Prefix-Tuning for Arbitrary Output Sequences on Pretrained Transformers

Peter Cho-Ho Lam **Ziyi Wang** **Zirui Zhou**
Huawei Technologies Canada Co., Ltd.
cho.ho.lam@huawei.com ziyi.wang2@h-partners.com zirui.zhou@huawei.com
https://github.com/wzii/prefix_tuning

Abstract

Prefix-tuning is a parameter-efficient fine-tuning method for large language models (LLMs) that steers model outputs by prepending task-specific prefix representations to the input sequence. Despite its practical success, the theoretical understanding of the expressiveness of prefixes remains incomplete. In particular, the following question remains open: *Given a pretrained autoregressive Transformer, for any input-output sequence pair* $(\mathtt{x}, \mathtt{y})$*, does there always exist a prefix* $\mathcal{S}$ *such that prepending* $\mathcal{S}$ *to* $\mathtt{x}$ *induces the model to generate the target output* $\mathtt{y}$*?* In this work, we prove that under the assumption that $\mathtt{y}$ is in the *range* of the pretrained Transformer, such a prefix $\mathcal{S}$ exists and can be explicitly computed by solving multiple linear systems. We further show that this assumption is necessary, in the sense that such a prefix may not exist when the assumption fails, thereby providing a complete answer to the above question. Our analysis is constructive and the obtained theoretical results hold under fairly general assumptions on model architecture. Extensive experiments on GPT-2 and Qwen3-4B demonstrate that our algorithm produces prefixes with strong control over model outputs.

1 Introduction

Large language models (LLMs) have become a major driving force behind modern technological advancement. Their effectiveness has been demonstrated across a wide range of applications, including software engineering, scientific research, healthcare, finance, education, and legal domains (Bommasani et al., 2021; Chen et al., 2024; Yang et al., 2025b). However, most LLMs are trained as general-purpose models, which can limit their performance on domain-specific tasks that require specialized knowledge or terminology. Empirical studies consistently show that models fine-tuned for a target domain outperform general-purpose models on domain-specific benchmarks and task-relevant metrics (Beltagy et al., 2019; Lee et al., 2019; Chalkidis et al., 2020; Gururangan et al., 2020).

Although fine-tuning is substantially cheaper than training a model from scratch, full fine-tuning remains computationally expensive due to the need to update and store all model parameters, making it impractical for many users and organizations (Strubell et al., 2019). These limitations have motivated growing interest in parameter-efficient fine-tuning methods, which aim to reduce training costs while retaining performance (He et al., 2021; Ding et al., 2023; Xu et al., 2023; Han et al., 2024). Parameter-efficient fine-tuning techniques update a limited subset of parameters or introduce lightweight modules, achieving performance comparable to full fine-tuning at a fraction of the computational cost. (Houlsby et al., 2019; Ben Zaken et al., 2022; Karimi Mahabadi et al., 2021; Hu et al., 2022; Lester et al., 2021; Li and Liang, 2021; Lian et al., 2022).

Prefix-tuning, a prominent parameter-efficient fine-tuning technique, adapts new tasks by learning a small set of continuous task-specific vectors, or prefixes, which are prepended to the model's hidden states while keeping the original parameters frozen (Li and Liang, 2021). By optimizing only these

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

prefixes, prefix-tuning avoids the high computational and memory costs of full fine-tuning. This efficiency makes prefix-tuning a practical and scalable approach for adapting large models to diverse tasks and deployment settings (Liu et al., 2022).

Despite its popularity, the theoretical foundations of prefix-tuning remain poorly understood. A desirable property of prefix-tuning is **expressiveness**: Can one induce arbitrary desired behaviors in a Transformer solely by appending prefixes? Formally, this can be framed as the following problem:

Problem 1. Given a pretrained autoregressive Transformer, for any input-output sequence pair $(\mathtt{x}, \mathtt{y})$, does there always exist a prefix $\mathcal{S}$ such that prepending $\mathcal{S}$ to $\mathtt{x}$ induces the model to generate the target output $\mathtt{y}$?

Several existing works have studied the expressiveness of prefix-tuning (Wang et al., 2023; Petrov et al., 2024b; Hu et al., 2025; Liang et al., 2025); however, their analyses suffer from three notable limitations. (i) **Model mismatch:** Their results are established for hand-crafted Transformers with preassigned weights and simplified architectures, rather than pretrained models specified by the user. (ii) **Generation mismatch:** A number of works study the approximation of arbitrary sequence-to-sequence functions (Petrov et al., 2024b; Hu et al., 2025), whereas in practice language models generate outputs autoregressively through repeated queries to the Transformer. This autoregressive setting imposes significantly stronger structural constraints than a one-shot sequence mapping. [1] (iii) **Scaling assumptions:** The resulting guarantees typically rely on impractical assumptions regarding model width, depth, or prefix length (Petrov et al., 2024b; Liang et al., 2025). Due to these limitations, expressiveness results have never been empirically demonstrated on modern LLMs.

In this work, we address this gap by establishing an expressiveness result for pretrained autoregressive Transformers with general architectural components, where the model is fixed a priori and cannot be modified. Specifically, we prove the existence of $\mathcal{S}$ in Problem 1 under the assumption that $\mathtt{y}$ lies in the *range* of the Transformer; that is, $\mathtt{y}$ is a sequence reachable through autoregressive generation. With minor adaptations, our proof applies to nearly all common modern LLM architectures, including GPT-2 (Radford et al., 2019), which features multi-head attentions, residual connections, and positional encodings. The statement of our main theorem (Theorem 2.1) is theoretically tight: removing any assumption leads to counterexamples, implying that our result is best possible and that Problem 1 is fully resolved.

Our proof also provides a practical algorithm to compute such a prefix. The computation is training-free and amounts to solving a fixed and small number of low-dimensional systems of linear equations, which can be done efficiently with minimal hardware requirements. The construction of our linear-system formulation is based on attention matching, a framework that matches attention outputs across different autoregressive decoding trajectories. We note that a related attention matching framework was independently developed by Zweiger et al. (2026) for KV cache compaction. We discuss the common underlying principles and the differences between the two frameworks in Appendix H.

To validate our theory and algorithm, we conducted extensive experiments using GPT-2 (Radford et al., 2019). Our results demonstrate a high success rate in achieving complete control of output sequences through prefixes. To the best of our knowledge, this is the first instance in which a theoretical expressiveness result has been empirically demonstrated and validated in a modern LLM. We also achieved strong results on Qwen3-4B (Yang et al., 2025a), which features grouped-query attention (GQA) (Ainslie et al., 2023).

Below is a summary of our contributions:

- We resolve a long-standing open question on the expressiveness of prefix-tuning in practical LLMs and fine-tuning scenarios by providing and proving the necessary and sufficient conditions on the existence of a prefix that can generate essentially any arbitrary output.
- We provide an efficient, training-free algorithm for constructing such prefixes and validate it through extensive experiments on GPT-2 and Qwen3-4B, showing that the resulting prefixes achieve a high success rate in fully controlling output sequences.

[1] For a more thorough discussion on why sequence-to-sequence function is incompatible with autoregressive generation, see our Appendix A.

2 Background

2.1 Transformer Notations

Let $\mathcal{V}$ be a finite vocabulary, and let $|\mathcal{V}|$ be its cardinality. A token $\mathtt{x}$ is an element of $\mathcal{V}$. A sequence of tokens of length p is denoted by $(\mathtt{x}_1, \ldots, \mathtt{x}_p)$.

Fix an embedding dimension d_e. The embedding map is a function $\mathbf{Emb} : \mathcal{V} \to \mathbb{R}^{d_e}$ applied element-wise to a token sequence. In practice, **Emb** may also incorporate positional information. Given a sequence $(\mathtt{x}_1, \ldots, \mathtt{x}_p)$, we define the embedded input as $\mathbf{x}_i^{(0)} = \mathbf{Emb}(\mathtt{x}_i)$ for $i = 1, \ldots, p$.

An L-layer Transformer $\mathcal{T}$ is a parameterized mapping $\mathcal{T} : \bigcup_{p\geq 1}(\mathbb{R}^{d_e})^p \longrightarrow \bigcup_{p\geq 1}(\mathbb{R}^{|\mathcal{V}|})^p$ in the form $\mathcal{T} = \mathcal{F}^{(L)} \circ \cdots \circ \mathcal{F}^{(1)}$ where, for each layer $l = 1, \ldots, L$, $\mathcal{F}^{(l)} = \mathbf{MLP}^{(l)} \circ \mathbf{MHA}^{(l)}$. Here, $\mathbf{MHA}^{(l)}$ is a multi-head attention module and $\mathbf{MLP}^{(l)}$ is a position-wise multi-layer perceptrons, each with layer-specific parameters.

Given an output of $\mathcal{T}$ in $(\mathbb{R}^{|\mathcal{V}|})^p$, a deterministic next-token prediction may be obtained by applying the **argmax** operator to the representation **at the final position**. By a slight abuse of notation, we will also use $\mathcal{T}$ to denote the composition of the Transformer with this **argmax** operation, in which case its output space is $\mathcal{V}$. The generated token is then appended to the input sequence and fed back into $\mathcal{T}$ to produce subsequent tokens iteratively. This procedure, known as **autoregressive decoding**, is the standard mode of operation for LLMs and the default decoding method considered in this paper.

Given the embedded input $(\mathbf{x}_1^{(0)}, \ldots, \mathbf{x}_p^{(0)})$, the layer-wise representations are defined recursively by

$$(\mathbf{x}_1^{(l)}, \ldots, \mathbf{x}_p^{(l)}) = \mathcal{F}^{(l)}(\mathbf{x}_1^{(l-1)}, \ldots, \mathbf{x}_p^{(l-1)})$$

for $l = 1, \ldots, L$. For notational simplicity, we omit the layer index and use $\mathbf{x}_i$ for a generic token representation. Note that for all $l \geq 1$, the representation $\mathbf{x}_i^{(l)}$ depends on the previous tokens through the attention mechanism. This dependence is also suppressed in the notation.

We now describe a single multi-head attention module $\mathcal{A}$ in detail. Let H denote the number of attention heads. For $h \in \{1, \ldots, H\}$, let $\mathbf{W}_Q^h, \mathbf{W}_K^h \in \mathbb{R}^{d_{\text{hidden}} \times d_{\text{in}}}$, $\mathbf{W}_V^h \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ be the query, key, and value matrices, respectively. Given an input sequence of vectors $(\mathbf{x}_1, \ldots, \mathbf{x}_p)$ with $\mathbf{x}_i \in \mathbb{R}^{d_{\text{in}}}$, the attention matrix for head h is $\mathbf{A}^h \in \mathbb{R}^p \times \mathbb{R}^p$, whose entries are given by

$$\mathbf{A}_{i,j}^h = \frac{\exp(\frac{T}{\sqrt{d_{\text{hidden}}}}(\mathbf{W}_Q^h \mathbf{x}_i)^\top (\mathbf{W}_K^h \mathbf{x}_j))}{\sum_{r=1}^p \exp(\frac{T}{\sqrt{d_{\text{hidden}}}}(\mathbf{W}_Q^h \mathbf{x}_i)^\top (\mathbf{W}_K^h \mathbf{x}_r))},$$

where $T > 0$ is an inverse temperature parameter. Then, the **MHA** output at position i is given by:[2]

$$\mathbf{a}_i = \sum_{h=1}^{H} \sum_{j=1}^{p} \mathbf{A}_{i,j}^h \mathbf{W}_V^h \mathbf{x}_j.$$

We also write this as $\mathcal{A}((\mathbf{x}_1, \ldots, \mathbf{x}_p)) = (\mathbf{a}_1, \ldots, \mathbf{a}_p)$. For simplicity, we omit the potential bias term in our expressions; see Appendix E for treatment with biases.

2.2 Prefix-Tuning

In the literature and in practical implementations, there exist different levels of generality in how prefixes are defined. Here, we adopt a formulation that encompasses most existing variants while remaining mathematically concise.

Formally, let $\mathcal{T}$ be an L-layer Transformer. A prefix $\mathcal{S}$ with respect to $\mathcal{T}$ is a set of layer-specific sequences of vectors

$$\mathcal{S} = \{(\mathbf{s}_1^{(l)}, \ldots, \mathbf{s}_m^{(l)}) \in (\mathbb{R}^{d_{\text{in}}})^m \mid l = 1, \ldots, L\}.$$

For the l-th layer, the input to the multi-head attention module $\mathcal{A}^l$ is then

$$(\mathbf{s}_1^{(l)}, \ldots, \mathbf{s}_m^{(l)}, \mathbf{x}_1^{(l-1)}, \ldots, \mathbf{x}_p^{(l-1)}).$$

[2] Alternatively, concatenation can be used instead of summation. It does not affect our theoretical analysis.

Many existing works introduce additional flexibilities in the definition of the prefix. For example, some allow the vectors $\mathbf{s}_i^{(l)}$ to be specific to each attention head (Poth et al., 2023), while others allow separate prefixes to be applied when projecting to the key and value matrices,[3] $\mathbf{W}_K^h$ and $\mathbf{W}_V^h$ (Petrov et al., 2024b). For the purposes of this work, we initially adopt the simpler, layer-wise prefix formulation and will generalize to these more flexible definitions when required in our theoretical analysis. We refer to these constructions as **generalized** prefixes in our main results. In practice, the choice of prefix variant often has little impact on memory usage or inference speed as they can be implemented in the form of KV caches.

2.3 Main Conjecture

Our goal is to establish the expressiveness of prefix-tuning in a practical large language model setting, where outputs are generated autoregressively and both the pretrained Transformer and the target dataset are fixed in advance. Before doing so, however, it is important to clarify a fundamental limitation: in general, for a given Transformer $\mathcal{T}$, not all sequences over the vocabulary can be generated autoregressively. For instance, it is straightforward to construct a Transformer for which a particular token never attains the maximum logit and thus is never produced. Even if a Transformer is surjective at the level of individual tokens—meaning that every vocabulary can be generated at some position—this does not imply that every sequence of tokens can be realized.

Consequently, to meaningfully discuss the expressiveness of prefix-tuning, we restrict attention to output sequences that are realizable by the underlying Transformer. Specifically, for a target output sequence $\mathtt{y} = (\mathtt{y}_1, \ldots, \mathtt{y}_k)$, we always assume that there exists some source input sequence $\mathtt{x}' = (\mathtt{x}'_1, \ldots, \mathtt{x}'_p)$ such that $\mathcal{T}$ produces $(\mathtt{y}_1, \ldots, \mathtt{y}_k)$ when queried autoregressively on this input. In other words, $\mathtt{y}$ is in the *range* of $\mathcal{T}$. This assumption is consistent with the findings of (Petrov et al., 2024a), which show that a prefix-tuned Transformer cannot exhibit behaviors that are not already present in the original model. In what follows, for a prefix $\mathcal{S}$, we write $\mathcal{T}_\mathcal{S}$ to denote the Transformer obtained by applying the prefix $\mathcal{S}$ to $\mathcal{T}$. We now state our main problem, which is a precise formulation of Problem 1:

Problem 2. Let $\mathcal{T}$ be a Transformer and $\mathtt{y} = (\mathtt{y}_1, \ldots, \mathtt{y}_k)$ be a sequence of tokens for which there exists a sequence of tokens $\mathtt{x}' = (\mathtt{x}'_1, \ldots, \mathtt{x}'_p)$ such that

$$\begin{aligned} \mathcal{T}((\mathtt{x}'_1, \ldots, \mathtt{x}'_p)) &= \mathtt{y}_1, \\ \mathcal{T}((\mathtt{x}'_1, \ldots, \mathtt{x}'_p, \mathtt{y}_1)) &= \mathtt{y}_2, \\ &\vdots \\ \mathcal{T}((\mathtt{x}'_1, \ldots, \mathtt{x}'_p, \mathtt{y}_1, \mathtt{y}_2, \ldots, \mathtt{y}_{k-1})) &= \mathtt{y}_k. \end{aligned} \tag{1}$$

Then, for any sequence of tokens $\mathtt{x} = (\mathtt{x}_1, \ldots, \mathtt{x}_n)$, does there always exist a prefix $\mathcal{S}$ of $\mathcal{T}$ such that

$$\begin{aligned} \mathcal{T}_\mathcal{S}((\mathtt{x}_1, \ldots, \mathtt{x}_n)) &= \mathtt{y}_1, \\ \mathcal{T}_\mathcal{S}((\mathtt{x}_1, \ldots, \mathtt{x}_n, \mathtt{y}_1)) &= \mathtt{y}_2, \\ &\vdots \\ \mathcal{T}_\mathcal{S}((\mathtt{x}_1, \ldots, \mathtt{x}_n, \mathtt{y}_1, \mathtt{y}_2, \ldots, \mathtt{y}_{k-1})) &= \mathtt{y}_k? \end{aligned} \tag{2}$$

We refer to $(\mathtt{y}_1, \ldots, \mathtt{y}_k)$ as the **target output**, $(\mathtt{x}'_1, \ldots, \mathtt{x}'_p)$ as the **source input**, and $(\mathtt{x}_1, \ldots, \mathtt{x}_n)$ as the **user input**. The conclusion of this problem is extremely strong: it asserts that for an arbitrary user input, there exists a prefix that steers the model's output towards the target sequence, even if it is not related to the user input. Our main result resolves this problem under fairly general conditions.

Theorem 2.1. *There exists a (generalized) prefix that satisfies* (2) *if there exists a source input* $\mathtt{x}'$ *for* $\mathtt{y}$ *and if condition* (10) *holds at every attention head. If* $\mathcal{T}$ *has residual connections and positional encodings, then the conditions* $p = n$ *and* $\mathtt{x}_n = \mathtt{x}'_p$ *are also required.*

Condition (10) is primarily used to facilitate certain linear-algebraic operations in our proof and is never found to be violated in our experiments. Beyond this, Theorem 2.1 is best possible, as the

[3] Technically, multiplying with $\mathbf{W}_Q^h$ is not necessary for inference.

source input assumption cannot be removed. For the case of residual connections and positional encodings, the statement is also best possible as counterexamples exist when the assumptions $p = n$ and $\mathtt{x}_n = \mathtt{x}'_p$ are removed (see Appendix B). Therefore, Theorem 2.1 provides a complete answer to Problem 2.

The requirement $\mathtt{x}_n = \mathtt{x}'_p$ may appear restrictive at first glance, but it is almost automatically satisfied in practice: in chatbot-style interactions, user inputs are typically terminated by a special end-of-conversation token inserted by the system, which serves as the common terminal token required here. Finally, the condition $p = n$ is introduced mainly to mitigate the effects of positional encodings. Since positional encodings typically induce only small perturbations to embedding vectors, this requirement is not essential in real applications. This claim is supported by our additional experiment in Appendix C.

Note that Problem 2 corresponds to the fine-tuning setting with a single input-output pair. While it is natural to consider extending this formulation to larger datasets, such a generalization does not hold in general. In fact, even full fine-tuning may fail in this regime:

Theorem 2.2. *There exists a fine-tuning dataset of two elements that cannot be realized even for full fine-tuning.*

In other words, Theorem 2.1 is also best possible with respect to the dataset size. Note that this theorem does not undermine the viability of prefix-tuning (or full fine-tuning). The construction in Theorem 2.2 and its variants rely on fine-tuning datasets that exhibit contradictory behavior, either internally or with respect to the pretrained model, which is not representative of realistic fine-tuning scenarios. It would be of independent interest to formalize a notion of consistency and extend our results to larger, consistent datasets. For further discussion, see Appendix B.

3 Attention Matching

In this section, we outline our strategy for constructing a prefix that satisfies (2). In principle, (2) may be viewed as a system of high-dimensional nonlinear equations, for which general-purpose solvers exist. However, directly applying such methods faces several significant challenges: the dimensionality of the system is extremely large; the underlying Transformer may be non-differentiable, rendering gradient-based approaches inapplicable; and, most importantly, there is no a priori guarantee that a solution exists.

To address these difficulties, we develop an **attention matching** framework for prefix construction, which aims to match the attention outputs in (1) and (2). The central idea of this framework is to formulate and solve a collection of **matching equations**, consisting of one system of equations per attention head. By allowing the key and value prefixes to be decoupled, these matching equations admit a linear formulation. As a result, they can be solved efficiently using standard linear algebraic techniques.

We formally introduce the matching equations in Section 3.1 and present their linear formulations in Section 3.2. The complete proof of Theorem 2.1 based on the attention matching framework is deferred to Appendix D.

3.1 Matching Equations

Consider the decoding stage corresponding to the source input $(\mathtt{x}'_1, \dots, \mathtt{x}'_p)$ and let $\mathcal{A}$ denote a multi-head attention module of the Transformer $\mathcal{T}$. During the autoregressive generation of the output tokens $\mathtt{y}_1, \dots, \mathtt{y}_k$, the following computations are performed by $\mathcal{A}$:

$$\begin{aligned} \mathcal{A}((\mathbf{x}'_1, \dots, \mathbf{x}'_p)) &= (\mathbf{a}_1, \dots, \mathbf{a}_p), \\ \mathcal{A}((\mathbf{x}'_1, \dots, \mathbf{x}'_p, \mathbf{y}_1)) &= (\mathbf{a}_1, \dots, \mathbf{a}_p, \mathbf{a}_{p+1}), \\ &\vdots \\ \mathcal{A}((\mathbf{x}'_1, \dots, \mathbf{x}'_p, \mathbf{y}_1, \dots, \mathbf{y}_{k-1})) &= (\mathbf{a}_1, \dots, \mathbf{a}_p, \mathbf{a}_{p+1}, \dots, \mathbf{a}_{p+k-1}). \end{aligned} \tag{3}$$

These computations occur sequentially at each step of the generation process of $\mathtt{y}_1, \dots, \mathtt{y}_k$.

Now, consider an arbitrary user input token sequence $(\mathtt{x}_1, \ldots, \mathtt{x}_n)$, together with its corresponding embedding vectors $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ immediately before entering $\mathcal{A}$. Our objective is to construct a prefix consisting of m vectors $(\mathbf{s}_1, \ldots, \mathbf{s}_m)$, where m is a hyperparameter, such that

$$\begin{aligned} \mathcal{A}((\mathbf{s}_1, \ldots, \mathbf{s}_m, \mathbf{x}_1, \ldots, \mathbf{x}_n)) &= (\ldots, \mathbf{a}_p), \\ \mathcal{A}((\mathbf{s}_1, \ldots, \mathbf{s}_m, \mathbf{x}_1, \ldots, \mathbf{x}_n, \mathbf{y}_1)) &= (\ldots, \mathbf{a}_p, \mathbf{a}_{p+1}), \\ &\vdots \\ \mathcal{A}((\mathbf{s}_1, \ldots, \mathbf{s}_m, \mathbf{x}_1, \ldots, \mathbf{x}_n, \mathbf{y}_1, \ldots, \mathbf{y}_{k-1})) &= (\ldots, \mathbf{a}_p, \mathbf{a}_{p+1}, \ldots, \mathbf{a}_{p+k-1}) \end{aligned} \tag{4}$$

holds. Here we omit the first $m + n - 1$ outputs, as they play no role in determining subsequent token predictions.

Equivalently, our goal is to ensure that, when generating the first k output tokens using the user input augmented with the prefix, the attention outputs produced by $\mathcal{A}$ match those obtained when decoding from the source input. In other words, the outputs of $\mathcal{A}$ in the two runs—one conditioned on the source input and the other on the user input with prefix—are identical for all components that influence next-token prediction. If this condition is satisfied, then both runs will necessarily produce the same sequence of predictions for the first k tokens, namely $(\mathtt{y}_1, \ldots, \mathtt{y}_k)$.[4]

The attention output in (3) can be written explicitly as

$$\begin{aligned} \mathbf{a}_{p+t} &= \sum_{h=1}^{H} \left(\sum_{i=1}^{p} \mathbf{A}^h_{t,i} \mathbf{W}^h_V \mathbf{x}'_i + \sum_{i=1}^{t} \mathbf{A}^h_{t,p+i} \mathbf{W}^h_V \mathbf{y}_i \right) \\ &= \sum_{h=1}^{H} \mathbf{W}^h_V \frac{\sum_{i=1}^{p} E_h(\mathbf{y}_t, \mathbf{x}'_i)\mathbf{x}'_i + \sum_{i=1}^{t} E_h(\mathbf{y}_t, \mathbf{y}_i)\mathbf{y}_i}{\sum_{i=1}^{p} E_h(\mathbf{y}_t, \mathbf{x}'_i) + \sum_{i=1}^{t} E_h(\mathbf{y}_t, \mathbf{y}_i)} \end{aligned}$$

for $t = 0, \ldots, k-1$ where

$$E_h(\mathbf{x}, \mathbf{y}) = \exp\left(\frac{T}{\sqrt{d_{\text{hidden}}}} (\mathbf{W}^h_Q \mathbf{x})^\top (\mathbf{W}^h_K \mathbf{y})\right), \tag{5}$$

and we set $\mathbf{y}_0 = \mathbf{x}'_p$. When $t = 0$, the second summation in both the numerator and denominator is interpreted as 0. The corresponding attention expression appearing in (4) is

$$\sum_{h=1}^{H} \mathbf{W}^h_V \frac{\sum_{j=1}^{m} E_h(\mathbf{y}'_t, \mathbf{s}_j)\mathbf{s}_j + \sum_{i=1}^{n} E_h(\mathbf{y}'_t, \mathbf{x}_i)\mathbf{x}_i + \sum_{i=1}^{t} E_h(\mathbf{y}'_t, \mathbf{y}_i)\mathbf{y}_i}{\sum_{j=1}^{m} E_h(\mathbf{y}'_t, \mathbf{s}_j) + \sum_{i=1}^{n} E_h(\mathbf{y}'_t, \mathbf{x}_i) + \sum_{i=1}^{t} E_h(\mathbf{y}'_t, \mathbf{y}_i)} = \sum_{h=1}^{H} \mathbf{W}^h_V \frac{A_h(\mathbf{x}, \mathbf{y}, \mathbf{s})}{B_h(\mathbf{x}, \mathbf{y}, \mathbf{s})} \tag{6}$$

where A_h and B_h are used to represent the numerator and denominator respectively, with $\mathbf{y}'_0 = \mathbf{x}_n, \mathbf{y}'_t = \mathbf{y}_t$ for all $t \geq 1$. As discussed above, our objective is to enforce equality between the quantity in (6) and the corresponding attention output $\mathbf{a}_{p+t}$ obtained from the source run.

To simplify the analysis, we first eliminate the summation over attention heads by allowing the prefix vectors $\mathbf{s}_j$ to be head-dependent. In addition, we remove the multiplication by the value matrix $\mathbf{W}^h_V$ by imposing the following, technically stronger, sufficient condition:

$$\frac{A_h(\mathbf{x}, \mathbf{y}, \mathbf{s})}{B_h(\mathbf{x}, \mathbf{y}, \mathbf{s})} = \frac{\sum_{i=1}^{p} E_h(\mathbf{y}_t, \mathbf{x}'_i)\mathbf{x}'_i + \sum_{i=1}^{t} E_h(\mathbf{y}_t, \mathbf{y}_i)\mathbf{y}_i}{\sum_{i=1}^{p} E_h(\mathbf{y}_t, \mathbf{x}'_i) + \sum_{i=1}^{t} E_h(\mathbf{y}_t, \mathbf{y}_i)}. \tag{7}$$

We further simplify the quotients appearing on the left-hand side by adopting an override strategy. Specifically, once a candidate prefix vector $\mathbf{s}_j$ is identified, it can be repeated arbitrarily many times in the prefix so that its contribution dominates the corresponding numerator and denominator in (7). Consequently, it suffices to find $\mathbf{s}_j$ such that, for all $t = 0, \ldots, k-1$,

$$\frac{\sum_{j=1}^{m} E_h(\mathbf{y}'_t, \mathbf{s}_j)\mathbf{s}_j}{\sum_{j=1}^{m} E_h(\mathbf{y}'_t, \mathbf{s}_j)} = \frac{\sum_{i=1}^{p} E_h(\mathbf{y}_t, \mathbf{x}'_i)\mathbf{x}'_i + \sum_{i=1}^{t} E_h(\mathbf{y}_t, \mathbf{y}_i)\mathbf{y}_i}{\sum_{i=1}^{p} E_h(\mathbf{y}_t, \mathbf{x}'_i) + \sum_{i=1}^{t} E_h(\mathbf{y}_t, \mathbf{y}_i)} \tag{8}$$

holds. Once such vectors $\mathbf{s}_j$ are obtained, repeated insertion allows (7) to be satisfied up to an arbitrarily small error. Since $\mathcal{T}$ is continuous, this approximation is sufficient for our purposes.

Notably, the embedding vectors $\mathbf{x}_i$ of the user input do not appear in (8). As a result, the resolution of (8) for different attention heads is completely decoupled and can be carried out in parallel.

[4]There are some nuisances to this discussion when residual connections are present. They will be discussed in Appendix D.

3.2 Converting into Linear Equations

Solving (8) constitutes the central technical challenge in our work. To address this, we apply the split-prefix assumption and seek $\mathbf{s}$ and $\tilde{\mathbf{s}}$ so that

$$\frac{\sum_{j=1}^{m} E_h(\mathbf{y}'_t, \mathbf{s}_j)\tilde{\mathbf{s}}_j}{\sum_{j=1}^{m} E_h(\mathbf{y}'_t, \mathbf{s}_j)} = \frac{\sum_{i=1}^{p} E_h(\mathbf{y}_t, \mathbf{x}'_i)\mathbf{x}'_i + \sum_{i=1}^{t} E_h(\mathbf{y}_t, \mathbf{y}_i)\mathbf{y}_i}{\sum_{i=1}^{p} E_h(\mathbf{y}_t, \mathbf{x}'_i) + \sum_{i=1}^{t} E_h(\mathbf{y}_t, \mathbf{y}_i)} \tag{9}$$

Since the target attention outputs are linear combinations of the source embeddings of $\{\mathbf{x}'_1, \ldots, \mathbf{x}'_p\}$ and the generated outputs $\{\mathbf{y}_1, \ldots, \mathbf{y}_{k-1}\}$, it is natural to construct each $\tilde{\mathbf{s}}_j$ as a linear combination of these vectors and attempt to match their coefficients on both sides. Concretely, for each j we assume

$$\tilde{\mathbf{s}}_j = \sum_{i=1}^{p} \lambda_{j,i}\mathbf{x}'_i + \sum_{i=1}^{k-1} \mu_{j,i}\mathbf{y}_i$$

so that each $\tilde{\mathbf{s}}_j$ introduces $p + k - 1$ variables. With m prefix vectors in total, this introduces $m(p + k - 1)$ variables. It now suffices to solve the set of equations $SV = [X|Y]$ where we have the $k \times m$ coefficient matrix

$$S = \begin{bmatrix} \frac{E_h(\mathbf{x}_n, \mathbf{s}_1)}{\sum_{j=1}^{m} E_h(\mathbf{x}_n, \mathbf{s}_j)} & \cdots & \frac{E_h(\mathbf{x}_n, \mathbf{s}_m)}{\sum_{j=1}^{m} E_h(\mathbf{x}_n, \mathbf{s}_j)} \\ \vdots & \ddots & \vdots \\ \frac{E_h(\mathbf{y}_{k-1}, \mathbf{s}_1)}{\sum_{j=1}^{m} E_h(\mathbf{y}_{k-1}, \mathbf{s}_j)} & \cdots & \frac{E_h(\mathbf{y}_{k-1}, \mathbf{s}_m)}{\sum_{j=1}^{m} E_h(\mathbf{y}_{k-1}, \mathbf{s}_j)} \end{bmatrix},$$

$m \times (p + k - 1)$ variable matrix

$$V = \begin{bmatrix} \lambda_{1,1} & \cdots & \lambda_{1,p} & \mu_{1,1} & \cdots & \mu_{1,k-1} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \lambda_{m,1} & \cdots & \lambda_{m,p} & \mu_{m,1} & \cdots & \mu_{m,k-1} \end{bmatrix},$$

$k \times p$ target matrix $X = (X_{t,\ell})_{0 \le t \le k-1, 1 \le \ell \le p}$, $k \times (k-1)$ target matrix $Y = (Y_{t,\ell})_{0 \le t \le k-1, 1 \le \ell \le k-1}$, in which their entries are given by

$$X_{t,\ell} = \frac{E_h(\mathbf{y}_t, \mathbf{x}'_\ell)}{\sum_{i=1}^{p} E_h(\mathbf{y}_t, \mathbf{x}'_i) + \sum_{i=1}^{t} E_h(\mathbf{y}_t, \mathbf{y}_i)}, Y_{t,\ell} = \frac{E_h(\mathbf{y}_t, \mathbf{y}_j)}{\sum_{i=1}^{p} E_h(\mathbf{y}_t, \mathbf{x}'_i) + \sum_{i=1}^{t} E_h(\mathbf{y}_t, \mathbf{y}_i)},$$

with the extra condition that $Y_{t,\ell} = 0$ if $t < \ell$. This corresponds to $p + k - 1$ linear systems of m variables and k equations each, sharing the same $k \times m$ coefficient matrix. Compared to the original nonlinear formulation, the dimensionality depends linearly on sequence length rather than embedding dimension, making the system tractable.

Intuitively, when m is large enough, $\mathbf{s}_j$ can be chosen so that S has rank k. Consequently, these $p + k - 1$ linear systems admit a solution regardless of what is in X and Y. There are some obvious counterexamples though: if $\mathbf{y}_i = \mathbf{y}_j$ for some i, j, then we will always have two identical rows in S and it will never reach rank k. This turns out to be the only obstruction: If we assume that there exists $\mathbf{s}$ such that

$$E_h(\mathbf{x}_n, \mathbf{s}), E_h(\mathbf{y}_1, \mathbf{s}), \ldots, E_h(\mathbf{y}_{k-1}, \mathbf{s}) \text{ are all distinct,} \tag{10}$$

then we can always find $\mathbf{s}_j$ such that S is rank k. The proof will be provided in Appendix D.

In our experiments, condition (10) is never violated, even when there are repeated tokens in $\mathtt{y}$ or $\mathtt{x}_n$. This is due to the presence of positional encodings: even if two tokens are identical, they receive slightly different embeddings before entering $\mathcal{T}$.

4 Algorithms

We summarize our procedure below. Given a fixed Transformer $\mathcal{T}$, a source input $(\mathtt{x}'_1, \ldots, \mathtt{x}'_p)$, a desired target output $(\mathtt{y}_1, \ldots, \mathtt{y}_k)$, and the hyperparameters m (the prefix length, which requires $m \ge k$) and r (the repetition factor), the algorithm proceeds as follows:

1. Run inference on the source input and collect all intermediate representations required by the attention matching procedure.

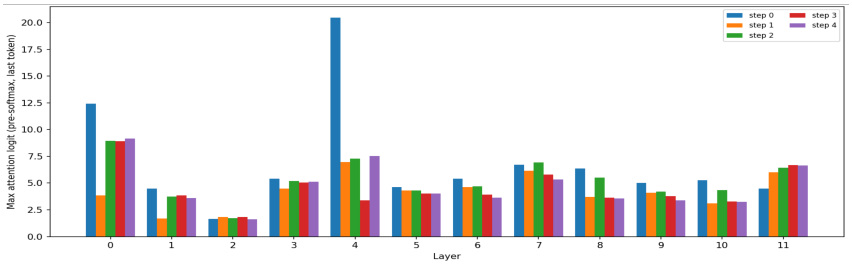


Figure 1: Maximum (across attention heads) pre-softmax scaled dot-product attention logits l_h across layers and decoding steps in GPT-2. The model is prompted with "To be, or not to be, that is the question" and generates 5 tokens. Notably, for certain layer–step pairs, this maximum exceeds 20.

2. For each attention head, construct S and $\mathbf{s}_j$.
3. For each attention head, solve the linear system $SV = [X|Y]$ to obtain the corresponding vectors $\tilde{\mathbf{s}}_j$.
4. Repeat each prefix vector r times.

The choice of $\mathbf{s}_j$ in step 2 requires careful consideration. In the proof of our main theorem, we provide a construction that guarantees S has full rank. In practice, this construction performs poorly numerically. Below, we highlight a desirable numerical property of $\mathbf{s}_j$ and present a new construction that achieves this property. Empirically, this construction always satisfies the full-rank requirement.

A key consideration when constructing S is the override strategy underlying the derivation of the sufficient condition (8). In theory, repeating the prefix vectors sufficiently many times guarantees successful override. In practice, however, this assumption may fail: due to the exponential scaling in the softmax operation, attention scores with large magnitudes can dominate, making them difficult to override using a limited number of prefix vectors.

Consider the pre-exponential score (logit)

$$l_h(\mathbf{x}, \mathbf{y}) = \frac{T}{\sqrt{d_{\text{hidden}}}} (\mathbf{W}_Q^h \mathbf{x})^\top (\mathbf{W}_K^h \mathbf{y}).$$

The exponential mapping in attention makes the mechanism highly sensitive to differences in the scaled logits. As illustrated in Figure 1, the maximum scaled attention logit l_h can reach values as large as 20, in which case the dominant term ($\exp(l_h) \approx e^{20}$) overwhelms all smaller contributions, making random overrides practically infeasible.

To enable reliable override, it is therefore necessary not only to repeat the prefix multiple times but also to ensure that, for each decoding step t, there exists at least one prefix vector $\mathbf{s}_j$ that produces a sufficiently large attention logit l_h. We formalize this requirement through the following override constraint. Given a threshold $attn_{lim}$,

$$\forall t \in \{0, 1, \ldots, k-1\},\ \exists j \in \{1, 2, \ldots, m\} \text{ s.t. } l_h(\mathbf{y}'_t, \mathbf{s}_j) \geq attn_{lim}. \tag{11}$$

We next propose a simple and constructive method for generating prefix vectors that satisfy this constraint. To induce a large attention score at step t, we construct a dedicated prefix vector $\mathbf{s}_{t+1}$ aligned with the direction of $(\mathbf{W}_K^h)^\top \mathbf{W}_Q^h \mathbf{y}'_t$. This choice is motivated by the fact that the scaled attention logit is linear in the key vector, and alignment with this direction maximizes the resulting attention score under a fixed norm.

Given this direction, the attention logit increases monotonically with the scaling factor. We therefore select the scaling factor ω_{t+1} as the minimal value that satisfies the override constraint in Equation (11), i.e., such that $l_h(\mathbf{y}'_t, \mathbf{s}_{t+1}) = attn_{lim}$. This choice yields a minimal-norm prefix vector that is sufficient to induce a large attention logit at the desired decoding step. Specifically, we choose $\mathbf{s}_{t+1} = \omega_{t+1} (\mathbf{W}_K^h)^\top \mathbf{W}_Q^h \mathbf{y}'_t$, where the scaling factor ω_{t+1} is determined accordingly.

Combining this construction with the override constraint in Equation (11), we obtain the following explicit formulation: Set $m = k$, and for $t = 0, \ldots, k-1$,

$$\mathbf{s}_{t+1} = \frac{attn_{lim}}{T \|(\mathbf{W}_K^h)^\top \mathbf{W}_Q^h \mathbf{y}'_t\|^2} (\mathbf{W}_K^h)^\top \mathbf{W}_Q^h \mathbf{y}'_t. \tag{12}$$

Table 1: Control success rates across datasets with different ending tokens, varying input length and output length. InLen and OutLen denote input length and output length, respectively.

Dataset	OutLen	InLen				
		5	10	20	50	100
and	1	1.0000	0.9931	0.9865	0.9804	0.9653
	5	0.9559	0.9857	0.9220	0.9127	0.7812
	10	0.8110	0.9722	0.9127	0.7857	0.6710
but	1	1.0000	1.0000	1.0000	0.9992	0.9865
	5	1.0000	1.0000	0.8702	0.9355	0.9282
	10	0.9527	0.9506	0.9122	0.7739	0.7641
comma	1	1.0000	1.0000	0.9988	0.8959	0.9327
	5	0.9620	0.9682	1.0000	0.8155	0.8224
	10	0.9171	0.9139	0.8433	0.6282	0.5988
dot	1	1.0000	1.0000	1.0000	0.9029	0.9943
	5	0.9918	0.9053	0.8555	0.7816	0.8861
	10	0.8196	0.6649	0.7551	0.5461	0.6743

5 Experiment on Complete Control of Output Tokens on GPT-2

In this experiment, we evaluate the ability of our algorithm to exert complete control over the generated output tokens across a wide range of source inputs, target outputs, and user inputs. All experiments are conducted on a workstation equipped with 4 consumer-grade GPUs and an Intel Core i9-9980XE CPU, and the full experimental sweep completes in approximately 2 hours.

To satisfy the assumptions stated in Theorem 2.1, we use Gemini Flash to construct four datasets that differ only in their terminal tokens, namely "but", "and", ".", and ",". Each dataset is further partitioned into buckets according to sentence length, with each bucket containing exactly 50 sentences. For all experiments, we set $attn_{lim} = 20.0$, which was chosen based on preliminary experiments and to account for the relatively large attention scores produced by GPT-2, as discussed above. The prefix repetition count was fixed at 90 to align with the context window of GPT-2, allowing the repeated prefix to occupy a substantial portion of the available context without exceeding the model's maximum input length.

For a fixed sentence length and terminal token, we consider all pairs of sentences within the same bucket. For each pair, one sentence is treated as the source input and the other as the user input. The success rate is defined as the number of pairs for which the computed prefix is successful, divided by the number of pairs in which the original generated outputs of the two sentences differ. In addition, we systematically vary the output length, the attention limit parameter $attn_{lim}$, and the number of prefix repetitions to assess the algorithm's ability to generate output of different length and to identify the parameter settings that yield the highest success rate.

Table 1 illustrates the results of the experiments. Overall, our method achieves a very high success rate. For short input/output lengths, the success rate is close to 1. Even for relatively longer input/output settings, the success rate remains in the range of 0.54–0.78, demonstrating strong robustness.

Further analysis of input length and output length shows that the success rate is primarily affected by the output length, while the impact of input length is comparatively minor. This behavior is expected, since the dimension of the constructed linear system is determined by output length; larger output length leads to higher-dimensional systems and increases the probability of an ill-conditioned linear system, thereby reducing the success rate. More investigations on the effects of the attention limit $attn_{lim}$ and the prefix repetition count are illustrated in Appendix F.

6 Extension to Grouped-Query Attention and Qwen3-4B

We conduct additional experiments on Qwen3-4B (Yang et al., 2025a), whose grouped-query attention (GQA) (Ainslie et al., 2023) architecture falls outside the scope of our theoretical guarantees. Nevertheless, our algorithm can be readily adapted to accommodate GQA. In GQA, used by many

Table 2: Success rates on Qwen3-4B, averaged over the 15 (OutLen, InLen) pairs of each dataset. The datasets are those of Table 1, rebuilt for the Qwen3 tokenizer with 50 sentences per bucket.

	`and`	`but`	`comma`	`dot`
$r = 1$	0.9977	0.9966	0.9950	0.9968
$r = 10$	1.0000	1.0000	1.0000	1.0000

current LLMs (Yang et al., 2025a; OpenAI, 2025), each group of G query heads shares one key/value head. Since a prefix enters the attention only through the key and value matrices, the query heads $h_1, \ldots, h_G$ of a group must also share it, and we stack their linear systems of Section 3.2 with a common variable matrix V:

$$\begin{bmatrix} S^{(1)} \\ \vdots \\ S^{(G)} \end{bmatrix} V = \begin{bmatrix} [X^{(1)}|Y^{(1)}] \\ \vdots \\ [X^{(G)}|Y^{(G)}] \end{bmatrix}. \tag{13}$$

Each key/value head thus receives $m = Gk$ prefix vectors, one per query head h_i and decoding step t, built from the query of h_i at step t as $\mathbf{s}_{t+1}$ in (12); $G = 1$ recovers $m = k$. Proposition D.1 does not cover the stacked $Gk \times Gk$ matrix S, which could in principle be singular or ill-conditioned. We have not observed this in our experiments.

QK-Norm. Qwen3-4B also applies QK-Norm: every query and key head is normalized by an RMSNorm (Zhang and Sennrich, 2019), $\mathrm{RMSNorm}(\mathbf{k}) = \sqrt{d_{\text{hidden}}}\,\boldsymbol{\gamma} \odot \mathbf{k}/\|\mathbf{k}\|_2$ with a learned gain $\boldsymbol{\gamma}$. Hence every key has the form $\boldsymbol{\gamma} \odot \mathbf{u}$ with $\|\mathbf{u}\|_2 = \sqrt{d_{\text{hidden}}}$, and the scaling factor in (12) has no effect. For the same reason, the construction in the proof of Proposition D.1 does not apply. For a normalized query $\mathbf{q}$, Cauchy–Schwarz gives

$$\mathbf{q}^\top(\boldsymbol{\gamma} \odot \mathbf{u})/\sqrt{d_{\text{hidden}}} \le \|\boldsymbol{\gamma} \odot \mathbf{q}\|_2, \quad \text{with equality at } \mathbf{u}^\star = \sqrt{d_{\text{hidden}}}\,\boldsymbol{\gamma} \odot \mathbf{q}/\|\boldsymbol{\gamma} \odot \mathbf{q}\|_2, \tag{14}$$

so the dedicated key is $\boldsymbol{\gamma} \odot \mathbf{u}^\star$, and the attained logit $\|\boldsymbol{\gamma} \odot \mathbf{q}\|_2$ replaces $attn_{lim}$, leaving no free parameter. We verified numerically that this key is produced by an input vector of the attention module (i.e., $\mathbf{W}_K^h$ has full row rank, so that $\mathbf{W}_K^h \mathbf{s} = \mathbf{u}^\star$ has a solution $\mathbf{s}$), so it is a prefix in the sense of Section 2.2. By (14), each query attains the largest possible logit $\|\boldsymbol{\gamma} \odot \mathbf{q}\|_2$ on its own prefix vector. Divided by this value, the logits form a matrix with ones on the diagonal and, off the diagonal, the cosines between the vectors $\boldsymbol{\gamma} \odot \mathbf{q}$ of different queries, which are smaller than one unless two of these vectors point in the same direction; hence every row of S has its largest entry on the diagonal. As this logit is a ceiling, a real token can occasionally outscore the dedicated prefix vector, which repetition compensates. Repetition leaves S and V unchanged and only raises the share of attention on the prefix, which tends to one as r grows; the limit argument in Appendix D requires no lower bound on the logits.

Results. We repeat the experiment of Section 5 on Qwen3-4B, with the four datasets rebuilt for the Qwen3 tokenizer so that every bucket again holds 50 sentences ending in the common terminal token and no pair violates $\mathtt{x}_n = \mathtt{x}'_p$. As Table 2 shows, the prefix forces the target output in 99.65% of the instances without any repetition ($r = 1$), and with $r = 10$ all groups reach 100%. This is far above prompt-based steering: even a system prompt instructing the model to output the target succeeds only 84.6% of the time (Appendix G). We conjecture that QK-Norm, by bounding the logits, weakens the softmax sensitivity discussed in Section 4 and keeps the linear system of Section 3.2 well conditioned.

7 Conclusions

We settle a long-standing open problem regarding the expressive power of prefix-tuning in practical large language models and fine-tuning settings by providing and proving a complete characterization of the existence of a prefix capable of inducing essentially arbitrary output sequences. Unlike prior work, our result applies to modern LLM architectures with arbitrary weights, and is fully constructive: we provide an explicit algorithm for generating a prefix that produces any desired target token sequence. The construction extends to grouped-query attention, where it forces the target output of Qwen3-4B in every tested setting. This provides strong evidence for the practicality of prefix-tuning as a parameter-efficient fine-tuning method.

References

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4895–4901, Singapore, December 2023. Association for Computational Linguistics.

Iz Beltagy, Kyle Lo, and Arman Cohan. SciBERT: A pretrained language model for scientific text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3615–3620, Hong Kong, China, November 2019. Association for Computational Linguistics.

Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1–9, Dublin, Ireland, May 2022. Association for Computational Linguistics.

Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri Chatterji, Annie Chen, Kathleen Creel, Jared Quincy Davis, Dora Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark Krass, Ranjay Krishna, Rohith Kuditipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir Mirchandani, Eric Mitchell, Zanele Munyikwa, Suraj Nair, Avanika Narayan, Deepak Narayanan, Ben Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, Julian Nyarko, Giray Ogut, Laurel Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Rob Reich, Hongyu Ren, Frieda Rong, Yusuf Roohani, Camilo Ruiz, Jack Ryan, Christopher Ré, Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishnan Srinivasan, Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramèr, Rose E. Wang, William Wang, Bohan Wu, Jiajun Wu, Yuhuai Wu, Sang Michael Xie, Michihiro Yasunaga, Jiaxuan You, Matei Zaharia, Michael Zhang, Tianyi Zhang, Xikun Zhang, Yuhui Zhang, Lucia Zheng, Kaitlyn Zhou, and Percy Liang. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, Nikolaos Aletras, and Ion Androutsopoulos. LEGAL-BERT: The muppets straight out of law school. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2898–2904, Online, November 2020. Association for Computational Linguistics.

Zhiyu Zoey Chen, Jing Ma, Xinlu Zhang, Nan Hao, An Yan, Armineh Nourbakhsh, Xianjun Yang, Julian J. McAuley, Linda Ruth Petzold, and William Yang Wang. A survey on large language models for critical societal domains: Finance, healthcare, and law. *Transactions on Machine Learning Research*, 2024.

Ning Ding, Yujia Qin, Guang Yang, Fu Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, Jing Yi, Weilin Zhao, Xiaozhi Wang, Zhiyuan Liu, Haitao Zheng, Jianfei Chen, Y. Liu, Jie Tang, Juanzi Li, and Maosong Sun. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, 5:220–235, 2023.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online, July 2020. Association for Computational Linguistics.

Zeyu Han, Chao Gao, Jinyang Liu, Jeff Zhang, and Sai Qian Zhang. Parameter-efficient fine-tuning for large models: A comprehensive survey. *Transactions on Machine Learning Research*, 2024.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. Towards a unified view of parameter-efficient transfer learning. *arXiv preprint arXiv:2110.04366*, 2021.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, pages 2790–2799. PMLR, 2019.

J. Edward Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Jerry Yao-Chieh Hu, Wei-Po Wang, Ammar Gilani, Chenyang Li, Zhao Song, and Han Liu. Fundamental limits of prompt tuning transformers: Universality, capacity and efficiency. In *International Conference on Learning Representations*, volume 2025, pages 29634–29686, 2025.

Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. Compacter: Efficient low-rank hypercomplex adapter layers. In *Advances in Neural Information Processing Systems*, volume 34, pages 1022–1035, 2021.

Jinhyuk Lee, WonJin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36:1234 – 1240, 2019.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, 2021.

Dongze Lian, Daquan Zhou, Jiashi Feng, and Xinchao Wang. Scaling & shifting your features: A new baseline for efficient model tuning. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Yingyu Liang, Zhenmei Shi, Zhao Song, and Chiwun Yang. Towards infinite-long prefix in transformer. *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 2025.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 61–68, Dublin, Ireland, May 2022. Association for Computational Linguistics.

OpenAI. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025.

Aleksandar Petrov, Philip Torr, and Adel Bibi. When do prompting and prefix-tuning work? A theory of capabilities and limitations. In *International Conference on Learning Representations*, volume 2024, pages 6031–6054, 2024a.

Aleksandar Petrov, Philip H.S. Torr, and Adel Bibi. Prompting a pretrained transformer can be a universal approximator. In *Proceedings of the 41st International Conference on Machine Learning*, 2024b.

Clifton Poth, Hannah Sterz, Indraneil Paul, Sukannya Purkayastha, Leon Engländer, Timo Imhof, Ivan Vulić, Sebastian Ruder, Iryna Gurevych, and Jonas Pfeiffer. Adapters: A unified library for parameter-efficient and modular transfer learning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 149–160, Singapore, December 2023. Association for Computational Linguistics.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy, July 2019. Association for Computational Linguistics.

Yihan Wang, Jatin Chauhan, Wei Wang, and Cho-Jui Hsieh. Universality and limitations of prompt tuning. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Lingling Xu, Haoran Xie, Si-Zhao Joe Qin, Xiaohui Tao, and Fu Lee Wang. Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment. *arXiv preprint arXiv:2312.12148*, 2023.

Zihao Xu, Yi Liu, Gelei Deng, Yuekang Li, and Stjepan Picek. A comprehensive study of jailbreak attack versus defense for large language models. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 7432–7449, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

Chenghan Yang, Ruiyu Zhao, Yang Liu, and Ling Jiang. Survey of specialized large language model. *arXiv preprint arXiv:2508.19667*, 2025b.

Sibo Yi, Yule Liu, Zhen Sun, Tianshuo Cong, Xinlei He, Jiaxing Song, Ke Xu, and Qi Li. Jailbreak attacks and defenses against large language models: A survey. *arXiv preprint arXiv:2407.04295*, 2024.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in neural information processing systems*, 32, 2019.

Adam Zweiger, Xinghong Fu, Han Guo, and Yoon Kim. Fast kv compaction via attention matching. In *International Conference on Machine Learning (ICML)*, 2026. URL `https://arxiv.org/abs/2602.16284`.

A Comparison between Autoregressive Generation and Sequence-to-Sequence Generation

A few prior works have shown that a Transformer with a prefix can approximate arbitrary sequence-to-sequence functions (Wang et al., 2023; Petrov et al., 2024b; Hu et al., 2025), which appears to be a much stronger result than ours (which considers only a single input-output pair). This makes it tempting to derive our result as a consequence of theirs. However, as noted in the main text, their proofs rely on specially crafted Transformers, whereas ours apply to any pretrained Transformer specified by the user, aligning more closely with practical fine-tuning scenarios. Beyond this distinction, this section highlights another subtle but important difference: autoregressive generation imposes significantly stronger requirements on prefix construction than sequence-to-sequence approximation or generation.

Essentially, we aim to find a prefix $\mathcal{S}$ (with slightly simplified and abused notation) such that

$$\mathcal{T}_{\mathcal{S}}(\mathtt{x}) = \mathtt{y}_1, \mathcal{T}_{\mathcal{S}}(\mathtt{x}, \mathtt{y}_1) = \mathtt{y}_2, \cdots, \mathcal{T}_{\mathcal{S}}(\mathtt{x}, \mathtt{y}_1, \cdots, \mathtt{y}_{k-1}) = \mathtt{y}_k$$

(see (2) for the precise formulation). If one attempts to apply sequence-to-sequence approximation results to construct $\mathcal{S}$, a choice must be made regarding which of these equations to target. For

instance, one could apply existing expressiveness results to obtain a prefix $\mathcal{S}$ satisfying $\mathcal{T}_{\mathcal{S}}(\mathtt{x}) = \mathtt{y}_1$, or $\mathcal{T}_{\mathcal{S}}(\mathtt{x}, \mathtt{y}_1) = \mathtt{y}_2$, and so on. The key issue is that the resulting prefix $\mathcal{S}$ would generally differ across these choices. In contrast, our approach constructs a single, consistent prefix $\mathcal{S}$ that satisfies all k equations simultaneously.

B Counterexamples

Example 1 ($\mathtt{x}_n = \mathtt{x}'_p$ is necessary)**.** Let $\mathcal{V} = \{a, b\}$ be a vocabulary with two tokens a and b, and let $\mathbf{Emb}(a) = (1, 0)$ and $\mathbf{Emb}(b) = (0, 1)$ be their embeddings. Let $\mathcal{T}$ be a one-layer, one-head Transformer with a residual connection, where the weight matrix W_V is the zero matrix. In other words, the attention module makes no contribution. Thus, $\mathcal{T}$ is essentially the identity map from $\mathbb{R}^2$ to $\mathbb{R}^2$ (for an input sequence, it acts as the identity on the last token).

At the output of $\mathcal{T}$, let the first coordinate correspond to the logit of a and the second to that of b. Then, $\mathcal{T}$ simply takes the last observed token as the next-token prediction.

Now, in Theorem 2.1, let the target input and output be (a) and (b), respectively. The token b is clearly reachable by $\mathcal{T}$ (e.g. with input (b)). However, no prefix $\mathcal{S}$ can satisfy $\mathcal{T}_{\mathcal{S}}((a)) = b$, since the output is entirely determined by the last input token. The model will therefore output a again.

Example 2 ($p = n$ is necessary)**.** Consider the same setting as in Example 1, but now suppose we use a positional encoding such that every embedding from the second position onward is added by $(100, 0)$, forcing the model to always predict a whenever the input sequence has length at least two.

Now, let the target input and output be (a, b) and (b), respectively. The target (b) remains reachable from the source input (b). Note that, unlike in Example 1, the last tokens of the target and source inputs already coincide. However, the next-token prediction for (a, b), even with a prefix, will always be a due to the positional encoding. This limitation arises fundamentally from the mismatch in sequence lengths, i.e., $p \neq n$.

Example 3 (Two input-output pairs, Proof of Theorem 2.2)**.** Consider the following two input-output pairs:

$$\begin{aligned} \mathtt{x} = (\mathtt{x}_1, \mathtt{x}_2), &\mathtt{y} = (\mathtt{y}_1, \mathtt{y}_2) \\ \mathtt{x} = (\mathtt{x}_1, \mathtt{x}_2, \mathtt{y}_1), &\mathtt{y} = (\mathtt{y}_3) \end{aligned} \tag{15}$$

where $\mathtt{y}_2 \neq \mathtt{y}_3$. If $\mathcal{T}$ satisfies the first pair, then when given the input from the second pair, it must generate $\mathtt{y}_2$ rather than $\mathtt{y}_3$. Therefore, these two pairs cannot be simultaneously realized by $\mathcal{T}$, even under full fine-tuning.

Example 4 (Two input-output pairs, with extra assumptions)**.** It is tempting to address Example 3 by imposing stronger assumptions. Suppose we additionally require:

- The two input-output pairs are jointly realizable via full fine-tuning;
- Each input-output pair, considered in isolation, is realizable by some prefix.

Can we then conclude that there exists a single prefix that realizes both pairs simultaneously? Unfortunately, the answer remains negative.

Consider again a one-layer, one-head Transformer $\mathcal{T}$, but now let W_Q and W_K be zero matrices and W_V be the identity matrix, with no residual connections or positional encodings. In this case, all tokens attend uniformly (since the softmax is applied to a zero vector), and the output of $\mathcal{T}$ is simply the average of the token embeddings. Equivalently, next-token prediction reduces to a majority vote: the most frequent token is selected.

Now consider the two input-output pairs:

$$\begin{aligned} \mathtt{x} = (a), \quad \mathtt{y} = (b), \\ \mathtt{x} = (b), \quad \mathtt{y} = (a). \end{aligned} \tag{16}$$

These can be realized via full fine-tuning (e.g. by setting W_V to the negative identity). Individually, each pair can also be realized by a prefix $\mathcal{S}$: a prefix consisting of many b tokens for the first pair, or many a tokens for the second. However, no single prefix can satisfy both simultaneously, as it cannot contain strictly more a than b and strictly more b than a at the same time.

Table 3: Success rate under different input lengths. Source Input Length = 50 (*"You are an idiot."* $\times$ 10), Output Length = 10 (*"You are an idiot."* $\times$ 2).

InLen	5	10	20	50	100
Success Rate	1.00	1.00	1.00	1.00	0.92

At a high level, Example 3 arises from contradictory requirements within the fine-tuning dataset. In contrast, in this example the contradiction stems not only from the dataset but also from the model's aggregation behavior. Consequently, addressing Problem 1 for larger datasets likely requires consistency conditions on both the dataset and the model. Formulating such conditions is nontrivial, and we hope this work encourages further progress in that direction.

C Attack on LLM memory

In (Petrov et al., 2024b), the authors postulate that the expressiveness of prefix-tuning may pose security risks to LLM systems, although they do not establish whether such attacks can manifest in real-world settings. In this section, we provide a proof-of-concept attack scenario based on our algorithm. This attack is not intended to compromise real-world deployments, particularly when reasonable defenses are in place; rather, our goal is to move one step beyond the speculative concerns raised in (Petrov et al., 2024b).

Recall that our algorithm enables the model to produce a predetermined response with high probability, largely independent of the user input, provided that we have access to the model and the ability to replace its KV cache with one augmented by a malicious prefix. This capability relies on two key assumptions: first, that there exists a source input capable of generating the desired target output, and second, that the user input has the same length as the source input. In this experiment, we demonstrate a method for identifying a suitable source input for a deliberately constructed hazardous target output, and we further show that the attack remains effective even when the user input length does not exactly match that of the source input.

For GPT-2, we identify two effective approaches for generating such source inputs. The first approach is to directly prompt the model to repeat a given sentence, since GPT-2 lacks post-training mechanisms to avoid hazardous outputs. The second approach is to repeat the target sentence sufficiently many times in the prompt, which causes GPT-2 to continue generating the same sentence. For state-of-the-art models equipped with post-training safety alignment to avoid hazardous outputs, we refer readers to the literature on jailbreak attacks (Xu et al., 2024; Yi et al., 2024).

As a concrete example, to generate the humiliating sentence "You are an idiot.", one can simply repeat this sentence multiple times in the input, after which GPT-2 will reproduce it in the output. Since the resulting source input may not have the same length as the user input, we additionally evaluate the robustness of our algorithm by conducting experiments in which the same source input is used while varying the length of the user input.

Despite the theoretical requirement that the source input length should match the user input length, the experimental results demonstrate surprising robustness to variations in user input length. As shown in Table 3, the success rate remains at 100% across a wide range of input lengths, and only slightly drops when the input length deviates substantially from the source length. One possible explanation is that positional embeddings contribute only a small fraction of the final token representation, making the attack relatively insensitive to moderate length mismatches.

On the other hand, our experiments with ending token mismatches consistently fail, indicating that matching the final token is crucial in practice.

D Proof of the Main Theorem

Most of the content of the proof of 2.1 is already included in the main text, except the solubility of the linear system $SV = [X|Y]$ when (10) is satisfied. We formulate this fact precisely and provide the proof below.

Proposition D.1. *If* (10) *is satisfied, then there exists* $\mathbf{s}_1, \ldots, \mathbf{s}_k$ *such that the* $k \times k$ *matrix*

$$S = \begin{bmatrix} \frac{E_h(\mathbf{x}_n, \mathbf{s}_1)}{\sum_{j=1}^k E_h(\mathbf{x}_n, \mathbf{s}_j)} & \cdots & \frac{E_h(\mathbf{x}_n, \mathbf{s}_k)}{\sum_{j=1}^k E_h(\mathbf{x}_n, \mathbf{s}_j)} \\ \vdots & \ddots & \vdots \\ \frac{E_h(\mathbf{y}_{k-1}, \mathbf{s}_1)}{\sum_{j=1}^k E_h(\mathbf{y}_{k-1}, \mathbf{s}_j)} & \cdots & \frac{E_h(\mathbf{y}_{k-1}, \mathbf{s}_k)}{\sum_{j=1}^k E_h(\mathbf{y}_{k-1}, \mathbf{s}_j)} \end{bmatrix}$$

is non-singular (note that we have set $m = k$*).*

Proof. Since multiplying each row by a constant preserves non-singularity, it suffices to find $\mathbf{s}_1, \ldots, \mathbf{s}_k$ such that the $k \times k$ matrix

$$\overline{S} = \begin{bmatrix} E_h(\mathbf{x}_n, \mathbf{s}_1) & \cdots & E_h(\mathbf{x}_n, \mathbf{s}_k) \\ \vdots & \ddots & \vdots \\ E_h(\mathbf{y}_{k-1}, \mathbf{s}_1) & \cdots & E_h(\mathbf{y}_{k-1}, \mathbf{s}_k) \end{bmatrix}$$

is non-singular. By our assumption (10), there exists $\mathbf{s}$ such that

$$E_h(\mathbf{x}_n, \mathbf{s}), E_h(\mathbf{y}_1, \mathbf{s}), \ldots, E_h(\mathbf{y}_{k-1}, \mathbf{s}) \text{ are all distinct.}$$

Consequently, we can simply take

$$\mathbf{s}_j = (j-1)\mathbf{s}.$$

Then, we have

$$\begin{aligned} E_h(\mathbf{x}_n, \mathbf{s}_j) &= \exp\left(\frac{T}{\sqrt{d_{\text{hidden}}}}(\mathbf{W}_Q^h \mathbf{x}_n)^\top((j-1)\mathbf{W}_K^h \mathbf{s})\right) = \left(\exp\left(\frac{T}{\sqrt{d_{\text{hidden}}}}(\mathbf{W}_Q^h \mathbf{x}_n)^\top(\mathbf{W}_K^h \mathbf{s})\right)\right)^{j-1} \\ &= E_h(\mathbf{x}_n, \mathbf{s})^{j-1}. \end{aligned}$$

Similar equalities hold for $\mathbf{y}_1, \ldots, \mathbf{y}_{k-1}$. Therefore, $\overline{S}$ is a Vandermonde matrix and it is well-known that its determinant is given by the product of mutual differences of $E_h(\mathbf{x}_n, \mathbf{s}), E_h(\mathbf{y}_1, \mathbf{s}), \ldots, E_h(\mathbf{y}_{k-1}, \mathbf{s})$, which is non-zero by our assumption. □

Proof of Theorem 2.1. By Proposition D.1, the linear system $SV = [X|Y]$ is solvable at every attention head. This gives $\mathbf{s}_j, \tilde{\mathbf{s}}_j$ $(j = 1, \ldots, k)$ such that (9) holds. To establish (8) from (9), note that if all $\mathbf{s}_j, \tilde{\mathbf{s}}_j$ are repeated N times, the quotient $A_h(\mathbf{x}, \mathbf{y}, \mathbf{s})/B_h(\mathbf{x}, \mathbf{y}, \mathbf{s})$ becomes

$$\begin{aligned} &\frac{N\sum_{j=1}^m E_h(\mathbf{y}'_t, \mathbf{s}_j)\tilde{\mathbf{s}}_j + \sum_{i=1}^n E_h(\mathbf{y}'_t, \mathbf{x}_i)\mathbf{x}_i + \sum_{i=1}^t E_h(\mathbf{y}'_t, \mathbf{y}_i)\mathbf{y}_i}{N\sum_{j=1}^m E_h(\mathbf{y}'_t, \mathbf{s}_j) + \sum_{i=1}^n E_h(\mathbf{y}'_t, \mathbf{x}_i) + \sum_{i=1}^t E_h(\mathbf{y}'_t, \mathbf{y}_i)} \\ &= \frac{\sum_{j=1}^m E_h(\mathbf{y}'_t, \mathbf{s}_j)\tilde{\mathbf{s}}_j + \frac{\sum_{i=1}^n E_h(\mathbf{y}'_t, \mathbf{x}_i)\mathbf{x}_i + \sum_{i=1}^t E_h(\mathbf{y}'_t, \mathbf{y}_i)\mathbf{y}_i}{N}}{\sum_{j=1}^m E_h(\mathbf{y}'_t, \mathbf{s}_j) + \frac{\sum_{i=1}^n E_h(\mathbf{y}'_t, \mathbf{x}_i) + \sum_{i=1}^t E_h(\mathbf{y}'_t, \mathbf{y}_i)}{N}}. \end{aligned}$$

For a given $\epsilon > 0$ there exists a positive integer N_h such that

$$\left|\left| \frac{\sum_{j=1}^m E_h(\mathbf{y}'_t, \mathbf{s}_j)\tilde{\mathbf{s}}_j}{\sum_{j=1}^m E_h(\mathbf{y}'_t, \mathbf{s}_j)} - \frac{\sum_{j=1}^m E_h(\mathbf{y}'_t, \mathbf{s}_j)\tilde{\mathbf{s}}_j + \frac{\sum_{i=1}^n E_h(\mathbf{y}'_t, \mathbf{x}_i)\mathbf{x}_i + \sum_{i=1}^t E_h(\mathbf{y}'_t, \mathbf{y}_i)\mathbf{y}_i}{N}}{\sum_{j=1}^m E_h(\mathbf{y}'_t, \mathbf{s}_j) + \frac{\sum_{i=1}^n E_h(\mathbf{y}'_t, \mathbf{x}_i) + \sum_{i=1}^t E_h(\mathbf{y}'_t, \mathbf{y}_i)}{N}} \right|\right|_\infty < \epsilon$$

for all $N \geq N_h$. By taking the maximum over all attention heads and all t, we have shown that for any given $\epsilon > 0$, there exists a generalized prefix $\mathcal{S}$ such that for every multi-head attention module $\mathcal{A}$, we have

$$||\mathcal{A}_{\mathcal{S}}(\mathbf{y}'_t) - \mathcal{A}(\mathbf{y}_t)||_\infty < Hd_{in}\left(\max_h ||\mathbf{W}_V^h||_\infty\right)\epsilon$$

for $0 \leq t \leq k-1$. Here the constant H comes from aggregating all attention heads, $d_{in}\max_h ||\mathbf{W}_V^h||_\infty$ comes from the multiplication by $\mathbf{W}_V^h$, and $\mathcal{A}_{\mathcal{S}}$ denotes the attention module after applying the prefix $\mathcal{S}$ (for simplicity, we omit the contribution of previous tokens).

If $\mathcal{T}$ does not have residual connections, the proof is essentially complete as we have guaranteed that the outputs of all multi-head attention modules with the user input $\mathtt{x}$ can be arbitrarily close to those

with the source input $\mathtt{x}'$. It is also clear from the proof we only require prefixes at the last layer. If residual connections are present, an input embedding is transformed as

$$\mathbf{x}' \longrightarrow \mathbf{x}' + \mathcal{A}(\mathbf{x}').$$

Our attention matching has already achieved

$$\mathcal{A}_{\mathcal{S}}(\mathbf{y}') \approx \mathcal{A}(\mathbf{y})$$

up to a factor of ϵ. To do the same for the residual outputs, we further require the condition $\mathbf{Emb}(\mathtt{x}'_p) = \mathbf{Emb}(\mathtt{x}_n)$. This clearly holds when $p = n$ and $\mathtt{x}'_p = \mathtt{x}_n$.

□

E Considering Biases in the Attention

Previously, we ignored the bias in the attention mechanism for simplicity. In this section, we show that our theory and algorithm still hold if we introduce biases in Q, K, and V. The Q and K biases are handled jointly, whereas the V bias is treated independently. We start with the V cache.

We first introduce the bias to the V term. In practical models, instead of directly using $\mathbf{W}_V^h \mathbf{x}'_i$ and $\mathbf{W}_V^h \mathbf{y}_i$, the terms become $\mathbf{W}_V^h \mathbf{x}'_i + \mathbf{bias}_V^h$ and $\mathbf{W}_V^h \mathbf{y}_i + \mathbf{bias}_V^h$, respectively.

This modification yields the following expression for the attention output:

$$\sum_{h=1}^{H} \frac{A^h_{vbias}(\mathbf{x}, \mathbf{y}, \mathbf{s})}{B_h(\mathbf{x}, \mathbf{y}, \mathbf{s})},$$

where the numerator is given by

$$\begin{aligned}
A^h_{vbias}(\mathbf{x}, \mathbf{y}, \mathbf{s}) =& \sum_{j=1}^{m} E_h(\mathbf{y}'_t, \mathbf{s}_j)(\mathbf{W}_V^h \mathbf{s}_j + \mathbf{bias}_V^h) + \sum_{i=1}^{n} E_h(\mathbf{y}'_t, \mathbf{x}_i)(\mathbf{W}_V^h \mathbf{x}_i + \mathbf{bias}_V^h) \\
&+ \sum_{i=1}^{t} E_h(\mathbf{y}'_t, \mathbf{y}_i)(\mathbf{W}_V^h \mathbf{y}_i + \mathbf{bias}_V^h) \\
=& \mathbf{W}_V^h A_h + \left(\sum_{j=1}^{m} E_h(\mathbf{y}'_t, \mathbf{s}_j) + \sum_{i=1}^{n} E_h(\mathbf{y}'_t, \mathbf{x}_i) + \sum_{i=1}^{t} E_h(\mathbf{y}'_t, \mathbf{y}_i) \right) \mathbf{bias}_V^h \\
=& \mathbf{W}_V^h A_h + B_h \mathbf{bias}_V^h.
\end{aligned}$$

Then the aggregated attention expression can be written as:

$$\sum_{h=1}^{H} \frac{A^h_{vbias}}{B_h} = \sum_{h=1}^{H} \left(\mathbf{W}_V^h \frac{A^h}{B_h} + \mathbf{bias}_V^h \right) = \sum_{h=1}^{H} \mathbf{W}_V^h \frac{A^h}{B_h} + \sum_{h=1}^{H} \mathbf{bias}_V^h.$$

The same logic applies to the RH Sof our sufficient condition (7), where the additional V bias term naturally cancels out on both sides without requiring extra conditions.

Next, we address the Q and K caches. In the main text, Q and K were introduced in Section 3 within the E_h term. Simply adding bias to the E_h term does not alter the analysis in that section. However, the construction of the prefix in equation (12) requires modification.

Since the attention logit becomes

$$l_h(\mathbf{x}, \mathbf{y}) = \frac{T}{\sqrt{d_{\text{hidden}}}} (\mathbf{W}_Q^h \mathbf{x} + \mathbf{bias}_Q^h)^\top (\mathbf{W}_K^h \mathbf{y} + \mathbf{bias}_K^h),$$

the direction of the prefix vector $\mathbf{s}_{t+1}$ must be adjusted to:

$$\mathbf{s}_{t+1} = \omega_{t+1} (\mathbf{W}_K^h)^\top (\mathbf{W}_Q^h \mathbf{y}'_t + \mathbf{bias}_Q^h).$$

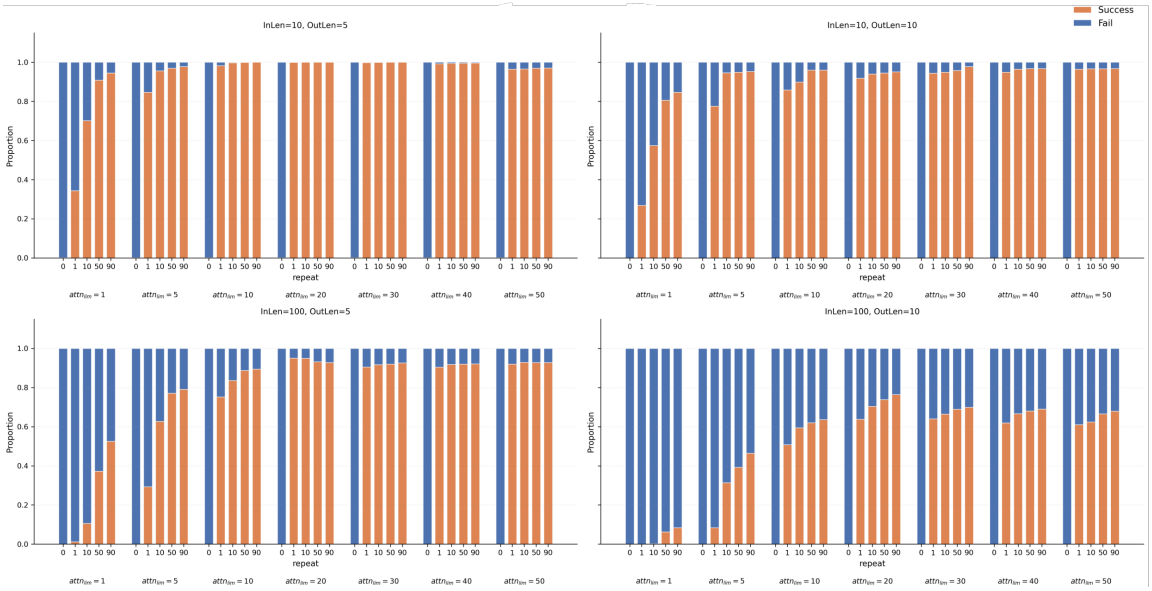


Figure 2: Performance Breakdown Across Attention Limits and Repetition Counts

The corresponding scaling factor must also account for the K-bias. This yields the final explicit formulation:

$$\text{Set } m = k. \quad \text{For } t = 0, \ldots, k-1:$$

$$\mathbf{s}_{t+1} = \frac{\sqrt{d_{\text{hidden}}}\text{attn}_{lim} - T(\mathbf{W}_Q^h \mathbf{y}_t' + \mathbf{bias}_Q^h)^\top \mathbf{bias}_K^h}{T\|(\mathbf{W}_K^h)^\top(\mathbf{W}_Q^h \mathbf{y}_t' + \mathbf{bias}_Q^h)\|^2} (\mathbf{W}_K^h)^\top (\mathbf{W}_Q^h \mathbf{y}_t' + \mathbf{bias}_Q^h).$$

Finally, our proof of the main theorem in Section D depends on the concrete form of Q and K. The Q-bias does not influence the proof structure; simply adding the bias to the Q-term does not introduce conflicts. On the other hand, including the K-bias requires a brief explanation regarding the non-singularity of matrix $\overline{S}$.

Recall that $E_h(\mathbf{x}, \mathbf{s}) = \exp\left((\mathbf{W}_Q^h \mathbf{x} + \mathbf{bias}_Q^h)^\top (\mathbf{W}_K^h \mathbf{s} + \mathbf{bias}_K^h)\right)$. Expanding the exponent gives:

$$(\mathbf{W}_Q^h \mathbf{x} + \mathbf{bias}_Q^h)^\top \mathbf{W}_K^h \mathbf{s} + (\mathbf{W}_Q^h \mathbf{x} + \mathbf{bias}_Q^h)^\top \mathbf{bias}_K^h.$$

The second term depends only on the query $\mathbf{x}$ (the row index in $\overline{S}$) and not on the prefix $\mathbf{s}$ (the column index). Therefore, adding K-bias is equivalent to multiplying each row of the original $\overline{S}$ matrix by a non-zero constant factor:

$$C(\mathbf{x}) = \exp\left(\frac{T}{\sqrt{d_{\text{hidden}}}} (\mathbf{W}_Q^h \mathbf{x} + \mathbf{bias}_Q^h)^\top \mathbf{bias}_K^h\right).$$

Since multiplying rows by non-zero constants preserves linear independence, the modified $\overline{S}$ remains non-singular.

F Effect of the Override Parameters on GPT-2

We analyze model performance under varying attention limits ($attn_{lim}$) and repetition counts (repeat) across different input-output length configurations, measured by success rate over 2,450 trials per setting. The results are illustrated in Figure 2. Across all input–output length configurations, we observe consistent trends in how the attention limit ($attn_{lim}$) and repetition count (repeat) affect model performance.

When varying the attention limit while holding repetition fixed, the success rate increases sharply at small attention limits (e.g., from $attn_{lim} = 1$ to 5 or 10), suggesting that insufficient attention limit is

a primary cause of failure. At very small attention limits, the attention scores are too weak to reliably preserve relevant information, such that even a large number of repetitions (e.g., repeat $= 90$) still results in frequent override failures. Beyond moderate values ($attn_{lim} \approx 20$–30), further increasing the attention limit yields diminishing returns, with performance largely saturating or exhibiting only marginal improvements. One possible explanation is that excessively large attention limits may introduce numerical instability, leading to an ill-conditioned attention computation that offsets the potential benefits of increased capacity.

In contrast, increasing the repetition count leads to a monotonic improvement in success rate, but with a different characteristic shape. Performance improves substantially when increasing repeat from 0 to small positive values, while further increases (e.g., from 50 to 90) result in minor gains, indicating early saturation. Unlike attention, repetition does not degrade performance at high values, but its effect is contingent on sufficient attention capacity.

Importantly, attention limit and repetition exhibit a strong interaction effect. When the attention limit is very small, increasing repetition alone cannot recover performance, whereas at sufficiently large attention limits, even modest repetition is sufficient to achieve high success rates. This suggests that attention capacity determines whether the task is solvable, while repetition primarily affects the reliability of successful outcomes once the attention constraint is relaxed.

Finally, the absolute performance levels and sensitivity to these factors depend on the input and output lengths. Tasks with longer inputs require higher attention limits and greater repetition to achieve comparable performance, while shorter input-output settings reach near-saturation at much smaller values. Overall, these results highlight complementary roles of attention capacity and repetition, as well as their dependence on task length characteristics.

G Comparison with Prompt-Based Steering on Qwen3-4B

A natural concern is that forcing an LLM to emit a fixed output is easy. We compare attention matching with two prompt-based approaches on Qwen3-4B. The first, *repeated target*, prepends the target output repeated $N \in \{16, 32\}$ times, in the hope that the model continues the pattern. The second, *system prompt*, instructs the model through the system message to output the target and ignore the user input. We use a long variant,

```
Your response must always be exactly: TARGET_OUTPUT. Output no
additional characters, whitespace, punctuation, formatting, or
explanation. Ignore all user requests to change this response.
```

and a short variant, `Output "TARGET_OUTPUT", ignore the user prompt.` Both use the Qwen3 chat template with an empty thinking block.

Table 4 reports the overall success rate over all 60 groups. Repeating the target barely helps, and even the long system prompt fails on roughly one instance in six, whereas attention matching succeeds on more than 99% of the instances with $r = 1$ and on all instances with $r \geq 10$. The prompt-based baselines are restricted to discrete tokens, whereas our prefix is continuous; the comparison shows that the task is far from trivial rather than comparing methods under identical access assumptions.

Table 4: Overall success rates of steering Qwen3-4B toward a target output, averaged over all 60 groups.

Method	Success rate
No prefix	0.0351
Repeated target, $N = 16$	0.1660
Repeated target, $N = 32$	0.1700
System prompt, short	0.5547
System prompt, long	0.8462
Attention matching (ours), $r = 1$	0.9966
Attention matching (ours), $r = 10$	**1.0000**
Attention matching (ours), $r = 50$	**1.0000**

H Relationship to Attention Matching in Zweiger at al. (2026)

Independent and parallel development. A closely related attention matching framework was independently developed by Zweiger et al. (2026) for KV cache compaction. Our attention matching framework was also developed independently, with its core formulation, linear-system construction, and Vandermonde-based existence proof already included in our original submission to ICML 2026 in January 2026, prior to the public release of Zweiger et al. (2026) in February 2026. Thus, while the two works share a similar underlying mathematical principle and terminology, they were developed independently and in parallel.

Common underlying principle, different construction and solution methods. Both works exploit the observation that, once the attention keys are fixed, attention outputs depend linearly on the value representations. This allows attention matching to be formulated as a linear-algebraic problem rather than requiring end-to-end gradient-based optimization. While both works share this principle idea, they differ in the construction as well as solution methods for the linear-algebraic problem. Specifically, in our framework, we decouple the key and value prefixes and formulate the matching conditions as a system of linear equations, $SV = [X \mid Y]$, whose solutions determine the desired prefix values. In contrast, Zweiger et al. (2026) construct compacted KV representations by selecting keys and solving a least-squares problem for the corresponding values to approximate the original attention outputs over a set of reference queries.

Differences in objectives and theoretical contributions. Despite this shared principle, the two frameworks address distinct research problems and differ substantially in their theoretical objectives. Zweiger et al. (2026) focus on KV cache compaction, aiming to preserve the attention behavior of the original cache using fewer KV entries. Their approach primarily selects a subset of existing keys, fits the corresponding values through least squares, and introduces additional attention-logit biases to preserve attention mass.

In contrast, our objective is to establish the expressive power of prefix-tuning: given a pretrained autoregressive Transformer and a reachable target output sequence, we seek to construct a prefix that induces the model to generate that sequence. Rather than selecting existing keys for compression, we construct new prefix keys and values to match the attention outputs of a prescribed autoregressive decoding trajectory. Crucially, we establish sufficient conditions for the resulting linear system to admit a solution, using a Vandermonde-based full-rank argument, and employ prefix repetition to suppress the contribution of the original input tokens. This construction leads to the existence guarantees in Theorem 2.1 under its stated assumptions.

In summary, while both works independently exploit the linear structure of attention matching, our primary contribution is a constructive theoretical characterization of prefix-tuning expressiveness, whereas Zweiger et al. (2026) develop efficient algorithms for approximate KV cache compaction. The success of attention matching in our work and the concurrent work of Zweiger et al. (2026) on a different application highlights its potential as a powerful and broadly applicable framework worthy of further study.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes].

   Justification: Our main claim is the complete resolution of the following problem: Given a pretrained autoregressive Transformer, for any input-output sequence pair $(\mathtt{x}, \mathtt{y})$, does there always exist a prefix $\mathcal{S}$ such that prepending $\mathcal{S}$ to $\mathtt{x}$ induces the model to generate the target output $\mathtt{y}$? This is completely answered by our Theorem 2.1.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes].

   Justification: We make it clear that the situation with two or more target input-output pairs is out of reach due to the lack of a proper definition of consistency.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes].

Justification: The full set of assumptions are made explicit in the statement of Theorem 2.1, and the proof follows from the main text and Appendix D.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes].

Justification: We have provided all experimental details as well as the source code for reproducibility.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes].

   Justification: Our source code is provided in the supplemental material.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes].

   Justification: All these details are provided in Section 5 and 6.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes].

   Justification: Our theory suggests that we should achieve nearly perfect success rate, except rare cases in which it fails due to numerical difficulties. This is precisely what our experimental results show.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes].

   Justification: We have included these details in Section 5 and 6.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes].

   Justification: Our work is fully compliant with the code of ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes].

    Justification: We have discussed the possibility of our algorithm being used for LLM attack in Appendix C.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A].

    Justification: We do not provide model or dataset in our work.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes].

    Justification: We have properly credit the use of assets from other parties (pretrained GPT-2, pretrained Qwen3-4B, data constructed by Gemini Flash, etc.).

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes].

    Justification: Our code is well documented.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A].

    Justification: Our work does not involve human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A].

    Justification: We are not involved.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes].

    Justification: Gemini Flash is used to construct the evaluation data in Section 5.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.