Where You Backpropagate Matters: Token Hypothesis for Memory-Efficient Fine-Tuning

Md Kowsher[1,2*] **Chen Chen**[2]
[1]Meta, [2]University of Central Florida

Project Page Code

Abstract

Fine-tuning large transformers routes gradients through every token position, paying the cost in activation memory that scales linearly with sequence length and dominates the GPU budget. Existing efficient methods reduce the parameter side—low-rank adapters, prefix tuning, weight-decomposed updates—but leave the activation footprint untouched. We present the Token Hypothesis, a memory-efficient fine-tuning procedure motivated by a Neural Tangent Kernel analysis of how fine-tuning signal is distributed across token positions. We prove that every position is a low-dimensional capacity bottleneck whose width does not grow with the model, that positions with aligned hidden-state geometry are interchangeable in bidirectional models, and that causal positions form a monotone capacity chain in which later positions strictly dominate earlier ones. The analysis provides an assumption-qualified criterion for estimating how many positions are needed and which ones, using quantities computable from forward passes alone: in bidirectional models, small arbitrary subsets are predicted to suffice when position-wise kernels are aligned, while in causal models, later positions are predicted to carry the richest fine-tuning signal. We translate this into a procedure that backpropagates through only a small subset of positions while running the full forward pass over the rest. Empirically, the procedure matches or exceeds full-sequence fine-tuning in nearly all settings across six task families—commonsense reasoning, mathematical reasoning, visual question answering, video question answering, GLUE, and image classification—and five backbones spanning bidirectional and causal architectures. It reduces activation memory substantially, translating to up to $\sim 19\%$ lower total GPU memory ($\sim$ 31 GB) on visual question answering while exceeding full-sequence accuracy, and composes cleanly with parameter-efficient methods such as LoRA and DoRA.

1 Introduction

Fine-tuning a pretrained transformer backpropagates the loss through every token position of every training sequence [45]. The resulting memory cost has three sources: **(i)** model parameters, **(ii)** gradients and optimizer states, and **(iii)** intermediate activations [61]. Most prior work on parameter-efficient fine-tuning targets the first two through low-rank adapters [19], prefix tuning [32], and related techniques [26], yet activations dominate GPU memory in practice [61, 51]. Activations scale linearly with sequence length T: fine-tuning an 8B-parameter model at $T = 2048$ consumes over 100 GB of activation memory alone.

A natural way to reduce this cost is to backpropagate through only a subset of token positions, and recent methods [51, 61, 44] have pursued increasingly sophisticated criteria for *which* tokens

*This work was not conducted at Meta.

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

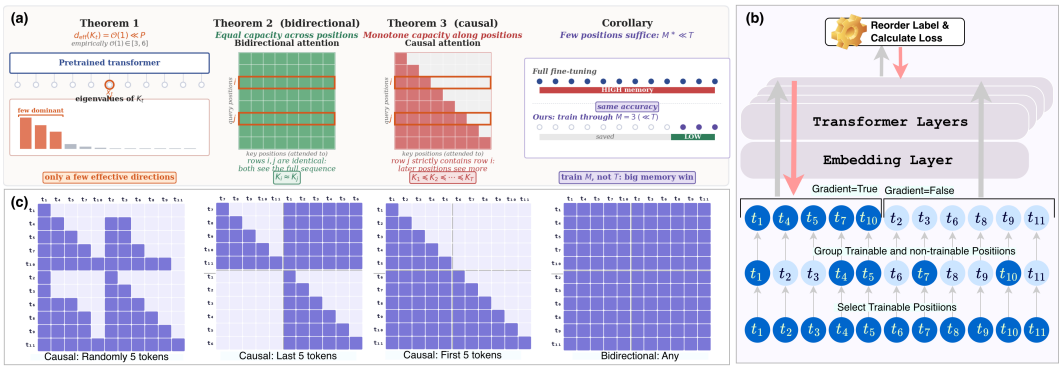

Figure 1: **The Token Hypothesis and our training procedure. (a)** Three theorems and a corollary characterize how fine-tuning signal is distributed across token positions: capacity per position is bounded, equal across positions in bidirectional models, and monotonically increasing in causal models—so a small subset $M \ll T$ of positions suffices to match full fine-tuning. **(b)** We select M trainable positions, group them against the non-trainable ones, and reorder the sequence so that only the trainable group receives gradients while the rest pass forward without storing activations. **(c)** Attention masks for four selection strategies, shown after reordering tokens so trainable positions are contiguous: *(i) Random* selects M positions uniformly at random each epoch; *(ii) Last-M* selects the final M positions of the sequence (optimal for causal models); *(iii) First-M* selects the initial M positions (baseline for causal models); *(iv) Any subset* selects any M positions in bidirectional models, where all positions are interchangeable. Darker rows mark trainable queries.

to select. We argue this framing skips the prior question: what determines a single position's capacity to drive fine-tuning, and how does that capacity depend on architecture? Once these are answered, both *how many* and *which* positions to train follow directly. We answer both theoretically and empirically through what we call the *Token Hypothesis* through a Neural Tangent Kernel (NTK) analysis. Training through a single token position confines every parameter update to a low-dimensional subspace whose size is set at initialization and does not grow with the model. Concretely, let $\theta \in \mathbb{R}^P$ denote the pretrained parameters initialized at θ_0, and let $o_t(x;\theta)$ denote the model output at position t for input x. Given a fine-tuning dataset $\mathcal{D} = \{(x^{(n)}, y^{(n)})\}_{n=1}^N$, define the position-$t$ feature map $\phi_t(x) := \nabla_\theta o_t(x;\theta_0) \in \mathbb{R}^P$ and Gram matrix $K_t \in \mathbb{R}^{N\times N}$ with $K_t[n,m] := \langle \phi_t(x^{(n)}), \phi_t(x^{(m)}) \rangle$. Fine-tuning through position t alone then confines updates to the span of $\{\phi_t(x^{(n)})\}_{n=1}^N$, a subspace of dimension at most $\mathrm{rank}(K_t) \le N$, independent of P. The effective rank $d_{\mathrm{eff}}(K_t) := \mathrm{tr}(K_t)/\|K_t\|_{\mathrm{op}}$, which counts how many gradient directions actually carry signal, is tighter still: across RoBERTa-base, SmolLM2-135M, and LLaMA-3-8B, we measure $d_{\mathrm{eff}}(K_t) \in [1.5, 5.2]$ at every tested position and dataset size (Figures 4, 14, 15). This single-position bottleneck raises three questions, which we resolve in turn.

Are all bottlenecks equally wide? In bidirectional models, two positions i and j that hold fixed tokens across samples and share aligned hidden-state geometry produce nearly identical fine-tuned predictors, with the gap controlled by quantities computable from forward passes alone (Theorem 2). Choosing among such positions therefore reduces to sampling. *What about positions with different information access?* In a causal model, under a contraction condition relating adjacent hidden states, the position-specific Gram matrices form a monotone chain $K_1 \preceq K_2 \preceq \cdots \preceq K_T$ (up to a vanishing slack term), so later positions carry at least as rich a gradient signal as earlier ones (Theorem 3). The natural trainable subset is therefore the last M positions. *How many positions are needed?* The minimum number M^* of positions whose combined Gram matrix spans the same image as the full sequence is governed by the architecture (Corollary 1). In bidirectional models, any small subset works whenever the position-wise Gram matrices are close in operator norm. In causal models the answer depends on the ordering: under strict prefix inclusion, the last single position spans the entire prediction space, while the forward ordering requires all T. Training the last M positions is therefore both architecturally principled and provably efficient. Every quantity in these bounds is measurable from the pretrained model with no fine-tuning (Figure 1(a)).

These results yield a memory-efficient fine-tuning procedure. Given a sequence of length T and a target $M \le T$, we select a subset $\mathcal{S} \subseteq \{1, \ldots, T\}$ of size M—any subset for bidirectional models,

the last M positions for causal models—and backpropagate only through $\mathcal{S}$ while running the full forward pass over all T positions. Positions outside $\mathcal{S}$ are evaluated inside a `no_grad` context (Algorithm 1), so their activations are never materialized. Peak activation memory is reduced by a fraction $1 - M/T$, the forward representation is unchanged, and inference is unaffected. The procedure composes with LoRA [19] and DoRA [34], adding activation-memory savings on top of the gradient and optimizer-state savings these methods already provide (Figure 1(b)).

Empirically, a small budget of trainable positions—M=25–100 for bidirectional models and M=100–500 for causal models—matches or exceeds full-sequence fine-tuning in nearly all settings across commonsense reasoning, mathematical reasoning, visual question answering, video reasoning, GLUE, and image classification, spanning RoBERTa-base, LLaMA-3-8B, Qwen3-VL-8B, LLaVA-OneVision-7B, and ViT-L/16. The empirical optimum lies above the worst-case M^* from Corollary 1, reflecting a practical preference for stable optimization over the minimal spanning subset.

Contributions. **(i)** We show that each token position is a capacity bottleneck with $\mathrm{rank}(K_t) \leq \min(N, P)$, independent of parameter count, and empirically observe $d_{\mathrm{eff}}(K_t) \in [1.5, 5.2]$ across tested models. **(ii)** We establish equal-capacity behavior for aligned bidirectional positions and monotone capacity for causal positions, yielding architecture-dependent selection rules. **(iii)** We characterize M^* as the smallest position budget whose combined Gram matrix spans the full prediction space, with all required quantities computable from forward passes. **(iv)** We turn this analysis into a fine-tuning procedure that matches or exceeds full-sequence fine-tuning across six task families and five architectures, while substantially reducing activation memory and lowering total GPU memory by up to 19% in measured long-context settings.

Related Work. We briefly discuss related work here and provide a detailed literature review in Appendix F. Most prior work on memory-efficient fine-tuning either reduces the number of trainable parameters [19, 34, 32, 26], reversibility, or quantization [6, 13, 10, 37], leaving the number of positions in the backward graph untouched. The line closest to ours selects a subset of token positions to backpropagate through [51, 61], but does so heuristically—random selection or attention-and-gradient scoring with a mandatory pre-stage and per-model hyperparameters—without explaining why a small subset suffices or how the optimal size depends on architecture. We use the NTK regime [21, 38, 1] as the theoretical lens for our analysis in large pretrained models. Within this regime, we ask a question prior work has not: how does the fine-tuning capacity of a *single token position* depend on architecture, and how few positions suffice to span the same achievable prediction space as the full sequence? The answer turns token selection from a theoretically heuristic into an empirical recipe.

2 Preliminaries

Let $f_\theta : \mathcal{V}^T \to \mathbb{R}^{T \times C}$ be a large pretrained transformer with vocabulary $\mathcal{V}$ and D layers, producing per-position outputs $o_t(x; \theta) := f_\theta(x)_t \in \mathbb{R}^C$ from token and positional embeddings $h_t^{(0)}(x) = E_{\mathrm{tok}}[x_t] + E_{\mathrm{pos}}[t]$. In a *bidirectional* model o_t depends on all positions; in a *causal* model it depends only on the prefix $(x_1, \ldots, x_t)$. For a subset $\mathcal{S} \subseteq \{1, \ldots, T\}$ of *trainable positions*, only positions in $\mathcal{S}$ contribute to parameter updates through the masked loss

$$\mathcal{L}_{\mathcal{S}}(\theta) \;:=\; \frac{1}{|\mathcal{S}|\, N} \sum_{n=1}^{N} \sum_{t \in \mathcal{S}} \ell\big(o_t(x^{(n)}; \theta),\, y^{(n)}\big),$$

with $\mathcal{L}_t$ denoting the single-position case $\mathcal{S} = \{t\}$. The gradient $\nabla_\theta \mathcal{L}_{\mathcal{S}}$ decomposes as a sum of position-wise terms over $t \in \mathcal{S}$, so only activations on backward paths from $t \in \mathcal{S}$ need to be stored—this is the source of the memory savings (§ 4).

Our theoretical analysis is carried out in the NTK regime for fine-tuning large-scale models around θ_0 [1, 38], under which gradient descent on $\mathcal{L}_t$ is equivalent to kernel regression with kernel function $\mathcal{K}_t(x, x') := \langle \phi_t(x), \phi_t(x') \rangle$. We empirically verify that this regime holds for the models we study: mean weight drift stays below 0.011 across every parameter group of LLaMA 3-8B during fine-tuning (Figure 28(d)), justifying the use of θ_0-Jacobians throughout. The corresponding Gram matrix $K_t \in \mathbb{R}^{N \times N}$ on $\mathcal{D}$ has entries $K_t[n, m] := \mathcal{K}_t(x^{(n)}, x^{(m)})$. We use $\mathcal{K}_t$ for the kernel function and K_t for its Gram matrix throughout.

3 The Token Hypothesis

We formalize the Token Hypothesis through three theorems characterizing the fine-tuning capacity of individual token positions under the NTK regime: every position is a bounded capacity bottleneck (Theorem 1), bidirectional positions with shared information access yield nearly identical predictors (Theorem 2), and in causal models, position-wise capacity is monotone (Theorem 3). We use the NTK analysis as a predictive design model rather than an exact description of fine-tuning dynamics: it suggests architecture-dependent selection rules—small arbitrary subsets for bidirectional models with aligned position-wise kernels, and last-M subsets for causal models—that we test empirically in Section 5. The theorems characterize capacity, not optimization dynamics, so the empirical choice of M reflects optimization and stability considerations on top of the predicted capacity structure. Full proofs and additional empirical validation are in the appendix.

Each Position Has Limited Capacity. Recall from Section 2 that fine-tuning through position t alone under the NTK regime is equivalent to kernel regression with kernel K_t. Every gradient update is therefore a linear combination of the features $\{\phi_t(x^{(n)})\}_{n=1}^N$, which confines all parameter changes to the subspace $\mathcal{G}_t := \mathrm{span}\{\phi_t(x^{(1)}), \ldots, \phi_t(x^{(N)})\} \subseteq \mathbb{R}^P$, whose dimension is at most $\mathrm{rank}(K_t) \leq \min(N, P)$. The ceiling $\min(N, P)$ is the standard representer bound for kernel regression on N samples and is loose whenever the spectrum of K_t is concentrated on a few directions. A tighter quantity is the *effective rank* $d_{\mathrm{eff}}(K_t) := \mathrm{tr}(K_t)/\|K_t\|_{\mathrm{op}} \in [1, \mathrm{rank}(K_t)]$, which counts how many gradient directions actually carry signal during training. The following theorem makes the underlying subspace structure precise.

Theorem 1: Limited capacity per position

Under the NTK regime, the set of predictions achievable by training through position t on $\mathcal{D} = \{(x^{(n)}, y^{(n)})\}_{n=1}^N$ is the affine subspace

$$\left\{ f_t(X; y) : y \in \mathbb{R}^N \right\} = o_t(X; \theta_0) + \mathrm{Im}(K_t),$$

where $X := (x^{(1)}, \ldots, x^{(N)})$ and $\mathrm{Im}(K_t)$ is the column space of K_t, with $\mathrm{rank}(K_t) \leq \min(N, P)$. The learned correction takes the representer form

$$g_t(x) = \sum_{n=1}^{N} \alpha_n \mathcal{K}_t(x, x^{(n)}), \qquad \alpha = K_t^+ \big(y - o_t(X; \theta_0)\big),$$

lying in the reproducing kernel Hilbert space (RKHS) $\mathcal{H}_{\mathcal{K}_t}$.

The proof, given in Appendix A.1, proceeds by showing that gradient descent on $\mathcal{L}_t$ is equivalent to kernel gradient flow in $\mathcal{H}_{K_t}$, whose minimum-norm solution is the representer expansion above. The theorem is position-specific through the feature map $\phi_t = \nabla_\theta o_t(\cdot; \theta_0)$: different positions induce different kernels, and Theorems 2 and 3 characterize how K_t varies with t across architectures. Empirically, $d_{\mathrm{eff}}(K_t)$ stays within $[1.5, 5.2]$ across every tested position, dataset size, and model scale (Section 5 and Appendix A.2), tightening the worst-case ceiling by orders of magnitude.

Equal Capacity Across Tokens. Two positions that attend to the same context tend to induce similar gradient structures, and hence similar fine-tuned predictors, whereas positions with different information access need not. Bidirectional models provide the natural setting for this comparison, since every position attends to the full sequence. The result below makes this intuition precise: for any two positions i and j whose tokens are fixed across samples and whose Gram matrices share a common column space, the gap between the corresponding fine-tuned predictors is bounded by quantities computable from the pretrained model.

We work in the *representation NTK regime*: at a fixed layer ℓ following an attention layer, the feature map is the pretrained hidden state $\phi_t(x) = h_t^{(\ell)}(x) \in \mathbb{R}^d$, and the Gram matrix

$$K_t[n, m] = \langle h_t^{(\ell)}(x^{(n)}),\, h_t^{(\ell)}(x^{(m)}) \rangle$$

is computable from a single forward pass. This regime models training as a linear readout from pretrained features; Appendix E.4 verifies that real fine-tuning stays close to it (mean weight drift below 0.011). Define the kernel-evaluation vector $k_t(x) \in \mathbb{R}^N$ and the residual $r_t \in \mathbb{R}^N$ with entries

$$k_t(x)[n] := \langle h_t^{(\ell)}(x),\, h_t^{(\ell)}(x^{(n)}) \rangle, \qquad r_t := y - o_t(X; \theta_0).$$

The capacity gap between two positions i and j is then controlled by four measurable quantities, all computed without any fine-tuning:

Gram matrix difference:	$\delta_K := \|K_i - K_j\|_{\mathrm{op}}$,
kernel vector difference:	$\delta_k(x) := \|k_i(x) - k_j(x)\|_2$,
residual difference:	$\delta_r := \|r_i - r_j\|_2$,
initial output gap:	$\delta_{\mathrm{init}}(x) := \|o_i(x;\theta_0) - o_j(x;\theta_0)\|_2$.

Theorem 2: Equal capacity under shared information access

Let f_i and f_j be the representation NTK predictors (scalar output, $C = 1$) obtained by training via gradient masking at positions i and j on $\mathcal{D}$, with $x_i^{(n)} = s_i$ and $x_j^{(n)} = s_j$ for all $n = 1, \ldots, N$. Suppose K_i and K_j share the same column space, and let $\lambda_0^+ := \min(\lambda_{\min}^+(K_i),\, \lambda_{\min}^+(K_j)) > 0$ denote the smallest nonzero eigenvalue of either. Then for all test inputs x with $x_i = s_i$ and $x_j = s_j$,

$$\|f_i(x) - f_j(x)\|_2 \;\le\; \delta_{\mathrm{init}}(x) \;+\; \frac{\delta_k(x)\, R}{\lambda_0^+} \;+\; \frac{\|k_j(x)\|_2\, \delta_K\, R}{(\lambda_0^+)^2} \;+\; \frac{\|k_j(x)\|_2\, \delta_r}{\lambda_0^+},$$

where $R := \|r_i\|_2$. Every quantity on the right-hand side is computable from the pretrained model.

The proof is given in Appendix B.1, and empirical validation is provided in Section 5 and Appendices B.2 and B.3.

Monotone Capacity in Causal Models. Theorems 1 and 2 established that every position is a bounded bottleneck and that positions with aligned information access are equivalent. What remains is the causal case, where each position attends only to its prefix: as the position index increases, the prefix grows, and position $t{+}1$ has access to information that position t does not. Causal attention provides the mechanism by which information at position t enters position $t{+}1$'s representation: at every layer, the update at $t{+}1$ reads from positions $1, \ldots, t{+}1$. This shared lineage links the hidden states at adjacent positions through a regularity that we formalize below. We show that this expanding information access translates into a monotone fine-tuning capacity, up to a slack term that vanishes when the residual $\sigma_t \to 0$, in the representation NTK regime where $K_t[n, m] = \langle h_t^{(\ell)}(x^{(n)}), h_t^{(\ell)}(x^{(m)})\rangle$. The positive semidefinite ordering $K_i \preceq K_j$ is the natural notion of capacity ordering: when it holds exactly, $\mathrm{Im}(K_i) \subseteq \mathrm{Im}(K_j)$, so every labeling of the training set achievable via position i is also achievable via position j.

Theorem 3: Monotone capacity in causal models

Suppose that for each $t \in \{1, \ldots, T{-}1\}$ there exists a matrix $A_t \in \mathbb{R}^{d\times d}$ with $\|A_t\|_{\mathrm{op}} \le 1$, **shared uniformly across all inputs**, and residuals $\varepsilon_t(x^{(n)}) \in \mathbb{R}^d$ such that the hidden states at a fixed layer ℓ of a pretrained causal transformer f_θ satisfy

$$h_t^{(\ell)}(x^{(n)}) \;=\; A_t\, h_{t+1}^{(\ell)}(x^{(n)}) + \varepsilon_t(x^{(n)}) \qquad \text{for all } n = 1, \ldots, N.$$

Define $\sigma_t := \max_n \|\varepsilon_t(x^{(n)})\|_2$ and $\beta_{t+1} := \max_n \|h_{t+1}^{(\ell)}(x^{(n)})\|_2$. Then for every $t \in \{1, \ldots, T{-}1\}$,

$$K_{t+1} \;\succeq\; K_t - \delta_t\, I_N, \qquad \delta_t := N\sigma_t(2\beta_{t+1} + \sigma_t),$$

where σ_t and β_{t+1} are computable from N forward passes. The bound is informative whenever $\delta_t \ll \lambda_{\min}(K_t)$.

The substantive restriction is the uniformity of A_t across inputs, and the bound is non-trivial when $\sigma_t \ll \beta_{t+1}$. Empirically, $\|A_t\|_{\mathrm{op}} \approx 1$ across the bulk of layers, $\sigma_t/\beta_{t+1} \lesssim 10^{-14}$, and $\delta_t \ll \lambda_{\min}^+(K_t)$ by at least six orders of magnitude on SmolLM2-135M and at least seven on LLaMA 3-8B (Appendices C.2 and C.3). When $\sigma_t = 0$ the bound holds with equality, recovering the exact chain $K_1 \preceq K_2 \preceq \cdots \preceq K_T$. The proof is given in Appendix C.1, and empirical validation in Section 5.

4 Memory-Efficient Fine-Tuning via the Token Hypothesis

The three theorems of Section 3 jointly imply that fine-tuning through every position is unnecessary: capacity is bounded per position (Theorem 1), equivalent across aligned positions (Theorem 2), and monotone in causal models (Theorem 3). A minimal number of positions M^* therefore suffices to span the same achievable prediction space as the full sequence, and its value depends on the architecture. Given an ordering $t_1, t_2, \ldots, t_T$ of the positions, let $\mathcal{S}_M := \{t_1, \ldots, t_M\}$ and define the combined Gram matrix $K^{(M)} := \sum_{m=1}^{M} K_{t_m} \in \mathbb{R}^{N\times N}$. Since $\mathrm{Im}(K^{(M)})$ is monotone non-decreasing in M (adding a PSD matrix cannot shrink the image), the minimum number of positions required to reach the full image is well-defined.

Corollary 1: Optimal position count

Under the conditions of Theorems 1–3 and in the output-head NTK regime, the quantity

$$M^* := \min\{\, M : \mathrm{Im}(K^{(M)}) = \mathrm{Im}(K^{(T)}) \,\}$$

satisfies $1 \le M^* \le T$, and its value depends on both architecture and ordering:

(i) Bidirectional. If $\|K_i - K_j\|_{\mathrm{op}} \le \epsilon_K$ for all i, j and $\lambda^+ := \lambda^+_{\min}(K^{(T)})$ satisfies $\epsilon_K < \lambda^+/(2T)$, then $M^* \le \lceil \epsilon_K T/\lambda^+ + 1 \rceil$, with $M^* = 1$ in the exact case $\epsilon_K = 0$.

(ii) Causal, last-to-first ordering. Under Theorem 3 with strict inclusion $\mathrm{Im}(K_t) \subsetneq \mathrm{Im}(K_{t+1})$, the ordering $t_1 = T$, $t_2 = T-1, \ldots$ gives $M^* = 1$: the single last position spans the same prediction space as all T positions combined.

(iii) Causal, first-to-last ordering. Under the same conditions, the forward ordering $t_m = m$ gives $M^* = T$: every position adds a strictly new direction.

Here M^* is the smallest number of positions at which all prediction directions reachable with the full sequence become reachable at convergence, whereas the number of positions required under finite-step training additionally depends on the conditioning of $K^{(M)}$, which in our measurements continues to improve well beyond M^*. The proof is given in Appendix D. All three quantities ϵ_K, λ^+, and the monotone chain of images are measurable from forward passes alone. In practice M^* is identified without fine-tuning through an *image-saturation criterion*: find the smallest M such that $\|P_{K^{(T)}} - P_{K^{(M)}} P_{K^{(T)}}\|_F / \|P_{K^{(T)}}\|_F \le \varepsilon$, where P_K is the orthogonal projector onto $\mathrm{Im}(K)$. Trace-based criteria are unreliable here: $\mathrm{tr}(K^{(M)})$ grows with M even when all K_t are identical, so it cannot detect image saturation.

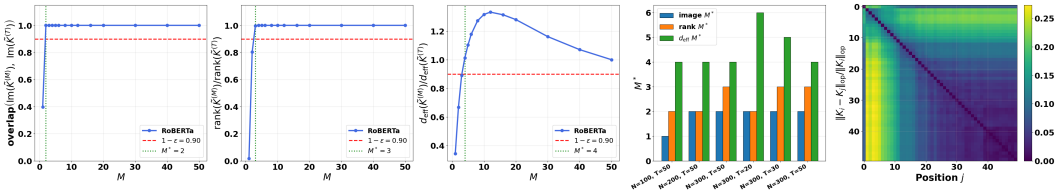

Figure 2: Image-space saturation on RoBERTa-base (bidirectional, N=300, T=50). **(a)** Image-space overlap crosses the $1-\varepsilon = 0.90$ threshold at $M^* = 2$. **(b)** Rank saturation at $M^* = 3$. **(c)** Effective rank $d_{\mathrm{eff}}(K^{(M)})$ saturates at $M^* = 4$. **(d)** M^* remains stable across (N, T), indicating an architecture-intrinsic threshold. **(e)** Pairwise relative Gram distance $\|K_i - K_j\|_{\mathrm{op}} / \|K_i\|_{\mathrm{op}}$ has mean 0.15 and max 0.27, supporting $\epsilon_K \ll 1$.

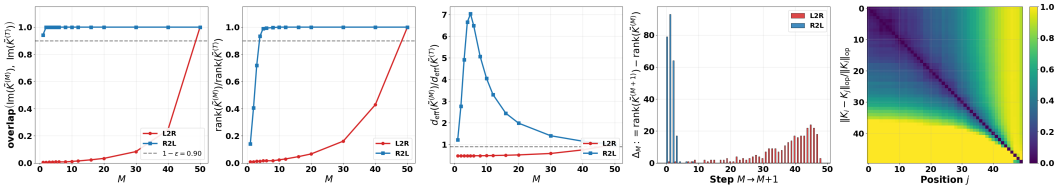

Figure 3: Image-space saturation on SmolLM2-135M (causal, N=300, T=50). **(a)** Forward ordering (L2R) reaches $\mathrm{Im}(K^{(T)})$ only at $M = T$; reverse ordering (R2L) reaches it at $M = 1$. **(b)** Rank saturation: L2R at $M^* = T$, R2L at $M^* = 4$. **(c)** Effective rank $d_{\mathrm{eff}}(K^{(M)})$ for both orderings. **(d)** Per-step rank increment $\Delta_M = \mathrm{rank}(\tilde{K}^{(M+1)}) - \mathrm{rank}(\tilde{K}^{(M)})$: under R2L, nearly all new directions are contributed by the first three positions, consistent with rank saturation at M^*=4; under L2R, increments are small and distributed across the sequence, growing toward its end. **(e)** Pairwise relative Gram distance shows a strong gradient, supporting strict inclusion $\mathrm{Im}(K_t) \subsetneq \mathrm{Im}(K_{t+1})$.

Figures 2 and 3 empirically confirm Corollary 1. On RoBERTa-base (bidirectional), all three saturation metrics cross the $1-\varepsilon = 0.90$ threshold at $M^* \le 4$, and the value of M^* remains stable as (N, T) vary—so a handful of positions suffices regardless of dataset size. On SmolLM2-135M (causal), the reverse ordering saturates immediately while the forward ordering requires the full sequence, directly matching the two causal cases of the corollary; kernel-ridge accuracy under R2L peaks at $M = 1$, while L2R continues improving up to $M = T$.

Corollary 1 translates into a memory-efficient fine-tuning recipe. Given T positions, we partition $\{1, \ldots, T\}$ into a trainable set $\mathcal{S}$ of size M and its complement $\bar{\mathcal{S}}$. The choice of $\mathcal{S}$ follows the

architecture: in *bidirectional* models, any subset of size $M \geq M^*$ achieves image equality with $K^{(T)}$, so we use a uniformly random subset resampled each epoch; in *causal* models, the last-M strategy $\mathcal{S} = \{T-M+1, \ldots, T\}$ is optimal under both the corollary and Theorem 3 (i.e., $\mathcal{S} = \mathcal{S}_M$ under the reverse ordering). The forward pass is unchanged: at each layer we compute $\hat{x}_t = Wx_t$ for every position, but retain gradients only for $t \in \mathcal{S}$, while positions $t \in \bar{\mathcal{S}}$ are computed inside a `no_grad` context. Their activations are therefore never materialized in the computation graph, reducing peak activation memory by a fraction $1 - M/T$. When composed with LoRA [19], trainable positions additionally receive the adapter update while frozen positions do not: $\hat{x}_t = Wx_t + (\alpha/r)\, BAx_t$ for $t \in \mathcal{S}$ and $\hat{x}_t = Wx_t$ for $t \in \bar{\mathcal{S}}$, where $A \in \mathbb{R}^{r \times d}$, $B \in \mathbb{R}^{d \times r}$ are the rank-r adapter factors and α is the scaling factor. The same construction applies to DoRA [34]. After training, the gradient-splitting mechanism is discarded and any low-rank updates are merged into W, leaving the original architecture with no inference overhead. In practice M can be used as a hyperparameter.

5 Experiments

We evaluate the Token Hypothesis across six task modalities—commonsense reasoning, mathematical reasoning, visual question answering, video question answering, the GLUE benchmark, and image classification. Dataset descriptions and preprocessing are provided in Appendix G. We use five pretrained backbones: LLaMA-3-8B, Qwen3-VL-8B, LLaVA-OneVision-7B, RoBERTa-base, and ViT-L/16-224, with details in Appendix I. Each modality is evaluated under full fine-tuning and a parameter-efficient setting (LoRA or DoRA), and every experiment sweeps the number of trainable positions M against a full-sequence baseline (*All*). Within each (modality, regime) pair, the baseline and all M-swept runs share identical learning rate, batch size, gradient accumulation, warmup schedule, mixed-precision setting, epoch count, and GPU configuration; the only variable is M. Hyperparameter details are provided in Appendix H.

Table 1: Unified comparison of *average* results across four modalities (Commonsense, Math, VQA, Video) and three fine-tuning regimes (Full FT, LoRA, DoRA) vs. number of trainable tokens. Each block reports mean accuracy, total GPU memory summed across all devices (GB), and training time (min). VQA and Video results use Qwen3-VL-8B. Commonsense FT averages five benchmarks and Commonsense DoRA averages four. **Bold** = best value per column; ties are bolded jointly. Detailed results are shown in Tables 11, 20, 16, 13, 14, 17, and 18.

Tokens	Commonsense FT			Commonsense DoRA			Math FT			Math DoRA			VQA FT			VQA LoRA			Video FT			Video LoRA		
	Acc	*GPU*	*Time*	*Acc*	*GPU*	*Time*	*Acc*	*GPU*	*Time*	*Acc*	*GPU*	*Time*	*Acc*	*GPU*	*Time*	*Acc*	*GPU*	*Time*	*Acc*	*GPU*	*Time*	*Acc*	*GPU*	*Time*
All	0.831	95.28	23.85	0.744	50.20	9.31	0.829	95.38	20.62	0.774	55.59	21.31	0.728	168.07	13.29	0.669	88.12	10.95	0.604	113.09	24.53	**0.579**	56.87	23.15
10	0.744	**83.69**	**19.35**	0.687	**49.24**	**9.23**	0.399	**83.93**	**16.73**	0.494	**49.70**	20.93	0.711	**135.29**	11.69	0.663	**76.22**	9.48	0.574	**101.03**	**21.29**	0.531	**49.78**	20.07
25	0.801	83.91	19.45	0.697	49.30	9.24	0.576	84.15	17.02	0.602	49.79	20.90	0.711	135.58	**11.30**	0.652	76.31	9.57	0.562	101.44	21.31	0.536	49.93	**20.00**
50	0.831	84.63	19.81	0.736	49.43	9.32	0.695	84.61	16.98	0.687	49.94	**20.84**	0.720	136.06	11.44	0.665	76.46	**9.18**	0.574	102.08	21.87	0.528	50.17	20.28
100	0.834	85.85	20.49	0.744	49.62	9.36	0.790	85.34	17.57	0.773	50.26	21.15	0.736	136.82	11.60	0.659	76.74	9.96	0.574	104.02	22.28	0.497	50.60	20.38
300	**0.838**	88.87	22.04	**0.747**	50.38	9.46	**0.834**	88.35	19.13	**0.790**	51.41	21.45	**0.739**	140.13	12.48	**0.676**	77.86	9.94	0.588	108.62	23.74	0.540	52.37	20.66
500	0.830	91.54	23.74	0.744	51.20	9.58	0.833	91.49	20.72	0.782	52.60	21.84	0.735	143.45	13.20	0.675	78.98	9.81	**0.609**	109.12	22.51	0.535	53.30	20.44
700	0.831	94.21	25.43	0.745	51.95	9.70	0.828	94.47	22.88	0.786	53.72	22.05	0.731	146.72	13.55	0.668	80.07	9.68	0.593	109.35	22.89	0.550	53.52	20.79
1000	0.832	96.62	27.16	0.746	52.72	9.79	**0.834**	97.50	25.15	0.782	54.87	22.52	0.724	151.70	13.68	0.663	81.77	9.70	0.597	109.70	22.48	0.563	53.84	20.61

Table 2: Accuracy comparison across six benchmarks on LLaMA-3.2-3B for varying numbers of trainable tokens. TT = Token Tuning, TS = TokenSeek, TH = Token Hypothesis. GPU = peak allocated memory (GB), shared across the three methods at each token count. Highest accuracy in each cell is shown in **bold**.

	ARC-Chal.			ARC-Easy			BoolQ			HellaSwag			OpenBookQA			WinoGrande			
Tokens	TT	TS	TH	TT	TS	TH	TT	TS	TH	TT	TS	TH	TT	TS	TH	TT	TS	TH	**GPU**
10	0.258	**0.519**	0.504	0.256	0.764	**0.768**	0.622	0.651	**0.678**	0.438	0.683	**0.776**	0.364	**0.630**	0.616	0.496	0.705	**0.728**	29.99
25	0.470	**0.607**	0.528	0.693	**0.816**	0.812	0.622	0.614	**0.719**	0.491	0.247	**0.892**	0.468	**0.730**	0.664	0.496	0.760	**0.775**	29.99
50	0.486	**0.643**	0.635	0.653	0.808	**0.841**	0.622	0.714	**0.717**	0.593	0.255	**0.903**	0.448	0.714	**0.772**	0.496	0.810	**0.829**	29.99
100	0.420	**0.672**	0.665	0.734	0.855	**0.859**	0.622	**0.722**	0.721	0.762	0.455	**0.924**	0.316	**0.784**	0.782	0.656	0.830	**0.834**	30.00
300	0.570	0.661	**0.666**	0.787	**0.858**	0.856	0.651	0.713	**0.717**	0.864	0.931	**0.932**	0.676	0.786	**0.788**	0.681	0.837	**0.838**	35.82
500	0.622	0.658	**0.664**	0.807	0.854	**0.857**	0.684	0.709	**0.719**	0.887	**0.933**	0.932	0.722	**0.782**	**0.782**	0.800	0.838	**0.838**	42.79
700	0.659	**0.666**	0.664	0.824	**0.859**	0.857	0.695	0.714	**0.719**	0.907	0.932	**0.935**	0.730	**0.786**	0.780	0.781	0.840	**0.841**	49.73
900	0.658	**0.666**	0.658	0.849	0.853	**0.858**	0.714	**0.717**	0.716	0.930	**0.934**	0.933	0.766	**0.782**	**0.782**	0.841	0.846	**0.848**	56.68

Main results. Table 1 reports accuracy averaged across benchmarks within each modality and summarizes how accuracy, memory, and time vary with M across eight (modality, regime) pairs on

causal backbones. Three observations stand out. First, accuracy matches or exceeds the full-sequence baseline at M far smaller than the sequence length in seven of the eight (modality, regime) pairs, with the saturation point varying by task: commonsense and visual QA saturate at M=100–300, math reasoning at M=300, and video reasoning under full fine-tuning first exceeds the baseline at M=500. Video LoRA is the exception: average accuracy remains below the baseline at every M (best 0.563 at M=1000 versus 0.579), with the gap driven primarily by VideoMathQA. Empirically, the required budget grows with the length of the supervised span, being larger for multi-step reasoning and video inputs than for short-answer tasks. Second, the empirical saturation point lies above the minimal position count M^* of Corollary 1, which equals one for causal models under last-M selection; the practical optimum reflects a preference for stable optimization and well-conditioned kernels over the minimal spanning subset. Third, in the settings that saturate, accuracy remains essentially flat beyond the saturation point, indicating that the additional gradient signal from earlier positions, which are added as M grows under last-M selection, is largely redundant rather than harmful—in line with the monotone capacity ordering $K_t \preceq K_{t+1}$. We also observe parallel results on bidirectional backbones: GLUE on RoBERTa-base (Tables 21 and 22) and image classification on ViT-L/16-224 (Table 19) saturate at M=50–100, consistent with Theorem 2's prediction that small subsets of aligned positions suffice in the bidirectional case. Detailed per-benchmark results are in Appendix J.

The memory reduction is driven directly by restricting the backward graph to the M trainable positions: backward-graph activations are stored only for these positions, while the remaining $T - M$ positions execute under `torch.no_grad()` and never enter the gradient buffer. The absolute saving therefore grows with the number of frozen positions in each sequence. At M=300, VQA achieves the largest reduction in Table 1 (27.9 GB under full fine-tuning and 10.3 GB under LoRA), compared with 6.4 GB for commonsense, 7.0 GB for math, and 4.5 GB for video. The reduction is orthogonal to parameter-side compression: applying token masking on top of LoRA or DoRA further reduces total memory while preserving the parameter savings of the adapter, since the two mechanisms target disjoint components of the GPU footprint. Training time tracks memory closely, dominated by the per-step backward pass over the trainable positions.

Comparison. Table 2 compares the Token Hypothesis (TH) against TokenTune (TT) [51] and TokenSeek (TS) [61] on six commonsense-reasoning benchmarks at matched token budgets, with shared GPU memory across all three methods at each token count. The Token Hypothesis achieves the highest accuracy in the majority of cells across all token counts, and matches TokenSeek closely where it does not lead. The gap to TokenTune is large at small token counts (e.g., $+0.34$ on HellaSwag and $+0.25$ on OpenBookQA at M=10), confirming that random selection is far from optimal when the budget is tight. Against TokenSeek, the Token Hypothesis achieves comparable or better accuracy without any pre-stage scoring or per-model hyperparameter search: TokenSeek requires a forward and partial backward pass over the entire training set to compute per-sample importance scores, plus a grid search over its weights for each backbone. The Token Hypothesis uses the last-M rule directly, motivated by Theorem 3, which carries information from earlier high-importance tokens without training them. The training-time and memory cost is identical across the three methods at each M, so the comparison isolates the effect of the position-selection rule.

Empirical validation of Theorem 1. Figure 4 validates both claims of Theorem 1 on RoBERTa-base. The worst-case rank bound $\mathrm{rank}(K_t) \le \min(N, P)$ is visibly slack: d_{95} stays well below N at every position, so K_t never approaches the ceiling. The stronger claim—that the *effective* rank is far smaller still—is confirmed directly: $d_{\mathrm{eff}}(K_t) \in [1.5, 3.9]$ across all eight positions, so fine-tuning through a single position uses fewer than four effective directions. This narrow band persists as we vary the dataset size from 30 to 2,000 samples and as we move from RoBERTa-base (125M) to SmolLM2-135M and LLaMA-3-8B (Appendix A.2). The bottleneck is therefore a property of the pretrained kernel itself: fixed at initialization, independent of dataset size or model scale.

Empirical validation of Theorem 2. Figure 5 confirms Theorem 2 on RoBERTa-base with a fixed 12-token prefix prompt. All 12 positions yield $d_{\mathrm{eff}} \in [1.02, 1.12]$ with coefficient of variation $\mathrm{CV} = 0.02$, and $d_{95} \in \{1, 2\}$ at almost every position. When the token at a position is identical across all N samples, the position-t gradient has one dominant direction set by that fixed token's embedding, with context variation contributing only a small perturbation through bidirectional attention; this concentrates the kernel onto a single eigendirection. The resulting $\lambda_0^+ \approx \lambda_1(K_t) \approx \mathrm{tr}(K_t)$ is large, making the second and third terms of the bound negligible. Full results for $N \in \{50, 300, 1000\}$ and a parallel experiment with a fixed suffix are deferred to Appendix B.2.

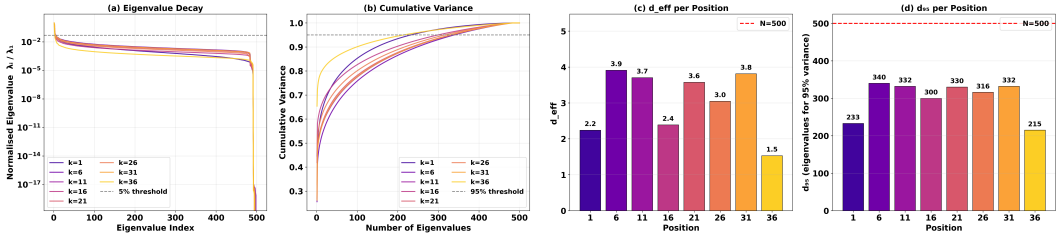

Figure 4: Empirical validation of Theorem 1 on RoBERTa-base (N=500, positions $t \in \{1, 6, 11, 16, 21, 26, 31, 36\}$; tokens at each position vary across samples). (a) Normalized eigenvalues of K_t decay gradually over most of the spectrum and drop by more than ten orders of magnitude near its tail, so K_t is numerically rank-deficient. (b) Cumulative variance crosses the 95% threshold at indices 215–340, below N=500. (c) The effective rank $d_{\text{eff}}(K_t)$ lies between 1.5 and 3.9 across all eight positions, indicating that the spectrum is dominated by a small number of leading directions. (d) d_{95}, the smallest number of top eigenvalues whose sum exceeds 95% of $\text{tr}(K_t)$, remains below N at every position.

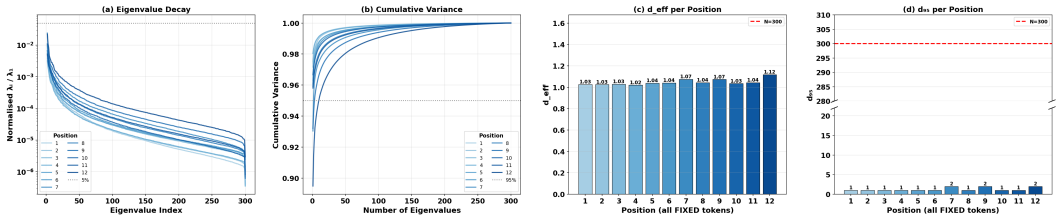

Figure 5: Empirical validation of Theorem 2 on RoBERTa-base (N=300; 12 fixed prefix-prompt positions). **(a)** Eigenvalue decay curves are nearly identical across all 12 positions, indicating $\delta_K \approx 0$. **(b)** Cumulative variance curves are indistinguishable, consistent with the shared column-space assumption. **(c)** $d_{\text{eff}} \in [1.02, 1.12]$ with CV $= 0.02$: every fixed position has essentially the same bottleneck width. **(d)** $d_{95} = 1$ at almost every position, confirming K_t is effectively rank-one and λ_0^+ is large, so the second and third terms of the bound are negligible.

We next test the prediction at the level of the fine-tuned predictor. Representation NTK predictors f_i and f_j are trained separately at ten token pairs spanning positions 1–8 of the same prompt, using 300 SST-2 training sentences and 50 held-out test sentences. Across all pairs, the mean prediction gap $\|f_i(x) - f_j(x)\|$ ranges from 18.0% to 27.5% of the label standard deviation $\hat{\sigma}_y$, and 86–98% of test sentences satisfy $\|f_i(x) - f_j(x)\| < 0.5\,\hat{\sigma}_y$. All ten pairs are statistically significant ($p < 0.001$) under a left-tail position-exchangeability permutation test with B=2000 resamples, confirming that the observed closeness reflects genuine equal capacity across positions rather than chance (Table 4 in Appendices B.2 and B.3).

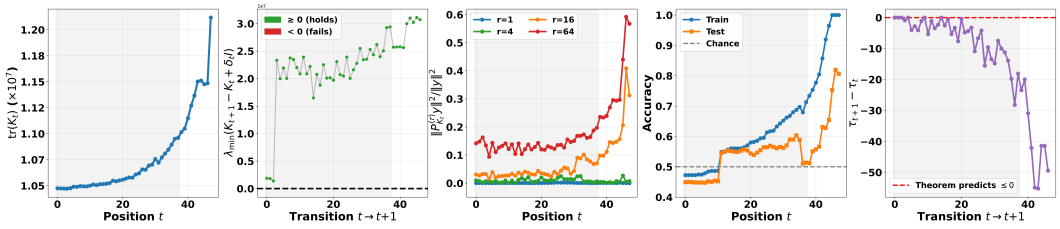

Figure 6: Empirical validation of Theorem 3 on LLaMA 3-8B (T=48, N_{train}=512, N_{test}=1024). **(a)** $\text{tr}(K_t)$ across positions. **(b)** $\lambda_{\min}(K_{t+1} - K_t + \delta_t I)$ at every transition $t \to t{+}1$. **(c)** Fraction of label variance captured by the top-r eigendirections of K_t, for $r \in \{1, 4, 16, 64\}$. **(d)** Training and test accuracy of a fine-tuned linear head reading from position t. **(e)** Increment $\tau_{t+1} - \tau_t$ of the prediction-error statistic $\tau_t := \mathbb{E}_x[\|y - K_t K_t^+ y\|^2]$, with the dashed line indicating the theorem's prediction $\tau_{t+1} \leq \tau_t$.

Empirical validation of Theorem 3. Figure 6 validates Theorem 3 along three complementary axes. The kernel-level claim $K_{t+1} \succeq K_t - \delta_t I$ is confirmed in two ways: $\mathrm{tr}(K_t)$ grows monotonically with position, and $\lambda_{\min}(K_{t+1} - K_t + \delta_t I) \geq 0$ at every transition, so the PSD bound holds along the entire sequence. The geometric consequence $\mathrm{Im}(K_t) \subseteq \mathrm{Im}(K_{t+1})$ shows up in eigendirection alignment with labels: the fraction of label variance captured by the top eigendirections of K_t grows with position, with the top 64 directions covering near zero at position 1 and 60% at position 45. Finally, the downstream prediction—that later positions admit strictly larger function classes—is confirmed by linear-probe behavior: held-out accuracy rises from near chance at position 1 to roughly 0.8 at position 45, and the prediction-error statistic τ_t decreases monotonically.

Ablations. We conduct five ablations to characterize the behavior of token-masked fine-tuning beyond the core accuracy-memory trade-off. *Batch-size scaling* (Appendix E.1) shows that activation memory grows linearly in the batch size with a slope proportional to M, so reducing M translates directly into a larger attainable batch size under a fixed memory budget—up to $3.6\times$ at batch size 64. *Sequence-length scaling* (Appendix E.2) shows that fixing $M{=}50$ while varying T largely decouples memory from sequence length: token-masked memory grows by only 2–3 GB from $T{=}100$ to $T{=}2048$, while the number of positions in the backward graph is reduced by $1 - M/T \approx 97.6\%$ at $T{=}2048$. *Convergence across M* (Appendix E.3) shows that on LLaVA-OneVision-7B, $M{=}50$–100 reaches or exceeds full-sequence accuracy on SafeTag and MMK12 within 1000 evaluation steps, while the reasoning-heavy M3CoT benefits from a larger budget, consistent with the bounded per-position capacity of Theorem 1. *NTK laziness across parameter groups* (Appendix E.4) verifies that weight drift stays at or below 0.011 across every parameter group and every value of M, empirically supporting the lazy regime Theorem 1 assumes. *Capability retention* (Appendix E.5) shows that fine-tuning with $M{=}100$ on ARC-C improves target accuracy while reducing forgetting on four untrained commonsense benchmarks by roughly $4\times$ relative to full fine-tuning, consistent with confining updates to a low-dimensional subspace of the pretrained tangent space.

6 Conclusion

We introduced the Token Hypothesis, a Neural Tangent Kernel analysis of how fine-tuning capacity is distributed across token positions. The analysis shows that each position acts as a low-dimensional capacity bottleneck, that aligned positions in bidirectional models have similar capacity, and that position-wise capacity increases along causal sequences. These predictions lead to a simple memory-efficient training procedure: backpropagate through only a small subset of positions while preserving the full forward pass. Empirically, a small number of trainable positions, ranging from fifty to five hundred depending on the task, matches or exceeds full-sequence fine-tuning in nearly all settings across six task families and five backbones, substantially reducing activation memory, enabling nearly four times larger batches, and composing cleanly with LoRA and DoRA. Together, the theory and experiments suggest that memory-efficient fine-tuning should be understood not only as a question of *which* tokens to select, but also *why* a small position subset can carry sufficient fine-tuning signal.

Acknowledgments and Disclosure of Funding

This work used Bridges-2 at the Pittsburgh Supercomputing Center through an allocation from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program, which is supported by U.S. National Science Foundation grants #2138259, #2138286, #2138307, #2137603, and #2138296. The authors declare no competing interests.

References

[1] Zahra Rahimi Afzal, Tara Esmaeilbeig, Mojtaba Soltanalian, and Mesrob I Ohannessian. Linearization explains fine-tuning in large language models. *arXiv preprint arXiv:2602.08239*, 2026.

[2] Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, et al. Smollm2: When smol goes big–data-centric training of a small language model. *arXiv preprint arXiv:2502.02737*, 2025.

[3] Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

[4] Ali Furkan Biten, Ruben Tito, Andres Mafla, Lluis Gomez, Marçal Rusinol, Ernest Valveny, CV Jawahar, and Dimosthenis Karatzas. Scene text visual question answering. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4291–4301, 2019.

[5] Qiguang Chen, Libo Qin, Jin Zhang, Zhi Chen, Xiao Xu, and Wanxiang Che. M^3CoT: A novel benchmark for multi-domain multi-step multi-modal chain-of-thought. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8199–8221, 2024.

[6] Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*, 2016.

[7] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 conference of the north American chapter of the association for computational linguistics: Human language technologies, volume 1 (long and short papers)*, pages 2924–2936, 2019.

[8] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

[9] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[10] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized llms. *Advances in neural information processing systems*, 36:10088–10115, 2023.

[11] Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[12] Qichen Fu, Minsik Cho, Thomas Merth, Sachin Mehta, Mohammad Rastegari, and Mahyar Najibi. LazyLLM: Dynamic token pruning for efficient long context LLM inference. *arXiv preprint arXiv:2407.14057*, 2024.

[13] Aidan N Gomez, Mengye Ren, Raquel Urtasun, and Roger B Grosse. The reversible residual network: Backpropagation without storing activations. *Advances in neural information processing systems*, 30, 2017.

[14] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[15] Adam W Harley, Alex Ufkes, and Konstantinos G Derpanis. Evaluation of deep convolutional nets for document image classification and retrieval. In *2015 13th international conference on document analysis and recognition (ICDAR)*, pages 991–995. IEEE, 2015.

[16] Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. EuroSAT: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12(7):2217–2226, 2019.

[17] Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 523–533, 2014.

[18] 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 *International conference on machine learning*, pages 2790–2799. PMLR, 2019.

[19] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. LoRA: Low-rank adaptation of large language models. *Iclr*, 1(2):3, 2022.

[20] Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Lee. LLM-Adapters: An adapter family for parameter-efficient fine-tuning of large language models. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pages 5254–5276, 2023.

[21] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

[22] Daniel Kermany. Labeled optical coherence tomography (oct) and chest x-ray images for classification. *Mendeley data*, 2018.

[23] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

[24] Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585–597, 2015.

[25] Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. Mawps: A math word problem repository. In *Proceedings of the 2016 conference of the north american chapter of the association for computational linguistics: human language technologies*, pages 1152–1157, 2016.

[26] Md Kowsher, Tara Esmaeilbeig, Chun-Nam Yu, Chen Chen, Mojtaba Soltanalian, and Niloofar Yousefi. RoCoFT: Efficient finetuning of large language models with row-column updates. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 26659–26678, 2025.

[27] Benno Krojer, Mojtaba Komeili, Candace Ross, Quentin Garrido, Koustuv Sinha, Nicolas Ballas, and Mahmoud Assran. A shortcut-aware video-qa benchmark for physical understanding via minimal video pairs. *arXiv preprint arXiv:2506.09987*, 2025.

[28] Jason J Lau, Soumya Gayen, Asma Ben Abacha, and Dina Demner-Fushman. A dataset of clinically generated visual questions and answers about radiology images. *Scientific data*, 5(1):180251, 2018.

[29] Heejun Lee, Geon Park, Jaduk Suh, and Sung Ju Hwang. InfiniteHiP: Extending language model context up to 3 million tokens on a single GPU. *arXiv preprint arXiv:2502.08910*, 2025.

[30] Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, et al. LLaVA-OneVision: Easy visual task transfer. *arXiv preprint arXiv:2408.03326*, 2024.

[31] Rongao Li, Jie Fu, Bo-Wen Zhang, Tao Huang, Zhihong Sun, Chen Lyu, Guang Liu, Zhi Jin, and Ge Li. TACO: Topics in algorithmic code generation dataset. *arXiv preprint arXiv:2312.14852*, 2023.

[32] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *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.

[33] Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Advances in Neural Information Processing Systems*, 35:1950–1965, 2022.

[34] Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. DoRA: Weight-decomposed low-rank adaptation. In *Forty-first International Conference on Machine Learning*, 2024.

[35] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[36] Lingkun Long, Rubing Yang, Yushi Huang, Desheng Hui, Ao Zhou, and Jianlei Yang. SlimInfer: Accelerating long-context LLM inference via dynamic token pruning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 32284–32292, 2026.

[37] Qijun Luo, Mengqi Li, Lei Zhao, and Xiao Li. StreamBP: Memory-efficient exact backpropagation for long sequence training of LLMs. *arXiv preprint arXiv:2506.03077*, 2025.

[38] Sadhika Malladi, Alexander Wettig, Dingli Yu, Danqi Chen, and Sanjeev Arora. A kernel-based view of language model fine-tuning. In *International Conference on Machine Learning*, pages 23610–23641. PMLR, 2023.

[39] Fanqing Meng, Lingxiao Du, Zongkai Liu, Zhixiang Zhou, Quanfeng Lu, Daocheng Fu, Tiancheng Han, Botian Shi, Wenhai Wang, Junjun He, et al. Mm-eureka: Exploring the frontiers of multimodal reasoning with rule-based reinforcement learning. *arXiv preprint arXiv:2503.07365*, 2025.

[40] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

[41] Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 2381–2391, 2018.

[42] Zizheng Pan, Peng Chen, Haoyu He, Jing Liu, Jianfei Cai, and Bohan Zhuang. Mesa: A memory-saving training framework for transformers. *arXiv preprint arXiv:2111.11124*, 2021.

[43] Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 2080–2094, 2021.

[44] Xiaohan Qin, Xiaoxing Wang, Ning Liao, Cancheng Zhang, Xiangdong Zhang, Mingquan Feng, Jingzhi Wang, and Junchi Yan. ssToken: Self-modulated and semantic-aware token selection for llm fine-tuning. *arXiv preprint arXiv:2510.18250*, 2025.

[45] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[46] Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models. In *SC20: international conference for high performance computing, networking, storage and analysis*, pages 1–16. IEEE, 2020.

[47] Hanoona Rasheed, Abdelrahman Shaker, Anqi Tang, Muhammad Maaz, Ming-Hsuan Yang, Salman Khan, and Fahad Shahbaz Khan. VideoMathQA: Benchmarking mathematical reasoning via multimodal understanding in videos. *arXiv preprint arXiv:2506.05349*, 2025.

[48] Xuankun Rong, Wenke Huang, Tingfeng Wang, Daiguo Zhou, Bo Du, and Mang Ye. SafeGRPO: Self-rewarded multimodal safety alignment via rule-governed policy optimization. *arXiv preprint arXiv:2511.12982*, 2025.

[49] Subhro Roy and Dan Roth. Solving general arithmetic word problems. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1743–1752, 2015.

[50] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial Winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

[51] Antoine Simoulin, Namyong Park, Xiaoyi Liu, and Grey Yang. Memory-efficient fine-tuning of transformers via token selection. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 21565–21580, 2024.

[52] Davinder Singh, Naman Jain, Pranjali Jain, Pratik Kayal, Sudhakar Kumawat, and Nipun Batra. PlantDoc: A dataset for visual plant disease detection. In *Proceedings of the 7th ACM IKDD CoDS and 25th COMAD*, pages 249–253. 2020.

[53] Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*, 2023.

[54] Akiyoshi Tomihari and Issei Sato. Understanding linear probing then fine-tuning language models from NTK perspective. *Advances in Neural Information Processing Systems*, 37:139786–139822, 2024.

[55] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP workshop BlackboxNLP: Analyzing and interpreting neural networks for NLP*, pages 353–355, 2018.

[56] Tuowei Wang, Xingyu Chen, Kun Li, Ting Cao, Ju Ren, and Yaoxue Zhang. LeMo: Enabling less token involvement for more context fine-tuning. *arXiv preprint arXiv:2501.09767*, 2025.

[57] Hou Xia, Zheren Fu, Fangcan Ling, Jiajun Li, Yi Tu, Zhendong Mao, and Yongdong Zhang. Video-LevelGauge: Investigating contextual positional bias in large video language models. *arXiv preprint arXiv:2508.19650*, 2025.

[58] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 3485–3492. IEEE, 2010.

[59] Zhewei Yao, Xiaoxia Wu, Conglong Li, Connor Holmes, Minjia Zhang, Cheng Li, and Yuxiong He. Random-LTD: Random and layerwise token dropping brings efficient training for large-scale transformers. *arXiv preprint arXiv:2211.11586*, 2022.

[60] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 4791–4800, 2019.

[61] Runjia Zeng, Qifan Wang, Qiang Guan, Ruixiang Tang, Lifu Huang, Zhenting Wang, Xueling Zhang, Cheng Han, and Dongfang Liu. Tokenseek: Memory efficient fine tuning via instance-aware token ditching. 2026.

Contents

1 Introduction **1**

2 Preliminaries **3**

3 The Token Hypothesis **4**

4 Memory-Efficient Fine-Tuning via the Token Hypothesis **5**

5 Experiments **7**

6 Conclusion **10**

A Capacity Bound **17**

A.1 Proof of Theorem 1 17

A.2 Detailed Empirical Validation of Theorem 1 18

B Equal Capacity Across Tokens: Proof and Validation **22**

B.1 Proof of Theorem 2 22

B.2 Empirical Validation of Theorem 2 23

B.3 Token Importance in the Low-Shot Regime 26

C Monotone Output-Head NTK **29**

C.1 Proof of Theorem 3 29

C.2 Empirical Validation of the A_t Assumption 31

C.3 Empirical Validation of Theorem 3's Contraction Assumptions 32

C.4 Empirical Validation of Theorem 3 33

D Optimal Position Count **36**

E Ablation Studies **38**

E.1 Batch size scaling 38

E.2 Sequence length scaling 38

E.3 Convergence across M 39

E.4 NTK laziness: weight drift and kernel stability 39

E.5 Capability retention under targeted fine-tuning 40

F Extended Related Work **41**

F.1 Parameter-Efficient Fine-Tuning 41

F.2 Activation-Memory Reduction 41

F.3 Token-Selective Fine-Tuning 42

F.4 Neural Tangent Kernel Theory of Fine-Tuning 43

F.5 Synthesis 43

G Datasets **44**
G.1 Commonsense Reasoning 44
G.2 Mathematical Reasoning 44
G.3 Visual Question Answering 44
G.4 Video Question Answering 45
G.5 Image Classification 45

H Hyperparameters and Training Configuration **47**

I Models **50**

J Detailed Experimental Results **52**
J.1 Commonsense Reasoning (LLaMA-3-8B) 52
J.2 Mathematical Reasoning (LLaMA-3-8B) 54
J.3 Visual QA Reasoning (LLaVA-OneVision-7B and Qwen3-VL-8B) 54
J.4 Video Reasoning (Qwen3-VL-8B) 57
J.5 Image Classification (ViT-L/16-224) 57
J.6 GLUE (RoBERTa-base) 57
J.7 Gradient Checkpointing 57
J.8 Long-Context Instruction 58
J.9 Code Generation 59
J.10 Compatibility with FlashAttention and QLoRA 59
J.11 Summary 60

K Implementation Details **61**

L Limitations **62**

A Capacity Bound

A.1 Proof of Theorem 1

We restate the theorem for convenience before giving the proof.

Theorem 1 (restated). *Fix a position $t \in \{1, \dots, T\}$ and suppose the model is trained on $\mathcal{L}_t$ (the single-position loss defined in Section 2) by gradient descent in the NTK regime, with the position-t Jacobians $\nabla_\theta o_t(\cdot; \theta)$ frozen at their initialization values throughout training. Let f_t denote the converged predictor and let $r_t := y - o_t(X; \theta_0) \in \mathbb{R}^N$ be the residual at initialization. Then the learned correction $g_t(x) := f_t(x) - o_t(x; \theta_0)$ admits the representer form*

$$g_t(x) \;=\; \sum_{n=1}^{N} \alpha_n \, \mathcal{K}_t(x, x^{(n)}), \qquad \alpha \;=\; K_t^{+} \, r_t,$$

where K_t^{+} is the Moore–Penrose pseudoinverse of the Gram matrix K_t. In particular $g_t \in \mathcal{H}_{K_t}$, the correction on the training set satisfies $g_t(X) \in \mathrm{Im}(K_t)$, and the prediction on the training set lies in the affine subspace

$$f_t(X) \;\in\; o_t(X; \theta_0) + \mathrm{Im}(K_t),$$

whose dimension is $\mathrm{rank}(K_t) \le \min(N, P)$, independent of P.

Proof. The loss functional is

$$\mathcal{L}_t(\theta) \;=\; \frac{1}{2N} \sum_{n=1}^{N} \big(o_t(x^{(n)}; \theta) - y^{(n)}\big)^2,$$

and one step of gradient descent with step size η produces

$$\theta_{s+1} - \theta_s \;=\; -\frac{\eta}{N} \sum_{n=1}^{N} \big(o_t(x^{(n)}; \theta_s) - y^{(n)}\big) \, \nabla_\theta o_t(x^{(n)}; \theta_s),$$

where s indexes training steps. Under the frozen-Jacobian assumption, $\nabla_\theta o_t(x^{(n)}; \theta_s) = \phi_t(x^{(n)})$ for every $s \ge 0$, so each update is a linear combination of the fixed vectors $\{\phi_t(x^{(n)})\}_{n=1}^N$. Summing over all steps gives

$$\theta_S - \theta_0 \;\in\; \mathcal{G}_t \;:=\; \mathrm{span}\{\phi_t(x^{(n)}) : n = 1, \dots, N\} \;\subseteq\; \mathbb{R}^P$$

for every $S \ge 0$, so parameter updates are confined to a subspace of dimension at most N, independent of the total parameter count P.

We pass to the continuous-time limit by rescaling $\tau := s\eta$ and taking $\eta \to 0$, which transforms gradient descent into gradient flow. Under the frozen-Jacobian assumption the predictor at any input x evolves as

$$\frac{d}{d\tau} o_t(x; \theta_\tau) \;=\; \phi_t(x)^\top \frac{d\theta_\tau}{d\tau} \;=\; -\frac{1}{N} \sum_{n=1}^{N} \big(o_t(x^{(n)}; \theta_\tau) - y^{(n)}\big) \, \mathcal{K}_t(x, x^{(n)}),$$

where the second equality uses $\mathcal{K}_t(x, x') = \langle \phi_t(x), \phi_t(x') \rangle$. Writing $\hat{g}(x, \tau) := o_t(x; \theta_\tau) - o_t(x; \theta_0)$ for the running correction, this becomes the kernel gradient flow

$$\frac{d}{d\tau} \hat{g}(x, \tau) \;=\; -\frac{1}{N} \sum_{n=1}^{N} \big(\hat{g}(x^{(n)}, \tau) - r_t^{(n)}\big) \, \mathcal{K}_t(x, x^{(n)}),$$

and on the training set reduces to the linear ODE

$$\frac{d}{d\tau} \hat{g}(X, \tau) \;=\; -\frac{1}{N} \, K_t \, \big(\hat{g}(X, \tau) - r_t\big).$$

Since $K_t \succeq 0$, the flow contracts on $\mathrm{Im}(K_t)$ (where K_t has strictly positive eigenvalues) and is stationary on $\ker(K_t)$. Decomposing $r_t = r_t^{\parallel} + r_t^{\perp}$ with $r_t^{\parallel} \in \mathrm{Im}(K_t)$ and $r_t^{\perp} \in \ker(K_t)$, the flow drives $\hat{g}(X, \tau)$ from zero to $r_t^{\parallel}$ and leaves the orthogonal component untouched, so

$$\hat{g}(X, \infty) \;=\; r_t^{\parallel} \;=\; K_t K_t^{+} \, r_t,$$

where $K_t K_t^+$ is the orthogonal projector onto $\mathrm{Im}(K_t)$.

Since $\theta_\infty - \theta_0 \in \mathcal{G}_t$, it can be written as $\theta_\infty - \theta_0 = \sum_{n=1}^N \alpha_n \, \phi_t(x^{(n)})$ for some $\alpha \in \mathbb{R}^N$. Evaluating the converged predictor at any input x yields the representer form

$$g_t(x) \;:=\; \hat{g}(x, \infty) \;=\; \phi_t(x)^\top (\theta_\infty - \theta_0) \;=\; \sum_{n=1}^{N} \alpha_n \, \mathcal{K}_t(x, x^{(n)}).$$

On the training set, $g_t(X) = K_t \alpha$, and combining with $\hat{g}(X, \infty) = K_t K_t^+ r_t$ gives $K_t \alpha = K_t K_t^+ r_t$. Among all α satisfying this equation, gradient flow from zero initialization selects the unique solution in $\mathrm{Im}(K_t)$, namely $\alpha = K_t^+ r_t$; equivalently, $\alpha = K_t^+ r_t$ minimizes the parameter-space displacement $\|\theta_\infty - \theta_0\|_2 = \|\Phi_t \alpha\|_2$ among interpolants on the training set [1, 38].

We now characterize the achievable prediction subspace. Evaluating the correction on the training set,

$$g_t(X) \;=\; K_t \alpha \;=\; K_t K_t^+ r_t \;\in\; \mathrm{Im}(K_t),$$

since the product of any matrix with any vector lies in its image. Conversely, for any $z \in \mathrm{Im}(K_t)$, choosing labels $y := o_t(X; \theta_0) + z$ gives $r_t = z$. By the convergence analysis above, $g_t(X) = K_t K_t^+ z$, and since $K_t K_t^+$ is the orthogonal projector onto $\mathrm{Im}(K_t)$ with $z \in \mathrm{Im}(K_t)$ already, $K_t K_t^+ z = z$, yielding $g_t(X) = z$. Hence the achievable set of corrections on the training set is exactly

$$\big\{ g_t(X; y) : y \in \mathbb{R}^N \big\} \;=\; \mathrm{Im}(K_t),$$

and the full prediction vector lies in the affine subspace $f_t(X) \in o_t(X; \theta_0) + \mathrm{Im}(K_t)$.

Finally we bound the dimension of this subspace. Let $\Phi_t := [\phi_t(x^{(1)}) \mid \cdots \mid \phi_t(x^{(N)})] \in \mathbb{R}^{P \times N}$ be the feature matrix, so that $K_t = \Phi_t^\top \Phi_t$. By the identity $\mathrm{rank}(M^\top M) = \mathrm{rank}(M)$,

$$\dim \mathrm{Im}(K_t) \;=\; \mathrm{rank}(K_t) \;=\; \mathrm{rank}(\Phi_t) \;\le\; \min(P, N),$$

so the achievable prediction subspace has dimension at most $\min(P, N)$. Since in all our experiments $N \le P$ by orders of magnitude, this reduces to $\mathrm{rank}(K_t) \le N$, independent of the model's parameter count P. □

A.2 Detailed Empirical Validation of Theorem 1

We validate Theorem 1 across three pretrained transformers—RoBERTa-base (125M parameters, bidirectional), SmolLM2-135M (135M, causal), and LLaMA 3-8B (8B, causal)—and across dataset sizes $N \in \{30, 50, 100, 200, 300, 500, 1000, 2000\}$. For each (model, dataset size, position) configuration we compute the output-head NTK Gram matrix at the pretrained initialization θ_0, before any fine-tuning: using the final-layer hidden states $h_t^{(D)}(x^{(n)}) \in \mathbb{R}^d$, we set

$$K_t[n, m] \;=\; \langle h_t^{(D)}(x^{(n)}),\, h_t^{(D)}(x^{(m)}) \rangle.$$

This is a tractable proxy for the full NTK Gram matrix; in the output-head NTK regime, where only the final projection is trained, it equals K_t exactly. For each configuration we report the effective rank $d_{\mathrm{eff}}(K_t) = \mathrm{tr}(K_t)/\|K_t\|_{\mathrm{op}}$, the d_{95} metric (number of eigenvalues required to capture 95% of the trace), the normalized eigenvalue spectrum, and the cumulative-variance curve. The theorem predicts $\mathrm{rank}(K_t) \le \min(P, N)$; in all experiments $N \ll P$, so the meaningful ceiling is N. The effective-rank metric d_{eff} is tighter still, and quantifies the number of directions with non-negligible eigenvalues.

RoBERTa-base across dataset sizes. Figures 7–13 show results for RoBERTa-base on SST-2 sentences tokenized to length T=50, at positions $t \in \{1, 6, 11, 16, 21, 26, 31, 36\}$. Tokens at each position vary across samples (the fixed-token setup of Theorem 2 is evaluated separately in Appendix B.2). Across all tested dataset sizes and positions, the normalized eigenvalue spectrum drops by more than ten orders of magnitude within the first 10–20 indices, and d_{eff} lies in the range $[1.5, 4.0]$ at every configuration. Table 3 summarizes the effective-rank values per position and per N. Despite N growing by a factor of 67 (from 30 to 2000), d_{eff} barely changes—confirming that the capacity bottleneck is a structural property of the pretrained kernel at initialization, not a finite-sample artifact.

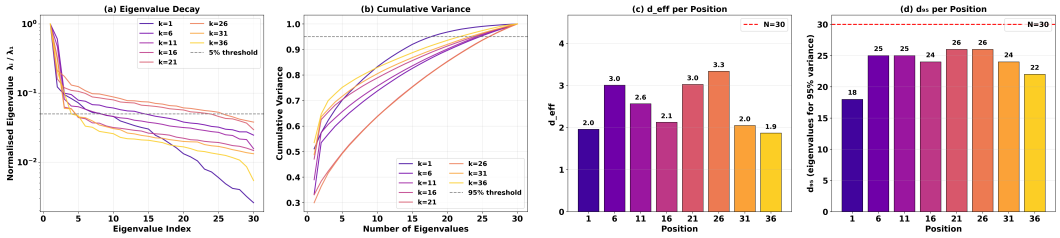

Figure 7: RoBERTa-base, N=30. $d_{\text{eff}} \in [1.9, 3.3]$, $d_{95} \in [18, 26]$, both well below the ceiling N=30.

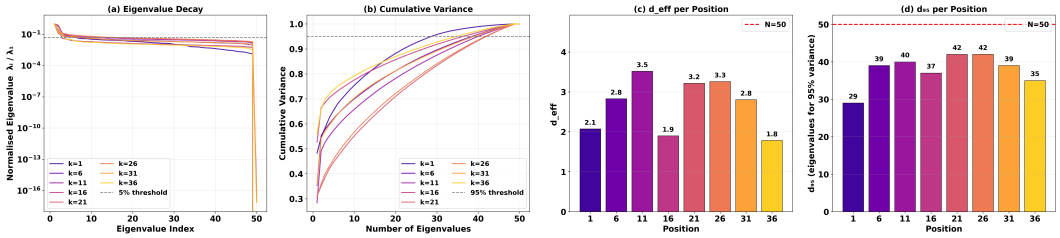

Figure 8: RoBERTa-base, N=50. $d_{\text{eff}} \in [1.8, 3.5]$, $d_{95} \in [29, 42]$.

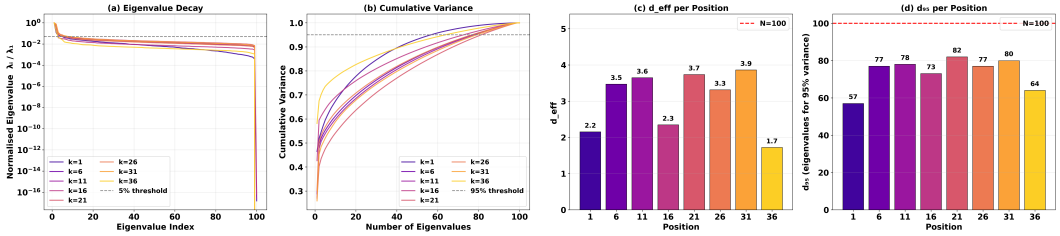

Figure 9: RoBERTa-base, N=100. $d_{\text{eff}} \in [1.7, 3.9]$, $d_{95} \in [57, 82]$. Values are nearly identical to those at N=50, confirming insensitivity to dataset size.

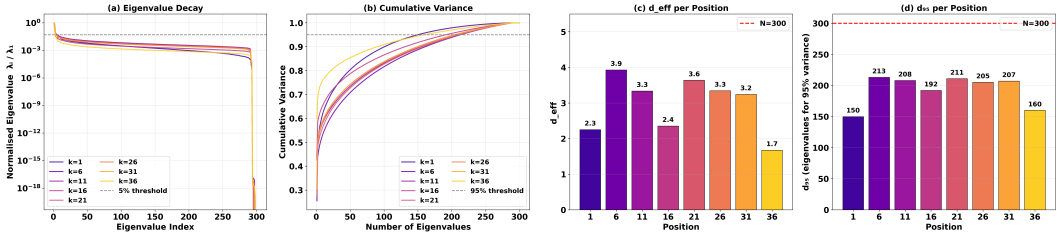

Figure 10: RoBERTa-base, N=200. $d_{\text{eff}} \in [1.9, 3.9]$, $d_{95} \in [100, 215]$.

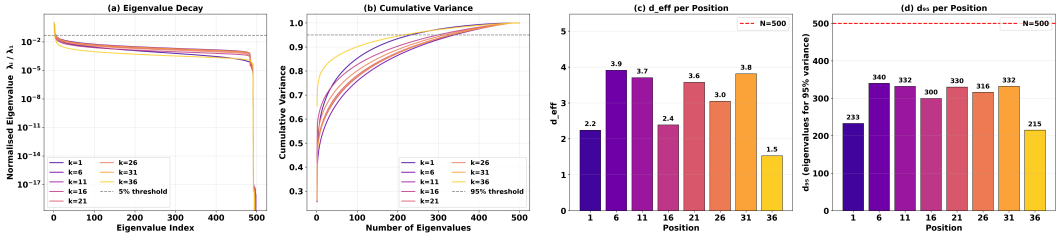

Figure 11: RoBERTa-base, N=500. $d_{\text{eff}} \in [1.5, 3.9]$, $d_{95} \in [215, 340]$. Reproduced in the main text as Figure 4.

SmolLM2-135M: causal model at comparable scale. Figure 14 validates Theorem 1 on SmolLM2-135M, a causal 135M-parameter model, at N=600 and positions $t \in$

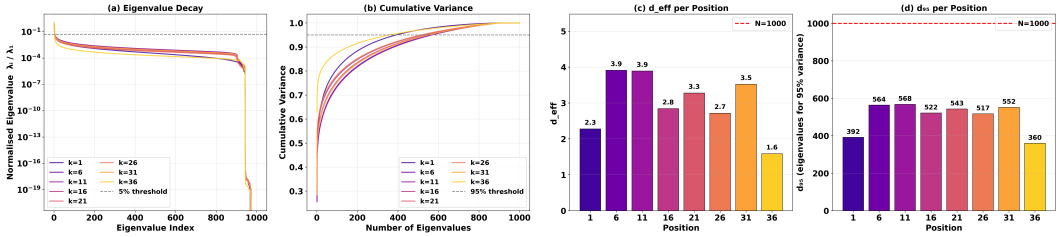

Figure 12: RoBERTa-base, N=1000. $d_{\text{eff}} \in [1.6, 3.9]$, $d_{95} \in [392, 568]$. A tenfold increase from N=100 produces no meaningful change in d_{eff}.

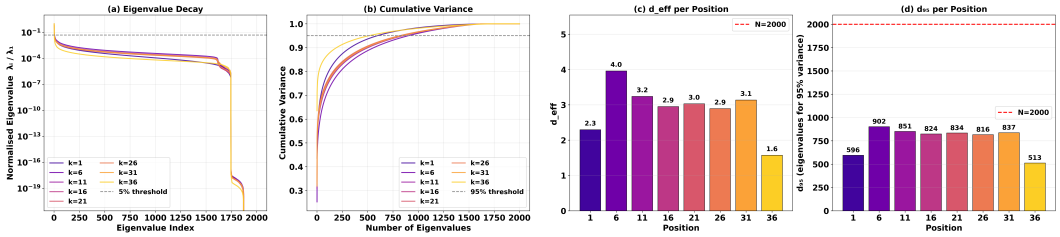

Figure 13: RoBERTa-base, N=2000. $d_{\text{eff}} \in [1.6, 4.0]$, $d_{95} \in [513, 902]$. At the largest tested size, d_{eff} stays below 4.0 and $d_{95}/N < 0.46$.

Table 3: Effective rank $d_{\text{eff}}(K_t)$ for RoBERTa-base across dataset sizes N and positions t. Values rounded to one decimal place. All entries are more than an order of magnitude below N.

N	t=1	t=6	t=11	t=16	t=21	t=26	t=31
30	2.0	3.0	2.6	2.1	3.0	2.0	1.9
50	2.1	2.8	3.5	1.9	3.2	3.3	2.8
100	2.2	3.5	3.6	2.3	3.7	3.3	3.9
200	2.2	3.9	3.3	2.4	3.6	3.3	3.2
500	2.2	3.9	3.7	2.4	3.6	3.0	3.8
1000	2.3	3.9	3.9	2.8	3.3	2.7	3.5
2000	2.3	4.0	3.2	2.9	3.0	2.9	3.1

$\{10, 30, 70, 100, 150, 200, 300, 500\}$. The d_{eff} values lie in a tight range $[2.4, 2.8]$—more concentrated than for RoBERTa, with less position-to-position variation. The normalized spectrum collapses by more than fourteen orders of magnitude within the first few indices, and d_{95} stays in $[296, 343]$, well below N=600. The capacity bottleneck therefore holds equally for causal architectures, and at a size comparable to RoBERTa the two models produce comparable effective ranks despite their different attention directionality.

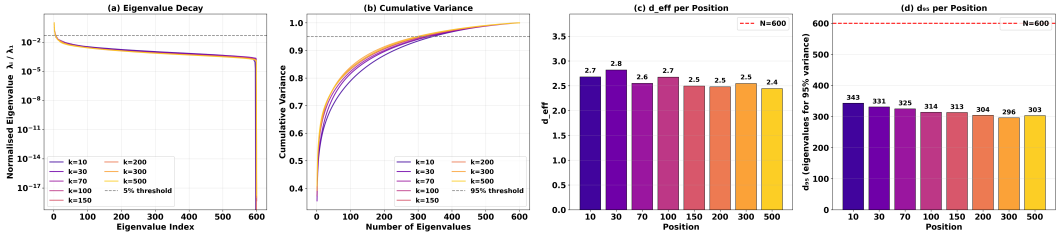

Figure 14: SmolLM2-135M (causal, 135M parameters), N=600, positions $t \in \{10, 30, 70, 100, 150, 200, 300, 500\}$. $d_{\text{eff}} \in [2.4, 2.8]$, $d_{95} \in [296, 343]$.

LLaMA 3-8B: large-scale validation. Figure 15 shows the same analysis on LLaMA 3-8B (8B parameters, causal) at N=500 and positions $t \in \{1, 10, 50, 100, 150, 300, 500, 700, 1000\}$, spanning a sequence of length T=1024. Despite a 64× increase in parameter count over RoBERTa-base,

d_{eff} remains in the narrow range $[2.4, 5.2]$, and d_{95} is strikingly uniform across positions (176–199). The uniformity of d_{95} across positions is a further empirical signal that the final-layer hidden-state geometry of LLaMA 3-8B is nearly position-independent—an observation that provides independent support for the aligned-geometry assumption of Theorem 2 in the causal setting.

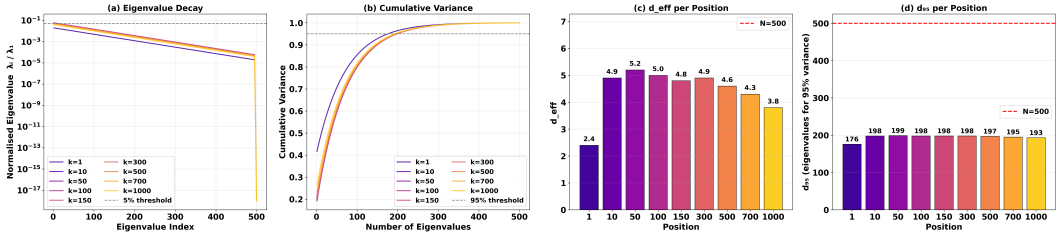

Figure 15: LLaMA 3-8B (causal, 8B parameters), $N{=}500$, positions $t \in \{1, 10, 50, 100, 150, 300, 500, 700, 1000\}$. $d_{\text{eff}} \in [2.4, 5.2]$, $d_{95} \in [176, 199]$. An 8B-parameter model fine-tuned through a single position admits fewer than six effective degrees of freedom.

Summary. Across all three models, all tested positions, and all dataset sizes, the three core predictions of Theorem 1 are confirmed. First, the achievable prediction vectors lie in the low-dimensional affine subspace $o_t(X; \theta_0) + \text{Im}(K_t)$ with $\text{rank}(K_t) \ll N$. Second, $d_{\text{eff}}(K_t)$ is determined by the pretrained kernel spectrum rather than by parameter count: it stays below 6 regardless of whether the model has 125M, 135M, or 8B parameters. Third, the capacity ceiling is set at initialization and is a structural property of the position-specific kernel, independent of the fine-tuning task, learning rate, or number of gradient steps.

B Equal Capacity Across Tokens: Proof and Validation

B.1 Proof of Theorem 2

Recall from Section 3 that in the output-head NTK regime the feature map reduces to the final-layer hidden state, $\phi_t(x) = h_t^{(D)}(x) \in \mathbb{R}^d$, and the Gram matrix and kernel-evaluation vector are computed from forward passes as $K_t[n,m] = \langle h_t^{(D)}(x^{(n)}), h_t^{(D)}(x^{(m)})\rangle$ and $k_t(x)[n] = \langle h_t^{(D)}(x), h_t^{(D)}(x^{(n)})\rangle$. By Theorem 1, for either choice $t \in \{i,j\}$ the converged predictor is $f_t(x) = o_t(x;\theta_0) + k_t(x)^\top K_t^+ r_t$, with $r_t := y - o_t(X;\theta_0)$. For brevity we write $R := \|r_i\|_2$ and $B_j(x) := \|k_j(x)\|_2$ throughout the proof.

Before the main argument we record two standard facts about the Moore–Penrose pseudoinverse of a symmetric positive semidefinite $K \in \mathbb{R}^{N\times N}$. Let $K = U\Lambda U^\top$ be its eigendecomposition, with $U = [U_r \mid U_0]$ orthogonal, $U_r \in \mathbb{R}^{N\times r}$ collecting the eigenvectors of the $r = \operatorname{rank}(K)$ nonzero eigenvalues $\lambda_1 \geq \cdots \geq \lambda_r > 0$ (gathered in $\Lambda_r := \operatorname{diag}(\lambda_1,\ldots,\lambda_r)$), and $U_0 \in \mathbb{R}^{N\times(N-r)}$ spanning $\ker(K)$. The pseudoinverse is $K^+ = U_r\Lambda_r^{-1}U_r^\top$.

Fact (i): operator-norm equality. K^+ is symmetric with nonzero eigenvalues $\{1/\lambda_1,\ldots,1/\lambda_r\}$, so

$$\|K^+\|_{\mathrm{op}} \;=\; \frac{1}{\lambda_r} \;=\; \frac{1}{\lambda^+_{\min}(K)}.$$

Fact (ii): K^+ annihilates $\ker(K)$. Let $\Pi := U_rU_r^\top$ be the orthogonal projector onto $\operatorname{Im}(K)$. Using $U_r^\top U_r = I_r$,

$$K^+(I-\Pi) \;=\; U_r\Lambda_r^{-1}U_r^\top(I-U_rU_r^\top) \;=\; U_r\Lambda_r^{-1}(U_r^\top - U_r^\top U_rU_r^\top) \;=\; U_r\Lambda_r^{-1}(U_r^\top - U_r^\top) \;=\; 0,$$

so $K^+ = K^+\Pi$.

Writing $\lambda_0^+ := \min(\lambda^+_{\min}(K_i),\, \lambda^+_{\min}(K_j))$, fact (i) gives $\|K_t^+\|_{\mathrm{op}} \leq 1/\lambda_0^+$ for $t \in \{i,j\}$.

Theorem 2 (restated). *Under the output-head NTK regime with scalar output ($C = 1$), suppose the fixed-token condition $x_i^{(n)} = s_i$ and $x_j^{(n)} = s_j$ holds for all n, and suppose K_i and K_j share the same column space,*

$$\operatorname{Im}(K_i) \;=\; \operatorname{Im}(K_j) \;=:\; \mathcal{W}, \tag{1}$$

with common orthogonal projector Π onto $\mathcal{W}$ and $\lambda_0^+ > 0$. Then for all test inputs x with $x_i = s_i$ and $x_j = s_j$,

$$\|f_i(x) - f_j(x)\|_2 \;\leq\; \delta_{\mathrm{init}}(x) \;+\; \frac{\delta_k(x)\,R}{\lambda_0^+} \;+\; \frac{B_j(x)\,\delta_K\,R}{(\lambda_0^+)^2} \;+\; \frac{B_j(x)\,\delta_r}{\lambda_0^+}. \tag{2}$$

The fixed-token hypothesis is not invoked directly in the proof; rather, it is the physical condition that makes the common-image assumption (1) plausible in a bidirectional model, where positions with aligned context produce hidden states spanning the same linear subspace.

Proof. The two converged predictors are $f_i(x) = o_i(x;\theta_0) + k_i(x)^\top K_i^+ r_i$ and $f_j(x) = o_j(x;\theta_0) + k_j(x)^\top K_j^+ r_j$. Subtracting and applying the triangle inequality,

$$\|f_i(x) - f_j(x)\|_2 \;\leq\; \underbrace{\|o_i(x;\theta_0) - o_j(x;\theta_0)\|_2}_{=\,\delta_{\mathrm{init}}(x)} \;+\; \big|k_i(x)^\top K_i^+ r_i - k_j(x)^\top K_j^+ r_j\big|. \tag{3}$$

To bound the second term, add and subtract $k_j(x)^\top K_i^+ r_i$:

$$\begin{aligned} &k_i(x)^\top K_i^+ r_i - k_j(x)^\top K_j^+ r_j \\ &\quad=\; (k_i(x) - k_j(x))^\top K_i^+ r_i \;+\; k_j(x)^\top\big(K_i^+ r_i - K_j^+ r_j\big). \end{aligned} \tag{4}$$

Applying the triangle inequality and Cauchy–Schwarz to the second summand,

$$\big|k_i(x)^\top K_i^+ r_i - k_j(x)^\top K_j^+ r_j\big| \;\leq\; \big|(k_i(x) - k_j(x))^\top K_i^+ r_i\big| \;+\; B_j(x)\cdot\|K_i^+ r_i - K_j^+ r_j\|_2. \tag{5}$$

Term 1. Cauchy–Schwarz combined with fact (i) gives

$$\left|(k_i(x) - k_j(x))^\top K_i^+ r_i\right| \;\le\; \|k_i(x) - k_j(x)\|_2 \cdot \|K_i^+\|_{\mathrm{op}} \cdot \|r_i\|_2 \;\le\; \frac{\delta_k(x)\, R}{\lambda_0^+}. \tag{6}$$

Term 2. Adding and subtracting $K_j^+ r_i$,

$$\|K_i^+ r_i - K_j^+ r_j\|_2 \;\le\; \|(K_i^+ - K_j^+) r_i\|_2 \;+\; \|K_j^+ (r_i - r_j)\|_2. \tag{7}$$

The second summand is bounded directly:

$$\|K_j^+ (r_i - r_j)\|_2 \;\le\; \|K_j^+\|_{\mathrm{op}} \cdot \|r_i - r_j\|_2 \;\le\; \frac{\delta_r}{\lambda_0^+}. \tag{8}$$

For the first summand, fact (ii) gives $K_t^+ = K_t^+ \Pi$ for $t \in \{i, j\}$ (the common-image assumption (1) makes Π the common projector), so

$$(K_i^+ - K_j^+) r_i \;=\; (K_i^+ - K_j^+) \Pi r_i,$$

with $\Pi r_i \in \mathcal{W}$. Restricted to $\mathcal{W}$, both K_i and K_j are strictly positive with eigenvalues at least λ_0^+, so the restricted operators $K_i|_{\mathcal{W}}$ and $K_j|_{\mathcal{W}}$ are invertible as maps $\mathcal{W} \to \mathcal{W}$. To see that $K_t^+|_{\mathcal{W}} = (K_t|_{\mathcal{W}})^{-1}$ on $\mathcal{W}$, write $K_t = U_r^{(t)} \Lambda_r^{(t)} (U_r^{(t)})^\top$ with $\mathrm{span}(U_r^{(t)}) = \mathcal{W}$. For any $v \in \mathcal{W}$,

$$K_t K_t^+ v \;=\; U_r^{(t)} \Lambda_r^{(t)} (U_r^{(t)})^\top U_r^{(t)} (\Lambda_r^{(t)})^{-1} (U_r^{(t)})^\top v \;=\; U_r^{(t)} (U_r^{(t)})^\top v \;=\; \Pi v \;=\; v,$$

using $(U_r^{(t)})^\top U_r^{(t)} = I_r$ and $\Pi v = v$ for $v \in \mathcal{W}$. The analogous computation gives $K_t^+ K_t v = v$ on $\mathcal{W}$, so $K_t^+|_{\mathcal{W}}$ and $K_t|_{\mathcal{W}}$ are mutual inverses on $\mathcal{W}$.

For any two invertible operators A, B on $\mathcal{W}$, the first-order resolvent identity

$$A^{-1} - B^{-1} \;=\; A^{-1}(B - A)B^{-1} \tag{9}$$

follows from $A^{-1}(B - A)B^{-1} = A^{-1}BB^{-1} - A^{-1}AB^{-1} = A^{-1} - B^{-1}$. Applying (9) with $A = K_i|_{\mathcal{W}}$, $B = K_j|_{\mathcal{W}}$, and evaluating both sides at $\Pi r_i \in \mathcal{W}$:

$$(K_i^+ - K_j^+) \Pi r_i \;=\; K_i^+ (K_j - K_i) K_j^+ \Pi r_i.$$

Taking norms, applying submultiplicativity, and using $\|\Pi r_i\|_2 \le \|r_i\|_2 = R$,

$$\|(K_i^+ - K_j^+) r_i\|_2 \;=\; \|(K_i^+ - K_j^+) \Pi r_i\|_2 \;\le\; \|K_i^+\|_{\mathrm{op}} \cdot \|K_j - K_i\|_{\mathrm{op}} \cdot \|K_j^+\|_{\mathrm{op}} \cdot \|\Pi r_i\|_2 \;\le\; \frac{\delta_K\, R}{(\lambda_0^+)^2}. \tag{10}$$

Substituting (8) and (10) into (7),

$$\|K_i^+ r_i - K_j^+ r_j\|_2 \;\le\; \frac{\delta_K\, R}{(\lambda_0^+)^2} \;+\; \frac{\delta_r}{\lambda_0^+}. \tag{11}$$

Substituting (6) and (11) into (5) and then into (3),

$$\|f_i(x) - f_j(x)\|_2 \;\le\; \delta_{\mathrm{init}}(x) \;+\; \frac{\delta_k(x)\, R}{\lambda_0^+} \;+\; B_j(x) \left(\frac{\delta_K\, R}{(\lambda_0^+)^2} + \frac{\delta_r}{\lambda_0^+} \right),$$

which rearranges to (2). □

B.2 Empirical Validation of Theorem 2

We validate Theorem 2 empirically using RoBERTa-base (125M parameters) on SST-2 sentences. The theorem predicts that two positions i and j sharing fixed anchor tokens in a bidirectional model yield approximately equal fine-tuning capacity, with the prediction gap $\|f_i(x) - f_j(x)\|_2$ controlled by the four measurable quantities $\delta_{\mathrm{init}}(x), \delta_k(x), \delta_K, \delta_r$ of Theorem 2. We measure the effective rank $d_{\mathrm{eff}}(K_t) = \mathrm{tr}(K_t)/\|K_t\|_{\mathrm{op}}$ and the d_{95} metric of the output-head NTK Gram matrix at each fixed position t, computed at the pretrained initialization θ_0 before any fine-tuning. We run three complementary experiments: two that confirm the theorem's geometric hypotheses by placing fixed tokens at opposite ends of the sequence, and one that confirms the theorem's conclusion by measuring the prediction gap between fine-tuned predictors directly.

Experiment 1: Fixed prefix prompt (right padding). We prepend the fixed natural-language prompt *"Read the sentence properly, and determine the sentiment for this sentence"* to every input, producing sequences of the form

$$\texttt{[CLS]}\ \underbrace{t_1\ t_2\ \cdots\ t_{12}}_{\text{fixed prompt tokens}}\ \underbrace{w_1\ w_2\ \cdots}_{\text{varying sentence}}\ \texttt{[SEP]}\ \texttt{[PAD]}\cdots$$

where the prompt tokens occupy positions $t = 1, \ldots, 12$ and are identical across all N samples, while the sentence tokens at positions $t > 12$ vary freely. This mirrors the prefix-tuning setup in which a fixed prompt is prepended to every input. We measure d_{eff} and d_{95} at all 12 prompt positions for $N \in \{50, 300, 1000\}$.

Figures 16, 17, and 18 show the results. Across all dataset sizes and all 12 prompt positions, d_{eff} lies in $[1.00, 1.12]$ with coefficient of variation $\text{CV} = \sigma/\mu \leq 0.02$. The eigenvalue decay curves and the cumulative variance curves are visually indistinguishable across the 12 positions. $d_{95} = 1$ at almost every position across all three dataset sizes, meaning a single eigenvalue explains 95% of the variance—the Gram matrix is effectively rank-one at each fixed position. The d_{eff} values are nearly identical across $N \in \{50, 300, 1000\}$, indicating that the observed equivalence is a structural property of the pretrained model, not a finite-sample artifact.

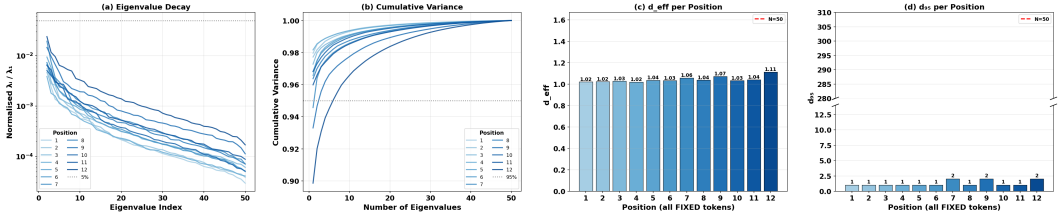

Figure 16: Experiment 1 (fixed prefix prompt, right padding), N=50. $d_{\text{eff}} \in [1.02, 1.11]$, CV $= 0.02$; $d_{95} = 1$ at all but two positions. Eigenvalue decay and cumulative variance curves are nearly identical across the 12 fixed prompt positions.

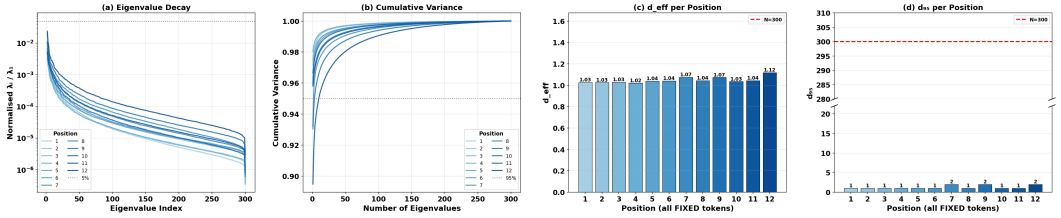

Figure 17: Experiment 1 (fixed prefix prompt, right padding), N=300. $d_{\text{eff}} \in [1.02, 1.12]$, CV $=$ 0.02; $d_{95} = 1$ at all but two positions. The pattern is essentially unchanged from N=50.

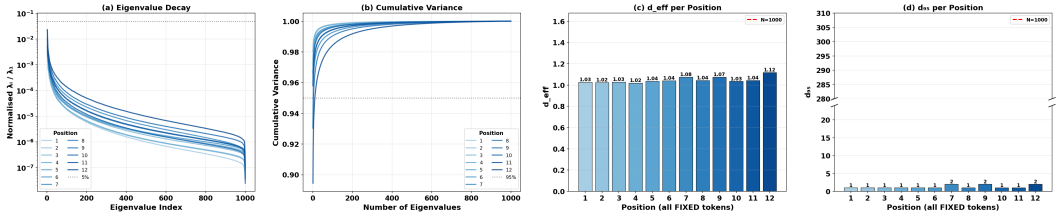

Figure 18: Experiment 1 (fixed prefix prompt, right padding), N=1000. $d_{\text{eff}} \in [1.02, 1.12]$, CV $=$ 0.02; $d_{95} = 1$ at all but two positions. Despite a twenty-fold increase in N from 50, the effective ranks are numerically identical.

Experiment 2: Fixed suffix prompt (left padding). We append the fixed natural-language prompt *"Based on the analysis of the given sentence, we can say the sentiment is"* to every input, producing sequences of the form

$$\texttt{[PAD]}\cdots\ \texttt{[CLS]}\ \underbrace{w_1\ w_2\ \cdots}_{\text{varying sentence}}\ \underbrace{t_1\ t_2\ \cdots\ t_{15}}_{\text{fixed answer prompt}}\ \texttt{[SEP]}$$

where the answer prompt occupies positions $t = 48, \ldots, 62$ and is identical across all N samples. This places the fixed tokens at the end of the sequence rather than the beginning, testing whether positional equivalence holds in a structurally different context. We again measure d_{eff} and d_{95} at all 15 suffix positions for $N \in \{50, 300, 1000\}$.

Figures 19, 20, and 21 show the results. All 15 suffix positions yield $d_{\text{eff}} \in [1.02, 1.05]$ with $\text{CV} \leq 0.01$—a tighter range than Experiment 1. The eigenvalue curves are visually indistinguishable across positions, and $d_{95} = 1$ at every position without exception across all three dataset sizes.

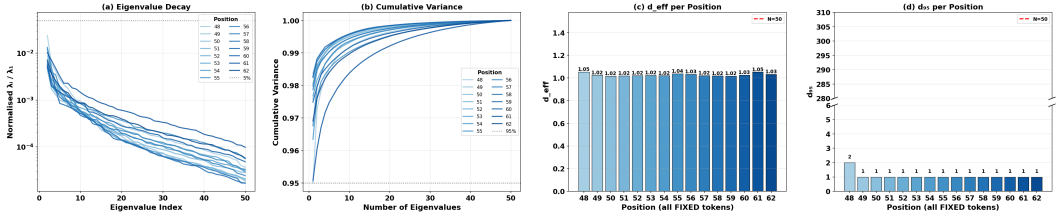

Figure 19: Experiment 2 (fixed suffix prompt, left padding), N=50. $d_{\text{eff}} \in [1.02, 1.05]$, $\text{CV} = 0.01$; $d_{95} = 1$ at every position.

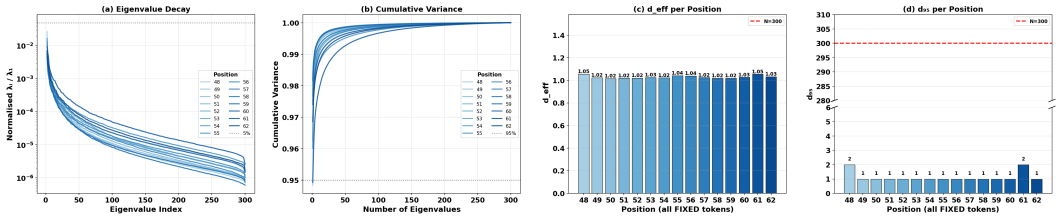

Figure 20: Experiment 2 (fixed suffix prompt, left padding), N=300. $d_{\text{eff}} \in [1.02, 1.05]$, $\text{CV} = 0.01$; $d_{95} = 1$ at every position. Results are unchanged from N=50.

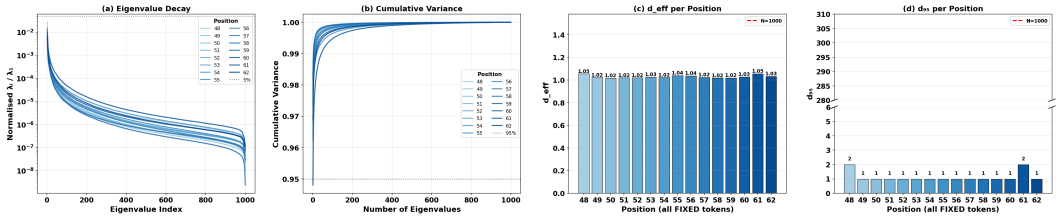

Figure 21: Experiment 2 (fixed suffix prompt, left padding), N=1000. $d_{\text{eff}} \in [1.02, 1.05]$, $\text{CV} = 0.01$; $d_{95} = 1$ at every position.

Experiment 3: Prediction-gap measurement. Experiments 1 and 2 confirmed the theorem's geometric hypotheses through the near-identity of eigenvalue spectra and effective ranks across fixed positions. We now measure the theorem's *conclusion* directly: that the converged predictors $f_i(x)$ and $f_j(x)$ are numerically close at test inputs, not just structurally similar in feature geometry. Using the fixed prompt *"Classify the sentiment of this movie review :"* prepended to every input on RoBERTa-base, we train output-head NTK predictors f_i and f_j separately at each of ten pairs of positions across the prompt (spanning position-index gaps from 1 to 7), using N=300 SST-2 training sentences and evaluating on 50 held-out test sentences. Table 4 reports the mean prediction gap $\|f_i(x) - f_j(x)\|$ normalized by the label standard deviation $\hat{\sigma}_y = 0.5$, the fraction of test sentences for which $\|f_i(x) - f_j(x)\| < 0.5\,\hat{\sigma}_y$, and the p-value from a left-tail position-exchangeability permutation test with B=2000 resamples.

Across all ten pairs, the mean prediction gap ranges from 0.180 to 0.275 standard deviations of the label—substantially below the $0.5\,\hat{\sigma}_y$ threshold—and between 86% and 98% of test sentences fall below that threshold per pair. The effect is statistically significant for every pair ($p < 0.001$), indicating that the observed closeness is not a chance consequence of arbitrary position assignment but reflects a genuine equivalence in converged predictors. The prediction gap shows no systematic

dependence on the position-index distance $|i-j|$: adjacent pairs and gap-7 pairs produce comparable gaps, consistent with the theorem's prediction that positional distance is irrelevant once the hidden-state geometry is aligned. This provides a direct downstream confirmation of the four-term bound of Theorem 2: the measurable quantities $\delta_K, \delta_k(x), \delta_r, \delta_{\text{init}}(x)$ jointly control the gap, and at the prediction level the gap is bounded uniformly across position pairs.

Table 4: Prediction gaps between fine-tuned predictors f_i and f_j across ten token pairs within the fixed prompt "`Classify the sentiment of this movie review :`" on RoBERTa-base (N=300 SST-2 training, 50 test sentences). *Gap* is the position-index distance $|i-j|$; *Frac* is the fraction of test sentences with $\|f_i(x) - f_j(x)\| < 0.5\,\hat{\sigma}_y$; p-values are from a left-tail position-exchangeability permutation test (B=2000).

Token s_i	Token s_j	Gap	$\|f_i - f_j\|/\hat{\sigma}_y$	Frac $< 0.5\hat{\sigma}_y$	p-value
Adjacent pairs (gap = 1)					
`Classify`	`the`	1	0.188	96%	<0.001
`the`	`sentiment`	1	0.245	92%	<0.001
`sentiment`	`of`	1	0.229	96%	<0.001
`of`	`this`	1	0.216	92%	<0.001
Short-range pairs (gap = 2)					
`Classify`	`sentiment`	2	0.250	88%	<0.001
`the`	`of`	2	0.246	94%	<0.001
`sentiment`	`this`	2	0.180	98%	<0.001
Medium-range pairs (gap = 4)					
`Classify`	`this`	4	0.275	86%	<0.001
`the`	`movie`	4	0.232	92%	<0.001
Long-range pair (gap = 7)					
`Classify`	`:`	7	0.273	86%	<0.001
All 10 pairs			0.188–0.275	86–98%	<0.001

Synthesis. The three experiments validate Theorem 2 at complementary levels. Experiments 1 and 2 confirm the theorem's geometric hypotheses: across the 12 prefix positions and 15 suffix positions respectively, the Gram matrices K_t have visually indistinguishable eigenvalue spectra and effective ranks (CV ≤ 0.02), empirically supporting the common-image assumption (Eq. 1) and $\delta_K \approx 0$. The rank-one collapse observed in both experiments is a consequence of token fixing that goes beyond what the theorem strictly requires: when the token at position t is identical across all N samples, the position-t gradient has one dominant direction determined by the fixed token embedding, with context variation contributing only a small perturbation through attention. This concentrates almost all variance into a single eigenvalue, making $\lambda^+_{\min}(K_t) \approx \lambda_1(K_t) \approx \text{tr}(K_t)$ large and driving the second and third terms of the bound of Theorem 2 (divided by $(\lambda^+_0)^2$ and λ^+_0) to be numerically negligible. Experiment 3 then confirms the theorem's conclusion directly: when the hypotheses hold, the converged predictors f_i and f_j are numerically close, with mean prediction gap below $0.28\,\hat{\sigma}_y$ and 86%–98% of test inputs falling within $0.5\,\hat{\sigma}_y$ across all ten tested position pairs. The consistency across early vs. late positions (Experiments 1 vs. 2) and across position-index distances from 1 to 7 (Experiment 3) confirms that positional equivalence is a global property of the pretrained model's hidden-state geometry, not an artifact of position index, padding convention, or adjacency.

B.3 Token Importance in the Low-Shot Regime

The three experiments above validate Theorem 2 in the regime where its hypotheses hold with wide margin. We now probe the opposite regime, where the bound becomes loose and the theorem's equal-capacity prediction weakens.

Table 5: Summary of geometric validation of Theorem 2 (Experiments 1 and 2). All fixed-token positions yield nearly identical d_{eff} values with CV ≤ 0.02 across both experiments and all dataset sizes.

Experiment	Positions	N	d_{eff} range	Mean d_{eff}	CV	d_{95} (modal)
Exp. 1 (prefix, right pad)	$t = 1$–12	50	$[1.02,\ 1.11]$	1.03	0.02	1
		300	$[1.02,\ 1.12]$	1.04	0.02	1
		1000	$[1.02,\ 1.12]$	1.04	0.02	1
Exp. 2 (suffix, left pad)	$t = 48$–62	50	$[1.02,\ 1.05]$	1.03	0.01	1
		300	$[1.02,\ 1.05]$	1.03	0.01	1
		1000	$[1.02,\ 1.05]$	1.03	0.01	1

Theorem 2 bounds the prediction gap between two fixed-token positions by $\delta_{\text{init}}(x) + \delta_k(x)R/\lambda_0^+ + B_j(x)\delta_K R/(\lambda_0^+)^2 + B_j(x)\delta_r/\lambda_0^+$, with the second and third terms scaling inversely in $\lambda_0^+ = \min_t \lambda_{\min}^+(K_t)$. In the generic training regime where N is reasonably large, λ_0^+ is comfortably bounded away from zero and these terms are negligible, so random selection suffices (as reported in Section 4). In this ablation we characterize the opposite regime—very low N—where λ_0^+ can be small and the error terms of the bound become non-negligible. We ask: when the bound is loose, does the choice of which positions to train through affect accuracy in the way the spectral structure would predict?

We use the Wanda [53] criterion—the product of weight magnitude and input activation norm—to rank token positions by how much they contribute to the model's output at initialization. Positions with larger activation norms produce Gram matrices with larger dominant eigenvalues, hence larger λ_0^+; positions with smaller activation norms produce Gram matrices with smaller λ_0^+. Theorem 2 predicts that when λ_0^+ is small the error terms of the bound blow up, so we expect that in low-shot training, high-λ_0^+ positions (Most Important) should outperform low-λ_0^+ positions (Least Important), with the gap shrinking as M increases and the combined kernel $K^{(M)}$ aggregates gradient signal across positions.

We test this on MRPC (paraphrase identification) and QQP (question-pair similarity) using RoBERTa-base, selecting the top-M, bottom-M, and a uniformly random M positions for $M \in \{1, 2, 3, 5, 10\}$. Accuracy is reported over 10 random seeds with error bars showing one standard deviation.

Figures 22 and 23 show the results. Across both datasets the ordering Most Important $\geq$ Random $\geq$ Least Important holds at every M. At $M = 1$, the gap between Most and Least Important is 0.018 on MRPC and 0.013 on QQP; at $M = 10$, the gap narrows to roughly 0.013 on MRPC and 0.015 on QQP. The narrowing is consistent with the theorem's structure: as M grows, the combined kernel $K^{(M)}$ inherits $\lambda_{\min}^+(K^{(M)}) \geq \sum_{m=1}^{M} \lambda_{\min}^+(K_{t_m})$, so the effective conditioning improves and the per-position importance matters less.

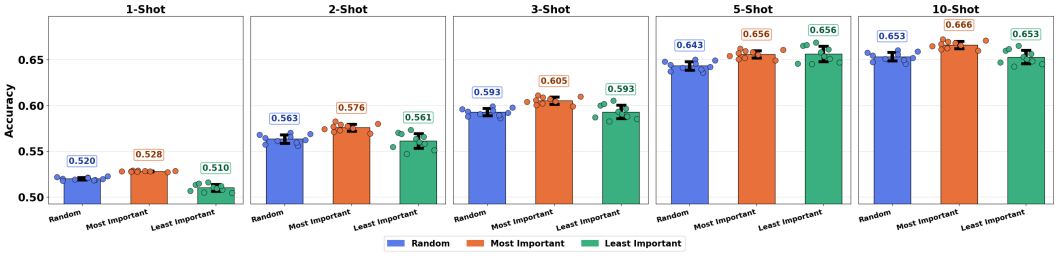

Figure 22: Token importance ablation on MRPC (RoBERTa-base, 10 seeds, error bars = 1σ). Most Important (high $\lambda_{\min}^+(K_t)$) consistently outperforms Random and Least Important. The gap is largest at $M = 1$ (0.018) where the bound of Theorem 2 is loosest, and narrows as M grows and $K^{(M)}$ aggregates gradient signal.

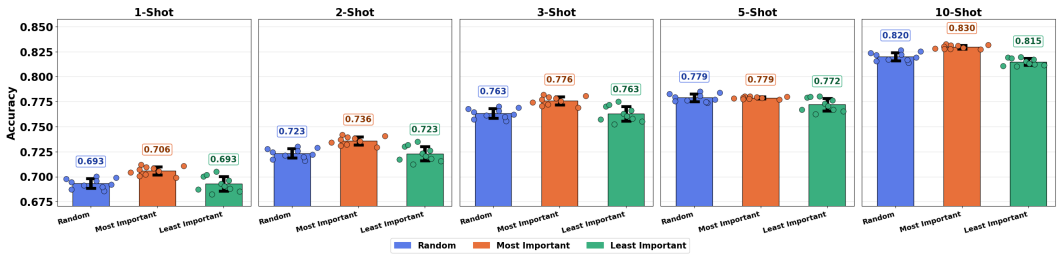

Figure 23: Token importance ablation on QQP (RoBERTa-base, 10 seeds, error bars = 1σ). Same ordering as MRPC: Most Important $\geq$ Random $\geq$ Least Important. At $M = 1$ the gap is 0.013; the ordering is preserved but with comparable magnitude at $M = 10$, consistent with the conditioning improvement predicted by spectral aggregation.

Interpretation. These findings refine rather than contradict the main theorem. Theorem 2 says that positions with aligned hidden-state geometry produce equivalent predictors *when the error terms of the bound are small*, which in practice holds once N is not pathologically small. The low-shot experiments probe the regime where the error terms become non-trivial—exactly the regime where the theorem predicts the λ_0^+ structure matters—and the empirical ordering matches: positions with larger λ_0^+ (Most Important) yield better fine-tuning, and the advantage shrinks as M grows. In the generic large-N regime used for all other experiments in this paper (Section 5), these effects are dominated by the primary capacity bottleneck of Theorem 1, and random position selection is both theoretically adequate and empirically sufficient (Section 4).

C Monotone Output-Head NTK

C.1 Proof of Theorem 3

Recall from Section 3 that we work in the output-head NTK regime where the feature map reduces to the final-layer hidden state, $\phi_t(x) = h_t^{(D)}(x) \in \mathbb{R}^d$, and the output-head Gram matrix is $K_t[n,m] = \langle h_t^{(D)}(x^{(n)}), h_t^{(D)}(x^{(m)})\rangle$, directly computable from forward passes. We restate Theorem 3 for convenience before giving the proof.

Theorem 3 (restated). *Suppose the final-layer hidden states of a causal pretrained transformer satisfy the adjacent-state contraction condition: for each position $t \in \{1,\dots,T-1\}$ there exists a single matrix $A_t \in \mathbb{R}^{d\times d}$ with $\|A_t\|_{\mathrm{op}} \le 1$, shared uniformly across all inputs x, and residual vectors $\varepsilon_t(x) \in \mathbb{R}^d$, such that*

$$h_t^{(D)}(x) \;=\; A_t\, h_{t+1}^{(D)}(x) + \varepsilon_t(x).$$

Define the pointwise bounds

$$\sigma_t \;:=\; \max_{n=1,\dots,N} \|\varepsilon_t(x^{(n)})\|_2, \qquad \beta_t \;:=\; \max_{n=1,\dots,N} \|h_t^{(D)}(x^{(n)})\|_2,$$

both computable directly from the data. Then the output-head Gram matrices satisfy the approximate positive semidefinite ordering

$$K_{t+1} \;\succeq\; K_t - \delta_t\, I_N, \qquad \delta_t \;:=\; N\sigma_t(2\beta_{t+1} + \sigma_t), \tag{12}$$

for every $t \in \{1,\dots,T-1\}$.

The adjacent-state contraction condition is a regularity property of the joint statistics of final-layer hidden states at adjacent positions, consistent with the way causal attention couples them: at every layer, the update at position $t{+}1$ reads from positions $1,\dots,t{+}1$, including position t, so adjacent hidden states share lineage rather than being independently produced. The direction $h_t^{(D)} = A_t h_{t+1}^{(D)} + \varepsilon_t$ should be read as a description of how the learned representations at adjacent positions are geometrically related, not as a forward-pass causal computation. The substantive restriction is the uniformity of A_t across inputs: it would fail if the attention-mediated relationship between $h_t^{(D)}(x)$ and $h_{t+1}^{(D)}(x)$ varied significantly with x. We define σ_t via the maximum over samples (rather than an average) because the proof uses $\|E_t\|_F^2 \le N\sigma_t^2$, which requires a uniform-over-samples bound. We empirically validate this assumption in Appendix C.4: composing the model's own (frozen) causal attention at the $(t, t{+}1)$ pair with a trainable linear map, we find $\|A_t\|_{\mathrm{op}} \approx 1$ across the bulk of layers in SmolLM2-135M.

Proof. Define the hidden-state matrix $H_t := [h_t^{(D)}(x^{(1)}), \dots, h_t^{(D)}(x^{(N)})] \in \mathbb{R}^{d\times N}$ and the residual matrix $E_t := [\varepsilon_t(x^{(1)}), \dots, \varepsilon_t(x^{(N)})] \in \mathbb{R}^{d\times N}$. Since A_t is the same matrix for all inputs, applying the contraction assumption column-by-column gives the matrix identity

$$H_t \;=\; A_t H_{t+1} + E_t,$$

whose n-th column is $A_t h_{t+1}^{(D)}(x^{(n)}) + \varepsilon_t(x^{(n)}) = h_t^{(D)}(x^{(n)})$. The uniformity of A_t across inputs is what allows this single matrix equation to hold; if A_t depended on x, the factorization would break. The output-head Gram matrix is $K_t = H_t^\top H_t$, so for any $v \in \mathbb{R}^N$,

$$v^\top K_t v \;=\; \|H_t v\|_2^2 \;=\; \|A_t H_{t+1} v + E_t v\|_2^2.$$

Expanding using $\|a+b\|_2^2 = \|a\|_2^2 + 2\langle a, b\rangle + \|b\|_2^2$ with $a := A_t H_{t+1} v \in \mathbb{R}^d$ and $b := E_t v \in \mathbb{R}^d$,

$$v^\top K_t v \;=\; \underbrace{\|A_t H_{t+1} v\|_2^2}_{T_1} \;+\; \underbrace{2\langle A_t H_{t+1} v,\, E_t v\rangle}_{T_2} \;+\; \underbrace{\|E_t v\|_2^2}_{T_3}. \tag{13}$$

We bound each term in turn.

Bounding T_1. Since $\|A_t\|_{\mathrm{op}} \le 1$, for any vector $w \in \mathbb{R}^d$ we have $\|A_t w\|_2 \le \|A_t\|_{\mathrm{op}}\|w\|_2 \le \|w\|_2$. Applying this with $w = H_{t+1} v$,

$$T_1 \;=\; \|A_t H_{t+1} v\|_2^2 \;\le\; \|H_{t+1} v\|_2^2 \;=\; v^\top H_{t+1}^\top H_{t+1} v \;=\; v^\top K_{t+1} v.$$

Bounding T_2. Since T_2 is real, $T_2 \le |T_2|$ trivially. This step is conservative when $T_2 < 0$, in which case δ_t is an over-estimate (the true quadratic form is smaller than what we bound); but in all cases the bound remains valid. By the Cauchy–Schwarz inequality,

$$|T_2| \;=\; 2|\langle A_t H_{t+1} v,\, E_t v\rangle| \;\le\; 2\|A_t H_{t+1} v\|_2 \cdot \|E_t v\|_2.$$

We bound each factor. Applying the contraction $\|A_t\|_{\mathrm{op}} \le 1$,

$$\|A_t H_{t+1} v\|_2 \;\le\; \|H_{t+1} v\|_2.$$

For the remaining operator-norm bounds we use $\|Mv\|_2 \le \|M\|_{\mathrm{op}}\|v\|_2 \le \|M\|_F\|v\|_2$, which combines the operator-Frobenius inequality $\|M\|_{\mathrm{op}} \le \|M\|_F$ with the standard operator-norm bound. This gives

$$\|H_{t+1} v\|_2 \;\le\; \|H_{t+1}\|_F \|v\|_2, \qquad \|E_t v\|_2 \;\le\; \|E_t\|_F \|v\|_2.$$

The Frobenius norms are controlled by the pointwise bounds:

$$\|H_{t+1}\|_F^2 \;=\; \sum_{n=1}^{N} \|h_{t+1}^{(D)}(x^{(n)})\|_2^2 \;\le\; N\beta_{t+1}^2, \qquad \|E_t\|_F^2 \;=\; \sum_{n=1}^{N} \|\varepsilon_t(x^{(n)})\|_2^2 \;\le\; N\sigma_t^2.$$

Hence $\|H_{t+1} v\|_2 \le \sqrt{N}\beta_{t+1}\|v\|_2$ and $\|E_t v\|_2 \le \sqrt{N}\sigma_t\|v\|_2$. This choice of the Frobenius bound (rather than the tighter $\|H_{t+1}\|_{\mathrm{op}} = \sqrt{\lambda_{\max}(K_{t+1})}$) is deliberate: it keeps δ_t expressible purely in terms of β_{t+1} and σ_t, avoiding circular dependence on the spectrum of K_{t+1}. Combining the two factor bounds,

$$T_2 \;\le\; |T_2| \;\le\; 2 \cdot \sqrt{N}\beta_{t+1}\|v\|_2 \cdot \sqrt{N}\sigma_t\|v\|_2 \;=\; 2N\beta_{t+1}\sigma_t\|v\|_2^2.$$

Bounding T_3. Squaring the bound on $\|E_t v\|_2$ established above,

$$T_3 \;=\; \|E_t v\|_2^2 \;\le\; N\sigma_t^2\|v\|_2^2.$$

Combining. Substituting the three bounds into (13),

$$v^\top K_t v \;\le\; v^\top K_{t+1} v + 2N\beta_{t+1}\sigma_t\|v\|_2^2 + N\sigma_t^2\|v\|_2^2 \;=\; v^\top K_{t+1} v + N\sigma_t(2\beta_{t+1} + \sigma_t)\|v\|_2^2.$$

Rearranging,

$$v^\top \big[K_{t+1} - K_t + N\sigma_t(2\beta_{t+1} + \sigma_t)\, I_N\big] v \;\ge\; 0.$$

Since $v \in \mathbb{R}^N$ was arbitrary, the matrix in brackets is positive semidefinite, which is the claim $K_{t+1} \succeq K_t - \delta_t I_N$ with $\delta_t = N\sigma_t(2\beta_{t+1} + \sigma_t)$. The coefficient δ_t depends only on β_{t+1} and σ_t, both computable directly from the data without reference to the spectrum of any Gram matrix. □

Remarks on the bound. *Informative regime.* The bound is informative when $\lambda_{\min}(K_t) \gg \delta_t$: in that regime every eigenvalue of K_t is positive and substantially larger than the spectral shift, so the ordering places genuine constraints on the eigenvalue spectra of the two Gram matrices. When $\delta_t \ge \lambda_{\min}(K_t)$, the shift can make $K_t - \delta_t I_N$ indefinite and the bound is vacuous. The assumption is also trivially satisfiable by taking $A_t = 0$ and $\varepsilon_t(x) = h_t^{(D)}(x)$, which gives $\sigma_t = \beta_t$ and an error coefficient $N\beta_t(2\beta_{t+1} + \beta_t)$ that renders the bound vacuous in most practical regimes. The theorem is therefore non-trivial only when a genuinely contractive A_t exists with $\sigma_t \ll \beta_t$; verifying this empirically before invoking the result is essential, and we do so in Appendix C.4. *Exact ordering when $\sigma_t = 0$.* When $\sigma_t = 0$ the residuals vanish so $H_t = A_t H_{t+1}$ exactly, giving $T_2 = 0$, $T_3 = 0$, and $\delta_t = 0$. The expansion (13) then gives $v^\top K_t v = T_1 \le v^\top K_{t+1} v$ for all v, which is exactly the PSD ordering $K_t \preceq K_{t+1}$. Applying this for every $t \in \{1, \dots, T-1\}$ and using transitivity of the PSD order yields the exact chain

$$K_1 \;\preceq\; K_2 \;\preceq\; \cdots \;\preceq\; K_T.$$

Capacity interpretation in the exact case. The PSD ordering $K_t \preceq K_{t+1}$ implies the image inclusion $\mathrm{Im}(K_t) \subseteq \mathrm{Im}(K_{t+1})$. We establish this via the null spaces. If $K_{t+1} v = 0$ then

$$0 \;\le\; v^\top K_t v \;\le\; v^\top K_{t+1} v \;=\; 0,$$

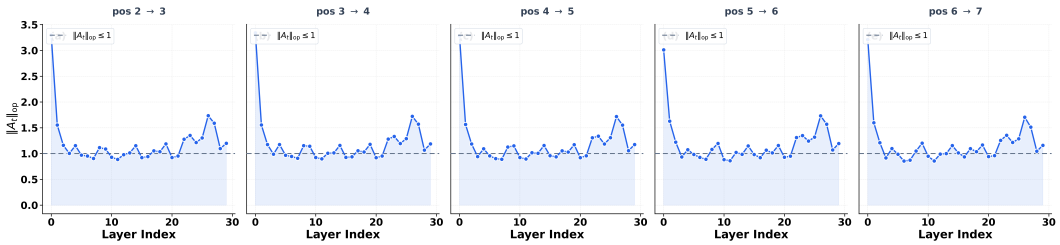

Figure 24: Operator norm $\|A_t\|_{\mathrm{op}}$ of the trained linear map across all 30 layers of SmolLM2-135M for five consecutive position pairs $(t, t{+}1)$ with $t \in \{2, 3, 4, 5, 6\}$. The dashed line marks the contraction threshold $\|A_t\|_{\mathrm{op}} = 1$ required by Theorem 3. Across all five pairs, $\|A_t\|_{\mathrm{op}}$ stays at or near the threshold throughout the bulk of the network (layers $\ell \in [3, 23]$), with violations confined to the embedding layer ($\ell = 0$, $\|A_t\|_{\mathrm{op}} \approx 3$) and a small late-layer bump ($\ell \in [24, 28]$, $\|A_t\|_{\mathrm{op}} \approx 1.7$).

forcing $v^\top K_t v = 0$. Since K_t is positive semidefinite, this implies $K_t v = 0$, establishing $\ker(K_{t+1}) \subseteq \ker(K_t)$. Since K_t and K_{t+1} are symmetric positive semidefinite, the fundamental theorem of linear algebra gives $\mathrm{Im}(K_t) = \ker(K_t)^\perp$ and $\mathrm{Im}(K_{t+1}) = \ker(K_{t+1})^\perp$. Using that taking orthogonal complements reverses inclusions ($A \subseteq B$ implies $B^\perp \subseteq A^\perp$),

$$\mathrm{Im}(K_t) \;=\; \ker(K_t)^\perp \;\subseteq\; \ker(K_{t+1})^\perp \;=\; \mathrm{Im}(K_{t+1}).$$

By Theorem 1, $\mathrm{Im}(K_t)$ is exactly the set of achievable learned corrections on the training set when training the output head through position t. The inclusion $\mathrm{Im}(K_t) \subseteq \mathrm{Im}(K_{t+1})$ therefore confirms that every labeling of the training set achievable by training via position t is also achievable via position $t + 1$. This is a statement about interpolation on the fixed training set $\{x^{(n)}\}_{n=1}^N$; it does not immediately extend to claims about new inputs (which would require RKHS containment as function spaces) or to the noisy case $\sigma_t > 0$, where the spectral shift $\delta_t I_N$ does not directly control kernel-regression prediction errors. Those depend additionally on the labeling vector y through the residual $r_t = y - o_t(X; \theta_0)$.

C.2 Empirical Validation of the A_t Assumption

We validate the adjacent-state contraction condition of Theorem 3—the existence of a matrix A_t with $\|A_t\|_{\mathrm{op}} \leq 1$ shared across inputs, satisfying $h_t^{(D)}(x) = A_t h_{t+1}^{(D)}(x) + \varepsilon_t(x)$—through a controlled experiment that grounds A_t in the model's own causal attention rather than fitting it from arbitrary linear regression.

We use SmolLM2-135M with 30 transformer layers and hidden dimension $d = 576$. The corpus consists of N=3,500 wikitext sentences, each tokenized to length T=16 with the model's standard pad token. For each layer $\ell \in \{0, 1, \ldots, 29\}$ and each consecutive position pair $(t, t{+}1)$ with $t \in \{2, 3, 4, 5, 6\}$, we collect the pre-attention hidden states $h_t^{(\ell)}$ and $h_{t+1}^{(\ell)}$ for every sample and proceed in two steps. First, we compute a frozen attention output by composing the pair $(h_t^{(\ell)}, h_{t+1}^{(\ell)})$ through the model's own causal self-attention block at layer ℓ, using the pretrained $W_Q^{(\ell)}, W_K^{(\ell)}, W_V^{(\ell)}, W_O^{(\ell)}$ projections and rotary position embeddings, with the query at position $t{+}1$ attending over $\{h_t^{(\ell)}, h_{t+1}^{(\ell)}\}$. The resulting attention output $\tilde{h}_t^{(\ell)}$ encodes how much of position t's representation is read into position $t{+}1$'s representation through the model's own attention computation. Second, we fit a trainable linear map $A_t \in \mathbb{R}^{d \times d}$ that maps this attention output back to the original $h_t^{(\ell)}$, using mean squared error on an 80%/20% train/test split, AdamW with learning rate $3{\times}10^{-4}$ and 60 epochs at batch size 256. The composition $A_t \circ \mathrm{Attn}^{(\ell)}$ is the attention-mediated map from $h_{t+1}^{(\ell)}$-region to $h_t^{(\ell)}$ that the theorem hypothesizes exists; we test whether it can be made contracting.

Figure 24 shows $\|A_t\|_{\mathrm{op}}$ across all 30 layers for each of the five position pairs. Three observations emerge clearly. First, across the bulk of the network (layers 3 through 23, roughly two-thirds of the layers), the trained A_t satisfies the contraction condition almost exactly: $\|A_t\|_{\mathrm{op}} \in [0.85, 1.15]$ at every layer, and several layers fall strictly below the threshold. Second, the pattern is highly consistent across all five position pairs—the operator norm at layer ℓ for pair $(2, 3)$ closely matches that for pair $(6, 7)$ at the same ℓ. This consistency directly supports the uniformity-across-positions

component of the assumption: the relationship between adjacent hidden states is structural rather than position-specific. Third, two regimes show meaningful violations: the embedding layer ($\ell = 0$) with $\|A_t\|_{\text{op}} \approx 3$, and a late-layer region ($\ell \in [24, 28]$) where the operator norm rises briefly to ≈ 1.7 before settling back. The first violation arises because at layer 0 the hidden states are the raw token embeddings $h_t^{(0)}(x) = E_{\text{tok}}[x_t] + E_{\text{pos}}[t]$, prior to any attention update. Different positions correspond to different tokens drawn from very different distributions over the vocabulary, and the embedding-layer hidden states have not yet been contextualized by attention. There is therefore no architectural mechanism connecting $h_t^{(0)}(x)$ and $h_{t+1}^{(0)}(x)$ except through the small positional-embedding component, which is shared across inputs but does not capture token-level correlations between adjacent positions. The trained A_t at this layer must reconstruct $h_t^{(0)}$ from a function of $h_{t+1}^{(0)}$ that has effectively no information about token t, and consequently must amplify in directions where the two are nearly orthogonal; this is the only layer where the architectural assumption underlying our theorem genuinely does not hold, and the operator norm reflects this honestly.

The bump at layers 24–28 has a different cause. In the final stages of a pretrained language model, hidden states begin to specialize toward the next-token prediction objective: the residual stream rotates from a representation of the current token's context toward a representation that is closer to the logits over the next token. This specialization is sharper at the latest position in the sequence (which produces the prediction) than at earlier positions, so $h_t^{(\ell)}$ and $h_{t+1}^{(\ell)}$ for ℓ near D encode increasingly different aspects of the prediction task: position $t{+}1$'s representation aligns more strongly with the output-head direction, while position t's remains tied to the contextual representation. Recovering $h_t^{(\ell)}$ from a linear map of position-$(t{+}1)$ attention output therefore requires a slightly amplifying A_t in the output-head direction, producing the observed late-layer bump. The violation is small ($\|A_t\|_{\text{op}} \leq 1.75$, not orders of magnitude) and confined to a narrow layer band, indicating that the contraction condition still holds approximately even in this specialized regime.

The empirical evidence therefore supports the adjacent-state contraction condition in the regime where Theorem 3 is most relevant: the bulk of the transformer's computation, where attention has contextualized representations and the residual stream is dominated by structured information sharing across positions. Where the condition is violated, the architectural cause is clear and localized—the embedding layer, which the theorem does not need to apply to since its conclusion concerns Gram matrices of hidden states after attention, and the output-head region, where the violation is small in magnitude. The uniformity across the five tested position pairs additionally supports that A_t is structurally determined rather than position-specific, addressing the substantive uniformity-across-inputs requirement of the theorem.

C.3 Empirical Validation of Theorem 3's Contraction Assumptions

Theorem 3 relies on two quantitative conditions: the existence of a contraction A_t with $\|A_t\|_{\text{op}} \leq 1$ relating adjacent hidden states, and a small residual σ_t ensuring the slack term $\delta_t = N\sigma_t(2\beta_{t+1} + \sigma_t)$ remains below $\lambda_{\min}^+(K_t)$. Appendix C.2 verified the first condition by composing the model's own causal attention with a trained linear map. Here we directly measure the four quantities entering the bound—σ_t, β_t, δ_t, and $\lambda_{\min}^+(K_t)$—across two pretrained causal transformers of different scales to confirm that the bound is non-vacuous in practice.

We use SmolLM2-135M [2] and LLaMA 3-8B [14] on $N{=}512$ WikiText-103 [40] sequences, extracting hidden states $h_t^{(\ell)}$ at a mid-network layer ($\ell{=}15$ for SmolLM2-135M, $\ell{=}16$ for LLaMA 3-8B) where $\|A_t\|_{\text{op}} \approx 1$ in Appendix C.2. Special tokens (BOS, EOS, padding) are excluded so each position corresponds to genuine content. For each position pair $(t, t{+}1)$, we fit the contraction by closed-form least squares, $A_t = (H_{t+1}^+ H_t)^\top$, and compute the per-sample residual $\varepsilon_t(x^{(n)}) = h_t^{(\ell)}(x^{(n)}) - A_t\, h_{t+1}^{(\ell)}(x^{(n)})$. We then evaluate $\sigma_t := \max_n \|\varepsilon_t(x^{(n)})\|_2$, $\beta_t := \max_n \|h_t^{(\ell)}(x^{(n)})\|_2$, $\delta_t = N\sigma_t(2\beta_{t+1} + \sigma_t)$, and $\lambda_{\min}^+(K_t)$, the smallest eigenvalue of $K_t = H_t H_t^\top$ above a relative threshold of $10^{-8} \cdot \lambda_{\max}(K_t)$.

Figures 25 and 26 confirm all four quantitative conditions of Theorem 3 on both architectures despite a $60\times$ difference in parameter count. The residual σ_t stays at the floating-point noise floor across all positions on both models ($\approx 10^{-12}$ on SmolLM2-135M, $\approx 10^{-13}$ on LLaMA 3-8B), indicating that the contraction $h_t^{(\ell)} = A_t h_{t+1}^{(\ell)} + \varepsilon_t$ holds essentially exactly on the data used to

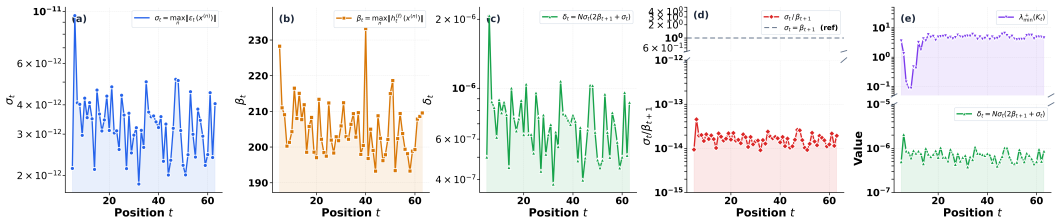

Figure 25: Empirical validation of Theorem 3's contraction assumptions on SmolLM2-135M (N=512, T=64, layer ℓ=15). **(a)** Residual size $\sigma_t \in [2, 10] \times 10^{-12}$, at the floating-point noise floor across all positions. **(b)** Hidden-state norm $\beta_t \approx 200$, bounded and stable. **(c)** Slack term $\delta_t \in [4, 20] \times 10^{-7}$, small in absolute scale. **(d)** Relative residual $\sigma_t/\beta_{t+1} \approx 10^{-14}$, far below the $\sigma_t = \beta_{t+1}$ threshold (dashed line, y=1); the contraction approximation is essentially exact. **(e)** The slack δ_t (green, $\approx 10^{-6}$) is six to seven orders of magnitude smaller than $\lambda^{+}_{\min}(K_t)$ (purple, $\approx 10^{0}$–10^{1}), so the PSD bound $K_{t+1} \succeq K_t - \delta_t I$ is informative at every position. Panels (d) and (e) use a broken y-axis to make the gap visible.

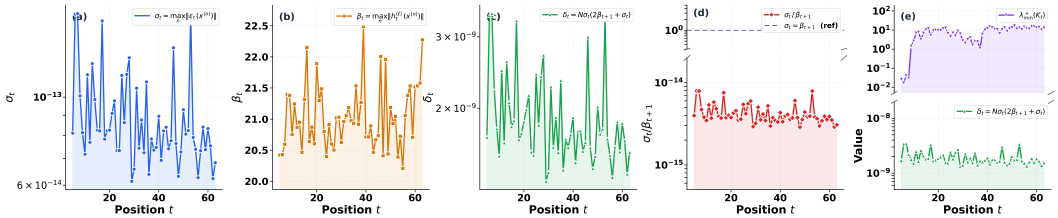

Figure 26: Empirical validation of Theorem 3's contraction assumptions on LLaMA 3-8B (N=512, T=64, layer ℓ=16). **(a)** Residual size $\sigma_t \in [6 \times 10^{-14}, 2 \times 10^{-13}]$, at the floating-point noise floor. **(b)** Hidden-state norm $\beta_t \in [20, 22.5]$, bounded and stable. **(c)** Slack term $\delta_t \in [1.5, 3.5] \times 10^{-9}$, small in absolute scale. **(d)** Relative residual $\sigma_t/\beta_{t+1} \approx 10^{-15}$, far below the $\sigma_t = \beta_{t+1}$ threshold; the contraction approximation is essentially exact. **(e)** The slack δ_t (green, $\approx 10^{-9}$) is seven to ten orders of magnitude smaller than $\lambda^{+}_{\min}(K_t)$ (purple, $\approx 10^{-2}$–10^{1}), so the PSD bound is informative at every position. Panels (d) and (e) use a broken y-axis to make the gap visible.

define K_t. The relative residual σ_t/β_{t+1} is $\approx 10^{-14}$ for SmolLM2 and $\approx 10^{-15}$ for LLaMA 3-8B, well below the $\sigma_t = \beta_{t+1}$ threshold at which the bound becomes non-trivial. Most importantly, δ_t is six to ten orders of magnitude smaller than $\lambda^{+}_{\min}(K_t)$ at every position on both models, so the slack term in $K_{t+1} \succeq K_t - \delta_t I$ never dominates the smallest nonzero eigenvalue. Combined with the operator-norm validation in Appendix C.4 (where $\|A_t\|_{\text{op}} \approx 1$ across the bulk of layers), this establishes that Theorem 3's assumptions hold with substantial margin in practice on both small and large pretrained causal models, and the resulting monotone-capacity ordering $K_1 \preceq K_2 \preceq \cdots \preceq K_T$ (up to vanishing slack) is empirically meaningful rather than a vacuous worst-case statement. The contraction is fitted on the same N samples that define K_t, which matches the theorem's statement: σ_t ranges over the training set indexing K_t, so in-sample residuals are exactly the relevant quantity, and out-of-distribution generalization of the contraction is not required for the bound to hold.

C.4 Empirical Validation of Theorem 3

We validate Theorem 3 via linear probing on a causal pretrained model. Linear probing is the direct empirical counterpart of the output-head NTK capacity at position t: training a linear classifier on the frozen hidden state $h_t^{(D)}(x)$ realizes the kernel regression solution of Theorem 1 with kernel $\mathcal{K}_t(x, x') = \langle h_t^{(D)}(x), h_t^{(D)}(x') \rangle$, and the resulting accuracy measures the capacity ceiling of the $\mathcal{K}_t$-induced RKHS for the given task. Theorem 3 predicts that, under the hidden-state contraction assumption, the Gram matrices satisfy $K_t \preceq K_{t+1}$ (in the exact case) or $K_{t+1} \succeq K_t - \delta_t I_N$ (in the approximate case), so linear-probe accuracy should be *monotonically non-decreasing* as we move from earlier to later positions in the sequence.

We evaluate on four GLUE classification tasks—SST-2, QNLI, QQP, and MRPC—using SmolLM2-360M as the causal backbone. For each (dataset, layer, position) triple, we freeze the backbone,

extract the final-layer hidden state $h_t^{(D)}(x)$ at layer ℓ and position t, and train a single linear classifier on top. We sweep position $t \in \{1, 2, 3, 4\}$ along the sequence (ordered from earliest to latest within the probed prefix) and repeat the sweep at four transformer depths: layers 24, 22, 18, and 15 of the 32-layer network. For comparison we also report three pooling baselines—mean pooling, max pooling, and attention pooling—which aggregate information over the entire sequence rather than a single position. Accuracy is reported on the validation split of each task.

Figure 27 shows linear-probe accuracy as a function of position at each of the four probed layers. The central finding is that accuracy at Token-t strictly increases in t for every (dataset, layer) combination tested: across all four tasks and all four layers, moving from Token-1 to Token-4 improves accuracy monotonically. At the final layer (panel a), SST-2 accuracy grows from 57.1% at Token-1 to 90.3% at Token-4 ($+33.2$ points); QNLI grows from 56.0% to 85.2% ($+29.2$ points); QQP grows from 50.7% to 82.4% ($+31.7$ points); MRPC grows from 51.8% to 82.5% ($+30.7$ points). The same monotone ordering holds at layers 22, 18, and 15 (panels b–d), with Token-4 exceeding Token-1 by 29–33 absolute points on every task-layer pair. No reported (dataset, layer, position) triple violates the predicted ordering.

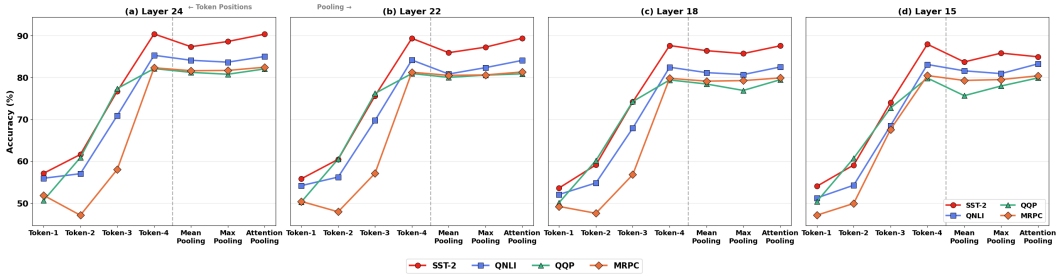

Figure 27: Empirical validation of Theorem 3 on SmolLM2-360M via linear probing of four GLUE classification tasks (SST-2, QNLI, QQP, MRPC). Each panel fixes a transformer layer (layers 24, 22, 18, and 15 of 32) and reports validation accuracy of a linear classifier trained on $h_t^{(\ell)}(x)$ at positions $t \in \{1, 2, 3, 4\}$ (left of the dashed line) and at three pooling baselines over the full sequence (right of the dashed line). Accuracy increases monotonically from Token-1 to Token-4 on every task at every layer, confirming the monotone-capacity ordering predicted by Theorem 3. Pooling baselines sit at or slightly above Token-4, reflecting that full-sequence aggregation has access to the same late-position information.

The monotone ordering is not specific to one depth. At layer 15, roughly the middle of the 32-layer network, the Token-1 to Token-4 gap is 31–33 points across tasks—comparable to the 29–33-point gap at layer 24. This indicates that the hidden-state contraction structure assumed by Theorem 3 is present across intermediate depths rather than at a single layer. Absolute Token-4 accuracy differs only modestly across depths (85.0–88.0% at layer 15 vs. 82.4–90.3% at layer 24), while the *relative* ordering across positions remains unchanged. The PSD ordering of Theorem 3, read at different depths, is therefore a robust geometric property of the learned representation rather than a single-layer phenomenon.

The three pooling baselines (mean, max, attention) aggregate over the whole sequence and are not directly predicted by Theorem 3, which is a single-position statement. At every layer, pooling accuracies sit in a narrow band at or slightly below Token-4. For example, at layer 24, SST-2 reaches 87.4% (mean), 88.6% (max), and 90.3% (attention), compared to 90.3% at Token-4; QNLI reaches 84.1%, 83.7%, and 85.0%, compared to 85.2% at Token-4. Pooling does not exceed the last probed position because the late-position hidden state already aggregates information from all earlier tokens through causal attention, whereas pooling mixes in early, low-capacity positions. This observation is consistent with Theorem 3: if $K_1 \preceq K_2 \preceq \cdots \preceq K_T$ (the exact case, approximately satisfied here), then $\mathrm{Im}(K_t) \subseteq \mathrm{Im}(K_T)$ for every t, so aggregating over earlier positions adds no new achievable directions and can only dilute the late-position signal.

Linear-probe accuracy is the empirical realization of the capacity quantity that Theorem 3 orders. The observed ordering $\mathrm{Accuracy}(K_1) \leq \mathrm{Accuracy}(K_2) \leq \cdots$ is the empirical counterpart of the predicted PSD ordering $K_1 \preceq K_2 \preceq \cdots$, with the image-inclusion argument of Theorem 3 guaranteeing that the set of achievable labelings does not shrink as t grows. The absence of any

ordering violation across 16 (task, layer) combinations provides direct evidence that, for this causal model and these tasks, the approximate hidden-state contraction assumption of Theorem 3 holds with σ_t small enough that the $\delta_t I_N$ shift does not disturb the qualitative ordering. Combined with the analysis of Theorem 1, this localizes the gains of token-masked fine-tuning: later positions in a causal sequence carry at least as much fine-tuning signal as earlier positions, and training through a small number of late positions captures essentially all the capacity available to full-sequence training.

D Optimal Position Count

Corollary 1 (restated) *Let $\{K_t\}_{t=1}^T$ be the output-head NTK Gram matrices on a fixed dataset $\{x^{(n)}\}_{n=1}^N$ as defined in Theorem 1. For any ordered subset $S_M = \{t_1, t_2, \ldots, t_M\} \subseteq \{1, \ldots, T\}$ with $S_1 \subset S_2 \subset \cdots \subset S_T$, define the cumulative combined Gram matrix*

$$\tilde{K}^{(M)} \;:=\; \sum_{m=1}^{M} K_{t_m} \;\in\; \mathbb{R}^{N\times N},$$

and let $\lambda^+ := \lambda^+_{\min}(\tilde{K}^{(T)}) > 0$ be the smallest nonzero eigenvalue of the full combined matrix. Then there exists a well-defined minimal position count

$$M^* \;:=\; \min\big\{M \in \{1,\ldots,T\} : \mathrm{Im}(\tilde{K}^{(M)}) = \mathrm{Im}(\tilde{K}^{(T)})\big\},$$

satisfying $1 \le M^* \le T$, such that adding positions beyond M^* does not expand the achievable prediction space on the training set. The optimal M^* depends on both the architecture and the selection strategy as follows. ***Bidirectional model (any selection strategy).*** *Under the conditions of Theorem 2, suppose $\|K_i - K_j\|_{\mathrm{op}} \le \epsilon_K$ for all i, j. Then*

$$M^* \;\le\; \left\lceil \frac{\epsilon_K \cdot T}{\lambda^+} + 1 \right\rceil,$$

a constant independent of N. In the exact case $\epsilon_K = 0$, $M^* = 1$.

Causal model, arbitrary selection. *Under the conditions of Theorem 3, if $\mathrm{Im}(K_t) \subsetneq \mathrm{Im}(K_{t+1})$ for all t, then for any selection strategy S_M that includes position $t < T$, the column space $\mathrm{Im}(\tilde{K}^{(M)})$ grows strictly with each new position added from S_M, so $M^* = T$.*

Causal model, last-M selection. *Under the same conditions, if instead we restrict to the last-M selection strategy $S_M = \{T - M + 1, \ldots, T\}$, then $\mathrm{Im}(\tilde{K}^{(M)}) = \mathrm{Im}(K_T)$ for every $M \ge 1$, so $M^* = 1$. In particular, fine-tuning through the single last position $t = T$ achieves the same achievable prediction space as fine-tuning through all T positions, making the last-M strategy memory-optimal for causal models.*

Proof. We work throughout with $\tilde{K}^{(M)} = \sum_{m=1}^{M} K_{t_m}$, where each K_{t_m} is symmetric positive semidefinite. We establish five claims: the image chain, well-definedness of M^*, the bidirectional bound, the causal arbitrary-selection result, and the causal last-M result.

Since each $K_{t_m} \succeq 0$, the cumulative matrix satisfies $\tilde{K}^{(M+1)} = \tilde{K}^{(M)} + K_{t_{M+1}} \succeq \tilde{K}^{(M)}$. Let $v \in \ker(\tilde{K}^{(M+1)})$. Then $v^\top \tilde{K}^{(M+1)} v = 0$, and since $\tilde{K}^{(M+1)} \succeq \tilde{K}^{(M)} \succeq 0$,

$$0 \;\le\; v^\top \tilde{K}^{(M)} v \;\le\; v^\top \tilde{K}^{(M+1)} v \;=\; 0,$$

forcing $v^\top \tilde{K}^{(M)} v = 0$, hence $\tilde{K}^{(M)} v = 0$ and $v \in \ker(\tilde{K}^{(M)})$. So $\ker(\tilde{K}^{(M+1)}) \subseteq \ker(\tilde{K}^{(M)})$. Since $\mathrm{Im}(K) = \ker(K)^\perp$ for symmetric positive semidefinite K, and $A \subseteq B \Rightarrow B^\perp \subseteq A^\perp$,

$$\mathrm{Im}(\tilde{K}^{(M)}) \;\subseteq\; \mathrm{Im}(\tilde{K}^{(M+1)}),$$

giving the nondecreasing chain

$$\mathrm{Im}(\tilde{K}^{(1)}) \;\subseteq\; \mathrm{Im}(\tilde{K}^{(2)}) \;\subseteq\; \cdots \;\subseteq\; \mathrm{Im}(\tilde{K}^{(T)}) \;\subseteq\; \mathbb{R}^N. \tag{14}$$

The rank sequence $\mathrm{rank}(\tilde{K}^{(1)}) \le \cdots \le \mathrm{rank}(\tilde{K}^{(T)}) \le N$ is nondecreasing and bounded.

Setting $R^* := \mathrm{rank}(\tilde{K}^{(T)})$, the rank sequence reaches R^* at some finite index. Once $\mathrm{rank}(\tilde{K}^{(M)}) = R^*$, both $\mathrm{Im}(\tilde{K}^{(M)})$ and $\mathrm{Im}(\tilde{K}^{(T)})$ are dimension-R^* subspaces with the former contained in the latter by (14), so they are equal. Hence $M^* = \min\{M : \mathrm{rank}(\tilde{K}^{(M)}) = R^*\}$ is well-defined with $1 \le M^* \le T$, and for all $M < M^*$ the image inclusion is strict, confirming minimality.

For the bidirectional bound, write $K_{t_m} = K_1 + E_m$ where $K_1 := K_{t_1}$ and $E_m := K_{t_m} - K_1$ with $\|E_m\|_{\mathrm{op}} \le \epsilon_K$. The full combined matrix satisfies $\tilde{K}^{(T)} = T \cdot K_1 + F$ where $F := \sum_{m=1}^{T} E_m$ and $\|F\|_{\mathrm{op}} \le T\epsilon_K$. For any unit vector $u \in \mathrm{Im}(\tilde{K}^{(T)})$, $u^\top \tilde{K}^{(T)} u \ge \lambda^+$, so

$$T \cdot u^\top K_1 u \;\ge\; u^\top \tilde{K}^{(T)} u - u^\top F u \;\ge\; \lambda^+ - T\epsilon_K,$$

giving $u^\top K_1 u \geq (\lambda^+ - T\epsilon_K)/T$. For $M_0 = \lceil \epsilon_K T/\lambda^+ + 1 \rceil$, the single-step matrix satisfies

$$u^\top \tilde{K}^{(M_0)} u \;\geq\; M_0 \cdot \frac{\lambda^+ - T\epsilon_K}{T} - M_0 \epsilon_K \;=\; M_0 \left(\frac{\lambda^+}{T} - 2\epsilon_K \right).$$

When $\epsilon_K < \lambda^+/(2T)$ this is strictly positive for all unit $u \in \mathrm{Im}(\tilde{K}^{(T)})$, so $\tilde{K}^{(M_0)}$ has no null vector in $\mathrm{Im}(\tilde{K}^{(T)})$. Since $\ker(\tilde{K}^{(M_0)}) \subseteq \ker(\tilde{K}^{(T)}) = \mathrm{Im}(\tilde{K}^{(T)})^\perp$ by the null space chain, we get $\ker(\tilde{K}^{(M_0)}) = \ker(\tilde{K}^{(T)})$ and hence $\mathrm{Im}(\tilde{K}^{(M_0)}) = \mathrm{Im}(\tilde{K}^{(T)})$, giving $M^* \leq M_0 = \lceil \epsilon_K T/\lambda^+ + 1 \rceil$. In the exact case $\epsilon_K = 0$, $\tilde{K}^{(M)} = M \cdot K_1$ for all M, so $\mathrm{Im}(\tilde{K}^{(1)}) = \mathrm{Im}(K_1) = \mathrm{Im}(\tilde{K}^{(T)})$ immediately, giving $M^* = 1$.

For the causal arbitrary-selection result, under $\mathrm{Im}(K_t) \subsetneq \mathrm{Im}(K_{t+1})$ for all t, any ordered selection $S_M = \{t_1 < \cdots < t_M\}$ satisfies $\mathrm{Im}(\tilde{K}^{(M)}) = \mathrm{Im}(K_{t_M})$ by the argument below. Since $\mathrm{Im}(K_{t_M}) \subsetneq \mathrm{Im}(K_{t_{M+1}}) \subseteq \mathrm{Im}(\tilde{K}^{(T)})$ when $t_M < T$, the image is strictly smaller than $\mathrm{Im}(\tilde{K}^{(T)})$ for all $M < T$, giving $M^* = T$.

For the causal last-M result, take $S_M = \{T - M + 1, \ldots, T\}$. By the monotone ordering from Theorem 3, $K_t \preceq K_T$ for all $t \leq T$, which by the image inclusion argument applied to the individual matrices gives $\mathrm{Im}(K_t) \subseteq \mathrm{Im}(K_T)$ for all $t \leq T$. Every column of each K_t therefore lies in $\mathrm{Im}(K_T)$, so every column of $\tilde{K}^{(M)} = \sum_{t=T-M+1}^{T} K_t$ lies in $\mathrm{Im}(K_T)$, giving $\mathrm{Im}(\tilde{K}^{(M)}) \subseteq \mathrm{Im}(K_T)$. Conversely, K_T is one of the summands in $\tilde{K}^{(M)}$ for every $M \geq 1$, so for any $z = K_T \alpha \in \mathrm{Im}(K_T)$,

$$\tilde{K}^{(M)} \alpha \;=\; K_T \alpha + \sum_{t=T-M+1}^{T-1} K_t \alpha \;=\; z + w,$$

where $w = \sum_{t=T-M+1}^{T-1} K_t \alpha \in \mathrm{Im}(K_{T-1}) \subseteq \mathrm{Im}(K_T)$. Hence $z = \tilde{K}^{(M)} \alpha - w$, and since $\tilde{K}^{(M)} \alpha \in \mathrm{Im}(\tilde{K}^{(M)})$ and $w \in \mathrm{Im}(K_T) \subseteq \mathrm{Im}(\tilde{K}^{(M)})$ (by the already-established forward inclusion), $z \in \mathrm{Im}(\tilde{K}^{(M)})$. Therefore $\mathrm{Im}(K_T) \subseteq \mathrm{Im}(\tilde{K}^{(M)})$, so $\mathrm{Im}(\tilde{K}^{(M)}) = \mathrm{Im}(K_T)$ for every $M \geq 1$. Since K_T is the dominant matrix in the PSD ordering, $\mathrm{Im}(K_T) = \mathrm{Im}(\tilde{K}^{(T)})$ by the same argument applied with $M = T$. Hence $\mathrm{Im}(\tilde{K}^{(M)}) = \mathrm{Im}(\tilde{K}^{(T)})$ for every $M \geq 1$, giving $M^* = 1$. □

E Ablation Studies

We conduct five ablation studies to characterize the memory efficiency, convergence behavior, NTK validity, and capability retention of the token-masking procedure across model families, sequence lengths, and tasks. All experiments use gradient masking on M trainable token positions out of a full sequence of length T; the remaining $T - M$ positions are evaluated without gradient tracking and contribute to neither loss nor gradient signal. The memory and convergence ablations are summarized in Figure 28; the capability retention ablation is reported in Table 8.

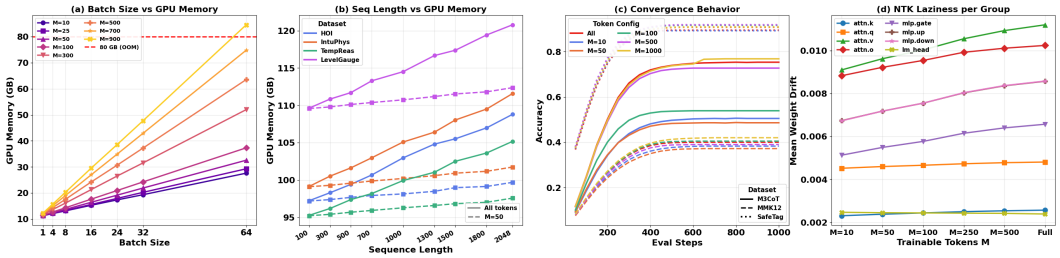

Figure 28: Ablation studies for token-masked fine-tuning. **(a)** GPU memory vs. batch size on LLaMA 3.2-1B for $M \in \{10, 25, 50, 100, 300, 500, 700, 900\}$, averaged over ARC-C, ARC-E, BoolQ, HellaSwag, and OpenBookQA. Memory scales linearly with batch size at a rate proportional to M; using $M{=}50$ instead of $M{=}900$ at batch size 64 saves roughly 55 GB. **(b)** GPU memory vs. sequence length on Qwen3-VL-8B for four video reasoning datasets (HOI, IntuPhys, TempReas, LevelGauge). Full fine-tuning (solid) grows steeply with T; $M{=}50$ token masking (dashed) is nearly flat, saving up to 97.6% of activation memory at $T{=}2048$. **(c)** Accuracy learning curves on LLaVA-OneVision-7B for VQA datasets (M3CoT, MMK12, SafeTag) under full-sequence and $M \in \{10, 50, 100, 500, 1000\}$ training, reported over 1000 evaluation steps. $M{=}50$–100 matches or exceeds full-sequence accuracy while using a fraction of the memory. **(d)** Mean weight drift per parameter group on LLaMA 3-8B (BoolQ) as M varies. Drift remains below 0.011 across every group and every M, empirically supporting the NTK regime assumed by Theorem 1.

E.1 Batch size scaling

Figure 28(a) shows GPU memory on LLaMA 3.2-1B across batch sizes $B \in \{1, 4, 8, 16, 24, 32, 64\}$ and trainable token counts $M \in \{10, 25, 50, 100, 300, 500, 700, 900\}$, averaged over ARC-C, ARC-E, BoolQ, HellaSwag, and OpenBookQA. At $B{=}1$ the differences across M are small (11.2–12.4 GB), because model weights and optimizer state dominate the memory footprint. As batch size grows the separation between curves widens sharply: at $B{=}64$, $M{=}10$ consumes 26.9 GB while $M{=}900$ consumes 82.7 GB—the latter crossing the 80 GB out-of-memory boundary marked in the figure. Memory grows linearly in B at a slope proportional to M, which is the expected behavior when activation memory scales approximately as $B \times M \times d \times D$ for hidden dimension d and layer count D. In concrete terms, using $M{=}50$ rather than $M{=}900$ at $B{=}64$ saves roughly 55 GB, equivalent to a $3.6\times$ increase in effective batch size under a fixed memory budget.

E.2 Sequence length scaling

Figure 28(b) compares full-sequence training (solid) against $M{=}50$ token masking (dashed) on Qwen3-VL-8B for four video reasoning datasets—HOI, IntuPhys, TempReas, and LevelGauge—at sequence lengths $T \in \{100, 300, 500, 700, 1000, 1300, 1500, 1800, 2048\}$ on two GPUs. Full-sequence memory grows steeply with T, reaching 108–121 GB at $T{=}2048$ depending on the dataset. The token-masked curves are nearly flat over the same range, growing by only 2–3 GB end-to-end and remaining below 112 GB at $T{=}2048$. The saving ratio predicted by the method is $1 - M/T$, which evaluates to $1 - 50/2048 \approx 97.6\%$ at the longest tested T—in agreement with the observed gap between solid and dashed curves. In practice, this means token masking largely decouples activation memory from sequence length, and long-context training runs that would otherwise trigger out-of-memory errors become feasible on the same hardware.

E.3 Convergence across M

Figure 28(c) shows accuracy learning curves on LLaVA-OneVision-7B for three VQA datasets—M3CoT, MMK12, and SafeTag—comparing full-sequence training to $M \in \{10, 50, 100, 500, 1000\}$ over 1000 evaluation steps. On SafeTag, $M{=}50$ and $M{=}100$ match or exceed the full-sequence accuracy by step 600 (0.907 and 0.916 respectively, versus 0.894 for full), and $M{=}500$ achieves the highest final accuracy (0.918). On M3CoT, $M{=}1000$ reaches 0.768 versus 0.753 for full by step 1000. On MMK12, $M{=}100$ matches the full baseline (0.405 vs. 0.400), and smaller M values converge to slightly lower but comparable values. Across the three datasets, $M{=}50$–100 provides the best trade-off: accuracy within 1–2% of full-sequence training at a fraction of the memory. This behavior is consistent with Theorem 1, which bounds the effective learning subspace of a single position by $d_{\text{eff}}(K_t)$. Once M is large enough to span the dominant gradient directions, further increases in M add redundant signal and carry only a memory cost.

E.4 NTK laziness: weight drift and kernel stability

Our analysis linearizes fine-tuning around θ_0. We test this assumption at two levels: the parameters, via weight drift, which bears on the full-NTK regime of Theorem 1; and the position-wise kernel itself, which is the object Theorems 2–3 depend on in the representation regime. Small weight drift is only indirect evidence for linearity, since small parameter changes can in principle rotate the kernel, so the kernel-level measurements below provide the more direct test.

Weight drift. Figure 28(d) reports the relative weight drift $\|W_\ell^{\text{final}} - W_\ell^0\|_F / \|W_\ell^0\|_F$ per parameter group on LLaMA 3-8B fine-tuned on BoolQ, as M varies from 10 to full fine-tuning. Across every group and every value of M, mean drift remains below 0.011, and the global drift $\|\Delta\theta\|_F / \|\theta_0\|_F$ stays under 0.008. The per-group picture reveals systematic structure: attention value and output weights (`attn.v`, `attn.o`) exhibit the largest drift (0.009–0.011), attention key and query weights (`attn.k`, `attn.q`) drift least (0.002–0.005), and MLP layers (`mlp.gate`, `mlp.up`, `mlp.down`) sit in between (0.005–0.009). This asymmetry is consistent with the interpretation that retrieval patterns (W_Q, W_K) are already well-formed at pretraining and require little adaptation, while the content pathway (W_V, W_O) carries most of the task-specific signal. The `lm_head` drift is small (≈ 0.0024) and slightly *decreases* as M grows, suggesting that additional gradient signal reduces reliance on shifting the output projection. Across all groups, drift grows slowly and sublinearly with M: from $M{=}10$ to full fine-tuning, the increase is below 25% for every group.

Kernel alignment. To assess the linearization directly at the level of the kernel, we compute the position-wise Gram matrix K_t on a fixed held-out set for the pretrained model (K_0) and for the model after fine-tuning (K_1), using LLaMA 3-8B on BoolQ with $M{=}100$, at four depths and five positions. We measure their similarity through the kernel alignment

$$\text{Align}(K_0, K_1) = \frac{\langle K_0, K_1 \rangle_F}{\|K_0\|_F \, \|K_1\|_F}. \tag{15}$$

As reported in Table 6, alignment exceeds 0.95 in every configuration, with a median of 0.980, indicating that the geometry of the position-wise kernel is largely preserved under fine-tuning. Alignment further exhibits a consistent dependence on position: it is highest at the final positions (0.978–0.990 at positions -1, -2, and -5) and decreases with distance from the end of the sequence (0.955–0.967 at position -20). The linear approximation is therefore most accurate at the positions selected by the last-M rule, which are precisely those to which Theorem 3 is applied.

To test whether the kernel continues to drift with additional training, we fine-tune LLaMA 3-8B on ARC-Challenge with $M{=}100$ for four epochs and measure $\text{Align}(K_0, K_e)$ after each epoch e on a fixed held-out set, at five positions and four depths (Table 7). Nearly all of the change occurs in the first epoch. From epoch 1 to epoch 4 the mean absolute change in alignment is 0.0015, and 19 of 20 cells change by less than 0.002, so the kernel stabilizes after the initial update rather than drifting further as training proceeds. Alignment on ARC-Challenge (0.92–0.97) is lower than on BoolQ (0.95–0.99), indicating that the quality of the approximation is task-dependent.

Together these results support the local linear approximation underlying Theorems 1–3. They do not establish exact laziness: the kernel moves by a task-dependent amount during early training, and we therefore use the NTK analysis as a predictive design model rather than an exact description of fine-tuning dynamics (Section 3).

Table 6: Kernel alignment $\mathrm{Align}(K_0, K_1)$ between the pretrained and fine-tuned position-wise Gram matrices (LLaMA 3-8B, BoolQ, $M{=}100$). Positions are indexed from the end of the sequence.

Position	Layer 8	Layer 16	Layer 24	Layer 32
-1	0.9899	0.9896	0.9798	0.9899
-2	0.9899	0.9897	0.9779	0.9898
-5	0.9799	0.9898	0.9795	0.9898
-10	0.9679	0.9685	0.9750	0.9680
-20	0.9648	0.9603	0.9668	0.9547

Table 7: Kernel alignment $\mathrm{Align}(K_0, K_e)$ after each epoch e (LLaMA 3-8B, ARC-Challenge, $M{=}100$), summarized over 5 positions $\times$ 4 layers.

Epoch	Min	Median	Max
1	0.921	0.942	0.972
2	0.922	0.944	0.972
3	0.922	0.942	0.973
4	0.922	0.944	0.972

Table 8: Capability retention analysis (Full Fine-tuning, LLaMA-3-8B). Model is fine-tuned on ARC-C only and evaluated on all five commonsense benchmarks. *Base* = pretrained LLaMA-3-8B zero-shot. Subscript deltas are absolute deviations from *Base*: ↑ green = gain, ↓ red = degradation. **Bold** = best per column. The Token Hypothesis preserves the base model's capabilities on the four untrained tasks while matching or exceeding full fine-tuning on the target task.

Method	**ARC-C** (target)	**ARC-E**	**BoolQ**	**HSwag**	**OBQA**	**Avg Drop**
Base	0.4445	**0.6768**	**0.5682**	**0.7836**	**0.4500**	—
Full FT	0.728 ↑0.284	0.615 ↓0.062	0.502 ↓0.066	0.711 ↓0.073	0.388 ↓0.062	0.554 ↓0.066
$M{=}100$	**0.764** ↑0.320	0.661 ↓0.016	0.554 ↓0.014	0.769 ↓0.015	0.435 ↓0.015	0.605 ↓0.015
$M{=}50$	0.761 ↑0.317	0.670 ↓0.007	0.561 ↓0.007	0.777 ↓0.007	0.443 ↓0.007	0.613 ↓0.007
$M{=}25$	0.719 ↑0.275	0.674 ↓0.003	0.565 ↓0.003	0.781 ↓0.003	0.447 ↓0.003	0.617 ↓0.003
$M{=}10$	0.662 ↑0.218	0.676 ↓0.001	0.567 ↓0.001	0.783 ↓0.001	0.449 ↓0.001	**0.619** ↓0.001

E.5 Capability retention under targeted fine-tuning

A concern with any fine-tuning procedure is *catastrophic forgetting*: training on a single target task can degrade the model's performance on tasks outside the fine-tuning distribution. We evaluate this on LLaMA-3-8B by fine-tuning only on ARC-C and then testing the resulting model on all five commonsense benchmarks; the pretrained LLaMA-3-8B zero-shot baseline (denoted *Base*) serves as the reference. Results are reported in Table 8. Full fine-tuning improves ARC-C substantially (from 0.4445 to 0.728, $+0.284$) but degrades all four untrained benchmarks, with an average drop of 0.066 relative to *Base*. Token masking produces a markedly different pattern. At $M{=}100$, the target accuracy reaches 0.764—higher than full fine-tuning—while the average drop on the four untrained tasks is only 0.015, roughly a $4\times$ reduction in forgetting. The effect strengthens as M decreases: at $M{=}10$, the drop on untrained tasks falls to 0.001 (essentially no forgetting), with ARC-C still improved by 0.218 over *Base*. This is consistent with Theorem 1: training through fewer

positions confines parameter updates to a lower-dimensional subspace of the pretrained tangent space, which limits the extent to which base-model representations can be disrupted. In practice, small M therefore offers a controllable trade-off between target-task improvement and preservation of general capabilities—a property that full-sequence fine-tuning does not provide.

F Extended Related Work

We position the Token Hypothesis against four lines of work: parameter-efficient fine-tuning (PEFT), activation-memory reduction techniques, token-selective backpropagation, and Neural Tangent Kernel theories of fine-tuning. The first two address the GPU memory cost of fine-tuning along axes that are largely orthogonal to ours; the third targets the same axis (which positions to backpropagate through) but does so heuristically; and the fourth provides the theoretical lens our analysis builds on. Across all four, what is missing is a principled account of *why* backpropagating through a small subset of positions suffices, what determines its optimal size, and how that optimum depends on the architecture. The Token Hypothesis fills this gap.

F.1 Parameter-Efficient Fine-Tuning

Parameter-efficient fine-tuning methods reduce the cost of adapting a large pretrained model by training only a small fraction of the parameters while keeping the backbone frozen. The dominant family is low-rank adaptation: LoRA [19] parameterizes weight updates as $\Delta W = BA$ with $A \in \mathbb{R}^{r\times d}$, $B \in \mathbb{R}^{d\times r}$ for rank $r \ll d$, training only A and B while W stays frozen. DoRA [34] extends this by decomposing the pretrained weight into magnitude and direction components and applying the low-rank update only to the direction, improving expressiveness at no extra inference cost. Prefix tuning [32] prepends a small set of learnable continuous vectors to each transformer layer's keys and values, leaving every model weight frozen. Adapter-based methods insert small trainable bottleneck modules between transformer sub-layers [18], while $(\mathrm{IA})^3$ [33] multiplies activations by learned scaling vectors. RoCoFT [26] restricts updates to a small number of rows and columns of weight matrices, achieving competitive accuracy at a tiny parameter footprint.

These methods all reduce the number of trainable parameters, which directly cuts gradient and optimizer-state memory: the optimizer no longer needs to store moments for the frozen backbone, and gradient buffers shrink in proportion. They do not, however, reduce the storage required for intermediate activations during the forward pass. Backpropagation through any low-rank adapter still requires the upstream activations of every layer to compute the chain rule through the adapter weights, so the activation footprint, which scales as batch size $\times$ sequence length $\times$ hidden dimension $\times$ depth, remains unchanged. As recent work notes, activations dominate the GPU memory budget for billion-scale models on long-context tasks [51, 61], which is precisely the regime where PEFT alone fails to deliver the expected savings. Our work targets this remaining bottleneck through position-wise gradient masking, and importantly, the two reductions compose without interference: applying token masking on top of LoRA or DoRA preserves the parameter savings while adding activation savings, as we demonstrate empirically in Section 5.

F.2 Activation-Memory Reduction

A separate body of work targets activation memory directly, independent of which parameters are trained. The classical approach is gradient checkpointing [6], which stores only a strategically chosen subset of intermediate activations during the forward pass and recomputes the rest during backward, reducing memory from $O(L)$ to $O(\sqrt{L})$ for an L-layer network at the cost of one extra forward pass per checkpointed segment. This technique is now standard practice in large-scale training and is supported natively in major frameworks. Reversible networks take a different route: prior work reformulates residual blocks so that activations can be reconstructed exactly from outputs [13], and this idea has been extended to transformer architectures [23]. Reversibility eliminates the need to cache activations entirely, but constrains architectural choices and typically incurs a modest accuracy cost.

Activation compression and quantization form another line. Mesa [42] stores low-precision approximate activations during the forward pass and dequantizes them in the backward pass, trading a small accuracy gap for substantial memory savings. ZeRO [46] and FSDP partition optimizer

states, gradients, and parameters across devices, reducing per-device memory at the cost of inter-device communication. Mixed-precision training (FP16 or BF16) and 8-bit optimizer quantization, combined as in QLoRA [10], further compress weight and optimizer storage. More recent work targets long-sequence training specifically: StreamBP [37] performs exact backpropagation while iteratively processing mini-sequences, reducing the per-step activation cost without approximation; sequence-parallel methods partition the sequence dimension across GPUs at the cost of inter-device communication.

All of these approaches treat the activation tensor as a monolithic object to be compressed, recomputed, partitioned, or reconstructed. None of them reduce the number of positions through which gradient information flows: every token position remains in the backward computation graph, and the savings come from how that computation is executed rather than from which subset of positions participates. Our work targets the orthogonal axis: by backpropagating through only M of T positions, we eliminate the activation cost of the remaining $T - M$ positions entirely, irrespective of how each retained activation is stored. The two strategies compose: gradient checkpointing on the M retained positions further reduces the memory of those positions, while reversibility could in principle eliminate the cached activations even for the trainable subset. Our experiments use no checkpointing or reversibility, so the reported reductions are pure consequences of position selection, but the framework is compatible with both.

F.3 Token-Selective Fine-Tuning

The closest line of work to ours selects a subset of token positions to backpropagate through, materializing the activation savings we describe but without theoretical foundation. TokenTune [51] backpropagates through k randomly chosen input tokens (with the [CLS] token always included) and demonstrates that this approximate-gradient strategy preserves accuracy on BERT and Llama-2 across few-shot benchmarks. Composed with LoRA and QLoRA, it reaches roughly 21% of full fine-tuning's memory at comparable accuracy. The selection is uniformly random per epoch, with no notion of which positions are more or less informative.

TokenSeek [61] introduces an instance-aware criterion: each token receives a score $\alpha \log(I_1) + \beta \cdot \mathrm{Norm}(I_2)$ combining attention column-sums (a context-importance signal) and gradient magnitudes (a task-importance signal), and the top-$r\%$ of tokens are kept per sample. Default settings retain 10% of tokens. The approach achieves strong results—on Llama-3.2-1B with QLoRA at 14.8% memory, it reaches an average of 52.61 on the few-shot benchmark suite, exceeding full token tuning. However, the criterion requires a pre-stage in which the model performs a forward and partial backward pass over the entire training set to compute scores, with the resulting per-sample score dictionary stored on disk. The hyperparameters α, β are grid-searched per model: $(1, 0)$ for Qwen-0.5B and Llama-1B, $(5, 5)$ for Llama-3B. The method is causal-only, exploiting the attention-sink structure of decoder models.

LeMo [56] takes an engineering-first approach: a small predictor network forecasts which tokens to keep at each layer, with elastic predictor sizing and kernel-level optimizations (permutation-free token movement, segment-based gradient computation) yielding end-to-end memory reductions of up to $1.93\times$ on long-context fine-tuning. Random-LTD [59] applies layerwise random token dropping during pretraining rather than fine-tuning, demonstrating that aggressive token-level sparsity is feasible during the forward pass.

These methods establish empirically that fine-tuning through a small subset of token positions preserves accuracy, but they leave the underlying mechanism unexplained. None offers a theoretical account of *why* a small subset suffices, what determines the size of the sufficient subset, or how that size depends on architectural properties of the backbone. Data-driven scoring approaches [61] implicitly assume that token importance varies systematically by content; our analysis shows that in causal models the variation is dominated by *position* rather than content, with later positions strictly dominating earlier ones under a contraction condition on hidden states (Theorem 3). This positional structure explains why gradient-based scoring concentrates on the response section of Alpaca-formatted training data, which is exactly the suffix of the sequence and exactly what our last-M rule selects without any per-sample computation. We therefore expect to match the accuracy of these methods at zero pre-stage cost and without per-model hyperparameter search.

A complementary thread on token pruning has emerged in the inference-time literature. LazyLLM [12] dynamically prunes prompt tokens during the prefill stage to accelerate first-token generation, achieving $2.34\times$ speedup on Llama-2-7B without fine-tuning. SlimInfer [36] extends pruning to intermediate hidden states under an information-diffusion observation, and InfiniteHiP [29] uses hierarchical pruning to extend the effective context to three million tokens on a single GPU. These methods address a different problem (inference compute and KV-cache memory rather than training activation memory) and operate without weight updates, but they corroborate the broader empirical finding that transformer computation is highly sparse in token utilization. Our work transports this insight to the training regime, where the relevant cost is activation storage rather than KV-cache size, and grounds it in NTK theory rather than attention-pattern heuristics.

F.4 Neural Tangent Kernel Theory of Fine-Tuning

The Neural Tangent Kernel framework [21] characterizes neural-network training in the infinite-width limit as kernel regression with the kernel induced by the network's Jacobians at initialization. Originally derived for randomly initialized networks, the framework has since been adapted to fine-tuning of pretrained models. Prior work [38] extends NTK analysis to the fine-tuning regime, demonstrating that prompt-based fine-tuning of large language models can exhibit kernel behavior under specific conditions on prompts and training scale, and proves that LoRA can be analyzed through a kernel-difference bound relative to full fine-tuning. A more recent line [1] introduces an explicit ℓ_2 inductive bias on parameters to ensure linearization is a faithful model of fine-tuning dynamics, and uses the eigenvalue spectrum of the resulting NTK to predict downstream performance and guide layer selection. Other work [54] applies NTK analysis to two-stage linear-probe-then-fine-tune training, decomposing the kernel matrix into pretrained-effective and fine-tuning-effective components.

These works address *which parameters* or *which layers* to fine-tune, treating the input sequence as a monolithic object whose tokens contribute uniformly to the kernel. Our analysis introduces a complementary axis: the kernel induced by training through a single *token position*, formalized as the Gram matrix K_t of position-t Jacobians. We show that this position-wise kernel is itself a low-dimensional bottleneck (Theorem 1), that positions with aligned hidden-state geometry yield equivalent kernels in bidirectional models (Theorem 2), and that position-wise kernels form a monotone chain in causal models under a contraction condition (Theorem 3). The output-head NTK regime in which our results are stated is closely aligned with the kernel formalism of prior fine-tuning NTK work [38], and the empirical NTK-laziness validation we provide on LLaMA-3-8B (Appendix E.4) is in the spirit of recent linearization measurements [1]. The novelty of our contribution is the position-wise refinement: existing NTK theories of fine-tuning are blind to which positions of the sequence drive learning, whereas we show that this question has both a clean theoretical answer (an architecture-dependent optimal position count M^*) and a directly actionable consequence (a memory-efficient training procedure).

F.5 Synthesis

The four lines of related work can be summarized along a single axis: each reduces fine-tuning cost by acting on a different dimension of the computation. PEFT methods reduce the parameter dimension. Activation-memory techniques (checkpointing, reversibility, compression) reduce per-position storage cost. Token-selective methods reduce the number of positions in the backward graph but rely on heuristics or auxiliary computation to choose which positions. NTK theories of fine-tuning explain training dynamics but treat positions uniformly. Our work occupies the intersection that none of the four cover: a theoretical account of position-wise capacity that yields, as a consequence, a principled criterion for how many and which positions to backpropagate through, with the resulting memory savings composing cleanly with PEFT methods on the parameter side and remaining orthogonal to activation-side techniques on the storage side. The empirical headline—fifty to one hundred trainable positions match or exceed full-sequence fine-tuning across six task families and five backbones, with up to ninety-seven percent activation-memory reduction—is the practical corollary of the theory rather than the result of an engineering search.

G Datasets

We evaluate on six task modalities spanning causal and bidirectional architectures. This section describes each dataset used in the paper.

G.1 Commonsense Reasoning

We use six benchmarks to evaluate general commonsense and factual reasoning abilities in pretrained language models by following work and setups [20].

- **ARC-Challenge (ARC-C)** [8]: a multiple-choice science QA dataset of grade-school questions that are hard for retrieval-based methods, requiring genuine reasoning. Each sample contains a question and four labeled options.
- **ARC-Easy (ARC-E)** [8]: the easier subset of the ARC benchmark, consisting of elementary-level science questions in the same four-option format.
- **BoolQ** [7]: a yes/no question answering dataset where each question is paired with a short passage, and the model must answer `true` or `false`.
- **HellaSwag** [60]: a commonsense sentence-completion benchmark where the model selects the most plausible continuation of a short context from four adversarially-generated candidates.
- **OpenBookQA (OBQA)** [41]: a multiple-choice elementary-science dataset designed to evaluate reasoning that combines a provided "open book" of core science facts with broader commonsense.
- **WinoGrande** [50]: a large-scale Winograd-style pronoun resolution dataset requiring commonsense understanding of events, objects, and everyday situations.

G.2 Mathematical Reasoning

We use six arithmetic and word-problem benchmarks, following the standard setup in prior PEFT literature by following work and setups [20].

- **AddSub** [17]: a collection of simple arithmetic word problems involving addition and subtraction over small integers.
- **MultiArith** [49]: multi-step arithmetic word problems that require combining several operations to reach the final answer.
- **SVAMP** [43]: a challenge benchmark constructed to reveal over-reliance on surface heuristics in word-problem solving, built from targeted perturbations of existing problems.
- **SingleEq** [24]: algebra word problems solvable by a single equation, covering linear and multi-operand cases.
- **GSM8K** [9]: grade-school math word problems that require multi-step natural-language reasoning with calculator-level arithmetic.
- **MAWPS** [25]: a unified math word-problem suite that aggregates several earlier arithmetic benchmarks into a common format.

G.3 Visual Question Answering

We evaluate five visual QA benchmarks spanning medical, scene-text, chain-of-thought, K–12, and safety-critical settings.

- **VQA-RAD** [28]: a radiology visual QA dataset containing clinician-generated questions paired with radiology images (CT, MRI, X-ray). Each question is labeled as closed-ended (yes/no or a small closed set) or open-ended.
- **STVQA** [4] : scene-text visual question answering, where answers depend on reading and interpreting text embedded in natural images.
- **M3CoT** [5] : a multi-modal chain-of-thought reasoning benchmark requiring intermediate reasoning steps grounded in both image and text.

- **MMK12** [39]: a K–12 multi-modal reasoning dataset covering science, math, and general knowledge with images. Since only a training split is released, we partition it $80/20$ into train/test with seed 42.
- **SafeTag-VL-3K** [48]: an image-grounded safety benchmark evaluating whether a vision-language model can identify safety-critical content in images. We filter out samples with empty image fields.

G.4 Video Question Answering

We evaluate six video-reasoning benchmarks spanning physics, interaction, temporal, and mathematical domains.

- **Human-Object Interactions (HOI)** [27]: a subset of `facebook/minimal_video_pairs` focused on identifying the correct description of how humans manipulate objects in short video clips.
- **Intuitive Physics** [27]: a subset of `facebook/minimal_video_pairs` testing intuitive physical reasoning about objects, forces, and motion.
- **Robot-Object Interactions** [27]: a subset of `facebook/minimal_video_pairs` evaluating understanding of robotic manipulation in controlled scenes.
- **Temporal Reasoning** [27]: a subset of `facebook/minimal_video_pairs` requiring the model to identify the temporal order or causal structure of events in a video.
- **VideoMathQA** [47] (`MBZUAI/VideoMathQA`, `multi_binary` subset): a mathematical-reasoning benchmark grounded in instructional videos, where the model must answer multi-step math questions about content demonstrated on video. Only a test split is released; we partition it $80/20$ with seed 42.
- **Video-LevelGauge** [57]: a long-video multiple-choice benchmark evaluating comprehension and reasoning over extended temporal contexts.

For all video datasets we decode frames with `decord` at 1 FPS, capped at 8 frames per video, and filter out samples whose videos are missing or cannot be decoded.

GLUE Benchmark. We use eight tasks from the General Language Understanding Evaluation (GLUE) benchmark [55] to evaluate natural-language understanding in bidirectional models.

- **SST-2**: binary sentiment classification of movie-review sentences.
- **QNLI**: question-answering natural language inference derived from SQuAD, recast as a sentence-pair entailment task.
- **MRPC**: paraphrase identification between pairs of sentences taken from news sources, evaluated with accuracy and F1.
- **QQP**: a large-scale paraphrase-identification dataset of Quora question pairs, evaluated with accuracy and F1.
- **CoLA**: grammatical-acceptability classification, evaluated with Matthews correlation coefficient.
- **MNLI**: multi-genre natural language inference, a three-way entailment task across ten genres of written and spoken English.
- **RTE**: recognizing textual entailment, a binary entailment task compiled from earlier textual-entailment challenges.
- **STS-B**: sentence-pair semantic similarity scored on a 0–5 scale, evaluated with Pearson and Spearman correlation.

G.5 Image Classification

We evaluate on five image-classification benchmarks at 224×224 resolution.

- **Tomato Leaves** [52] (`Vlone571/tomato-leaves-dataset`): an 8-class plant-disease dataset of tomato-leaf images, covering healthy leaves and seven common diseases.

- **SUN397** [58]: a large-scale scene-classification benchmark with 397 scene categories covering indoor, outdoor, and urban environments.
- **RVL-CDIP** [15]: a 16-class document-image classification dataset spanning letters, forms, emails, invoices, and other common document types.
- **EuroSAT** [16]: a 10-class satellite-image classification benchmark based on Sentinel-2 imagery, covering land-use and land-cover categories.
- **ChestX-Ray (Pneumonia)** [22] : a binary medical-imaging dataset classifying chest X-rays as normal or pneumonia.

H Hyperparameters and Training Configuration

This section documents the hyperparameters used across all experiments in the paper. All training runs use a single set of shared defaults for optimization, mixed precision, and scheduling; dataset- and architecture-specific deviations are noted below. Every run uses the random seed 42 for dataset splitting and trainer initialization.

Shared optimization settings. All experiments use the AdamW optimizer with a cosine learning-rate schedule, 100 linear warmup steps, bfloat16 mixed precision, and no gradient clipping (except image classification with ViT, which uses the default 1.0). Training runs for 1 epoch unless otherwise specified. We disable checkpoint saving during sweeps (`save_strategy = "no"`) to avoid disk overhead. Evaluation uses greedy decoding for all generative tasks.

Full fine-tuning. Full fine-tuning updates all trainable parameters with learning rate 2×10^{-5} and weight decay 0. We freeze all `Conv3d` layers in vision-language models before training to avoid corrupting the pretrained vision patch embeddings.

LoRA. LoRA [19] adapters use rank $r = 4$, scaling factor $\alpha = 5$, and dropout 0.05, with learning rate 1×10^{-4} and weight decay 0. We apply LoRA only to the attention projection matrices (`q_proj`, `k_proj`, `v_proj`, `o_proj`) of the language-model backbone (`model.language_model`); the vision encoder remains frozen throughout training. After training, LoRA updates are merged into the base weights, restoring the original architecture with no inference overhead.

DoRA. DoRA [34] uses the same rank, scaling, dropout, and learning rate as LoRA, and is applied to the same attention projections. The only difference is the weight-decomposition step introduced by DoRA itself.

Table 9 summarizes the per-modality training settings; Tables 10 report sequence-length and generation settings.

Table 9: Per-modality training hyperparameters. "bs" is per-device batch size; "accum" is gradient-accumulation steps; "eff. bs" is the effective batch size (bs × accum). Multi-GPU runs shard the model across devices (`device_map="auto"`), so the GPU count does not change the effective batch size. GPUs are NVIDIA H100 80GB.

Modality	Backbone	Regime	LR	bs	accum	eff. bs	Epochs	GPUs
Commonsense reasoning	LLaMA-3-8B	Full	2e−5	4	4	16	1	2
	LLaMA-3-8B	DoRA	1e−4	4	4	16	1	1
Math reasoning	LLaMA-3-8B	Full	2e−5	4	4	16	1	2
	LLaMA-3-8B	DoRA	1e−4	4	4	16	1	1
Visual QA (image)	Qwen3-VL-8B	Full	2e−5	3	4	12	1	4
	Qwen3-VL-8B	LoRA	1e−4	2	4	8	1	2
	LLaVA-OneVision-7B	Full	1e−4	3	4	12	1	2
	LLaVA-OneVision-7B	LoRA	1e−4	3	4	12	1	1
Video QA (minimal pairs)	Qwen3-VL-8B	Full	2e−5	8	4	32	1	2
	Qwen3-VL-8B	LoRA	1e−4	8	4	32	1	1
Video QA (VideoMathQA)	Qwen3-VL-8B	Full	2e−5	1	4	4	1	2
	Qwen3-VL-8B	LoRA	1e−4	1	4	4	1	1
Video QA (LevelGauge)	Qwen3-VL-8B	Full	2e−5	2	4	8	1	2
	Qwen3-VL-8B	LoRA	1e−4	2	4	8	1	2
GLUE	RoBERTa-base	Full	2e−5	32	1	32	3	1
Image classification	ViT-L/16-224	Full	1e−4	32	2	64	1	1

Table 10: Sequence lengths and generation settings per dataset.

Dataset	Input max length	Frames / FPS	Max new tokens
Commonsense (all)	1000	–	16
Math (all)	1000	–	1000
VQA-RAD	4048	–	64
STVQA	4048	–	512
M3CoT	4048	–	512
MMK12	8096	–	512
SafeTag-VL-3K	4024	–	16
HOI / IntuPhys / RobotOI / TempReas	2048	8 frames at 1 FPS	8
VideoMathQA	4096	8 frames at 1 FPS	512
Video-LevelGauge	4096	8 frames at 1 FPS	64
GLUE (all)	128	–	–
Image classification	224×224	–	–

LoRA / DoRA target specification. For all vision-language models (Qwen3-VL-8B and LLaVA-OneVision-7B), adapters target only the language-model attention projections:

- `target_modules`: `[q_proj, k_proj, v_proj, o_proj]`
- `name_prefix`: `model.language_model`

For ViT (bidirectional), full fine-tuning targets the encoder attention and MLP projections:

- `target_modules`: `[query, key, value, dense]`
- `name_prefix`: `vit.encoder`

For LLaMA-3-8B (full fine-tuning and DoRA), all linear layers in every transformer block are trainable (`target_modules = "all"`), with frozen embedding layers.

Sweep over trainable token counts. For each experiment, we sweep the number of trainable positions M over the following values plus a full-sequence baseline:

- Text (commonsense, math): $M \in \{10, 25, 50, 100, 300, 500, 700, 1000\}$.
- Visual QA: $M \in \{10, 25, 50, 100, 200, 300, 500, 700, 1000, 1500\}$.
- Video QA: $M \in \{10, 25, 50, 100, 300, 500, 700, 1000, 1500\}$.
- GLUE: $M \in \{5, 10, 25, 50, 75, 100\}$.
- Image classification: $M \in \{10, 25, 50, 100, 150, 180\}$.

The full-sequence baseline (labeled "All" in every table) corresponds to standard training with no position masking: all label positions contribute to the loss.

Memory and timing measurement. Peak GPU memory is measured with `torch.cuda.max_memory_allocated`, after resetting the peak-memory statistics immediately before `trainer.train()` is invoked. The reported value therefore captures the complete training footprint, comprising the resident model parameters together with the gradients, optimizer states, and activations allocated during training, while excluding transient allocations incurred during model loading. For multi-GPU configurations, we report the sum of the peak allocations across all devices; reported values for sharded models thus correspond to total rather than per-device memory. Training time is the wall-clock duration of the training loop and excludes model loading and evaluation.

Hardware. With the exception of the long-context and gradient-checkpointing experiments, all experiments are conducted on NVIDIA H100 GPUs with 80 GB of memory. Multi-GPU configurations

employ the HuggingFace `device_map="auto"` option to shard the model across devices. Because this scheme is model-parallel rather than data-parallel, the number of GPUs does not affect the effective batch size. A single GPU is used for DoRA fine-tuning of LLaMA-3-8B, LoRA fine-tuning of LLaVA-OneVision-7B, LoRA fine-tuning on the video benchmarks other than Video-LevelGauge, the GLUE experiments with RoBERTa-base, and image classification with ViT-L/16-224. Two GPUs are used for full fine-tuning of LLaMA-3-8B and LLaVA-OneVision-7B, LoRA fine-tuning of Qwen3-VL-8B on image QA, full fine-tuning of Qwen3-VL-8B on all video benchmarks, and LoRA fine-tuning on Video-LevelGauge. Four GPUs are used for full fine-tuning of Qwen3-VL-8B on image QA. Table 9 summarizes the GPU configuration for each setting. The long-context and gradient-checkpointing experiments on LongAlpaca-12k (Appendices J.7 and J.8) are conducted on eight NVIDIA A100 GPUs with 80 GB of memory each.

Precision and inference. All models are loaded and trained in bfloat16. Training uses automatic mixed precision via the HuggingFace `Trainer` with `bf16=True`. After training, adapter weights (for LoRA and DoRA) are merged back into the base parameters via the `merge` operation, so the inference-time architecture is identical to the pretrained model—no additional adapter modules remain at test time and no latency overhead is introduced.

Parameter freezing for vision-language models. We emphasize that for all vision-language experiments (Qwen3-VL-8B and LLaVA-OneVision-7B), the vision patch embeddings (`Conv3d` for Qwen3-VL, `Conv2d` for LLaVA) are frozen before training to avoid perturbing the pretrained patch-tokenization. Adapter updates target exclusively the language-model attention projections; this isolation ensures that any observed accuracy changes under token masking are attributable to the language-side training signal, not to shifts in visual representation.

I Models

We evaluate the Token Hypothesis across five backbone architectures spanning text-only causal language models, multimodal vision-language models, bidirectional encoders, and vision transformers. This section describes each model's architecture, pretraining, and the components that are modified or frozen during fine-tuning.

LLaMA-3-8B (causal). LLaMA-3-8B [14] is an 8-billion-parameter causal transformer trained on approximately 15 trillion tokens of publicly available text. The architecture consists of 32 transformer blocks, each with grouped-query attention (32 query heads and 8 key-value heads), SwiGLU activations, rotary positional embeddings, and RMSNorm. The hidden dimension is 4096 and the feed-forward dimension is 14,336. The vocabulary has 128,256 tokens with a byte-pair encoded tokenizer. We use the base (non-instruction-tuned) checkpoint for all commonsense and mathematical reasoning experiments and load it in bfloat16 precision. Under the token-masking protocol, the causal attention ensures that each hidden state at position t depends only on the prefix $(x_1, \ldots, x_t)$, which is the setting assumed by Theorem 3.

Qwen3-VL-8B-Instruct (causal, vision-language). Qwen3-VL-8B-Instruct [3] is an instruction-tuned multimodal model that extends the Qwen3 language backbone with a native vision encoder. The language model is an 8-billion-parameter causal transformer with grouped-query attention and rotary embeddings, and the vision encoder is a dynamic-resolution vision transformer that tokenizes images and videos into variable-length patch sequences. Visual tokens are projected into the language-model embedding space through an adaptor and interleaved with text tokens in the unified sequence. For video inputs, temporal patches are produced by a `Conv3d` patch embedding. In all our Qwen3-VL experiments, we freeze every `Conv3d` layer in the vision encoder before training and apply LoRA only to the language-model attention projections under `model.language_model`, following the orthogonality principle of the method: parameter-efficient fine-tuning targets the language pathway, while the vision encoder remains untouched.

LLaVA-OneVision-7B (causal, vision-language). LLaVA-OneVision-7B [30] is a vision-language model that couples a Qwen2-7B causal language model with a SigLIP vision encoder. Images are processed at 384×384 resolution and projected into the language-model token space via a two-layer MLP adaptor. For all LLaVA experiments we fix `image_grid_pinpoints = [[384, 384]]` to disable dynamic tiling, and we apply LoRA/DoRA to the same language-model attention projections as for Qwen3-VL. The `Conv2d` patch embedding in the vision tower is frozen in the DoRA setting; for LoRA and full fine-tuning the vision encoder is trainable but empirically moves negligibly (consistent with the NTK-laziness analysis in Appendix E.4).

RoBERTa-base (bidirectional). RoBERTa-base [35] is a 125-million-parameter bidirectional encoder pretrained on approximately 160 GB of English text with the masked-language-modeling objective. The architecture consists of 12 transformer blocks with multi-head self-attention (12 heads), GELU activations, learned absolute positional embeddings, and LayerNorm. Hidden dimension is 768, feed-forward dimension is 3072, and the tokenizer vocabulary has 50,265 byte-pair tokens. Since every position attends to the full sequence, RoBERTa is the canonical bidirectional setting for Theorem 2. We use RoBERTa-base for all GLUE experiments and for the empirical validation of Theorems 1 and 2 (Section 5 and Appendices A.2 and B.2), and for the image-saturation analysis of Corollary 1 (Figure 2).

SmolLM2-135M (causal). SmolLM2-135M [2] is a 135-million-parameter causal language model pretrained on curated web data. We use it only for the multi-scale validation of Theorem 1 (Appendix A.2), where it provides a mid-size causal architecture between RoBERTa-base (125M) and LLaMA-3-8B (8B), allowing us to verify that the effective-rank bottleneck $d_{\text{eff}}(K_t) < 6$ holds across a $64\times$ range in parameter count. SmolLM2-135M is not used for any downstream fine-tuning experiment.

ViT-L/16-224 (bidirectional). ViT-L/16-224 [11] is a 307-million-parameter vision transformer pretrained on ImageNet-21k at 224×224 resolution with 16×16 patches, yielding a sequence of 196 visual tokens plus a `[CLS]` token. The architecture consists of 24 encoder blocks with 16 attention

heads, hidden dimension 1024, and feed-forward dimension 4096. Like RoBERTa, self-attention is fully bidirectional over the token sequence, so all image patch positions share information access. For each classification task we replace the pretrained classification head with a randomly initialized linear layer of the appropriate output dimension, ignoring mismatched sizes in the state dict. Fine-tuning targets the attention projections and the feed-forward `dense` layers across all encoder blocks.

Table 11: Full fine-tuning accuracy, total GPU memory (GB), and training time (min) across five commonsense reasoning benchmarks vs. number of trainable tokens. Subscript deltas are absolute deviations from *All*: ↑ green = improvement, ↓ red = degradation. **Bold** = best value per column.

Tokens	ARC-C			ARC-E			BoolQ			HSwag			OBQA			Avg		
	Acc	*GPU*	*Time*	*Acc*	*GPU*	*Time*	*Acc*	*GPU*	*Time*	*Acc*	*GPU*	*Time*	*Acc*	*GPU*	*Time*	*Acc*	*GPU*	*Time*
All	0.728	94.68	2.32	0.863	94.68	4.66	0.738	94.68	19.53	0.951	97.67	82.44	0.874	94.68	10.32	0.831	95.28	23.85
10	0.662 ↓0.066	**83.42** ↑11.26	**1.89** ↑0.43	0.811 ↓0.052	**83.42** ↑11.26	**3.79** ↑0.87	0.700 ↓0.038	**83.42** ↑11.26	**15.81** ↑3.72	0.835 ↓0.116	**84.77** ↑12.90	**66.95** ↑15.49	0.712 ↓0.162	**83.42** ↑11.26	**8.29** ↑2.03	0.744 ↓0.087	**83.69** ↑11.59	**19.35** ↑4.51
25	0.719 ↓0.008	83.65 ↑11.03	**1.89** ↑0.43	0.853 ↓0.010	83.65 ↑11.03	3.80 ↑0.86	0.732 ↓0.006	83.65 ↑11.03	15.89 ↑3.64	0.903 ↓0.048	84.94 ↑12.73	67.34 ↑15.10	0.796 ↓0.078	83.65 ↑11.03	8.34 ↑1.98	0.801 ↓0.030	83.91 ↑11.37	19.45 ↑4.40
50	0.761 ↑0.033	84.07 ↑10.61	1.93 ↑0.39	**0.879** ↑0.016	84.07 ↑10.61	3.87 ↑0.79	0.730 ↓0.008	84.07 ↑10.61	16.18 ↑3.35	0.952 ↑0.001	86.89 ↑10.78	68.56 ↑13.88	0.834 ↓0.040	84.07 ↑10.61	8.49 ↑1.83	0.831 ↑0.001	84.63 ↑10.65	19.81 ↑4.05
100	**0.764** ↑0.036	84.84 ↑9.85	1.99 ↑0.33	0.872 ↑0.009	84.84 ↑9.85	4.00 ↑0.66	**0.745** ↑0.007	84.84 ↑9.85	16.78 ↑2.75	0.941 ↓0.010	89.90 ↑7.78	70.87 ↑11.57	0.848 ↓0.026	84.84 ↑9.85	8.79 ↑1.53	0.834 ↑0.003	85.85 ↑9.43	20.49 ↑3.37
300	0.761 ↑0.033	87.85 ↑6.83	2.14 ↑0.18	0.852 ↓0.011	87.85 ↑6.83	4.30 ↑0.36	0.744 ↑0.007	87.85 ↑6.83	18.01 ↑1.52	0.951 ↑0.000	92.96 ↑4.71	76.32 ↑6.12	**0.883** ↑0.009	87.85 ↑6.83	9.44 ↑0.88	**0.838** ↑0.008	88.87 ↑6.40	22.04 ↑1.81
500	0.733 ↑0.005	90.96 ↑3.72	2.32 ↑0.00	0.874 ↑0.011	90.96 ↑3.72	4.65 ↑0.01	0.734 ↓0.004	90.96 ↑3.72	19.42 ↑0.11	**0.959** ↑0.008	93.87 ↑3.80	82.13 ↑0.31	0.852 ↓0.022	90.96 ↑3.72	10.20 ↑0.12	0.830 ↑0.000	91.54 ↑3.74	23.74 ↑0.11
700	0.730 ↑0.002	93.97 ↑0.71	2.57 ↓0.25	0.873 ↑0.010	93.97 ↑0.71	4.73 ↓0.07	0.732 ↓0.006	93.97 ↑0.71	21.49 ↓1.96	0.949 ↓0.002	95.20 ↑2.47	87.09 ↓4.65	0.872 ↓0.002	93.97 ↑0.71	11.28 ↓0.96	0.831 ↑0.000	94.21 ↑1.07	25.43 ↓1.58
1000	0.754 ↑0.026	97.01 ↓2.33	2.85 ↓0.53	0.867 ↑0.004	95.01 ↓0.33	4.96 ↓0.30	0.739 ↑0.001	95.01 ↓0.33	23.95 ↓4.42	0.949 ↓0.002	99.06 ↓1.39	91.49 ↓9.05	0.852 ↓0.022	97.01 ↓2.33	12.57 ↓2.25	0.832 ↑0.002	96.62 ↓1.34	27.16 ↓3.31

Table 12: Full fientuning Accuracy, total GPU memory (GB), and training time (min) across five visual QA reasoning benchmarks (LLaVA-OneVision-7B) vs. number of trainable tokens. Subscript deltas are absolute deviations from *All*: ↑ green = improvement, ↓ red = degradation. **Bold** = best value per column.

Tokens	M3CoT			MMK12			SafeTag			STVQA			VQA-R			Avg		
	Acc	*GPU*	*Time*	*Acc*	*GPU*	*Time*	*Acc*	*GPU*	*Time*	*Acc*	*GPU*	*Time*	*Acc*	*GPU*	*Time*	*Acc*	*GPU*	*Time*
All	0.753	97.08	38.94	0.400	96.42	7.15	0.894	97.09	11.29	0.844	97.08	29.11	0.577	89.02	7.30	0.693	95.34	18.76
10	0.505 ↓0.249	**85.42** ↑11.66	31.45 ↑7.49	0.383 ↓0.018	**84.94** ↑11.48	5.75 ↑1.40	0.892 ↓0.001	**85.41** ↑11.68	9.33 ↑1.96	0.737 ↓0.106	**85.40** ↑11.67	24.02 ↑5.09	0.573 ↓0.003	**79.76** ↑9.26	**6.26** ↑1.04	0.618 ↓0.075	**84.19** ↑11.15	**15.36** ↑3.40
25	0.502 ↓0.251	85.55 ↑11.54	32.85 ↑6.09	0.394 ↓0.007	85.07 ↑11.35	5.64 ↑1.51	0.909 ↑0.015	85.54 ↑11.55	9.22 ↑2.07	0.821 ↓0.023	85.54 ↑11.53	**23.98** ↑5.13	0.574 ↓0.002	79.90 ↑9.12	6.31 ↑0.99	0.640 ↓0.054	84.32 ↑11.02	15.60 ↑3.16
50	0.486 ↓0.267	85.81 ↑11.27	**30.65** ↑8.29	0.372 ↓0.028	85.31 ↑11.11	**5.55** ↑1.60	0.907 ↑0.014	85.80 ↑11.29	**9.21** ↑2.08	0.851 ↑0.007	85.79 ↑11.28	26.71 ↑2.40	**0.588** ↑0.011	80.11 ↑8.91	6.30 ↑1.00	0.641 ↓0.053	84.56 ↑10.77	15.68 ↑3.07
100	0.539 ↓0.215	86.28 ↑10.80	33.93 ↑5.01	0.405 ↑0.005	85.82 ↑10.61	5.64 ↑1.51	0.916 ↑0.023	86.27 ↑10.81	9.26 ↑2.03	**0.896** ↑0.052	86.28 ↑10.80	24.82 ↑4.29	0.587 ↑0.010	80.63 ↑8.39	6.34 ↑0.96	0.669 ↓0.025	85.06 ↑10.28	16.00 ↑2.76
200	0.595 ↓0.158	87.21 ↑9.88	31.59 ↑7.35	**0.422** ↑0.022	86.71 ↑9.72	5.78 ↑1.37	0.906 ↑0.012	87.17 ↑9.92	9.86 ↑1.43	0.893 ↑0.049	87.18 ↑9.90	26.51 ↑2.60	0.587 ↑0.010	81.51 ↑7.51	6.58 ↑0.72	0.681 ↓0.013	85.96 ↑9.38	16.06 ↑2.69
300	0.621 ↓0.132	88.04 ↑9.04	35.14 ↑3.80	0.415 ↑0.015	87.55 ↑8.87	5.92 ↑1.23	0.904 ↑0.011	88.01 ↑9.08	10.08 ↑1.21	0.869 ↑0.025	88.02 ↑9.05	25.70 ↑3.41	0.581 ↑0.004	82.37 ↑6.65	6.74 ↑0.56	0.678 ↓0.015	86.80 ↑8.54	16.72 ↑2.04
500	0.727 ↓0.026	89.86 ↑7.22	33.96 ↑4.98	0.390 ↓0.010	89.35 ↑7.08	6.17 ↑0.98	**0.918** ↑0.024	89.84 ↑7.25	10.22 ↑1.07	0.841 ↓0.003	89.85 ↑7.22	27.61 ↑1.50	0.585 ↑0.009	84.19 ↑4.84	6.97 ↑0.33	0.692 ↓0.001	88.62 ↑6.72	16.99 ↑1.77
700	0.742 ↓0.012	91.65 ↑5.43	37.53 ↑1.41	0.407 ↑0.007	91.19 ↑5.24	6.52 ↑0.63	0.910 ↑0.017	91.67 ↑5.42	10.74 ↑0.55	0.854 ↑0.010	91.65 ↑5.43	27.93 ↑1.18	0.585 ↑0.009	86.01 ↑3.01	7.19 ↑0.11	0.700 ↑0.006	90.43 ↑4.91	17.98 ↑0.78
1000	**0.768** ↑0.015	93.36 ↑3.72	37.32 ↑1.62	0.420 ↑0.020	92.86 ↑3.56	6.93 ↑0.22	0.909 ↑0.015	93.38 ↑3.71	11.12 ↑0.17	0.830 ↓0.014	93.37 ↑3.70	28.39 ↑0.72	0.581 ↑0.004	87.67 ↑1.35	7.79 ↓0.49	**0.702** ↑0.008	92.13 ↑3.21	18.31 ↑0.45
1500	0.762 ↑0.009	96.57 ↑0.51	38.03 ↑0.91	0.403 ↑0.003	95.12 ↑1.30	7.82 ↓0.67	0.900 ↑0.006	96.59 ↑0.50	11.41 ↓0.12	0.843 ↓0.001	96.56 ↑0.51	32.44 ↓3.33	0.578 ↑0.001	87.67 ↑1.35	7.99 ↓0.69	0.697 ↑0.004	94.50 ↑0.83	19.54 ↓0.78

J Detailed Experimental Results

This section provides the full per-benchmark results used to produce the aggregate numbers in Section 5. Each table sweeps the number of trainable positions M across a wide range, with the full-sequence baseline (*All*) provided as the reference against which all other rows are compared. Subscript deltas on every cell denote the absolute deviation from *All*, color-coded in green when the token-masked run improves on the baseline and red when it degrades. Memory and time columns are reported so that accuracy and resource cost can be read jointly.

J.1 Commonsense Reasoning (LLaMA-3-8B)

Table 11 reports per-benchmark accuracy, peak GPU memory, and training time for full fine-tuning on five commonsense reasoning benchmarks. The full-sequence baseline averages 0.831 accuracy at 95.28 GB and 23.85 min. Token masking with $M = 50$ already matches this average (0.831) at 84.63 GB and 19.81 min, a 1.12 GB reduction in memory and 17% speedup. At $M = 100$ the average accuracy rises to 0.834 and at $M = 300$ it peaks at 0.838, recovering or exceeding the baseline on every benchmark while still saving roughly 6.4 GB. Memory and time grow monotonically with

Table 13: Accuracy, total GPU memory (GB), and training time (min) across five visual QA reasoning benchmarks (Full fine-tuning, Qwen3-VL-8B) vs. number of trainable tokens. Subscript deltas are absolute deviations from *All*: ↑ green = improvement, ↓ red = degradation. **Bold** = best value per column. GPU is the sum of all GPU peak allocations.

Tokens	M3CoT			MMK12			SafeTag			STVQA			VQA-R			Avg		
	Acc	*GPU*	*Time*	*Acc*	*GPU*	*Time*	*Acc*	*GPU*	*Time*	*Acc*	*GPU*	*Time*	*Acc*	*GPU*	*Time*	*Acc*	*GPU*	*Time*
All	0.782	207.54	34.18	0.568	188.12	4.13	0.918	102.76	5.94	0.852	148.69	13.69	0.519	193.25	8.52	0.728	168.07	13.29
10	0.769 ↓0.012	**161.61** ↑45.93	29.74 ↑4.44	0.548 ↓0.019	**148.10** ↑40.03	**3.70** ↑0.43	0.895 ↓0.023	**90.34** ↑12.42	**5.16** ↑0.78	0.814 ↓0.038	**121.51** ↑27.18	13.13 ↑0.56	0.528 ↑0.009	**154.88** ↑38.37	**6.72** ↑1.80	0.711 ↓0.017	**135.29** ↑32.78	11.69 ↑1.60
25	0.723 ↓0.058	161.89 ↑45.65	**28.70** ↑5.48	0.561 ↓0.006	148.33 ↑39.79	3.71 ↑0.42	0.909 ↓0.009	90.64 ↑12.12	5.18 ↑0.76	0.813 ↓0.039	121.84 ↑26.85	12.20 ↑1.49	0.550 ↑0.031	155.21 ↑38.04	6.73 ↑1.79	0.711 ↓0.016	135.58 ↑32.49	**11.30** ↑1.99
50	0.725 ↓0.057	162.34 ↑45.20	29.37 ↑4.81	0.555 ↓0.013	148.77 ↑39.35	3.73 ↑0.40	0.918 ↓0.000	91.14 ↑11.62	5.20 ↑0.74	0.857 ↑0.005	122.37 ↑26.31	**12.16** ↑1.53	0.545 ↑0.027	155.67 ↑37.58	6.75 ↑1.77	0.720 ↓0.008	136.06 ↑32.01	11.44 ↑1.85
100	0.752 ↓0.030	163.08 ↑44.46	29.42 ↑4.76	0.574 ↑0.007	149.48 ↑38.64	3.81 ↑0.32	0.910 ↓0.008	91.81 ↑10.95	5.36 ↑0.58	0.887 ↑0.036	123.41 ↑25.28	12.53 ↑1.16	**0.554** ↑0.035	156.34 ↑36.91	6.89 ↑1.63	0.736 ↑0.008	136.82 ↑31.25	11.60 ↑1.69
200	0.775 ↓0.007	164.61 ↑42.93	30.54 ↑3.64	**0.588** ↑0.020	151.01 ↑37.11	3.96 ↑0.17	0.921 ↑0.003	93.39 ↑9.37	5.46 ↑0.48	**0.904** ↑0.052	125.46 ↑23.23	14.64 ↓0.95	0.530 ↑0.011	157.91 ↑35.34	7.08 ↑1.44	**0.743** ↑0.016	138.48 ↑29.59	12.34 ↑0.96
300	0.783 ↑0.001	166.14 ↑41.39	31.35 ↑2.83	0.573 ↑0.005	152.61 ↑35.51	4.00 ↑0.13	**0.929** ↑0.011	94.99 ↑7.77	5.75 ↑0.19	0.890 ↑0.038	127.50 ↑21.19	14.12 ↓0.43	0.519 ↓0.000	159.41 ↑33.84	7.20 ↑1.32	0.739 ↑0.011	140.13 ↑27.94	12.48 ↑0.81
500	0.771 ↓0.010	169.22 ↑38.32	34.12 ↑0.06	0.575 ↑0.007	155.70 ↑32.42	4.00 ↑0.13	0.919 ↑0.002	98.12 ↑4.64	5.89 ↑0.05	0.881 ↑0.029	131.69 ↑17.00	14.19 ↓0.50	0.530 ↑0.011	162.53 ↑30.72	7.78 ↑0.74	0.735 ↑0.008	143.45 ↑24.62	13.20 ↑0.10
700	**0.789** ↑0.008	172.38 ↑35.15	34.49 ↓0.31	0.574 ↑0.007	158.75 ↑29.37	4.10 ↑0.03	**0.929** ↑0.011	101.13 ↑1.63	5.51 ↑0.43	0.837 ↓0.015	135.73 ↑12.96	15.23 ↓1.54	0.525 ↑0.007	165.59 ↑27.65	8.44 ↑0.08	0.731 ↑0.003	146.72 ↑21.35	13.55 ↓0.26
1000	0.779 ↓0.003	176.98 ↑30.55	35.47 ↓1.29	0.570 ↑0.002	163.45 ↑24.67	4.13 ↓0.00	0.921 ↑0.003	105.83 ↓3.07	5.93 ↑0.01	0.835 ↓0.017	141.91 ↑6.78	14.63 ↓0.94	0.514 ↓0.004	170.32 ↑22.93	8.24 ↑0.28	0.724 ↓0.004	151.70 ↑16.37	13.68 ↓0.39
1500	0.788 ↑0.006	184.67 ↑22.87	35.87 ↓1.69	0.570 ↑0.002	171.06 ↑17.06	4.14 ↓0.01	0.918 ↓0.000	102.76 ↑0.00	5.87 ↑0.07	0.840 ↓0.011	151.61 ↓2.92	13.92 ↓0.23	0.532 ↑0.013	177.94 ↑15.31	7.73 ↑0.79	0.730 ↑0.002	157.61 ↑10.46	13.51 ↓0.21

Table 14: Accuracy, total GPU memory (GB), and training time (min) across five visual QA reasoning benchmarks (LoRA, Qwen3-VL-8B) vs. number of trainable tokens. Subscript deltas are absolute deviations from *All*: ↑ green = improvement, ↓ red = degradation. **Bold** = best value per column. GPU is the sum of both GPU peak allocations.

Tokens	M3CoT			MMK12			SafeTag			STVQA			VQA-R			Avg		
	Acc	*GPU*	*Time*	*Acc*	*GPU*	*Time*	*Acc*	*GPU*	*Time*	*Acc*	*GPU*	*Time*	*Acc*	*GPU*	*Time*	*Acc*	*GPU*	*Time*
All	0.676	93.26	28.32	0.544	89.63	3.91	0.813	101.60	4.09	0.810	76.14	12.15	0.503	79.99	6.28	0.669	88.12	10.95
10	0.669 ↓0.007	**80.58** ↑12.68	25.16 ↑3.16	0.550 ↑0.006	**77.53** ↑12.10	**3.02** ↑0.89	0.807 ↓0.006	**87.37** ↑14.23	**3.77** ↑0.32	0.807 ↓0.002	**66.16** ↑9.97	**10.22** ↑1.93	0.479 ↓0.024	**69.45** ↑10.54	5.21 ↑1.07	0.663 ↓0.007	**76.22** ↑11.91	9.48 ↑1.47
25	0.596 ↓0.080	80.62 ↑12.63	25.57 ↑2.75	0.522 ↓0.022	77.58 ↑12.05	3.03 ↑0.88	0.818 ↑0.004	87.58 ↑14.03	3.79 ↑0.30	0.819 ↑0.009	66.24 ↑9.90	10.26 ↑1.89	0.505 ↑0.002	69.50 ↑10.49	**5.18** ↑1.10	0.652 ↓0.017	76.31 ↑11.82	9.57 ↑1.38
50	0.677 ↑0.001	80.71 ↑12.55	**22.48** ↑5.84	0.515 ↓0.029	77.66 ↑11.97	3.04 ↑0.87	0.816 ↑0.003	87.97 ↑13.63	3.81 ↑0.28	0.804 ↓0.006	66.37 ↑9.77	11.36 ↑0.79	**0.512** ↑0.009	69.59 ↑10.41	5.21 ↑1.07	0.665 ↓0.004	76.46 ↑11.67	9.18 ↑1.77
100	0.674 ↓0.002	80.88 ↑12.38	26.87 ↑1.45	0.530 ↓0.014	77.84 ↑11.79	3.03 ↑0.88	0.810 ↓0.003	88.60 ↑13.01	3.81 ↑0.28	0.772 ↓0.038	66.63 ↑9.51	10.82 ↑1.33	0.508 ↑0.005	69.75 ↑10.24	5.25 ↑1.03	0.659 ↓0.011	76.74 ↑11.39	9.96 ↑0.99
200	0.626 ↓0.050	81.22 ↑12.03	22.72 ↑5.60	0.546 ↑0.002	78.19 ↑11.44	3.07 ↑0.84	0.804 ↓0.009	89.90 ↑11.71	3.88 ↑0.21	0.810 ↑0.000	67.12 ↑9.02	10.53 ↑1.62	0.503 ↓0.000	70.08 ↑9.91	5.28 ↑1.00	0.658 ↓0.011	77.30 ↑10.82	**9.10** ↑1.85
300	**0.696** ↑0.020	81.55 ↑11.71	26.66 ↑1.66	**0.559** ↑0.015	78.52 ↑11.11	3.05 ↑0.86	0.791 ↓0.023	91.19 ↑10.41	3.96 ↑0.13	**0.825** ↑0.015	67.60 ↑8.54	10.72 ↑1.43	**0.512** ↑0.009	70.46 ↑9.53	5.29 ↑0.99	**0.676** ↑0.007	77.86 ↑10.26	9.94 ↑1.01
500	0.684 ↑0.008	82.20 ↑11.06	24.48 ↑3.84	0.552 ↑0.007	79.15 ↑10.48	3.06 ↑0.85	0.819 ↑0.005	93.85 ↑7.75	4.06 ↑0.03	0.818 ↑0.008	68.62 ↑7.51	12.11 ↑0.04	0.503 ↓0.000	71.08 ↑8.91	5.33 ↑0.95	0.675 ↑0.006	78.98 ↑9.14	9.81 ↑1.14
700	0.641 ↓0.035	82.84 ↑10.42	25.27 ↑3.05	0.553 ↑0.009	79.80 ↑9.83	3.07 ↑0.84	**0.825** ↑0.011	96.44 ↑5.17	3.97 ↑0.12	0.812 ↑0.003	69.57 ↑6.57	10.71 ↑1.44	0.508 ↑0.005	71.72 ↑8.27	5.39 ↑0.89	0.668 ↓0.002	80.07 ↑8.05	9.68 ↑1.27
1000	0.650 ↓0.026	83.84 ↑9.41	25.18 ↑3.14	0.540 ↓0.004	80.80 ↑8.83	3.08 ↑0.83	0.813 ↓0.000	100.35 ↑1.26	3.97 ↑0.12	0.810 ↑0.000	71.11 ↑5.02	10.75 ↑1.40	0.501 ↓0.002	72.73 ↑7.26	5.53 ↑0.75	0.663 ↓0.006	81.77 ↑6.36	9.70 ↑1.25
1500	0.674 ↓0.002	85.46 ↑7.79	25.40 ↑2.92	0.543 ↓0.001	82.42 ↑7.21	3.10 ↑0.81	0.818 ↑0.004	101.60 ↓0.00	3.97 ↑0.12	0.811 ↑0.002	73.54 ↑2.60	11.97 ↑0.18	0.501 ↓0.002	74.36 ↑5.64	5.48 ↑0.80	0.669 ↑0.000	83.48 ↑4.65	9.98 ↑0.97

M: at $M = 1000$, memory (96.62 GB) has surpassed the baseline and accuracy has no longer improved, confirming that beyond a small fraction of positions additional gradient signal yields negligible returns. Per-benchmark behavior is consistent: on ARC-C, BoolQ, and OBQA the best result is achieved in the $M \in \{100, 300\}$ range, while HellaSwag—a harder task requiring more context—peaks at $M = 500$.

Table 20 reports the parallel DoRA experiment on four commonsense benchmarks. The baseline averages 0.744 accuracy; token masking at $M = 100$ matches this (0.744) and at $M = 300$ slightly exceeds it (0.747), with memory growing from 49.24 GB at $M = 10$ to 52.72 GB at $M = 1000$. The gap in peak memory between full fine-tuning and DoRA (roughly 45 GB) confirms that the two reductions compose: DoRA shrinks the parameter footprint, and token masking shrinks the activation footprint on top of it.

Table 15: Accuracy, GPU memory (GB), and training time (min) across five visual QA reasoning benchmarks (LoRA, LLaVA-OneVision-7B) vs. number of trainable tokens. Subscript deltas are absolute deviations from *All*: ↑ green = improvement, ↓ red = degradation. **Bold** = best value per column.

Tokens	M3CoT			MMK12			SafeTag			STVQA			VQA-R			Avg		
	Acc	*GPU*	*Time*	*Acc*	*GPU*	*Time*	*Acc*	*GPU*	*Time*	*Acc*	*GPU*	*Time*	*Acc*	*GPU*	*Time*	*Acc*	*GPU*	*Time*
All	0.541	65.06	32.57	0.367	64.15	5.64	0.862	65.06	8.85	0.854	65.14	24.49	0.619	53.23	5.93	0.649	62.53	15.50
10	0.502 ↓0.039	**55.91** ↑9.15	**30.35** ↑2.22	0.348 ↓0.020	**55.16** ↑8.99	5.31 ↑0.33	0.801 ↓0.061	**55.91** ↑9.15	**8.38** ↑0.47	0.782 ↓0.072	**55.96** ↑9.18	**20.98** ↑3.51	0.608 ↓0.011	**46.23** ↑7.00	**5.15** ↑0.78	0.608 ↓0.041	**53.83** ↑8.69	**14.03** ↑1.46
25	0.504 ↓0.037	56.03 ↑9.03	30.86 ↑1.71	0.364 ↓0.004	55.27 ↑8.88	**5.30** ↑0.34	0.821 ↓0.041	56.03 ↑9.03	8.45 ↑0.40	0.820 ↓0.034	56.07 ↑9.07	21.01 ↑3.48	0.610 ↓0.009	46.36 ↑6.87	5.36 ↑0.57	0.624 ↓0.025	53.95 ↑8.58	14.20 ↑1.30
50	0.513 ↓0.029	56.16 ↑8.90	30.71 ↑1.86	0.376 ↑0.009	55.34 ↑8.80	5.32 ↑0.32	0.838 ↓0.024	56.10 ↑8.96	8.49 ↑0.36	0.826 ↓0.028	56.14 ↑9.00	21.65 ↑2.84	0.614 ↓0.004	46.44 ↑6.79	5.75 ↑0.18	0.633 ↓0.015	54.04 ↑8.49	14.38 ↑1.11
100	0.509 ↓0.032	56.31 ↑8.74	31.17 ↑1.40	0.370 ↑0.003	55.57 ↑8.58	5.37 ↑0.27	0.822 ↓0.039	56.31 ↑8.74	8.53 ↑0.32	0.819 ↓0.035	56.37 ↑8.78	22.42 ↑2.07	0.614 ↓0.004	46.65 ↑6.58	5.86 ↑0.07	0.627 ↓0.022	54.24 ↑8.28	14.67 ↑0.83
200	0.523 ↓0.018	56.76 ↑8.30	31.22 ↑1.35	0.382 ↑0.015	56.01 ↑8.13	5.52 ↑0.12	0.841 ↓0.021	56.76 ↑8.29	8.56 ↑0.29	0.839 ↓0.015	56.80 ↑8.35	22.86 ↑1.63	0.618 ↓0.000	47.09 ↑6.14	5.93 ↓0.00	0.641 ↓0.008	54.68 ↑7.84	14.82 ↑0.68
300	0.535 ↓0.007	57.21 ↑7.85	31.12 ↑1.45	0.386 ↑0.018	56.46 ↑7.69	5.40 ↑0.24	0.849 ↓0.013	57.21 ↑7.85	8.56 ↑0.29	0.857 ↑0.003	57.25 ↑7.90	23.30 ↑1.19	0.620 ↑0.002	47.54 ↑5.69	6.01 ↓0.08	0.649 ↑0.001	55.13 ↑7.40	14.88 ↑0.62
500	0.539 ↓0.002	58.13 ↑6.92	31.85 ↑0.72	0.380 ↑0.012	57.36 ↑6.78	5.44 ↑0.20	0.858 ↓0.003	58.13 ↑6.93	8.76 ↑0.09	**0.858** ↑0.004	58.17 ↑6.97	23.45 ↑1.04	**0.626** ↑0.008	48.44 ↑4.80	6.02 ↓0.09	0.652 ↑0.004	56.05 ↑6.48	15.10 ↑0.39
700	0.546 ↑0.004	59.00 ↑6.05	31.90 ↑0.67	0.385 ↑0.018	58.26 ↑5.89	5.54 ↑0.10	0.864 ↑0.002	59.00 ↑6.05	8.82 ↑0.03	0.856 ↑0.002	59.08 ↑6.07	24.16 ↑0.33	0.616 ↓0.002	49.33 ↑3.90	6.10 ↓0.17	**0.654** ↑0.005	56.94 ↑5.59	15.30 ↑0.19
1000	**0.558** ↑0.016	60.41 ↑4.65	32.41 ↑0.16	**0.398** ↑0.030	59.20 ↑4.95	5.60 ↑0.04	0.849 ↓0.013	60.37 ↑4.69	8.89 ↓0.04	0.851 ↓0.003	60.44 ↑4.70	24.51 ↓0.02	0.610 ↓0.009	49.91 ↑3.32	6.25 ↓0.32	0.653 ↑0.004	58.07 ↑4.46	15.53 ↓0.04
1500	0.550 ↑0.009	60.41 ↑4.65	32.61 ↓0.04	0.384 ↑0.017	61.87 ↑2.28	5.65 ↓0.01	**0.865** ↑0.003	61.62 ↑3.44	8.95 ↓0.10	0.854 ↓0.000	61.66 ↑3.49	24.84 ↓0.35	0.606 ↓0.013	51.19 ↑2.04	6.35 ↓0.42	0.652 ↑0.003	59.35 ↑3.18	15.68 ↓0.18

Table 16: Math reasoning accuracy across six benchmarks (LLaMA3-8B) for Full Fine-tuning and DoRA vs. number of trainable tokens. Avg = mean accuracy over all six tasks. Subscript deltas are absolute deviations from *All*: ↑ green = improvement, ↓ red = degradation. **Bold** = best value per column. Full FT GPU = gpu0+gpu1; DoRA GPU = gpu0.

Tokens	Full Fine-tuning									DoRA								
	AddSub	*MultiA*	*SVAMP*	*SingleEq*	*GSM8K*	*MAWPS*	*Avg*	*GPU*	*Time*	*AddSub*	*MultiA*	*SVAMP*	*SingleEq*	*GSM8K*	*MAWPS*	*Avg*	*GPU*	*Time*
All	0.881	0.978	0.749	0.886	0.618	0.861	0.829	95.38	20.62	0.876	0.835	0.710	0.866	0.544	0.815	0.774	55.59	21.31
10	0.698 ↓0.184	0.155 ↓0.823	0.551 ↓0.198	0.232 ↓0.653	0.238 ↓0.380	0.523 ↓0.339	0.399 ↓0.430	**83.93** ↑11.45	**16.73** ↑3.89	0.787 ↓0.089	0.322 ↓0.513	0.495 ↓0.215	0.621 ↓0.245	0.266 ↓0.278	0.471 ↓0.345	0.494 ↓0.281	**49.70** ↑5.89	20.93 ↑0.38
25	0.808 ↓0.074	0.280 ↓0.698	0.609 ↓0.140	0.772 ↓0.114	0.333 ↓0.285	0.655 ↓0.206	0.576 ↓0.253	84.15 ↑11.23	17.02 ↑3.60	0.825 ↓0.051	0.410 ↓0.425	0.589 ↓0.121	0.791 ↓0.075	0.325 ↓0.219	0.672 ↓0.143	0.602 ↓0.172	49.79 ↑5.80	20.90 ↑0.41
50	0.891 ↑0.010	0.678 ↓0.300	0.655 ↓0.094	0.843 ↓0.043	0.313 ↓0.305	0.790 ↓0.071	0.695 ↓0.134	84.61 ↑10.77	16.98 ↑3.64	0.863 ↓0.013	0.688 ↓0.147	0.623 ↓0.087	0.833 ↓0.033	0.322 ↓0.222	0.790 ↓0.025	0.687 ↓0.088	49.94 ↑5.66	**20.84** ↑0.47
100	0.878 ↓0.003	0.940 ↓0.038	0.677 ↓0.072	0.884 ↓0.002	0.499 ↓0.119	0.861 ↑0.000	0.790 ↓0.039	85.34 ↑10.04	17.57 ↑3.05	0.878 ↑0.003	**0.882** ↑0.047	0.683 ↓0.027	0.864 ↓0.002	0.526 ↓0.018	0.807 ↓0.008	0.773 ↓0.001	50.26 ↑5.34	21.15 ↑0.16
300	0.894 ↑0.012	0.983 ↑0.005	0.749 ↑0.000	**0.892** ↑0.006	0.619 ↑0.001	0.866 ↑0.004	**0.834** ↑0.005	88.35 ↑7.03	19.13 ↑1.49	0.881 ↑0.005	0.880 ↑0.045	0.730 ↑0.020	0.870 ↑0.004	0.549 ↑0.005	**0.832** ↑0.017	**0.790** ↑0.016	51.41 ↑4.18	21.45 ↓0.14
500	0.894 ↑0.012	**0.987** ↑0.008	0.751 ↑0.002	0.890 ↑0.004	**0.627** ↑0.009	0.849 ↓0.013	0.833 ↑0.004	91.49 ↑3.89	20.72 ↓0.10	**0.889** ↑0.013	0.850 ↑0.015	0.721 ↑0.011	0.862 ↓0.004	0.550 ↑0.005	0.821 ↑0.006	0.782 ↑0.008	52.60 ↑2.99	21.84 ↓0.53
700	0.894 ↑0.012	0.982 ↑0.003	0.753 ↑0.004	0.878 ↓0.008	0.617 ↓0.001	0.845 ↓0.017	0.828 ↓0.001	94.47 ↑0.91	22.88 ↓2.26	0.876 ↑0.000	0.865 ↑0.030	0.729 ↑0.019	**0.876** ↑0.010	0.552 ↑0.007	0.821 ↑0.006	0.786 ↑0.012	53.72 ↑1.88	22.05 ↓0.74
1000	**0.901** ↑0.020	0.970 ↓0.008	**0.754** ↑0.005	0.882 ↓0.004	0.616 ↓0.002	**0.878** ↑0.017	0.834 ↑0.005	97.50 ↓2.12	25.15 ↓4.53	0.876 ↑0.000	0.850 ↑0.015	**0.735** ↑0.025	0.862 ↓0.004	**0.556** ↑0.011	0.815 ↑0.000	0.782 ↑0.008	54.87 ↑0.72	22.52 ↓1.21

J.2 Mathematical Reasoning (LLaMA-3-8B)

Table 16 reports accuracy on six arithmetic and word-problem benchmarks for both full fine-tuning and DoRA. The math setting is the most demanding: with $M = 10$, full-FT average accuracy drops to 0.399 from the baseline 0.829—a 0.43 absolute drop, reflecting that single-digit M is insufficient to span the reasoning structure needed for multi-step arithmetic. The picture changes rapidly as M grows: at $M = 100$ average accuracy reaches 0.790, at $M = 300$ it equals or exceeds the baseline on every benchmark (0.834 average), and at $M = 1000$ it stabilizes at 0.834. The pattern is similar under DoRA, where $M = 300$ achieves the best average (0.790, above the baseline 0.774) while saving 4 GB. GSM8K is the hardest individual benchmark, requiring $M \geq 500$ to cross the baseline; the shorter-form datasets (AddSub, SingleEq) saturate much earlier, consistent with their lower per-step reasoning depth.

J.3 Visual QA Reasoning (LLaVA-OneVision-7B and Qwen3-VL-8B)

Table 12 reports full fine-tuning on LLaVA-OneVision-7B across five visual QA benchmarks. The baseline averages 0.693 accuracy at 95.34 GB; token masking crosses this threshold at $M = 700$ (0.700, 90.43 GB), and the best average is 0.702 at $M = 1000$. Per-benchmark results reveal task-

Table 17: Video reasoning results (Full fine-tuning, Qwen3-VL-8B) across five benchmarks vs. number of trainable tokens. HOI row 100 is approximate (†). ↑ green = improvement over *All*, ↓ red = degradation. **Bold** = best per column. GPU = gpu0+gpu1 (GB).

Tokens	VideoMathQA			LevelGauge			HOI			IntuPhys			TempReas			Avg		
	Acc	*GPU*	*Time*	*Acc*	*GPU*	*Time*	*Acc*	*GPU*	*Time*	*Acc*	*GPU*	*Time*	*Acc*	*GPU*	*Time*	*Acc*	*GPU*	*Time*
All	0.59	120.53	37.86	0.83	120.49	18.47	0.4067	108.20	42.46	0.5787	111.31	19.93	0.6158	104.93	3.94	0.604	113.09	24.53
10	0.50 ↓0.08	**109.44** ↑11.10	34.64 ↑3.22	0.78 ↓0.05	**109.43** ↑11.06	16.48 ↑1.99	**0.4671** ↑0.0604	**95.44** ↑12.76	37.25 ↑5.21	0.5354 ↓0.0433	**97.37** ↑13.95	**14.33** ↑5.60	0.5857 ↓0.0301	**93.46** ↑11.47	3.75 ↑0.19	0.574 ↓0.031	**101.03** ↑12.07	**21.29** ↑3.24
25	0.53 ↓0.06	109.47 ↑11.06	34.50 ↑3.36	0.78 ↓0.04	109.47 ↑11.02	**16.17** ↑2.30	0.4134 ↑0.0067	96.10 ↑12.10	**36.17** ↑6.29	0.5384 ↓0.0403	98.00 ↑13.31	16.77 ↑3.16	0.5463 ↓0.0695	94.16 ↑10.77	**2.92** ↑1.02	0.562 ↓0.043	101.44 ↑11.65	21.31 ↑3.23
50	0.53 ↓0.06	109.54 ↑10.99	35.08 ↑2.78	0.79 ↓0.04	109.54 ↑10.95	16.45 ↑2.02	0.4121 ↑0.0054	97.12 ↑11.08	37.28 ↑5.18	0.5366 ↓0.0421	99.04 ↑12.27	17.55 ↑2.38	0.6031 ↓0.0127	95.18 ↑9.75	2.98 ↑0.96	0.574 ↓0.030	102.08 ↑11.01	21.87 ↑2.66
100	0.54 ↓0.04	109.70 ↑10.83	34.76 ↑3.10	0.78 ↓0.05	109.70 ↑10.79	16.31 ↑2.16	0.4100 ↑0.0033†	102.32 ↑5.88	40.00 ↑2.46	0.5318 ↓0.0469	101.16 ↑10.15	17.29 ↑2.64	0.6093 ↓0.0065	97.22 ↑7.70	3.03 ↑0.91	0.574 ↓0.030	104.02 ↑9.07	22.28 ↑2.25
300	0.57 ↓0.01	110.26 ↑10.27	**34.48** ↑3.38	0.80 ↓0.03	110.27 ↑10.22	16.53 ↑1.94	0.4094 ↑0.0027	107.53 ↑0.67	43.68 ↓1.22	0.5564 ↓0.0223	109.51 ↑1.81	20.70 ↓0.77	0.6019 ↓0.0139	105.53 ↓0.61	3.31 ↑0.63	0.588 ↓0.017	108.62 ↑4.47	23.74 ↑0.79
500	**0.59** ↑0.00	110.80 ↑9.73	34.63 ↑3.23	0.82 ↓0.01	110.79 ↑9.70	16.71 ↑1.76	0.4121 ↑0.0054	108.01 ↑0.19	39.57 ↑2.89	**0.6061** ↑0.0274	111.31 ↑0.00	18.53 ↑1.40	0.6157 ↓0.0001	104.68 ↑0.25	3.13 ↑0.81	**0.609** ↑0.005	109.12 ↑3.97	22.51 ↑2.02
700	0.59 ↑0.00	111.36 ↑9.17	34.97 ↑2.89	0.78 ↓0.04	111.34 ↑9.14	16.88 ↑1.59	0.4060 ↓0.0007	108.01 ↑0.19	39.96 ↑2.50	0.5659 ↓0.0128	111.31 ↑0.00	19.47 ↑0.46	0.6226 ↑0.0068	104.73 ↑0.19	3.17 ↑0.77	0.593 ↓0.011	109.35 ↑3.74	22.89 ↑1.64
1000	0.58 ↑0.00	112.17 ↑8.36	35.37 ↑2.49	**0.84** ↑0.01	112.20 ↑8.29	16.98 ↑1.49	0.4121 ↑0.0054	108.01 ↑0.19	38.40 ↑4.06	0.5282 ↓0.0505	111.32 ↑0.00	18.53 ↑1.40	**0.6233** ↑0.0075	104.78 ↑0.14	3.10 ↑0.84	0.597 ↓0.008	109.70 ↑3.40	22.48 ↑2.06
1500	0.57 ↓0.01	113.57 ↑6.96	36.39 ↑1.47	0.82 ↓0.01	113.56 ↑6.93	17.57 ↑0.90	0.4174 ↑0.0107	108.01 ↑0.19	38.78 ↑3.68	0.5827 ↑0.0040	111.31 ↑0.00	18.32 ↑1.61	0.6180 ↑0.0022	104.91 ↑0.02	3.11 ↑0.83	0.602 ↓0.003	110.27 ↑2.82	22.83 ↑1.70

Table 18: Video reasoning results (LoRA, Qwen3-VL-8B) across five benchmarks vs. number of trainable tokens. ↑ green = improvement over *All*, ↓ red = degradation. **Bold** = best per column. GPU: VideoMathQA, HOI, IntuPhys, TempReas use gpu0; LevelGauge uses gpu0+gpu1 (GB).

Tokens	VideoMathQA			LevelGauge			HOI			IntuPhys			TempReas			Avg		
	Acc	*GPU*	*Time*	*Acc*	*GPU*	*Time*	*Acc*	*GPU*	*Time*	*Acc*	*GPU*	*Time*	*Acc*	*GPU*	*Time*	*Acc*	*GPU*	*Time*
All	**0.53**	53.40	34.00	0.74	97.89	15.17	0.4403	44.33	41.94	0.5516	47.17	21.15	0.6343	41.56	3.48	**0.579**	56.87	23.15
10	0.51 ↓0.02	**46.71** ↑6.69	**31.97** ↑2.03	0.72 ↓0.02	**84.51** ↑13.38	**14.46** ↑0.71	0.3383 ↓0.1020	**39.26** ↑5.07	36.64 ↑5.30	0.4466 ↓0.1050	**41.44** ↑5.73	14.56 ↑6.59	0.6389 ↑0.0046	**36.98** ↑4.58	**2.70** ↑0.78	0.531 ↓0.048	**49.78** ↑7.09	20.07 ↑3.08
25	0.51 ↓0.02	46.73 ↑6.67	32.25 ↑1.75	0.72 ↓0.02	84.56 ↑13.33	14.53 ↑0.64	0.3208 ↓0.1195	39.49 ↑4.84	**35.91** ↑6.03	0.4748 ↓0.0768	41.68 ↑5.49	14.59 ↑6.56	**0.6528** ↑0.0185	37.20 ↑4.35	**2.70** ↑0.78	0.536 ↓0.044	49.93 ↑6.94	**20.00** ↑3.15
50	0.51 ↓0.03	46.78 ↑6.62	32.37 ↑1.63	0.72 ↓0.02	84.64 ↑13.25	14.59 ↑0.58	0.3208 ↓0.1195	39.82 ↑4.51	36.96 ↑4.98	0.4610 ↓0.0906	42.03 ↑5.13	14.75 ↑6.40	0.6296 ↓0.0047	37.56 ↑3.99	2.72 ↑0.76	0.528 ↓0.051	50.17 ↑6.70	20.28 ↑2.87
100	0.29 ↓0.24	46.86 ↑6.54	32.47 ↑1.53	0.73 ↑0.00	84.82 ↑13.07	14.52 ↑0.65	0.3289 ↓0.1114	40.46 ↑3.88	37.73 ↑4.21	0.5000 ↓0.0516	42.68 ↑4.49	**14.44** ↑6.71	0.6343 ↑0.0000	38.18 ↑3.38	2.76 ↑0.72	0.497 ↓0.083	50.60 ↑6.27	20.38 ↑2.76
300	0.39 ↓0.14	47.25 ↑6.15	32.78 ↑1.22	**0.74** ↑0.00	85.47 ↑12.42	14.57 ↑0.60	0.3899 ↓0.0504	43.06 ↑1.27	37.59 ↑4.35	0.5352 ↓0.0164	45.27 ↑1.89	15.61 ↑5.54	0.6435 ↑0.0092	40.80 ↑0.76	2.74 ↑0.74	0.540 ↓0.040	52.37 ↑4.50	20.66 ↑2.49
500	0.31 ↓0.22	47.53 ↑5.87	32.88 ↑1.12	0.73 ↓0.01	86.15 ↑11.74	14.55 ↑0.62	0.4416 ↑0.0013	44.12 ↑0.21	36.89 ↑5.05	0.5522 ↑0.0006	47.16 ↑0.01	15.08 ↑6.07	0.6389 ↑0.0046	41.52 ↑0.04	2.79 ↑0.69	0.535 ↓0.045	53.30 ↑3.57	20.44 ↑2.71
700	0.36 ↓0.18	47.85 ↑5.55	34.67 ↓0.67	0.73 ↓0.01	86.77 ↑11.12	14.75 ↑0.42	**0.4572** ↑0.0169	44.30 ↑0.03	36.90 ↑5.04	**0.5561** ↑0.0045	47.16 ↑0.01	14.90 ↑6.25	0.6443 ↑0.0100	41.52 ↑0.04	2.74 ↑0.74	0.550 ↓0.030	53.52 ↑3.35	20.79 ↑2.36
1000	0.46 ↓0.08	48.37 ↑5.03	33.52 ↑0.48	0.72 ↓0.01	87.81 ↑10.08	14.68 ↑0.49	0.4436 ↑0.0033	44.33 ↑0.00	37.09 ↑4.85	**0.5561** ↑0.0045	47.16 ↑0.00	15.07 ↑6.08	0.6343 ↑0.0000	41.54 ↑0.02	2.71 ↑0.77	0.563 ↓0.016	53.84 ↑3.03	20.61 ↑2.53
1500	0.33 ↓0.20	49.18 ↑4.23	33.57 ↑0.43	0.73 ↓0.01	89.42 ↑8.47	14.94 ↑0.23	0.4376 ↓0.0027	44.33 ↑0.00	37.31 ↑4.63	0.5557 ↑0.0041	47.16 ↑0.00	15.60 ↑5.55	0.6343 ↑0.0000	41.55 ↑0.01	2.76 ↑0.72	0.538 ↓0.042	54.33 ↑2.54	20.84 ↑2.31

Table 19: Image classification results (Full fine-tuning, ViT-L/16-224) across five benchmarks vs. number of trainable tokens. ↑ green = improvement over *All*, ↓ red = degradation. **Bold** = best per column. GPU = gpu0 peak allocation (GB).

Tokens	Tomato			SUN397			RVL-CDIP			EuroSAT			ChestXray			Avg		
	Acc	*GPU*	*Time*	*Acc*	*GPU*	*Time*	*Acc*	*GPU*	*Time*	*Acc*	*GPU*	*Time*	*Acc*	*GPU*	*Time*	*Acc*	*GPU*	*Time*
All	0.9705	8.44	1.27	0.4096	8.32	1.55	0.5125	5.34	0.34	0.9819	8.32	0.76	0.8606	5.34	0.66	0.7470	7.15	0.92
10	0.9633 ↓0.0072	**6.18** ↑2.26	**0.48** ↑0.79	0.2415 ↓0.1681	**6.18** ↑2.14	1.91 ↓0.36	0.2734 ↓0.2391	**4.23** ↑1.11	**0.24** ↑0.10	0.9811 ↓0.0008	**6.18** ↑2.14	**0.70** ↑0.06	0.7804 ↓0.0802	**4.23** ↑1.11	**0.50** ↑0.16	0.6479 ↓0.0991	**5.40** ↑1.75	**0.77** ↑0.15
25	0.9605 ↓0.0100	6.37 ↑2.07	0.51 ↑0.76	0.3373 ↓0.0723	6.37 ↑1.95	1.84 ↓0.29	0.3969 ↓0.1156	4.33 ↑1.00	0.26 ↑0.08	0.9700 ↓0.0119	6.37 ↑1.95	0.74 ↑0.02	0.7853 ↓0.0753	4.33 ↑1.00	0.52 ↑0.14	0.6900 ↓0.0570	5.56 ↑1.60	0.77 ↑0.14
50	0.9725 ↑0.0020	6.71 ↑1.73	0.53 ↑0.74	0.3785 ↓0.0311	6.71 ↑1.61	1.92 ↓0.37	0.4266 ↓0.0859	4.51 ↑0.83	0.28 ↑0.06	0.9767 ↓0.0052	6.71 ↑1.61	0.74 ↑0.02	0.7612 ↓0.0994	4.52 ↑0.82	0.56 ↑0.10	0.7031 ↓0.0439	5.83 ↑1.32	0.81 ↑0.11
100	**0.9764** ↑0.0059	7.37 ↑1.07	0.58 ↑0.69	0.4222 ↑0.0126	7.37 ↑0.95	2.85 ↓1.30	0.4872 ↓0.0253	4.85 ↑0.49	0.29 ↑0.05	0.9841 ↑0.0022	7.37 ↑0.95	0.75 ↑0.01	**0.8862** ↑0.0256	4.85 ↑0.49	0.59 ↑0.07	0.7512 ↑0.0042	6.36 ↑0.79	1.01 ↓0.10
150	0.9748 ↑0.0043	8.03 ↑0.41	0.66 ↑0.61	**0.4416** ↑0.0320	8.03 ↑0.29	1.91 ↓0.36	0.5134 ↑0.0009	5.22 ↑0.12	0.32 ↑0.02	**0.9844** ↑0.0025	8.03 ↑0.29	0.75 ↑0.01	0.8606 ↑0.0000	5.22 ↑0.12	0.64 ↑0.02	**0.7550** ↑0.0079	6.91 ↑0.25	0.86 ↑0.06
180	0.9716 ↑0.0011	8.44 ↑0.00	0.72 ↑0.55	0.4136 ↑0.0040	8.14 ↑0.18	**1.48** ↑0.07	**0.5136** ↑0.0011	5.34 ↑0.00	0.34 ↑0.00	0.9837 ↑0.0018	8.44 ↓0.12	0.78 ↓0.02	0.8638 ↑0.0032	5.34 ↑0.00	0.66 ↑0.00	0.7493 ↑0.0022	7.14 ↑0.01	0.80 ↑0.12

specific saturation: SafeTag and VQA-R are nearly saturated at $M = 10$ (differences of 0.001 and 0.003 respectively), while M3CoT and STVQA require $M \geq 500$ to recover. MMK12 is unusually

Table 20: Accuracy, GPU memory (GB), and training time (min) across four commonsense reasoning benchmarks (DoRA) vs. number of trainable tokens. Subscript deltas are absolute deviations from *All*: ↑ green = improvement, ↓ red = degradation. **Bold** = best value per column.

Tokens	ARC-E			ARC-C			BoolQ			OBQA			Avg		
	Acc	*GPU*	*Time*	*Acc*	*GPU*	*Time*	*Acc*	*GPU*	*Time*	*Acc*	*GPU*	*Time*	*Acc*	*GPU*	*Time*
All	0.866	50.20	4.82	0.602	50.20	2.53	0.684	50.20	19.65	0.826	50.20	**10.24**	0.744	50.20	9.31
	0.783	**49.24**	**4.67**	0.595	**49.24**	**2.33**	0.681	**49.24**	**19.63**	0.690	**49.24**	10.28	0.687	**49.24**	**9.23**
10	↓0.083	↑0.96	↑0.15	↓0.007	↑0.96	↑0.20	↓0.003	↑0.96	↑0.02	↓0.136	↑0.96	↓0.04	↓0.057	↑0.96	↑0.08
	0.811	49.30	4.68	0.604	49.30	**2.33**	0.682	49.30	19.65	0.692	49.30	10.31	0.697	49.30	9.24
25	↓0.054	↑0.90	↑0.14	↑0.003	↑0.90	↑0.20	↓0.002	↑0.90	↓0.00	↓0.134	↑0.90	↓0.07	↓0.047	↑0.90	↑0.07
	0.856	49.43	4.69	**0.608**	49.43	2.34	0.678	49.43	19.93	0.800	49.43	10.33	0.736	49.43	9.32
50	↓0.010	↑0.77	↑0.13	↑0.007	↑0.77	↑0.19	↓0.006	↑0.77	↓0.28	↓0.026	↑0.77	↓0.09	↓0.009	↑0.77	↓0.01
	0.870	49.62	4.74	0.602	49.62	2.36	0.680	49.62	19.93	0.824	49.62	10.40	0.744	49.62	9.36
100	↑0.004	↑0.58	↑0.08	↑0.001	↑0.58	↑0.17	↓0.004	↑0.58	↓0.28	↓0.002	↑0.58	↓0.16	↓0.000	↑0.58	↓0.05
	0.869	50.38	4.78	0.608	50.38	2.38	0.682	50.38	20.21	**0.828**	50.38	10.49	**0.747**	50.38	9.46
300	↑0.003	↓0.17	↑0.04	↑0.006	↓0.17	↑0.15	↓0.002	↓0.17	↓0.56	↑0.002	↓0.17	↓0.25	↑0.002	↓0.17	↓0.15
	0.864	51.20	4.81	0.607	51.20	2.41	0.680	51.20	20.44	0.826	51.20	10.66	0.744	51.20	9.58
500	↓0.002	↓1.00	↑0.01	↑0.005	↓1.00	↑0.12	↓0.004	↓1.00	↓0.79	↓0.000	↓1.00	↓0.42	↓0.000	↓1.00	↓0.27
	0.865	51.94	4.93	0.604	51.95	2.45	**0.692**	51.95	20.64	0.818	51.95	10.79	0.745	51.95	9.70
700	↓0.000	↓1.74	↓0.11	↑0.003	↓1.75	↑0.08	↑0.008	↓1.75	↓0.99	↓0.008	↓1.75	↓0.55	↑0.001	↓1.74	↓0.39
	0.865	52.72	4.99	0.603	52.72	2.49	0.690	52.72	20.75	0.826	52.72	10.95	0.746	52.72	9.79
1000	↓0.000	↓2.51	↓0.17	↑0.002	↓2.51	↑0.04	↑0.006	↓2.51	↓1.10	↓0.000	↓2.51	↓0.71	↑0.002	↓2.51	↓0.48

Table 21: GLUE benchmark results – part 1 (RoBERTa-base, Full fine-tuning). Metrics: Acc and F1. ↑ green = improvement over *All*, ↓ red = degradation. **Bold** = best per column.

Tokens	SST-2				QNLI				MRPC				QQP				Avg			
	Acc	*F1*	*GPU*	*Time*	*Acc*	*F1*	*GPU*	*Time*	*Acc*	*F1*	*GPU*	*Time*	*Acc*	*F1*	*GPU*	*Time*	*Acc*	*F1*	*GPU*	*Time*
All	0.9381	0.9380	3.91	2.86	0.9220	0.9210	6.39	10.26	0.8750	0.8564	3.43	0.61	0.8974	0.8910	6.27	25.72	0.9081	0.9016	5.00	9.86
	0.9209	0.9208	**3.18**	**1.99**	0.8468	0.8466	**4.78**	**6.60**	0.7598	0.6479	**2.79**	**0.41**	0.7809	0.7740	**4.67**	**17.70**	0.8271	0.7973	**3.86**	**6.67**
5	↓0.0172	↓0.0172	↑0.72	↑0.87	↓0.0752	↓0.0744	↑1.61	↑3.66	↓0.1152	↓0.2085	↑0.64	↑0.20	↓0.1165	↓0.1170	↑1.59	↑8.02	↓0.0810	↓0.1043	↑1.14	↑3.19
	0.9278	0.9277	3.30	2.24	0.8770	0.8770	4.83	7.02	0.7647	0.6551	2.85	0.44	0.8274	0.8167	4.72	19.26	0.8492	0.8191	3.92	7.24
10	↓0.0103	↓0.0103	↑0.61	↑0.62	↓0.0450	↓0.0440	↑1.56	↑3.24	↓0.1103	↓0.2013	↑0.58	↑0.17	↓0.0700	↓0.0743	↑1.54	↑6.46	↓0.0589	↓0.0825	↑1.07	↑2.62
	0.9392	**0.9392**	3.56	2.50	0.9169	0.9169	4.99	8.39	0.8284	0.7756	3.00	0.53	0.8974	0.8918	4.88	22.08	0.8955	0.8809	4.11	8.38
25	↑0.0011	↑0.0012	↑0.35	↑0.36	↓0.0051	↓0.0041	↑1.40	↑1.87	↓0.0466	↓0.0808	↑0.43	↑0.08	↑0.0000	↑0.0008	↑1.39	↑3.64	↓0.0127	↓0.0207	↑0.89	↑1.49
	0.9388	0.9357	4.04	2.78	0.9280	0.9260	5.22	10.16	0.8715	0.8524	3.25	0.67	0.8979	0.8922	5.12	23.62	0.9091	0.9016	4.41	9.31
50	↑0.0007	↓0.0023	↓0.13	↑0.08	↑0.0060	↑0.0050	↑1.16	↑0.10	↓0.0035	↓0.0040	↑0.18	↓0.06	↑0.0005	↑0.0012	↑1.15	↑2.10	↑0.0009	↑0.0000	↑0.59	↑0.55
	0.9416	0.9346	3.90	2.84	**0.9291**	**0.9289**	5.47	10.59	**0.8778**	**0.8596**	3.39	0.75	0.8975	0.8921	5.38	25.04	**0.9115**	**0.9038**	4.54	9.80
75	↑0.0035	↓0.0034	↑0.01	↑0.02	↑0.0071	↑0.0079	↑0.91	↓0.33	↑0.0028	↑0.0032	↑0.04	↓0.14	↑0.0001	↑0.0011	↑0.89	↑0.68	↑0.0034	↑0.0022	↑0.46	↑0.06
	0.9346	0.9346	3.92	2.88	0.9217	0.9223	5.71	10.87	0.8705	0.8560	3.45	0.61	**0.8989**	**0.8929**	5.62	25.79	0.9064	0.9014	4.68	10.04
100	↓0.0035	↓0.0034	↓0.01	↓0.02	↓0.0003	↑0.0013	↑0.67	↓0.61	↓0.0045	↓0.0004	↓0.03	↑0.00	↑0.0015	↑0.0019	↑0.65	↓0.07	↓0.0017	↓0.0001	↑0.32	↓0.18

Table 22: GLUE benchmark results – part 2 (RoBERTa-base, Full fine-tuning). CoLA reports Matthews corr.; STS-B reports Pearson and Spearman corr. ↑ green = improvement over *All*, ↓ red = degradation. **Bold** = best per column. Avg GPU and Time are averaged over all four tasks.

Tokens	CoLA			MNLI				RTE				STS-B				Avg			
	Mcc	*GPU*	*Time*	*Acc*	*F1*	*GPU*	*Time*	*Acc*	*F1*	*GPU*	*Time*	*Prs*	*Spr*	*GPU*	*Time*	*Main*	*F1/Spr*	*GPU*	*Time*
All	0.5778	2.35	0.84	0.8687	0.8678	6.35	34.79	0.5993	0.5957	3.85	1.26	0.9043	0.9015	2.64	1.45	0.7375	0.7883	3.80	9.58
	0.5533	**2.08**	**0.80**	0.5169	0.5174	**4.76**	**22.57**	0.5668	0.5605	**3.05**	**0.83**	0.8636	0.8622	**2.26**	**1.21**	0.6251	0.6467	**3.04**	**6.35**
5	↓0.0245	↑0.27	↑0.04	↓0.3518	↓0.3504	↑1.59	↑12.22	↓0.0325	↓0.0352	↑0.80	↑0.43	↓0.0407	↓0.0393	↑0.38	↑0.24	↓0.1124	↓0.1416	↑0.76	↑3.23
	0.5521	2.13	0.82	0.7221	0.7175	4.81	24.19	0.5993	0.5989	3.07	0.86	0.8625	0.8624	2.28	1.24	0.6840	0.7263	3.07	6.78
10	↓0.0257	↑0.22	↑0.02	↓0.1466	↓0.1503	↑1.54	↑10.60	↑0.0000	↑0.0032	↑0.78	↑0.40	↓0.0418	↓0.0391	↑0.35	↑0.21	↓0.0535	↓0.0621	↑0.72	↑2.81
	0.5799	2.27	0.84	0.8467	0.8462	4.97	29.26	0.5993	0.5858	3.18	0.97	0.9055	0.9030	2.38	1.34	0.7329	0.7783	3.20	8.10
25	↑0.0021	↑0.08	↑0.00	↓0.0220	↓0.0216	↑1.38	↑5.53	↑0.0000	↓0.0099	↑0.67	↑0.29	↑0.0012	↑0.0015	↑0.26	↑0.11	↓0.0047	↓0.0100	↑0.60	↑1.48
	0.5838	2.34	0.84	0.8699	0.8683	5.21	32.84	0.6787	0.6732	3.28	1.12	0.9068	0.9068	2.48	1.38	0.7598	0.8161	3.33	9.05
50	↑0.0060	↑0.01	↑0.00	↑0.0012	↑0.0005	↑1.14	↑1.95	↑0.0794	↑0.0775	↑0.57	↑0.14	↑0.0025	↑0.0053	↑0.16	↑0.07	↑0.0223	↑0.0278	↑0.47	↑0.54
	0.5778	2.34	0.86	**0.8737**	**0.8699**	5.47	33.94	0.7076	0.7020	3.42	1.24	**0.9118**	**0.9120**	2.61	1.42	0.7677	0.8280	3.46	9.37
75	↑0.0000	↑0.01	↓0.02	↑0.0050	↑0.0021	↑0.88	↑0.85	↑0.1083	↑0.1063	↑0.44	↑0.02	↑0.0075	↑0.0105	↑0.03	↑0.03	↑0.0302	↑0.0396	↑0.34	↑0.22
	0.5778	2.34	0.87	0.8691	0.8684	5.71	35.24	**0.7509**	**0.7456**	3.52	1.26	0.9023	0.9008	2.72	1.46	**0.7750**	**0.8383**	3.57	9.71
100	↑0.0000	↑0.01	↓0.03	↑0.0004	↑0.0006	↑0.64	↓0.45	↑0.1516	↑0.1499	↑0.33	↑0.00	↓0.0020	↓0.0007	↓0.08	↓0.01	↑0.0375	↑0.0499	↑0.22	↓0.12

noisy—its peak (0.422) occurs at $M = 200$ and the baseline itself (0.400) is beaten at almost every M, likely reflecting noise in its single-train-split structure.

Table 13 reports the same experiments on Qwen3-VL-8B under full fine-tuning. The baseline accuracy is higher (0.728) and the memory cost is much larger (168.07 GB summed over four GPUs). Token masking is therefore more impactful in absolute terms: $M = 10$ saves 32.78 GB while only losing 0.017 accuracy, and $M = 200$ matches the baseline ($0.743 > 0.728$) at 138.48 GB, a 29.59 GB

reduction. The best single-M result is $M = 200$ (0.743); larger M values continue to improve per-task peaks (M3CoT at $M = 700$, SafeTag at $M = 300/700$) but saturate on average.

Table 14 reports LoRA on Qwen3-VL-8B. The baseline accuracy drops to 0.669 (reflecting LoRA's reduced capacity), and the memory cost drops substantially as well (88.12 GB baseline). Token masking compounds: at $M = 300$ accuracy rises to 0.676 at 77.86 GB—a 10.26 GB saving on top of the already-compressed LoRA footprint, validating the orthogonality claim of Section 4.

Table 15 reports the LLaVA-OneVision LoRA experiments. The pattern is consistent: baseline 0.649 at 62.53 GB, matched or exceeded from $M = 300$ onward (0.649) and peaking at $M = 700$ (0.654). Memory at the best M is 56.94 GB, a 5.6 GB saving.

J.4 Video Reasoning (Qwen3-VL-8B)

Table 17 reports full fine-tuning on five video benchmarks. At M=500, average accuracy (0.609) exceeds the baseline (0.604) while reducing total GPU memory from 113.09 to 109.12 GB, a 3.97 GB saving. The per-task picture is task-dependent: HOI peaks at M=10 (0.467 versus a baseline of 0.407), whereas IntuPhys and TempReas saturate near the baseline. The HOI entry at M=100 is approximate (†) owing to a partial-run restart.

Table 18 reports video LoRA. Memory savings remain substantial (49.78 GB at M=10 versus 56.87 GB for the baseline), but average accuracy stays below the baseline at every M, with the smallest gap at M=1000 (0.563 versus 0.579). On HOI, IntuPhys, and TempReas, accuracy recovers to or above the baseline from M=500 onward; the remaining gap is driven by VideoMathQA, which degrades sharply for M between 100 and 500 (as low as 0.29) and recovers only partially at M=1000. VideoMathQA requires multi-step mathematical reasoning over long videos, and under the reduced capacity of LoRA, restricting supervision to a subset of positions appears insufficient to recover full-sequence performance on this task.

J.5 Image Classification (ViT-L/16-224)

Table 19 reports full fine-tuning results on five image classification benchmarks. The ViT setting is instructive for two reasons: the baseline is small (context length 197, not 2048), so the absolute memory savings are modest (5.40 GB at $M = 10$ versus 7.15 GB baseline). But the relative savings are substantial ($\sim 25\%$), and accuracy is preserved from $M = 100$ onward: the best average is 0.7550 at $M = 150$ (versus baseline 0.7470), a slight improvement with a 0.24 GB memory reduction. RVL-CDIP and SUN397 are the hardest benchmarks and require the largest M to recover baseline performance; Tomato, EuroSAT, and ChestXray saturate earlier. Since ViT is bidirectional, this aligns with Theorem 2: a small number of positions spans the achievable prediction space, and the optimal M is dataset-specific rather than position-specific.

J.6 GLUE (RoBERTa-base)

Tables 21 and 22 report GLUE results on RoBERTa-base under full fine-tuning, reported across two tables because the evaluation metrics differ: Table 21 reports accuracy and F1 on SST-2, QNLI, MRPC, and QQP; Table 22 reports Matthews correlation on CoLA, accuracy on MNLI and RTE, and Pearson/Spearman correlation on STS-B. The GLUE setting is where our bidirectional theory (Theorem 2) makes its strongest prediction: positions with aligned hidden-state geometry are interchangeable, and the optimum M should be small. The data confirms this directly. On Table 21, $M = 50$ matches the baseline average accuracy (0.9091 versus 0.9081) at 4.41 GB, and $M = 75$ slightly improves on it (0.9115). On the harder-metric tasks in Table 22, $M = 50$ already matches the baseline average (0.7598 versus 0.7375), and $M = 100$ produces the best overall result (0.7750) at 3.57 GB—a 0.23 GB saving and a 0.0375 improvement. The striking observation is that $M = 25$–50 is sufficient across all eight GLUE tasks despite their varying difficulty, directly reflecting the bidirectional equal-capacity prediction.

J.7 Gradient Checkpointing

Gradient checkpointing (GC) [6] and the Token Hypothesis reduce activation memory through distinct mechanisms. GC discards intermediate activations during the forward pass and recomputes them

during the backward pass, reducing the memory cost of every position at the expense of additional computation. The Token Hypothesis instead restricts the set of positions that participate in the backward pass, so that activations of the $T - M$ frozen positions are never retained. The two techniques are therefore complementary and can be applied jointly. To quantify their combined effect, we fine-tune LLaMA-3-8B on LongAlpaca-12k with sequence length T=8192 on 8×A100-80GB GPUs, enabling GC in every configuration, including the full-sequence baseline. We evaluate both full fine-tuning and LoRA and report total GPU memory, training time, and perplexity in Table 23.

Under full fine-tuning, GC alone reduces total memory from 292.8 GB to 193.1 GB relative to training without checkpointing. Applying the Token Hypothesis on top of GC yields a further reduction that GC cannot achieve by itself: at M=2500, memory decreases to 182.9 GB while perplexity improves slightly (8.103 versus 8.143). The combined configuration is also faster than GC alone (338.7 versus 372.4 min), since the backward computation through per-token modules is confined to the M trainable positions. Smaller budgets continue to reduce memory and time, reaching 178.3 GB and 317.2 min at M=500, but at a cost of approximately one point of perplexity. Under LoRA, where the baseline perplexity is higher owing to the reduced capacity of the adapter, the Token Hypothesis at M=2500 reduces memory by 6.9 GB with a perplexity increase of 0.056, and smaller values of M trade additional perplexity for further savings. With the exception of M=4000 under LoRA, every configuration trains faster than its GC baseline. These results indicate that the savings of the Token Hypothesis are additive to those of gradient checkpointing, and that M should scale with the length of the supervised sequence in long-context generative settings.

Table 23: Composition of the Token Hypothesis with gradient checkpointing (LLaMA-3-8B, LongAlpaca-12k, T=8192, 8×A100-80GB). Gradient checkpointing is enabled in every row, including *All*. GPU is the sum of all GPU peak allocations (GB); Time is total training time (min); PPL is perplexity (lower is better). For reference, full fine-tuning without checkpointing requires 292.8 GB and 301.5 min at a PPL of 8.14. ↑ green = improvement over *All*, ↓ red = degradation. **Bold** = best per column.

Tokens	Full FT + GC			LoRA + GC		
	GPU	*Time*	*PPL*	*GPU*	*Time*	*PPL*
All	193.1	372.4	8.143	148.9	344.6	**9.177**
	178.3	**317.2**	9.169	**137.4**	**321.8**	10.158
500	↑14.8	↑55.2	↓1.026	↑11.5	↑22.8	↓0.981
	179.4	322.6	9.357	138.6	327.1	9.977
1000	↑13.7	↑49.8	↓1.214	↑10.3	↑17.5	↓0.800
	180.6	327.6	9.101	139.7	332.3	9.796
1500	↑12.5	↑44.8	↓0.958	↑9.2	↑12.3	↓0.619
	182.9	338.7	**8.103**	142.0	342.8	9.233
2500	↑10.2	↑33.7	↑0.040	↑6.9	↑1.8	↓0.056
	186.4	353.9	8.147	145.5	358.6	9.290
4000	↑6.7	↑18.5	↓0.004	↑3.4	↓14.0	↓0.113

J.8 Long-Context Instruction

The memory savings of the Token Hypothesis grow with sequence length, since the activations of the $T - M$ frozen positions are never retained. To evaluate the method in a long-context generative setting, we fine-tune LLaMA-3-8B on LongAlpaca-12k, a free-form instruction-following dataset, with sequence length T=8192 on 8×A100-80GB GPUs. The training loss is computed over the response tokens, and we report perplexity together with total GPU memory and training time in Table 24.

At M=5000 (M/T=0.61), the Token Hypothesis matches the full-sequence baseline, reaching a perplexity of 8.11 compared with 8.14, while reducing memory by 16.4 GB and training time by 14.8 min. Smaller budgets yield substantially larger savings: at M=2000, memory decreases by 68.4 GB (23%) with a perplexity increase of 0.47, and at M=500 memory and training time decrease by 28% and 24%, respectively, at a cost of approximately one point of perplexity. The reduction in memory is not linear in M: most of the saving is obtained between M=5000 and M=2000, after which the footprint approaches the floor set by model weights, optimizer states, and attention. Compared with the classification and short-answer tasks in the preceding sections, long-form generation requires a considerably larger M, consistent with the observation that the required budget scales with the length of the supervised span.

Table 24: Long-context instruction following (LLaMA-3-8B, LongAlpaca-12k, T=8192, 8×A100-80GB). GPU is the sum of all GPU peak allocations (GB); Time is total training time (min); PPL is perplexity (lower is better). ↑ green = improvement over *All*, ↓ red = degradation. **Bold** = best per column.

Tokens	*M/T*	*GPU*	*Time*	*PPL*
All	1.00	292.8	301.5	8.14
		211.0	**230.2**	9.12
500	0.06	↑81.8	↑71.3	↓0.98
		215.4	239.5	9.04
1000	0.12	↑77.4	↑62.0	↓0.90
		224.4	248.4	8.61
2000	0.24	↑68.4	↑53.1	↓0.47
		276.4	286.7	**8.11**
5000	0.61	↑16.4	↑14.8	↑0.03

J.9 Code Generation

We further evaluate the Token Hypothesis on open-ended code generation using TACO [31], which groups programming problems into five difficulty levels. We fine-tune LLaMA-3-8B and follow the evaluation protocol of the original benchmark, reporting Pass@1, Pass@10, and Pass@100 (%) for each difficulty level together with training time and total GPU memory (Table 25). Because TACO is a challenging benchmark, absolute scores are low across all configurations, and the comparison of interest is between values of M under an identical protocol.

At M=300, Pass@1 matches or exceeds the full-sequence baseline at every difficulty level, with the largest gains on the medium-hard (1.00 versus 0.50) and very hard (0.46 versus 0.36) problems. Pass@100 follows the same pattern and improves most markedly on medium-hard problems (12.55 versus 6.24). These results are obtained with 7.04 GB less memory and a 24% reduction in training time. M=500 performs comparably and achieves the best scores on the hard and very hard levels. In contrast, M=100 is insufficient, particularly on the easy and medium levels, where Pass@100 decreases by 4.29 and 7.43 points, respectively. This behavior is consistent with the long-context results in Appendix J.8: code solutions are long, and the required budget scales with the length of the supervised output.

Table 25: Open-ended code generation on TACO (LLaMA-3-8B). Metrics: Pass@1, Pass@10, and Pass@100 (%) per difficulty level; GPU is total GPU memory (GB); Time is total training time (min). ↑ green = improvement over *All*, ↓ red = degradation. **Bold** = best per column.

Tokens	Easy			Medium			Medium-hard			Hard			Very hard			*Time*	*GPU*
	P@1	*P@10*	*P@100*	*P@1*	*P@10*	*P@100*	*P@1*	*P@10*	*P@100*	*P@1*	*P@10*	*P@100*	*P@1*	*P@10*	*P@100*		
All	9.50	15.44	20.38	**2.00**	**6.96**	**14.77**	0.50	2.44	6.24	0.50	3.11	8.86	0.36	1.02	2.23	79.4	95.07
	7.50	12.19	16.09	1.00	3.48	7.34	0.50	2.44	6.28	0.47	2.94	8.23	0.50	2.92	6.25	**52.9**	**85.00**
100	↓2.00	↓3.25	↓4.29	↓1.00	↓3.48	↓7.43	↑0.00	↑0.00	↑0.04	↓0.03	↓0.17	↓0.63	↑0.14	↑1.90	↑4.02	↑26.5	↑10.07
	9.53	**15.49**	**20.44**	**2.00**	**6.96**	14.71	**1.00**	**4.88**	**12.55**	0.50	3.11	8.89	0.46	2.68	5.75	60.1	88.03
300	↑0.03	↑0.05	↑0.06	↑0.00	↑0.00	↓0.06	↑0.50	↑2.44	↑6.31	↑0.00	↑0.00	↑0.03	↑0.10	↑1.66	↑3.52	↑19.3	↑7.04
	9.52	15.48	20.42	1.89	6.58	13.91	0.52	2.54	6.53	**0.51**	**3.18**	**9.06**	**0.51**	**2.98**	**6.38**	69.0	91.15
500	↑0.02	↑0.04	↑0.04	↓0.11	↓0.38	↓0.86	↑0.02	↑0.10	↑0.29	↑0.01	↑0.07	↑0.20	↑0.15	↑1.96	↑4.15	↑10.4	↑3.92

J.10 Compatibility with FlashAttention and QLoRA

The Token Hypothesis is implemented as a wrapper around per-token modules and leaves the attention operation unchanged, so it is compatible with any attention backend. All main experiments use the default scaled dot-product attention (SDPA) backend of Hugging Face Transformers, which already employs memory-efficient attention kernels; the reported baselines therefore include these savings. To verify compatibility with FlashAttention, we repeat the BoolQ experiment with `flash_attention_2` enabled for both the baseline and the Token Hypothesis (Table 26, left). At M=50, the Token Hypothesis matches the baseline accuracy (0.629) while reducing memory by 15.17 GB (27%) and training time by 25%, and M=100 achieves the highest accuracy (0.643). The two techniques address different components of memory: FlashAttention reduces the memory associated with attention scores, whereas the Token Hypothesis reduces the per-token activations stored for backpropagation.

We also combine the Token Hypothesis with QLoRA [10], which applies 4-bit quantization to the base model, on ARC-Challenge with LLaMA-3-8B under the same setup as the main experiments (Table 26, right). At M=100, accuracy slightly exceeds the baseline (0.688 versus 0.683) with 10.49 GB less memory, and M=300 achieves the same accuracy. Because quantization reduces the memory occupied by model weights, activations account for a larger share of the remaining footprint, which is precisely the component that the Token Hypothesis reduces.

Table 26: Compatibility with FlashAttention and QLoRA. *Left:* BoolQ with `flash_attention_2` enabled for all rows. *Right:* ARC-Challenge with QLoRA (LLaMA-3-8B). GPU is total GPU memory (GB); Time is total training time (min). ↑ green = improvement over *All*, ↓ red = degradation. **Bold** = best per column.

Tokens	FlashAttention-2 (BoolQ)		
	GPU	*Acc*	*Time*
All	56.67	0.629	7.99
	40.85	0.621	**5.81**
10	↑15.82	↓0.008	↑2.18
	41.50	0.629	5.97
50	↑15.17	↑0.000	↑2.02
	42.29	**0.643**	6.14
100	↑14.38	↑0.014	↑1.85
	44.46	0.623	6.97
300	↑12.21	↓0.006	↑1.02
	48.64	0.633	7.74
500	↑8.03	↑0.004	↑0.25

Tokens	QLoRA (ARC-Challenge)		
	GPU	*Acc*	*Time*
All	65.19	0.683	4.00
	53.65	0.619	**3.60**
10	↑11.54	↓0.064	↑0.40
	53.83	0.651	3.62
25	↑11.36	↓0.032	↑0.38
	54.12	0.653	3.63
50	↑11.07	↓0.030	↑0.37
	54.70	**0.688**	3.65
100	↑10.49	↑0.005	↑0.35
	57.02	**0.688**	3.72
300	↑8.17	↑0.005	↑0.28

J.11 Summary

Across six modalities, five architectures, and three training regimes, the picture is consistent. The optimal M varies by task and architecture—$M = 50$–100 for bidirectional GLUE and image classification, $M = 100$–300 for causal reasoning and visual QA, $M = 300$–700 for math and long-context video—but in every case the improvement or match to the baseline comes at a memory cost that is substantially lower than full-sequence fine-tuning. The exact savings depend on sequence length and architecture: for causal transformers with long video inputs the saving is on the order of 10–30 GB; for bidirectional models operating on short sequences the saving is modest in absolute terms but matches or exceeds the baseline on every task. These per-benchmark results collectively confirm that the optimal M^* predicted by Corollary 1 is achievable across real training runs, and that accuracy parity with full fine-tuning is attainable at significantly lower memory cost.

Algorithm 1 Wrapper-based Token Hypothesis (one per-token module)

Require: Input X, base module f (a per-token op: Linear, Embedding, LayerNorm, RMSNorm)
Require: Number of trainable tokens M, adapter delta $\Delta(\cdot)$
$T_f \leftarrow T - M$ ▷ *frozen segment length*
Split X along its sequence axis: $X_f \leftarrow X[:, :T_f],\ X_p \leftarrow X[:, T_f:]$
with `torch.no_grad()`:
  $Y_f \leftarrow f(X_f)$ ▷ *frozen forward, no activations retained*
$Y_p \leftarrow f(X_p) + \Delta(X_p)$ ▷ *base + adapter, full gradient flow*
return CONCAT$(Y_f, Y_p,$ seq axis$)$

Note: The wrapper replaces each per-token module in the model. Method specialization is in $\Delta(\cdot)$: identity for full-FT, and PEFT such as BA for LoRA, $\frac{m \cdot (W+BA)}{\|W+BA\|_{\text{col}}}$ for DoRA. Cross-token operations (attention scores, softmax) are not wrapped—they run on the rejoined full-length sequence with gradients enabled, leaving the $\mathcal{O}(B \cdot H \cdot T^2)$ attention-score term unaffected. This term can be recovered by composing with gradient checkpointing at $\sim$33% additional compute per step.

K Implementation Details

We implement the Token Hypothesis through two complementary strategies that differ in how aggressively they confine the frozen-token path to the no-gradient regime. In what follows, M denotes the number of trainable positions and $\mathcal{S}, \bar{\mathcal{S}}$ denote the trainable and frozen index sets, matching the notation introduced in Section 4. Within the algorithm pseudocode, $\mathbf{K}$ and $\mathbf{V}$ (boldface) refer to the attention keys and values, distinct from the Gram matrices K_t of our theoretical analysis.

The *wrapper-based* approach (Algorithm 1) replaces every per-token module in the model—linear projections, token embeddings, RMSNorm, and LayerNorm—with a thin wrapper that splits its input along the sequence dimension into trainable and frozen segments, runs the frozen segment under `torch.no_grad()`, runs the trainable segment with the method-specific delta function, and rejoins the outputs. The wrapper applies cleanly to any per-token operation because each output position depends only on the corresponding input position. This strategy is architecture-agnostic and supports a uniform interface across full fine-tuning, LoRA, LoRA-FA, DoRA, and QLoRA: each method is realized by changing only the delta $\Delta(\cdot)$ inside the wrapper. It preserves the original attention implementation, retaining compatibility with SDPA and FlashAttention. Cross-token operations such as attention scores cannot be split this way, however, so the dominant $\mathcal{O}(B \cdot H \cdot T^2)$ activation memory term remains on the gradient-bearing path.

The *deep dual-path* approach (Algorithm 2) instead splits the sequence once during data collation and maintains two parallel hidden-state streams throughout the entire forward pass. Following prior work on token-selective fine-tuning [51, 61], the collator selects M trainable positions per sample using one of three strategies—the last M tokens, M random positions plus the boundary tokens, or top-M by precomputed attention–gradient importance scores keyed on the input hash—and bakes the next-token label shift into the prefix labels at this stage, so the cross-entropy loss requires no further offset. Inside each decoder layer, the trainable stream's queries attend to keys and values concatenated from both streams, preserving the full attention context, while every frozen-stream intermediate—attention scores, softmax outputs, residuals, RMSNorm activations, MLP intermediates—is kept under `no_grad`. This eliminates the dominant attention-score memory term without recomputation, yielding substantially larger savings at small M. The trade-off is that dual-path attention does not compose with fused kernels and requires eager attention.

We use the deep dual-path approach for our main fine-tuning experiments on LLaMA-3-8B and LLaMA-3.2-3B, where attention-score memory dominates and the savings are largest, and use the wrapper-based approach for cross-method comparisons with LoRA, DoRA, and QLoRA and for vision-language backbones, where compatibility with fused-attention kernels and a uniform PEFT interface are more important than maximizing attention-memory savings. For our head-to-head comparisons with TokenTune [51] and TokenSeek [61] (Section 5), we follow the same training protocol as these prior works to ensure fair comparison: the deep dual-path implementation matches their two-stream design while the position-selection rule is determined by our theoretical results (last-M for causal, random for bidirectional) rather than per-sample heuristic scoring. All experiments use bf16 mixed precision and the AdamW optimizer with a cosine learning-rate schedule; multi-

Algorithm 2 Deep dual-path Token Hypothesis (collation and decoder layer)

Stage 1 — Data collation (per sample, once)

Require: Input ids $\mathbf{x} \in \mathbb{Z}^T$, labels $\mathbf{y} \in \mathbb{Z}^T$, target M, mode $\mathtt{m} \in \{\text{last, random, attention}\}$

select index set $\mathcal{S} \subset \{0, \dots, T-1\}$ with $|\mathcal{S}| = M$:
  if $\mathtt{m}$ = `last`: $\mathcal{S} = \{T-M, \dots, T-1\}$
  if $\mathtt{m}$ = `random`: $\mathcal{S} = \{0\} \cup \textsc{rand}(M-2) \cup \{T-1\}$ ▷ *boundary tokens fixed*
  if $\mathtt{m}$ = `attention`: $\mathcal{S} = \textsc{topk}_M\big(\textsc{importance}\big[\textsc{hash}(\mathbf{x})\big]\big)$ ▷ *precomputed token-importance map*
$\bar{\mathcal{S}} \leftarrow \{0, \dots, T-1\} \setminus \mathcal{S}$
$\mathbf{x}^{\mathcal{S}} \leftarrow \mathbf{x}[\mathcal{S}], \quad \mathbf{x}^{\bar{\mathcal{S}}} \leftarrow \mathbf{x}[\bar{\mathcal{S}}]$
$\mathbf{y}^{\mathcal{S}} \leftarrow \mathbf{y}[\mathcal{S}+1]$ ▷ *shift baked in: position $t \in \mathcal{S}$'s logit predicts $\mathbf{y}_{t+1}$*
return $(\mathbf{x}^{\mathcal{S}}, \mathbf{x}^{\bar{\mathcal{S}}}, \mathbf{y}^{\mathcal{S}}, \text{positions})$

Stage 2 — Decoder layer forward

Require: Frozen stream $H_f \in \mathbb{R}^{B\times(T-M)\times d}$, trainable stream $H_p \in \mathbb{R}^{B\times M\times d}$
Require: Position ids P_f, P_p, attention masks M_f, M_p

with `torch.no_grad()`:
  $\tilde{H}_f \leftarrow \textsc{RMSNorm}(H_f)$
  $\mathbf{Q}_f, \mathbf{K}_f, \mathbf{V}_f \leftarrow W_q\tilde{H}_f, W_k\tilde{H}_f, W_v\tilde{H}_f$
  $\mathbf{Q}_f, \mathbf{K}_f \leftarrow \textsc{RoPE}(\mathbf{Q}_f, \mathbf{K}_f, P_f)$
$\tilde{H}_p \leftarrow \textsc{RMSNorm}(H_p)$
$\mathbf{Q}_p, \mathbf{K}_p, \mathbf{V}_p \leftarrow W_q\tilde{H}_p, W_k\tilde{H}_p, W_v\tilde{H}_p$
$\mathbf{Q}_p, \mathbf{K}_p \leftarrow \textsc{RoPE}(\mathbf{Q}_p, \mathbf{K}_p, P_p)$
$\mathbf{K} \leftarrow [\mathbf{K}_f \mid \mathbf{K}_p], \quad \mathbf{V} \leftarrow [\mathbf{V}_f \mid \mathbf{V}_p]$ ▷ *shared $\mathbf{K}/\mathbf{V}$ across both streams*
with `torch.no_grad()`:
  $\mathbf{A}_f \leftarrow \textsc{softmax}\big(\mathbf{Q}_f\mathbf{K}^\top/\sqrt{d} + M_f\big)$
  $H_f \leftarrow H_f + W_o\,(\mathbf{A}_f\mathbf{V})$
  $H_f \leftarrow H_f + \text{MLP}(\textsc{RMSNorm}(H_f))$
$\mathbf{A}_p \leftarrow \textsc{softmax}\big(\mathbf{Q}_p\mathbf{K}^\top/\sqrt{d} + M_p\big)$
$H_p \leftarrow H_p + W_o\,(\mathbf{A}_p\mathbf{V})$
$H_p \leftarrow H_p + \text{MLP}(\textsc{RMSNorm}(H_p))$
return (H_f, H_p)

Note: The trainable queries $\mathbf{Q}_p$ attend to $\mathbf{K}, \mathbf{V}$ from *both* streams, preserving the full attention context. All frozen-stream tensors—$\mathbf{A}_f$, softmax outputs, residuals, RMSNorm activations, MLP intermediates—are kept under `no_grad`, eliminating the dominant $\mathcal{O}(B \cdot H \cdot T^2)$ memory term without recomputation. Because the label shift in Stage 1 is baked into $\mathbf{y}^{\mathcal{S}}$, the loss is a plain cross-entropy on $H_p \to \mathbf{y}^{\mathcal{S}}$ at the LM head with no further offset.

GPU configurations shard the model across devices using HuggingFace `device_map="auto"`, as described in Appendix H.

L Limitations

Our analysis has three main limitations. First, the framework relies on the lazy NTK regime, which requires parameter updates to remain close to initialization. This holds for fine-tuning strong pretrained models—we measure mean weight drift below 0.011 across every parameter group of LLaMA-3-8B (Appendix E.4)—but is expected to break under large representational shifts, such as adapting a model to a substantially new domain, where the linearization around θ_0 no longer captures training dynamics and the per-position capacity bound need not control the achievable predictor. Second, all of our theoretical and empirical results assume a strong pretrained initialization; the behavior of token-wise capacity during training from scratch is not analyzed in this work and is not implied by our results. Third, our analysis is restricted to transformer architectures with standard attention, both bidirectional and causal. Other architectural families—state-space models, mixture-of-experts, and recurrent or convolutional sequence models—fall outside the scope of Theorems 2 and 3, whose proofs rely on properties of attention-mediated information flow, and would require a separate analysis.

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: The abstract's claims map directly to Theorems 1–3 (Section 3, proofs in Appendices A–C) and Corollary 1 (Section 4, proof in Appendix D), and to the empirical results across six task families and five backbones in Section 5 and Appendix J. The underlying assumptions (lazy NTK regime, adjacent-state contraction) are stated in Section 3 and discussed as limitations in Appendix L.

2. **Limitations**

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

   Answer: [Yes]

   Justification: Limitations are discussed in Appendix L: the analysis assumes the lazy NTK regime (validated by mean weight drift below 0.011 in Appendix E.4) and a strong pretrained initialization, and is restricted to transformer architectures with standard bidirectional or causal attention.

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: Theorems 1–3 and Corollary 1 are stated in Section 3 with all assumptions explicit, and complete proofs are given in Appendices A.1, B.1, C.1, and D respectively.

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: Datasets and preprocessing are described in Appendix G, backbones in Appendix I, and full hyperparameters (optimizer, learning rate, batch size, sequence length, M sweep, hardware) in Appendix H. The token-masking procedure is specified in Section 4 and Appendix K (Algorithms 1 and 2).

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: Code is publicly available at https://github.com/Kowsher/Token-Hypothesis. All datasets used are publicly available and described in Appendix G; preprocessing, hyperparameters, and hardware are specified in Appendix H.

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: Appendix H specifies the optimizer (AdamW), learning rates, batch sizes, gradient accumulation, schedule, mixed precision, epochs, GPU configuration, and the M sweep for every (modality, regime) pair. Data splits and preprocessing are described in Appendix G.

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: The token-importance ablation (Appendix B.3, Figures 22–23) reports mean accuracy with 1σ error bars over 10 random seeds. The prediction-gap experiment (Appendix B.2, Table 4) reports p-values from a left-tail position-exchangeability permutation test with B=2000 resamples. Main fine-tuning tables report accuracies averaged over 5 random seeds, limited by the compute cost of large-scale fine-tuning across six modalities and five backbones.

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: Appendix H specifies the hardware for every experimental setting. Most experiments are conducted on one, two, or four NVIDIA H100 GPUs (80 GB), depending on the backbone and fine-tuning regime, as listed per setting in Table 9. The long-context and gradient-checkpointing experiments use eight NVIDIA A100 GPUs (80 GB). Peak GPU memory and wall-clock training time are reported for every (modality, regime, M) configuration in Table 1 and Tables 11–25.

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: The research conforms with the NeurIPS Code of Ethics. All experiments use publicly available datasets and pretrained models, and the work involves no human subjects or personally identifiable data.

10. **Broader impacts**

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

    Answer: [N/A]

    Justification: The paper proposes a memory-efficient fine-tuning procedure for transformers. It is methodological foundational research with no direct path to specific negative societal applications beyond those of fine-tuning in general.

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: The paper does not release any new pretrained models, generative models, or scraped datasets. It proposes a fine-tuning method evaluated on existing public benchmarks.

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: All datasets and pretrained models used are publicly available and cited with their original sources in Appendices G and I (LLaMA-3, Qwen3-VL, LLaVA-OneVision, RoBERTa, SmolLM2, ViT, GLUE, ARC, BoolQ, HellaSwag, OpenBookQA, WinoGrande, GSM8K, and others). They are used in accordance with their respective licenses and intended research use.

13. **New assets**

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

    Answer: [N/A]

    Justification: The paper does not release new datasets or models. The contribution is a fine-tuning procedure evaluated on existing public benchmarks.

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: The paper does not involve crowdsourcing or research with human subjects.

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: The paper does not involve research with human subjects.

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: [N/A]

    Justification: Not relevant