE

Our embeddings will be split into 4 components: token, context matrix indicator, start and end indicator, and segment indicator. To maintain simplicity in our discussion, we will present them as 4 separate embeddings.

For a module that will represent $\texttt{HardMax}(\boldsymbol{C}_i + \boldsymbol{W}_i)\,\texttt{Shift}(\boldsymbol{V}_i)$, we assume the input will be provided as 2 major segments.

1. **First segment: in-context matrices** We will give the in-context information $\{\boldsymbol{C}_{p_1}\}_{p_1=1}^{d}$ as follows.: each context matrix $\boldsymbol{C}_{p_1}$ will be fed as n^2 token embeddings, $\{[\boldsymbol{e}_j; \boldsymbol{C}_{p_1}\boldsymbol{e}_j] \in \mathbb{R}^{2n^2}\}_{j=1}^{n^2}$, where $\boldsymbol{e}_j$ represents a one-hot n^2 dimensional vector that contains 1 in position j and $[\boldsymbol{e}_j; \boldsymbol{C}_{p_1}\boldsymbol{e}_j]$ represents a concatenation of $\boldsymbol{e}_j$ and $\boldsymbol{C}_i\boldsymbol{e}_j$. Thus, the in-context information will look as follows

$$\{[\boldsymbol{e}_j; \boldsymbol{C}_1\boldsymbol{e}_j]\}_{j=1}^{n^2}, \cdots, \{[\boldsymbol{e}_j; \boldsymbol{C}_d\boldsymbol{e}_j]\}_{j=1}^{n^2}$$

   Additional embedding: context matrix level indicator In order to differentiate the different context matrices, we will use an additional d dimensions in the embeddings to represent a one-hot vector that indicates the index of the corresponding step they will be used for. For simplicity, we will represent these dimensions separately as separate embeddings: $\boldsymbol{l}_{p_1} \in \mathbb{R}^d$, which are one-hot vectors that contain 1 in position p_1 and 0 otherwise.

2. **Second segment: input query** For a length-L input sequence $\boldsymbol{s}_i = (\boldsymbol{s}_{i,1}, \boldsymbol{s}_{i,2}, \ldots, \boldsymbol{s}_{i,L})$, we will use the sequence of columns of its matrix representation $\boldsymbol{V}_i \in \mathbb{R}^{n^2 \times L/2}$, appended by 0s to match embedding sizes, as token embeddings for the input query. A padding embedding <P>, containing 0s, follows this sequence as an end of sequence embedding.

   Additional embedding: start and end indicator embedding We will have 2 additional dimensions representing whether a token represents the start or the end of the input query sequence (first or second dimension activated respectively). Start of the input query sequence is determined by the first embedding in the input query, while end of the input query sequence is determined by the first padding embedding containing 0s after the input query sequence. We will represent these dimensions separately as separate embeddings: $\{\boldsymbol{b}_1, \boldsymbol{b}_2, \mathbf{0} \in \mathbb{R}^2\}_{i \in L}$, which are one-hot vectors. $\boldsymbol{b}_1$ contains 1 in dimension 1 if the embeddings represents start of the sequence, $\boldsymbol{b}_2$ contains 1 in dimension 2 if the embeddings represent end of the sequence. Other embeddings have **0**s.

Additional embedding: segment embedding We will differentiate the input embeddings in the two segments using 2 dimensions that represent one-hot vectors indicating segment indices. We will represent these dimensions separately as separate embeddings: $\boldsymbol{g}_1, \boldsymbol{g}_2 \in \mathbb{R}^2$, where both are one-hot vectors, with $\boldsymbol{g}_1$ containing 1 in dimension 1 and $\boldsymbol{g}_2$ containing 1 in dimension 2.

All the notations have been summarized in Table 5.

Additional optional inputs: There might be additional input embeddings, represented as <THINK> in the first segment, which are null inputs and are ignored during self-attention computation. As discussed next, we will right shift the sequence by 1 at each step, in order to handle *Circular shift* operation with causal masking in transformers.

H.4 STRUCTURE OF THE OUTPUT EMBEDDINGS FROM MLT-MODULE

All the embeddings in the first segment are kept intact. In the second segment, the module outputs a null output <THINK>, followed by $L/2$ output embeddings that represent the columns of $\boldsymbol{V}_i$. We will require <THINK> to represent *Circular shift* with causal self-attention in transformers. <THINK> will be ignored in self-attention computation and so, we will ignore their discussion for simplicity of presentation. Other embedding values are kept intact for the generated output, except start and end indicator embeddings $\boldsymbol{b}_1, \boldsymbol{b}_2$, which need to be right shifted at each step. The right shift operation can be handled similar to our computations on the token embeddings and so, we ignore them in our construction below. We summarize these in Table 6.

Embedding Name	Dimension size	First segment values (In-context information)	Second segment values (Input sequence embeddings)
Token	$2n^2$	$\{[\boldsymbol{e}_j; \boldsymbol{C}_1\boldsymbol{e}_j]\}_{j=1}^{n^2}, \cdots, \{[\boldsymbol{e}_j; \boldsymbol{C}_d\boldsymbol{e}_j]\}_{j=1}^{n^2}$	$[\boldsymbol{V}_{i-1}^{(1)}; \boldsymbol{0}], \cdots, [\boldsymbol{V}_{i-1}^{(L/2)}; \boldsymbol{0}]$, <P>
Context matrix index indicator	d	$\{\boldsymbol{l}_1\}_{[1,n^2]}, \{\boldsymbol{l}_2\}_{[1,n^2]}, \cdots \{\boldsymbol{l}_d\}_{[1,n^2]}$	$\boldsymbol{0}, \cdots, \boldsymbol{0}$
Start and End indicator	2	$\boldsymbol{0}, \cdots, \boldsymbol{0}$	$\boldsymbol{b}_1, \boldsymbol{0}, \cdots, \boldsymbol{0}, \boldsymbol{b}_2$
Segment	2	$\boldsymbol{g}_1, \cdots, \boldsymbol{g}_1$	$\boldsymbol{g}_2, \cdots, \boldsymbol{g}_2$

Table 5: Input embeddings to MLT-MODULE that simulates $\texttt{HardMax}(\boldsymbol{C}_i + \boldsymbol{W}_i)\,\texttt{Shift}(\boldsymbol{V}_i)$. $\boldsymbol{e}_j$ indicates a one-hot n^2 dimensional vector that contains 1 in dimension j.

Embedding Name	Dimension size	First segment values (In-context information)	Second segment values (Input sequence embeddings)
Token	$2n^2$	$\{[\boldsymbol{e}_j; \boldsymbol{C}_1\boldsymbol{e}_j]\}_{j=1}^{n^2}, \cdots, \{[\boldsymbol{e}_j; \boldsymbol{C}_d\boldsymbol{e}_j]\}_{j=1}^{n^2}$	<THINK>, $[\boldsymbol{V}_i^{(1)}; \boldsymbol{0}], \cdots, [\boldsymbol{V}_i^{(L/2)}; \boldsymbol{0}]$
Context matrix index indicator	d	$\{\boldsymbol{l}_1\}_{[1,n^2]}, \{\boldsymbol{l}_2\}_{[1,n^2]}, \cdots \{\boldsymbol{l}_d\}_{[1,n^2]}$	$\boldsymbol{0}, \boldsymbol{0}, \cdots, \boldsymbol{0}$
Start and End indicator	2	$\boldsymbol{0}, \cdots, \boldsymbol{0}$	$\boldsymbol{0}, \boldsymbol{b}_1, \boldsymbol{0}, \cdots, \boldsymbol{0}, \boldsymbol{b}_2$
Segment	2	$\boldsymbol{g}_1, \cdots, \boldsymbol{g}_1$	$\boldsymbol{0}, \boldsymbol{g}_2, \cdots, \boldsymbol{g}_2$

Table 6: Output of the transformer module MLT-MODULE that simulates $\texttt{HardMax}(\boldsymbol{C}_i + \boldsymbol{W}_i)\,\texttt{Shift}(\boldsymbol{V}_i)$. <THINK> represents a null output and won't be attended to in the future modules. We ignore this symbol for simplicity, when analyzing any module. $\boldsymbol{e}_j$ indicates a one-hot n^2 dimensional vector that contains 1 in dimension j.

Constructing the MLT-MODULE: The MLT-MODULE consists of 2 self-attention layers and 2 MLP layers. We use one self-attention layer and an MLP layer to represent *Circular shift* operation, one self-attention layer to represent $\boldsymbol{C}_i\,\texttt{Shift}(\boldsymbol{V}_i)$, and one MLP layer to represent $\texttt{HardMax}(\boldsymbol{C}_i + \boldsymbol{W}_i)\,\texttt{Shift}(\boldsymbol{V}_i)$. We name the two modules for *Circular shift* and *Translate* as CIRCULAR SHIFT-MODULE and TRANSLATE-MODULE respectively. We have outlined their constructions in Algorithms 5 and 6.

H.5 STEP 1 (CIRCULAR SHIFT-MODULE): REPRESENT *Circular shift* USING A SELF-ATTENTION AND AN MLP LAYER

As *Circular shift* only focuses on the input query sequence and not the in-context matrices, we will simply focus the module's operation on embeddings in the second segment. The effect of the operation on embeddings in the first segment can be removed using a gated residual connection. From Lemma G.3, we have that the output of the *Circular shift* operation on any sequence, represented by its matrix representation $\boldsymbol{V}_i$, can be written as

$$\tilde{\boldsymbol{V}}_i^{(j)} = Q\boldsymbol{V}_i^{(j)} \odot Q^\top \boldsymbol{V}_i^{((j+1)\%L)}, \text{ for all } 1 \leq j \leq L/2.$$

where $Q = (I_n \otimes \mathbf{1}_n)(\mathbf{1}_n \otimes I_n)^\top$, $\mathbf{1}_n \in \mathbb{R}^{n\times 1}$ is the all-ones vector, and $\odot$ is the Hadamard product.

In order to represent the operation, we will first use a self-attention layer to compute $(\mathbf{1}_n \otimes I_n)^\top \boldsymbol{V}_i^{(j)}$ and $(I_n \otimes \mathbf{1}_n)\boldsymbol{V}_i^{(j+1\%L)}$ at each column j. Because the computation of $\tilde{\boldsymbol{V}}_i^{(j)}$ requires the model to look forward to $\boldsymbol{V}_i^{(j+1)}$, we need to shift the computation of $\tilde{\boldsymbol{V}}_i^{(j)}$ to position $j+1$, as a causal attention mask is involved in self-attention computation.

After right shift operation, we will represent the output of the self-attention computation as <THINK>, $[\mathbf{o}_2; \mathbf{0}], [\mathbf{o}_3; \mathbf{0}], \cdots, [\mathbf{o}_{L/2+1}; \mathbf{0}]$, and we will ignore the <THINK> embedding. Note that the second half of the output embeddings will still contain 0s and we will ignore them in the current computation. Then, the above computation can be rephrased as

$$\mathbf{o}_j = Q\boldsymbol{V}_i^{(j-1)} \odot Q^\top \boldsymbol{V}_i^{(j)}, \text{ for all } 2 \le j \le L/2.$$
$$\mathbf{o}_{L/2+1} = Q\boldsymbol{V}_i^{(L/2)} \odot Q^\top \boldsymbol{V}_i^{(1)}$$

Self-attention layer: The computation of $\mathbf{o}_j$, for $2 \le j \le L/2$, requires the computation of $(\mathbf{1}_n \otimes I_n)^\top \boldsymbol{V}_i^{(j-1)}$ and $(I_n \otimes \mathbf{1}_n)\boldsymbol{V}_i^{(j)}$. This will require 2 attention heads, one head that attends to itself, and another that attends to previous embedding at each position. We will outline both below. We will require one additional head, as computing $\boldsymbol{o}_{L/2+1}$ will require the model to compute $(I_n \otimes \mathbf{1}_n)\boldsymbol{V}_i^{(1)}$.

1. Attention Head 1 computes $(\mathbf{1}_n \otimes I_n)^\top \boldsymbol{V}_i^{(j-1)}$ at position j for all $2 \le j \le L/2+1$. This can be done using a self-attention head (Definition H.1) that sets query and key matrices $\boldsymbol{W}_{query}$, $\boldsymbol{W}_{key}$, and biases $\{b_i\}_{t\le i\le 0}$ such that the attention score between embeddings at any two positions p_1, p_2 is given as follows:

   $$a_{p_1,p_2} = 1, \text{if } p_2 - p_1 = -1, \quad 0 \text{ otherwise}$$

   $\boldsymbol{W}_{value}$ is set such that for any input $\boldsymbol{x}$, the output of $\boldsymbol{W}_{value}\boldsymbol{x}$ is given by

   $$(\boldsymbol{W}_{value}\boldsymbol{x})_{p_1} = \sum_{j=0}^{n-1} x_{n\cdot j+p_1}.$$

   In simple words, this operation simply adds up the values in dimensions $p_1, p_1+n, p_1+2n, \cdots$ and stores them at position p_1 for all $1 \le p_1 \le n$.

2. Attention Head 2 computes $(I_n \otimes \mathbf{1}_n)\boldsymbol{V}_i^{(j)}$ at position j for all $2 \le j \le L/2$. This can be done using a self-attention head (Definition H.1) that sets $\boldsymbol{W}_{query}$, $\boldsymbol{W}_{key}$, $\boldsymbol{W}_{value}$ and biases $\{b_i\}_{t\le i\le 0}$ such that the attention score between embeddings at any two positions p_1, p_2 is given as follows:

   $$a_{p_1,p_2} = 1, \text{if } p_2 - p_1 = 0, \quad 0 \text{ otherwise}$$

   $\boldsymbol{W}_{value}$ is set such that for any input $\boldsymbol{x}$, the output of $\boldsymbol{W}_{value}\boldsymbol{x}$ is given by

   $$(\boldsymbol{W}_{value}\boldsymbol{x})_{p_1+n} = \sum_{j=1}^{n} x_{j+p_1 n-n}.$$

   In simple words, this operation simply adds up the values in dimensions $1+(p_1-1)n, 2+(p_1-1)n, \cdots$ and stores them at dimension p_1+n for all $1 \le p_1 \le n$.

3. Attention head 3 will compute $(I_n \otimes \mathbf{1}_n)\boldsymbol{V}_i^{(1)}$ and store in $\mathbf{o}_{L/2+1}$. This can be done by a self-attention layer which activates only between positions p_1 and p_2 that represent the start and the end tokens of the sequence, i.e. contain $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$ as start and end indicator embeddings, and is 0 otherwise. $\boldsymbol{W}_{value}$ is set same as attention head 2.

The output of the three heads are simply added up. Hence, at each position $2 \le j \le L/2+1$, the output $\boldsymbol{o}_j$ has $(\mathbf{1}_n \otimes I_n)^\top \boldsymbol{V}_i^{(j-1)}$ in $[1, n]$ dimensions and $(I_n \otimes \mathbf{1}_n)\boldsymbol{V}_i^{(j\%L)}$ in $[n+1, 2n]$ dimensions.

MLP layer: The objective with the MLP layer (Definition H.2) will be to multiply $(\mathbf{1}_n \otimes I_n)^\top \boldsymbol{V}_i^{(j-1)}$ present in $[1, n]$ dimensions and $(I_n \otimes \mathbf{1}_n)\, \boldsymbol{V}_i^{(j\%L)}$ present in $[n+1, 2n]$ dimensions in each position j. This can be done by using an MLP layer with GELU activation by using Lemma H.3. The weights of the MLP layer are set as follows: $\boldsymbol{W}_{inner}$ is set such that for all input $\boldsymbol{x}$, we have

$$(\boldsymbol{W}_{inner}\boldsymbol{x})_i = \frac{1}{N}x_i + \frac{1}{N}x_{n+i}, \quad \text{for all } 1 \le i \le n,$$
$$(\boldsymbol{W}_{inner}\boldsymbol{x})_{i+n} = \frac{1}{N}x_i, \quad \text{for all } 1 \le i \le n,$$
$$(\boldsymbol{W}_{inner}\boldsymbol{x})_{i+2n} = \frac{1}{N}x_{n+i}, \quad \text{for all } 1 \le i \le n.$$

$\boldsymbol{W}_{outer}$ is set such that for all $\boldsymbol{x} \in \mathbb{R}^{3n}$

$$(\boldsymbol{W}_{outer}\boldsymbol{x})_i = N^2(x_i - x_{i+n} - x_{i+2n}), \quad \text{for all } 1 \le i \le n.$$

All other coordinates in these matrices are set as 0s. N is set as a large number (say 100). By Lemma H.3, the output of the MLP layer will be <THINK>, $\mathbf{o}_2, \cdots, \mathbf{o}_{L/2+1}$, with $\mathbf{o}_j$ containing

$$\tilde{\boldsymbol{V}}_i^{(j-1)} + \mathcal{O}(N^{-4}) := (\mathbf{1}_n \otimes I_n)^\top \boldsymbol{V}_i^{(j-1)} \odot (I_n \otimes \mathbf{1}_n)\, \boldsymbol{V}_i^{(j\%L)} + \mathcal{O}(N^{-4})$$

at each position $2 \le j \le L/2$.

Embedding Name	Dimension size	First segment values (In-context information)	Second segment values (Input query sequence embeddings)
Token	$2n^2$	$\{[\boldsymbol{e}_j; \boldsymbol{C}_1\boldsymbol{e}_j]\}_{j=1}^{n^2}, \cdots, \{[\boldsymbol{e}_j; \boldsymbol{C}_d\boldsymbol{e}_j]\}_{j=1}^{n^2}$	<THINK>, $[\tilde{\boldsymbol{V}}_i^{(1)}; \mathbf{0}], \cdots, [\tilde{\boldsymbol{V}}_i^{(L/2)}; \mathbf{0}]$
Context matrix index indicator	d	$\{\boldsymbol{l}_1\}_{[1,n^2]}, \{\boldsymbol{l}_2\}_{[1,n^2]}, \cdots \{\boldsymbol{l}_d\}_{[1,n^2]}$	$\mathbf{0}, \mathbf{0}, \cdots, \mathbf{0}$
Start and End indicator	2	$\mathbf{0}, \cdots, \mathbf{0}$	$\mathbf{0}, \boldsymbol{b}_1, \mathbf{0}, \cdots, \mathbf{0}, \boldsymbol{b}_2$
Segment	2	$\boldsymbol{g}_1, \cdots, \boldsymbol{g}_1$	$\mathbf{0}, \boldsymbol{g}_2, \cdots, \boldsymbol{g}_2$

Table 7: Output of CIRCULAR SHIFT-MODULE in MLT-MODULE that simulates *Circular shift*, i.e. computes $\texttt{Shift}(\boldsymbol{V}_i)$ at second segment token embeddings. <THINK> represents a null output and won't be attended to in the future modules. We ignore this symbol for simplicity, when analyzing any module. $\boldsymbol{e}_j$ indicates a one-hot n^2 dimensional vector that contains 1 in dimension j.

H.6 STEP 2 (TRANSLATE-MODULE): *Translate* AS A MODULE CONTAINING A SELF-ATTENTION AND AN MLP LAYER

Our current token embeddings are given as <THINK>, $[\mathbf{o}_2; \mathbf{0}], [\mathbf{o}_3; \mathbf{0}], \cdots, [\mathbf{o}_{L/2+1}; \mathbf{0}]$, where each $\mathbf{o}_j$ contain $\tilde{\boldsymbol{V}}_i^{(j-1)}$. Other embeddings have been kept intact. The in-context information are given as $\{[\boldsymbol{e}_j; \boldsymbol{C}_1\boldsymbol{e}_j]\}_{j=1}^{n^2}, \cdots, \{[\boldsymbol{e}_j; \boldsymbol{C}_d\boldsymbol{e}_j]\}_{j=1}^{n^2}$. We will first use a self-attention layer to compute $[\boldsymbol{C}_i\tilde{\boldsymbol{V}}_i^{(j-1)}; \tilde{\boldsymbol{V}}_i^{(j-1)}]$ at position $2 \le j \le L/2+1$. We then use an MLP layer to represent $\texttt{HardMax}(\boldsymbol{C}_i + \boldsymbol{W}_i)\tilde{\boldsymbol{V}}_i^{(j-1)}$.

Self-attention layer to represent $[\boldsymbol{C}_i\tilde{\boldsymbol{V}}_i^{(j-1)}; \tilde{\boldsymbol{V}}_i^{(j-1)}]$: We will use two attention heads.

1. **The first attention head computes $\boldsymbol{C}_i\tilde{\boldsymbol{V}}_i^{(j-1)}$:** Matrices $\boldsymbol{W}_{query}, \boldsymbol{W}_{key}$ are set such that the attention score between an token embedding in first segment $[\boldsymbol{e}_r; \boldsymbol{C}_\ell\boldsymbol{e}_r]$ and a token embedding in second segment $\mathbf{o}_j$ is given by

   $$\langle \boldsymbol{e}_r, \tilde{\boldsymbol{V}}_i^{(j-1)} \rangle, \text{ if } \ell = i, \text{ and 0 otherwise.}$$

   for any $j \in [2, L/2+1], r \in [1, d]$. The condition requires the model to attend to $\boldsymbol{C}_i$ and ignore other in-context information. The condition can be set using the Context matrix index indicator vectors $\boldsymbol{l}_\ell$ which is present in each in-context information embedding.

   The attention between any two input sequence embedding $\mathbf{o}_j$ and $\mathbf{o}_{j'}$ is computed as 0s. The distinction between the attention scores of pairs of embeddings in second segment, $\mathbf{o}_j$

and $\mathbf{o}_{j'}$, v/s attention scores between a token embedding in second segment and a token embedding in first segment, $\mathbf{o}_j$ and $[\boldsymbol{e}_r; \boldsymbol{C}_\ell \boldsymbol{e}_r]$, can be done by using the segment indicator embeddings $\boldsymbol{g}_1$ and $\boldsymbol{g}_2$ used to differentiate token embeddings in first segment and the input sequence embedding vectors.

Matrix $\boldsymbol{W}_{value}$ is set such that the columns of each $\boldsymbol{C}_\ell$s are picked from the token embeddings in the first segment: $\{\{[\boldsymbol{e}_r; \boldsymbol{C}_\ell \boldsymbol{e}_r]\}_{r=1}^{n^2}\}_{\ell=1}^{d}$.

2. **The second attention head simply copies the input $\tilde{\boldsymbol{V}}_i^{(j-1)}$:** This can be done with an attention head that attends to itself at each position j and copies $\tilde{\boldsymbol{V}}_i^{(j-1)}$ to output.

The output of the two attention heads are simply added up. The output embeddings will now look as follows: `<THINK>`, $\{[\boldsymbol{C}_i \tilde{\boldsymbol{V}}_i^{(j-1)}; \tilde{\boldsymbol{V}}_i^{(j-1)}]\}_{j=1}^{L/2}$.

MLP to represent $\texttt{HardMax}(\boldsymbol{C}_i + \boldsymbol{W}_i)\tilde{\boldsymbol{V}}_i^{(j-1)}$: Our current token embeddings at any position j contain both $\boldsymbol{C}_i \tilde{\boldsymbol{V}}_i^{(j-1)}$ and $\tilde{\boldsymbol{V}}_i^{(j-1)}$. The first layer of MLP can be used to compute $(\boldsymbol{C}_i + \boldsymbol{W}_i)\tilde{\boldsymbol{V}}_i^{(j-1)}$ by setting the weights of the layer using $\boldsymbol{W}_i$. We simulate `HardMax` operation as follows:

$$\tilde{\mathbf{o}}_j / \left\|\tilde{\mathbf{o}}_j\right\|_2, \text{ where } \tilde{\mathbf{o}}_j = \text{GELU}((\boldsymbol{C}_i + \boldsymbol{W}_i)\tilde{\boldsymbol{V}}_i^{(j-1)})$$

The ℓ_2 normalization is equivalent to RMSnorm operation (Zhang & Sennrich, 2019). This is an approximation of the `HardMax` function, which are equivalent only under the following conditions: for each column j

1. either $\boldsymbol{C}_i^{(j)}$ or $\boldsymbol{W}_i^{(j)}$ are all 0s.
2. $\boldsymbol{C}_i^{(j)}$ and $\boldsymbol{W}_i^{(j)}$ are both one-hot vectors and they match at the corresponding activated dimension.

Algorithm 5 CIRCULAR SHIFT-MODULE: Self-attention and MLP layers for *Circular shift*

Require: Input embeddings (Token, Context matrix index indicator, Start and End Indicator, and Segment embeddings split into 2 segments) (Table 5). Important ones (for the current module) are

1. Token embeddings: First segment contains $\{[\boldsymbol{e}_j; \boldsymbol{C}_1\boldsymbol{e}_j]\}_{j=1}^{n^2}, \cdots, \{[\boldsymbol{e}_j; \boldsymbol{C}_d\boldsymbol{e}_j]\}_{j=1}^{n^2}$ and the second segment contains $[\texttt{Shift}(\boldsymbol{V}_{i-1}^{(1)}); \boldsymbol{0}], \cdots, [\texttt{Shift}(\boldsymbol{V}_{i-1}^{(L/2)}); \boldsymbol{0}], \boldsymbol{0}$
2. Start and End indicator: First segment contains all 0s and second segments contains $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$ at first and last embedding, while containing all 0s everywhere else.

Step a: Using a self-attention layer with 3 attention heads, change the token embeddings in second segment as <THINK>, $\{\mathbf{o}_j\}_{j=2}^{L/2+1}$ s.t. $\boldsymbol{o}_j$ has $(\mathbf{1}_n \otimes I_n)^\top \boldsymbol{V}_i^{(j-1)}$ in $[1, n]$ dimensions and $(I_n \otimes \mathbf{1}_n)\, \boldsymbol{V}_i^{(j\%L)}$ in $[n+1, 2n]$ dimensions. Primarily,

- Attention head 1: Computes attention score between any two positions p_1, p_2 as $a_{p_1,p_2} = 1$ iff $p_2 - p_1 = -1$ and 0 otherwise. Value matrix $\boldsymbol{W}_{value}$ is set such that for any input $\boldsymbol{x}$, the output of $\boldsymbol{W}_{value}\boldsymbol{x}$ is given by (for all $p_1 \in [1, n]$)

$$(\boldsymbol{W}_{value}\boldsymbol{x})_{p_1} = \sum_{j=0}^{n-1} x_{n\cdot j+p_1}.$$

- Attention head 2: Computes attention score between any two positions p_1, p_2 as $a_{p_1,p_2} = 1$ iff $p_2 - p_1 = 0$ and 0 otherwise. $\boldsymbol{W}_{value}$ is set such that for any input $\boldsymbol{x}$, the output of $\boldsymbol{W}_{value}\boldsymbol{x}$ is given by (for all $p_1 \in [1, n]$)

$$(\boldsymbol{W}_{value}\boldsymbol{x})_{p_1+n} = \sum_{j=1}^{n} x_{j+p_1 n-n}.$$

- Attention head 3: Computes attention score between any two positions p_1, p_2 as $a_{p_1,p_2} = 1$ iff $\boldsymbol{b}_2$ and $\boldsymbol{b}_1$ are present as at positions p_1 and p_2 respectively and 0 otherwise. $\boldsymbol{W}_{value}$ is set such that for any input $\boldsymbol{x}$, the output of $\boldsymbol{W}_{value}\boldsymbol{x}$ is given by (for all $p_1 \in [1, n]$)

$$(\boldsymbol{W}_{value}\boldsymbol{x})_{p_1+n} = \sum_{j=1}^{n} x_{j+p_1 n-n}.$$

Sum the output of the three heads.

Step b: Use MLP layer to change the token embeddings in second segment as <THINK>, $\{\mathbf{o}_j\}_{j=2}^{L/2+1}$ s.t. $\boldsymbol{o}_j$ has $\tilde{\boldsymbol{V}}_i^{(j-1)}$ with some small error. Primary computation at each position j is given as (for a large N)

$$\sqrt{2/\pi}N^2 \Big(\text{GELU}(\frac{1}{N}(\mathbf{1}_n \otimes I_n)^\top \boldsymbol{V}_i^{(j-1)} + \frac{1}{N}(I_n \otimes \mathbf{1}_n)\, \boldsymbol{V}_i^{(j\%L)}) \\ -\text{GELU}(\frac{1}{N}(\mathbf{1}_n \otimes I_n)^\top \boldsymbol{V}_i^{(j-1)}) - \text{GELU}(\frac{1}{N}(I_n \otimes \mathbf{1}_n)\, \boldsymbol{V}_i^{(j\%L)}) \Big),$$

which will return

$$\tilde{\boldsymbol{V}}_i^{(j-1)} + \mathcal{O}(N^{-4}) := (\mathbf{1}_n \otimes I_n)^\top \boldsymbol{V}_i^{(j-1)} \odot (I_n \otimes \mathbf{1}_n)\, \boldsymbol{V}_i^{(j\%L)} + \mathcal{O}(N^{-4})$$

Return the output embeddings (as given in Table 7).

Algorithm 6 TRANSLATE-MODULE: Self-attention and MLP layers for *Translate*

Require: We will require an index, and input embeddings as input:

- Index i (indicating index of the MLT-MODULE it is a part of),
- Embeddings (Token, Context matrix index indicator, Start and End Indicator, and Segment embeddings split into 2 segments) from the output of its preceding CIRCULAR SHIFT-MODULE (Table 7). Important ones (for the current module) are
    1. Token embeddings: First segment contains $\{[\boldsymbol{e}_j; \boldsymbol{C}_1\boldsymbol{e}_j]\}_{j=1}^{n^2}, \cdots, \{[\boldsymbol{e}_j; \boldsymbol{C}_d\boldsymbol{e}_j]\}_{j=1}^{n^2}$ and the second segment contains <THINK>, $[\tilde{\boldsymbol{V}}_i^{(1)}; \boldsymbol{0}], \cdots, [\tilde{\boldsymbol{V}}_i^{(L/2)}; \boldsymbol{0}]$
    2. Context matrix index indicator: First segment contains $\{\boldsymbol{l}_1\}_{[1,n^2]}, \{\boldsymbol{l}_2\}_{[1,n^2]}, \cdots \{\boldsymbol{l}_d\}_{[1,n^2]}$ and second segments contains all 0s vectors.

Step a: Using a self-attention layer, change token embeddings in the second segment as <THINK>, $\{[\boldsymbol{C}_i\tilde{\boldsymbol{V}}_i^{(j-1)}; \tilde{\boldsymbol{V}}_i^{(j-1)}]\}_{j=1}^{L/2}$.

- Primarily, the self-attention score between embeddings that contain a second segment token embedding $[\tilde{\boldsymbol{V}}_i^{(p_2)}; \boldsymbol{0}]$ and a first segment token embedding $[\boldsymbol{e}_{p_1}; \boldsymbol{C}_r\boldsymbol{e}_{p_1}]$ (for any $r \in [1, d]$, $p_1 \in [1, n^2], p_2 \in [1, L]$) is computed as

$$\langle \boldsymbol{e}_{p_1}, \tilde{\boldsymbol{V}}_i^{(p_2)} \rangle \cdot \langle \boldsymbol{l}_r, \boldsymbol{l}_i \rangle,$$

  where $\boldsymbol{l}_r$ is the corresponding context matrix index indicator for the first segment embedding under consideration and $\boldsymbol{l}_i$ is constructed using the index i.
- $\boldsymbol{C}_r\boldsymbol{e}_{p_1}$ is used as value vector from each first segment embeddings.

Step b: Using an MLP layer, change token embeddings in the second segment to contain <THINK>, $\{\mathbf{o}_j\}_{j=2}^{L/2+1}$, where

$$\mathbf{o}_j = \tilde{\mathbf{o}}_j / \left\|\tilde{\mathbf{o}}_j\right\|_2, \text{ where } \tilde{\mathbf{o}}_j = \text{GELU}((\boldsymbol{C}_i + \boldsymbol{W}_i)\tilde{\boldsymbol{V}}_i^{(j-1)})$$

Return the output embeddings (as given in Table 6).

I Additional Experiment Setups

I.1 Format of Training Text

Here we present examples of training data used for experiments in Section 3

```
INPUT:
"<|begin_of_text|> <|begin_of_text|> <|begin_of_text|> <|start_header_id|> user
<|end_header_id|>

You are performing a special translation task called language_task_2.  The subset of
dictionaries used are as follows:
Dictionary used from language 1 to language 2:  ;
Dictionary used from language 2 to language 3:  ;
Dictionary used from language 3 to language 4:  ;
Dictionary used from language 4 to language 5:  ;
Dictionary used from language 5 to language 6:;
Now please perform language_task_2 translation from the following sequence in language
1 to language 6.  Do not use code!  You must only reponse in the form:  "Sequence in
language 1:  [the sequence in language 1]; Sequence in language 2:  [the sequence
in language 2]; Sequence in language 3:  [the sequence in language 3]; Sequence in
language 4:  [the sequence in language 4]; Sequence in language 5:  [the sequence in
language 5]; Sequence in language 6:  [the sequence in language 6]".  The sequence you
need to translate from language 1 is:  C B E F E B D E C B C A H E F B C A D F G B D G
H E D E.<|eot_id|><|start_header_id|>assistant<|end_header_id|>"

LABEL:
"<|begin_of_text|>The translation result is:  Sequence in language 1:  C B E F E B D E
C B C A H E F B C A D F G B D G H E D E; Sequence in language 2:  <T> <T> <T> <T> <T>
<T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T>
<T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T>; Sequence in language 3:  <T> <T>
<T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T>
<T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T>; Sequence in language
4:  <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T>
<T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T>; Sequence
in language 5:  <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T>
<T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T>
<T> <T>; Sequence in language 6:  q o v t t u t o s u s s s o p p r o q s o q r q p s t
p;<|eot_id|>" ,
```

Figure 13: Input with completely masked context and internalized chain-of-thought. We use this data format to evaluate the model's capability on conducting translation without anything (useful information) in context.

INPUT:

"<|begin_of_text|> <|begin_of_text|> <|begin_of_text|> <|start_header_id|> user <|end_header_id|>

You are performing a special translation task called language_task_2. The subset of dictionaries used are as follows:
Dictionary used from language 1 to language 2:
D A -> N J; A D -> J I; B C -> O I; C C -> N N; H E -> P K; F E -> M L; G H -> L N; E G -> P J; A F -> K M; E C -> K P; B E -> K I; F D -> M N; E D -> P O; B A -> P L; B D -> I P; D G -> I I; D H -> I O; E F -> L M; H B -> J K; E A -> K J; A H -> L J; G C -> I K; F B -> P I; F G -> J O;
Dictionary used from language 2 to language 3:
L P -> V X; M K -> R W; L I -> X R; N O -> R R; M O -> U Q; I L -> W V; O K -> Q W; P M -> R U; J I -> V S; I M -> X W; O I -> Q U; N P -> R Q; I N -> Q X; N L -> R S; I J -> Q T; N J -> S T; L O -> U V; P J -> T X; J L -> Q R; P K -> W R; N M -> S U; M I -> Q S; P O -> S S; K K -> U U; P L -> X U;
Dictionary used from language 3 to language 4:
X U -> Y d; R W -> a c; T Q -> Y Y; U R -> Z Y; T X -> e d; W W -> e Y; U X -> f f; X R -> b Y; Q Q -> b c; S S -> c c; V U -> Z d; R S -> f a; V W -> Y e; R U -> f c; U S -> c a; U W -> c f; W X -> d Y; R Q -> c Z; V Q -> Z f; W S -> b d; V R -> a a; R X -> f Z; U Q -> d c; Q V -> d Z; S W -> Y b;
Dictionary used from language 4 to language 5:
c a -> m j; b Y -> n k; c f -> k m; c b -> l n; Z d -> g h; d c -> n g; b f -> i k; f Z -> i i; Z Y -> g l; Y a -> l l; b e -> l i; Y Y -> m g; d Z -> g g; d e -> k l; f c -> j g; f f -> k n; b Z -> n h; e a -> i m; c e -> j h; Y d -> i h; Y b -> h i; Y f -> h k; a Y -> k j; a a -> m k; c Y -> h m;
Dictionary used from language 5 to language 6:
k g -> p s; h h -> r q; j m -> o o; i h -> q s; l m -> p o; i j -> o q; l j -> q u; k m -> v u; g h -> p p; g n -> v p; n h -> s s; m m -> s o; m k -> v t; m i -> u t; h k -> t o; n k -> r u; j n -> t u; g l -> t s; j g -> v v; h g -> u s; n l -> s u; l k -> q o; h l -> t p; i i -> p q; k i -> r o; l h -> u p;
Now please perform language_task_2 translation from the following sequence in language 1 to language 6. Do not use code! You must only reponse in the form: "Sequence in language 1: [the sequence in language 1]; Sequence in language 2: [the sequence in language 2]; Sequence in language 3: [the sequence in language 3]; Sequence in language 4: [the sequence in language 4]; Sequence in language 5: [the sequence in language 5]; Sequence in language 6: [the sequence in language 6]". The sequence you need to translate from language 1 is: C B E F E B D E C B C A H E F B C A D F G B D G H E D E.<|eot_id|><|start_header_id|>assistant<|end_header_id|>"

LABEL:

"<|begin_of_text|>The translation result is: Sequence in language 1: C B E F E B D E C B C A H E F B C A D F G B D G H E D E; Sequence in language 2: K I M L I P K P O I L J L M O I J I J O I P L N P O K P * * * * * * * * * * * * *; Sequence in language 3: X W X R W R S S W V Q R U Q Q T Q T Q U X U R Q Q W W R * * * * * * * * * * * * * *; Sequence in language 4: d Y a c f a Y b Z f f c b c Y Y Y Y f f Z Y b c e Y f Z * * * * * * * * * * * * * *; Sequence in language 5: l l k m k j n h k n l n h m m g h k i i h i j h h k g h * * * * * * * * * * * * *; Sequence in language 6: q o v t t u t o s u s s s o p p r o q s o q r q p s t p;<|eot_id|>" ,

Figure 14: Input with complete context and explicit chain-of-thought. We use this data format at the begining of stage 1 training.

```
INPUT:
"<|begin_of_text|> <|begin_of_text|> <|begin_of_text|> <|start_header_id|> user
<|end_header_id|>

You are performing a special translation task called language_task_2.  The subset of
dictionaries used are as follows:
Dictionary used from language 1 to language 2:
D A -> N J; A D -> J I; B C -> O I; C C -> N N; H E -> P K; F E -> M L; G H -> L N; E G
-> P J; A F -> K M; E C -> K P; B E -> K I; F D -> M N; E D -> P O; B A -> P L; B D ->
I P; D G -> I I; D H -> I O; E F -> L M; H B -> J K; E A -> K J; A H -> L J; G C -> I
K; F B -> P I; F G -> J O;
Dictionary used from language 2 to language 3:
L P -> V X; M K -> R W; L I -> X R; N O -> R R; M O -> U Q; I L -> W V; O K -> Q W; P M
-> R U; J I -> V S; I M -> X W; O I -> Q U; N P -> R Q; I N -> Q X; N L -> R S; I J ->
Q T; N J -> S T; L O -> U V; P J -> T X; J L -> Q R; P K -> W R; N M -> S U; M I -> Q
S; P O -> S S; K K -> U U; P L -> X U;
Dictionary used from language 3 to language 4:
X U -> Y d; R W -> a c; T Q -> Y Y; U R -> Z Y; T X -> e d; W W -> e Y; U X -> f f; X R
-> b Y; Q Q -> b c; S S -> c c; V U -> Z d; R S -> f a; V W -> Y e; R U -> f c; U S ->
c a; U W -> c f; W X -> d Y; R Q -> c Z; V Q -> Z f; W S -> b d; V R -> a a; R X -> f
Z; U Q -> d c; Q V -> d Z; S W -> Y b;
Dictionary used from language 4 to language 5:
c a -> m j; b Y -> n k; c f -> k m; c b -> l n; Z d -> g h; d c -> n g; b f -> i k; f Z
-> i i; Z Y -> g l; Y a -> l l; b e -> l i; Y Y -> m g; d Z -> g g; d e -> k l; f c ->
j g; f f -> k n; b Z -> n h; e a -> i m; c e -> j h; Y d -> i h; Y b -> h i; Y f -> h
k; a Y -> k j; a a -> m k; c Y -> h m;
Dictionary used from language 5 to language 6:
k g -> p s; h h -> r q; j m -> o o; i h -> q s; l m -> p o; i j -> o q; l j -> q u; k m
-> v u; g h -> p p; g n -> v p; n h -> s s; m m -> s o; m k -> v t; m i -> u t; h k ->
t o; n k -> r u; j n -> t u; g l -> t s; j g -> v v; h g -> u s; n l -> s u; l k -> q
o; h l -> t p; i i -> p q; k i -> r o; l h -> u p;
Now please perform language_task_2 translation from the following sequence in language
1 to language 6.  Do not use code!  You must only reponse in the form:  "Sequence in
language 1:  [the sequence in language 1]; Sequence in language 2:  [the sequence
in language 2]; Sequence in language 3:  [the sequence in language 3]; Sequence in
language 4:  [the sequence in language 4]; Sequence in language 5:  [the sequence in
language 5]; Sequence in language 6:  [the sequence in language 6]".  The sequence you
need to translate from language 1 is:  C B E F E B D E C B C A H E F B C A D F G B D G
H E D E.<|eot_id|><|start_header_id|>assistant<|end_header_id|>"

LABEL:
"<|begin_of_text|>The translation result is:  Sequence in language 1:  C B E F E B D E
C B C A H E F B C A D F G B D G H E D E; Sequence in language 2:  <T> <T> <T> <T> <T>
<T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T>
<T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T>; Sequence in language 3:  <T> <T>
<T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T>
<T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T>; Sequence in language
4:  <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T>
<T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T>; Sequence
in language 5:  <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T>
<T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T> <T>
<T> <T>; Sequence in language 6:  q o v t t u t o s u s s s o p p r o q s o q r q p s t
p;<|eot_id|>" ,
```

Figure 15: Input with complete context and internalized chain-of-thought. We use this data format by the end of stage 1 training as well as the base format for stage 2 training (before dropout)