FREEING ACTIVATION MEMORY FOR TASK ADAPTATION OF PATHOLOGY FOUNDATION MODELS: LAYER-SELECTIVE FREEZING AND PARTIAL LOW-RANK ADAPTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Task Adaptation of Pathology Foundation Models (TAPFM, Kumar & Vanderbilt, 2025) showed that a vision-transformer (ViT) pathology foundation model (PFM) and an attention-based multiple instance learning (MIL) aggregator can be jointly fine-tuned on a single GPU by detaching gradients between two separate computational graphs. This makes end-to-end adaptation practical without the multi-GPU clusters used by prior work, but the number of tiles that can be sampled per whole-slide image (WSI) per training step remains small (e.g., 100 tiles for GigaPath), because activation memory for backpropagation through a giant ViT scales with both the number of tiles and the number of transformer blocks. Motivated by the observation in TAPFM that task-adaptation gradients concentrate in the final transformer blocks with negligible updates to earlier layers, we translate this empirical finding into an actionable memory-saving mechanism. We introduce and compare three single-GPU strategies for GigaPath: (i) low-rank adaptation (LoRA) applied to all transformer blocks, which reduces gradient and optimizer-state memory; (ii) freezing all but the last N transformer blocks, which additionally removes the frozen prefix from the autograd graph and eliminates its activation memory; and (iii) a hybrid that freezes all but the last N blocks and applies LoRA only within those unfrozen blocks, combining both memory savings. Empirically, these strategies raise the maximum tiles-per-WSI-per-epoch achievable on a single H200 from TAPFM's original 100 to 300 (LoRA-only), 500 (freeze-last-N only), and 1000 (freeze-last-N + LoRA) with $N \in \{9, 11\}$, respectively. We provide a memory-decomposition analysis that explains why LoRA alone does not reduce activation memory while layer freezing does, formalize the hybrid strategy as a generalization of TAPFM's detached dual-gradient update restricted to a block subset, and show across institutional and external cohorts that the proposed configurations retain strong FGFR3 and EGFR mutation-prediction performance while offering task-dependent trade-offs between predictive accuracy and resource efficiency. Our results suggest that removing activation-memory bottlenecks is a viable path toward more scalable and parameter-efficient single-GPU adaptation of large PFMs.

1 INTRODUCTION

Whole slide images (WSIs) are gigapixel objects that cannot be processed directly by a neural network. The standard multiple instance learning (MIL) formulation instead represents a WSI as a bag of K tiles, extracts a per-tile embedding with a pretrained encoder, and aggregates the tile embeddings into a slide-level representation for a weakly supervised prediction task such as gene-mutation status (Ilse et al., 2018; Campanella et al., 2024). The recent generation of pathology foundation models (PFMs), e.g., UNI (Chen et al., 2024a), GigaPath (Xu et al., 2024), Virchow/Virchow2 (Vorontsov et al., 2024), and H-Optimus-0 (Bioptimus, 2024), replace hand-designed or lightly supervised tile encoders with ViTs pretrained by self-supervision on hundreds of thousands to millions of WSIs, and consistently outperform earlier encoders as *fixed* feature extractors (Kumar & Vanderbilt, 2025). Because these PFMs are not trained for any specific downstream clinical task, adapting

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

their parameters rather than only training a MIL aggregator on top of frozen features has been shown to improve downstream performance (Campanella et al., 2025; Li et al., 2023), but full end-to-end adaptation of a billion-parameter ViT jointly with a MIL aggregator is memory-hungry: activation memory for a single tile scales with the number of transformer blocks and the square of the number of tokens per tile, and this cost is paid once for every tile sampled from every WSI in a batch.

TAPFM (Kumar & Vanderbilt, 2025) addressed the resulting optimization instability and memory pressure with two ideas: (1) using the ViT's own class-token attention weights as the MIL aggregation mechanism, avoiding a separately parameterized aggregator that must be reconciled with the encoder during backpropagation, and (2) *detaching* the aggregator's computational graph from the PFM's computational graph and defining a dedicated task-adaptation loss for the PFM from the aggregator's (detached) gradients with respect to tile features and attention weights. This lets TAPFM jointly adapt a PFM and MIL aggregator on a single 80GB H100 GPU, but the number of tiles processed per WSI per epoch is still capped at 100 (GigaPath), 300 (UNI), or 75 (H-Optimus-0) — "the maximum number of tiles processable for each PFM on a single GPU while maintaining end-to-end fine-tuning capability" (Kumar & Vanderbilt, 2025). Because random tile sampling with a small, resampled-each-epoch budget is known to leave information on the table relative to using all diagnostically relevant tiles (Keshvarikhojasteh et al., 2024), and because fully end-to-end fine-tuning with more tiles has been shown to outperform frozen-feature MIL (Campanella et al., 2024), increasing the tile budget without leaving the single-GPU regime is a practically important open problem that TAPFM itself identifies as future work: "investigations into which specific transformer layers benefit most from task adaptation could optimize the approach by selectively updating only those parameters" (Kumar & Vanderbilt, 2025).

In this paper we pursue exactly that direction, motivated by an empirical observation from applying TAPFM to GigaPath: the gradient signal delivered by the task-adaptation loss $\mathcal{L}_{PFM}$ (Equation 3) is concentrated in the last several transformer blocks, while the earliest blocks receive comparatively little effective update over the course of training. This is consistent with a broader literature showing that self-supervised ViT representations are hierarchical, with early blocks encoding generic, low-level structure and late blocks encoding task- and semantics-relevant structure (Dosovitskiy et al., 2021), as well as with the "surgical fine-tuning" finding that tuning only the last layers of a network matches or exceeds full fine-tuning when distribution shifts primarily affect semantic or output-level features rather than input-level features (Lee et al., 2023). We show that this observation can be converted directly into GPU memory savings, *if the freezing is implemented correctly*: freezing only the block-level parameters (leaving `patch_embed`, the class token, and positional embeddings trainable) does not save activation memory, because autograd must still retain the backward graph for every block downstream of a trainable upstream parameter. Freezing the entire prefix of the network ensures that `requires_grad=False` propagates cleanly from the first operation, which completely removes the frozen blocks from the backward graph and eliminates their activation memory (Pfeiffer et al., 2024; Zhang et al., 2024).

We combine this observation with low-rank adaptation (LoRA, Hu et al., 2022), which reduces a complementary memory term (gradients and optimizer state for the *trainable* weights) though does *not* by itself reduce activation memory, since the frozen base weights of every LoRA-augmented layer are still on the backward path (Zhang et al., 2023a; Pfeiffer et al., 2024). Figure 1 illustrates the three configurations.

Contributions.

1. A memory decomposition for TAPFM-style joint PFM+MIL adaptation that separates the *parameter/gradient/optimizer-state* memory term (addressed by LoRA) from the *activation* memory term (addressed only by removing frozen blocks from the backward graph), and an explanation, grounded in prior PEFT-memory analyses, of why the two are not redundant.
2. Two extensions of TAPFM's detached dual-gradient update, LoRA-TAPFM (Section 4.2) and Freeze-N-TAPFM (Section 4.3), together with a hybrid, Freeze+LoRA-TAPFM (Section 4.5), that restricts both the trainable-parameter set and the low-rank adapters to the last N blocks, formalized as a block-restricted special case of TAPFM's Task Adaptation Loss.
3. A principled, low-cost diagnostic for choosing N itself (Section 4.4), based on a per-block *representation-drift* score computed from a short calibration probe *before* committing to a full training run: we snapshot each block's representation on a held-out tile set, run a brief real

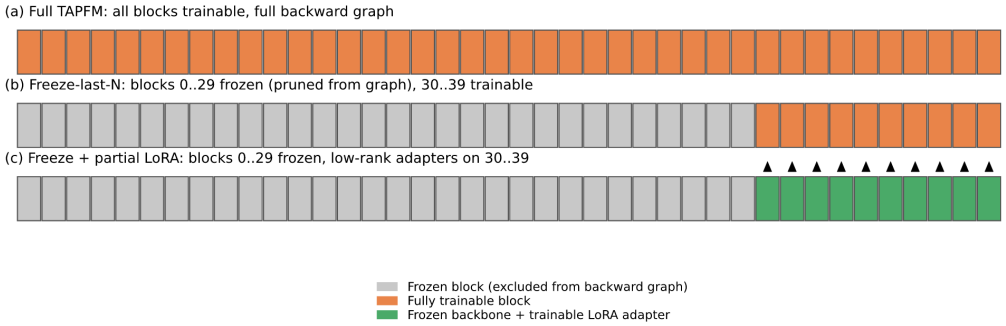

Figure 1: Schematic of the three configurations studied for the GigaPath tile encoder ($n = 40$ transformer blocks). Gray blocks are frozen *and* excluded from the autograd backward graph (no activations retained); orange blocks are fully trainable; green blocks are frozen at base-weight resolution but carry a trainable low-rank adapter (triangle marker). All three configurations use TAPFM's attention-based aggregation and detached dual-gradient update (Section 3); only the encoder's trainable-parameter set and backward-graph extent differ.

training probe, re-snapshot, and score each block by how much its representation changed (via linear Centered Kernel Alignment), then select N from the resulting cumulative-score "elbow" (Algorithm 2).

4. An empirical demonstration, on GigaPath, that the hybrid strategy enables full-dataset training with 1000 tiles/WSI on a single H200, supported by matched-budget GPU-memory measurements and BLCA-FGFR3 and LUAD-EGFR mutation-prediction evaluations on the Institute 1 validation cohort and independent external cohorts.

2 Related Work

PFM and MIL. Early WSI-analysis pipelines used ImageNet-pretrained or lightly self-supervised tile encoders with attention-based MIL aggregators such as ABMIL (Ilse et al., 2018) and its successors (CLAM, DSMIL, VarMIL, TransMIL); see Chen et al. (2024b) for a benchmark comparing aggregators on fixed pathology features. The current generation of ViT-based PFMs (Chen et al., 2024a; Xu et al., 2024; Bioptimus, 2024) are pretrained with DINOv2-style self-supervision on hundreds of thousands to millions of WSIs and are typically deployed as *frozen* feature extractors, with only a lightweight MIL aggregator trained per downstream task. Li et al. (2023) and Campanella et al. (2024) instead adapt the encoder itself, but the latter requires distributed training across multiple GPUs (up to 8 GPUs for 4900 tiles/slide, and 24 H100s for the deployed EGFR model of Campanella et al., 2025) to reach large tile budgets with full end-to-end backpropagation. TAPFM (Kumar & Vanderbilt, 2025) is, to our knowledge, the first method to jointly adapt a large PFM and a MIL aggregator on a *single* GPU, via a detached dual-gradient update; we build directly on this method and its own noted limitation of a small per-epoch tile budget.

Parameter-efficient fine-tuning. LoRA (Hu et al., 2022) freezes the pretrained weight matrix W_0 of a linear layer and injects a trainable low-rank update $\Delta W = BA$ with $B \in \mathbb{R}^{d\times r}$, $A \in \mathbb{R}^{r\times k}$, $r \ll \min(d, k)$, reducing trainable parameters and optimizer-state memory by orders of magnitude with no added inference latency. Variants include AdaLoRA (Zhang et al., 2023b), which allocates rank adaptively via an importance score and empirically assigns higher rank to deeper layers; QLoRA (Dettmers et al., 2023), which additionally quantizes the frozen base weights; DoRA (Liu et al., 2024), which decomposes weights into magnitude and direction and applies LoRA only to the latter; and VeRA (Kopiczko et al., 2024), which shares frozen random projections across layers and trains only per-layer scale vectors, including experiments on ViT image classification. BitFit (Ben-Zaken et al., 2022) fine-tunes only bias terms and is interpreted by its authors as "exposing knowledge induced by [pre-]training, rather than learning new task-specific knowledge." He et al. (2023) establish PEFT/LoRA benchmarks specifically for vision transformers. Within computational pathology, LoRA has recently been applied to Virchow2 for domain/continual adaptation, with rank-8, α=16 adapters on the query/value projections (Zhao et al., 2025; Guo et al., 2026); our

work differs in applying LoRA (and layer freezing) specifically to *raise the achievable tile budget* for joint encoder+aggregator MIL adaptation on a single GPU, rather than for continual pretraining or feature-space domain adaptation.

Layer-selective and surgical fine-tuning. Lee et al. (2023) show that fine-tuning only a contiguous block of layers can match or exceed full fine-tuning, and that the optimal block depends on the type of distribution shift: input-level shifts favor tuning early layers, feature-level shifts favor middle layers, and output-level/semantic shifts favor tuning the last layers. This is consistent with probing evidence that ViT representations move from generic, low-level structure in early blocks to semantic, task-relevant structure in later blocks (Dosovitskiy et al., 2021), and with the "stratified" fine-tuning heuristic of Zhu & Xu (2024). AdaLoRA's learned importance scores independently corroborate that deeper layers receive higher effective budget during adaptation (Zhang et al., 2023b). Gene-mutation prediction from WSIs is closer to an output-level/semantic task than an input-level one, which motivates our choice to freeze the earliest blocks and adapt only the last N. We select N using a representation-drift diagnostic based on Centered Kernel Alignment (Kornblith et al., 2019), related to SVCCA-based layer-convergence analysis (Raghu et al., 2017) and CKA studies of ViT layer specialization (Raghu et al., 2021) (Section 4.4).

3 Background: TAPFM

We briefly recap the components of TAPFM (Kumar & Vanderbilt, 2025) that our extensions build on.

Attention-based aggregation. For a WSI represented as a bag of K tiles $X = \{x_1, \ldots, x_K\}$, a ViT-based PFM f_θ maps each tile to a CLS-token feature $z_i \in \mathbb{R}^D$ and produces class-token attention weights $A^h_{\text{cls},j}$ over N tokens and H heads. TAPFM defines the per-tile attention weight as the attention-head- and token-averaged CLS attention,

$$a_i = \frac{1}{H}\sum_{h=1}^{H}\frac{1}{N}\sum_{j=1}^{N} A^h_{\text{cls},j}, \tag{1}$$

and uses these weights, after min-max scaling and softmax normalization, directly as the MIL aggregation weights: $\bar{Z} = Z^\top a = \sum_i a_i z_i$, avoiding a separately parameterized attention MIL head.

Detached dual-gradient optimization. Let $\mathcal{L}_{\text{agg}}$ be the weighted cross-entropy loss on the bag-level prediction $\hat{y} = \sigma(W\bar{Z} + b)$. TAPFM detaches Z and a from the PFM's computational graph before computing $\mathcal{L}_{\text{agg}}$, so that $\nabla_{\theta_{\text{agg}}}\mathcal{L}_{\text{agg}}$ updates only the aggregator. It then detaches the resulting feature gradients $g_{z_i} = \partial\mathcal{L}_{\text{agg}}/\partial z_i$ and attention gradients $g_{a_i} = \partial\mathcal{L}_{\text{agg}}/\partial a_i$ from the aggregator's graph and defines a Task Adaptation Loss (TAL) for the PFM:

$$\mathcal{L}_{\text{feature}} = -\text{tr}(ZG_z^\top) = \sum_i \langle z_i, g_{z_i}\rangle, \qquad \mathcal{L}_{\text{attention}} = a^\top g_a = \sum_i a_i g_{a_i}, \tag{2}$$

$$\mathcal{L}_{PFM} = \mathcal{L}_{\text{feature}} + \lambda\,\mathcal{L}_{\text{attention}}. \tag{3}$$

This decoupling into two separate computational graphs, each with its own backward pass, is what allows TAPFM to jointly train θ_{agg} and θ_{PFM} on a single GPU without the instability of a unified graph (TAPFM's Proposition 1). Both of our extensions (Sections 4.2–4.5) leave this detached dual-gradient mechanism entirely unchanged; they only change *which parameters of* θ_{PFM} *are updated by* $\nabla_{\theta_{PFM}}\mathcal{L}_{PFM}$, and, in the LoRA variants, *how* those parameters are represented.

Memory scaling. TAPFM's own complexity analysis (its Theorem 3) gives the space complexity of processing a bag of K tiles as $O(|\theta_{PFM}| + K \cdot (S_{\text{act}} + S_{\text{grad}}))$, where S_{act} and S_{grad} are the per-tile activation and gradient memory of the ViT. Because S_{act} scales with the number of transformer blocks L and quadratically with the number of tokens per tile N ($S_{\text{act}} = O(L \cdot N \cdot D + L \cdot H \cdot N^2)$), and because this cost is paid for *every* sampled tile, S_{act} — not the PFM's total parameter count $|\theta_{PFM}|$ — is the term that determines the maximum tile budget K on a fixed-memory GPU once $|\theta_{PFM}|$ itself is small relative to available memory. Our contribution is precisely to reduce the effective L in this term.

4 METHOD

We study three modifications to `setup_encoder` and the PFM backward pass of TAPFM, all of which leave the aggregator, the detached dual-gradient mechanism, and the TAL (Equation 3) unchanged.

4.1 MEMORY DECOMPOSITION

Following Zhang et al. (2023a); Pfeiffer et al. (2024); Zhang et al. (2024), we decompose the per-tile GPU memory cost of backpropagating through the PFM encoder into three terms:

$$S_{\text{tile}} = \underbrace{S_{\text{param}}}_{\text{model weights (shared across tiles)}} + \underbrace{S_{\text{opt+grad}}(\theta_{\text{train}})}_{\text{scales with \#trainable params}} + \underbrace{S_{\text{act}}(L_{\text{train-onward}})}_{\text{scales with \#blocks on the backward path}} . \quad (4)$$

S_{param} is fixed by the choice of PFM and is amortized across all sampled tiles. $S_{\text{opt+grad}}$ scales with the number of *trainable* parameters $|\theta_{\text{train}}|$: AdamW stores two optimizer moments plus a gradient per trainable parameter, so this term is $\propto |\theta_{\text{train}}|$ regardless of which blocks those parameters live in. $S_{\text{act}}(L_{\text{train-onward}})$ is the activation memory that must be retained *per tile* for the backward pass, and — critically — it depends not on how many parameters are trainable, but on how many transformer blocks lie *at or after* the first block containing a trainable parameter, because autograd must retain activations for every block on the path from that first trainable block to the loss (Zhang et al., 2024). This decomposition immediately implies:

- **LoRA reduces $S_{\text{opt+grad}}$ but not S_{act}**, because LoRA adapters are typically inserted into every block (or every attention block), so $L_{\text{train-onward}} = L$ is unchanged even though $|\theta_{\text{train}}| \ll |\theta_{PFM}|$.
- **Freezing the first $L - N$ blocks reduces S_{act} but not $S_{\text{opt+grad}}$ per remaining trainable parameter**, because it reduces $L_{\text{train-onward}}$ from L to N, but the last N blocks are still fully trainable (dense weight updates).
- **Freezing the first $L - N$ blocks *and* applying LoRA only within the last N** reduces both terms simultaneously: $L_{\text{train-onward}} = N$ (activation saving) and $|\theta_{\text{train}}| = |\theta_{\text{LoRA}}| \ll |\theta_{\text{last-}N}|$ (gradient/optimizer saving), which is why this configuration achieves the largest tile budget in our experiments (Section 5).

The same decomposition applies to training *time*: because forward propagation always traverses all L blocks regardless of freezing, only the backward pass shrinks with N, so memory savings from freezing are substantially larger than time savings (Appendix B).

4.2 LORA-TAPFM: LOW-RANK ADAPTATION OF ALL ENCODER BLOCKS

We insert LoRA adapters $\Delta W = BA$ (rank r, scaling α) into the query, key and value projections of every self-attention block of the GigaPath tile encoder, following the standard recipe used elsewhere in pathology PFM adaptation (Zhao et al., 2025; Guo et al., 2026), and freeze all original weight matrices W_0. The PFM's trainable parameter set becomes $\theta_{\text{train}} = \{B_\ell, A_\ell\}_{\ell=1}^{L}$, i.e., $O(L \cdot r \cdot d)$ parameters instead of $O(L \cdot d^2)$. TAPFM's TAL backward pass (Equation 3) is otherwise unchanged: $\nabla_{\theta_{PFM}} \mathcal{L}_{PFM}$ now only has nonzero components for $\{B_\ell, A_\ell\}$. Because every block still contains at least one trainable parameter, $L_{\text{train-onward}} = L$: this configuration is expected (Section 4.1) to increase the tile budget through the $S_{\text{opt+grad}}$ term alone, and indeed raises GigaPath's tile budget from 100 to 300 tiles/WSI.

4.3 FREEZE-N-TAPFM: LAYER-SELECTIVE FREEZING

We first freeze *every* parameter of the GigaPath tile encoder, including `patch_embed`, the class token, the positional embedding, and all L=40 transformer blocks. We then selectively set `requires_grad=True` for only the last N blocks and the final LayerNorm before constructing the optimizer (Algorithm 1). This ordering matters at the autograd level, not just at the optimizer level: if upstream modules such as `patch_embed` are left trainable, autograd must still build and retain the backward graph, and hence the forward activations, for every block downstream of them, including the "frozen" early blocks, because an operation's output requires a gradient if *any* of its

Algorithm 1 FREEZE-N encoder setup (modifies TAPFM's `setup_encoder`)

Input: pretrained PFM $f_{\theta_{PFM}}$ with L transformer blocks, number of trainable blocks $N \leq L$
for each parameter $p \in \theta_{PFM}$ **do**
  p.`requires_grad` $\leftarrow$ `False`
end for
$j_{\min} \leftarrow L - N$ {index of first trainable block}
for $\ell = j_{\min}$ **to** $L - 1$ **do**
  for each parameter $p \in \text{block}_\ell$ **do**
    p.`requires_grad` $\leftarrow$ `True`
  end for
end for
for each parameter $p \in$ `norm` **do**
  p.`requires_grad` $\leftarrow$ `True`
end for
Build optimizer over $\{p : p$.`requires_grad` $=$ `True`$\}$ only
(Downstream: TAPFM's `encoder_backward` and TAL are unchanged; frozen blocks' gradients are `None` and their optimizer step is a no-op, but critically they were never materialized in the backward graph in the first place.)

inputs or parameters does. Freezing top-to-bottom first ensures `requires_grad=False` propagates cleanly from the very first operation, so the frozen prefix is forward-only and its activations can be released immediately rather than retained for backward. This is the mechanism identified by Pfeiffer et al. (2024) and Zhang et al. (2024) as the source of activation-memory savings from freezing, and is what we verified empirically raises GigaPath's tile budget from 100 (full fine-tuning) to 500 tiles/WSI at N=10.

4.4 CHOOSING N: A REPRESENTATION-DRIFT (CKA) DIAGNOSTIC

Rather than choosing N by trial and error, we estimate it from a short calibration probe. We measure *realized* representation change throughout the PFM. For each block ℓ, let X_ℓ, Y_ℓ be its pooled output on a held-out tile set before and after a short real AdamW probe (encoder fully unfrozen; full procedure in Algorithm 2, Appendix C), and define the drift score via linear CKA (Kornblith et al., 2019):

$$D_\ell = 1 - \frac{\|Y_\ell^\top X_\ell\|_F^2}{\|X_\ell^\top X_\ell\|_F \|Y_\ell^\top Y_\ell\|_F} \in [0, 1]. \tag{5}$$

Normalizing, $\tilde{s}_\ell = D_\ell / \sum_k D_k$, we select the smallest N capturing a target cumulative fraction: $N^\star = \min\{N : \sum_{\ell=L-N}^{L-1} \tilde{s}_\ell \geq \tau\}$.

4.5 FREEZE+LORA-TAPFM: THE HYBRID STRATEGY

Our hybrid strategy composes Sections 4.2 and 4.3, with N set either empirically or via the diagnostic of Section 4.4: we apply Algorithm 1 with trainable-block set $\{L-N, \ldots, L-1\}$, but instead of making those N blocks fully dense-trainable, we replace their query, key and value projections with frozen base weight + trainable low-rank adapter, exactly as in Section 4.2 but restricted to the last N blocks. All blocks before $L - N$ remain fully frozen and, as in Algorithm 1, are excluded from the backward graph entirely. This is the configuration referred to in Section 4.1 as reducing both S_{act} (via $L_{\text{train-onward}} = N$) and $S_{\text{opt+grad}}$ (via $|\theta_{\text{train}}| = |\theta_{\text{LoRA}}|$), and empirically achieves the largest tile budget of the four configurations we study: 1000 tiles/WSI for GigaPath on a single 141GB H200. We note that partial/subset LoRA restricted to a network's deepest layers has independently been found competitive with or superior to all-layer LoRA in other domains, which is consistent with, though not identical to, our motivation of freeing activation memory in the frozen prefix.

5 EXPERIMENTAL SETUP

Foundation model, hardware, and baseline. We use GigaPath (Xu et al., 2024) as the tile encoder (L=40 transformer blocks, patch/tile size $224 \times 224 \times 3$) and a single NVIDIA H200 (141GB)

GPU. We use the released Prov-GigaPath tile encoder, a DINOv2-style Vision Transformer with $L = 40$ transformer blocks and hidden dimension $D = 1536$. The encoder uses 24 attention heads, giving a per-head dimension of 64. The baseline is TAPFM's original detached dual-gradient update with full fine-tuning of all L blocks, which we reproduce at its reported 100-tiles/WSI budget.

Datasets and tasks. We study binary mutation prediction in two institutional whole-slide image cohorts: FGFR3 in bladder urothelial carcinoma (BLCA) and EGFR in lung adenocarcinoma (LUAD). We curate data from three independent sources: Institute 1, The Cancer Genome Atlas (TCGA), and Institute 2. Institute 1 data are split, patient-disjoint, into training and validation cohorts: the training set is used for model training and the validation set is used for model selection (checkpoint with the best validation AUC). TCGA and Institute 2 data are never used for training or model selection; they serve solely as external evaluation cohorts. Table 1 summarizes sample counts and mutation-positive prevalence for each cohort and role.

Table 1: Cohort sample counts by role and mutation-positive prevalence in parenthesis.

	Institute 1		TCGA (model evaluation)	Institute 2 (model evaluation)
Task	Training	Validation		
BLCA-FGFR3	1,551 (16%)	194 (16%)	260 (14%)	236 (5%)
LUAD-EGFR	7,865 (26%)	873 (26%)	463 (11.0%)	278 (35%)

Configurations compared. We compare four GigaPath configurations under an otherwise identical training recipe (AdamW (Loshchilov & Hutter, 2019), differential learning rates for encoder vs. aggregator, cosine annealing with warm restarts (Loshchilov & Hutter, 2017), mixed precision):

1. **TAPFM (full FT)** — all 40 blocks trainable; 100 tiles/WSI/epoch (baseline, reproduced from Kumar & Vanderbilt, 2025).
2. **LoRA-TAPFM** (Section 4.2) — LoRA rank r=8, all 40 blocks; 300 tiles/WSI/epoch.
3. **Freeze-N-TAPFM** (Section 4.3) — last N of 40 blocks trainable; 500 tiles/WSI/epoch.
4. **Freeze-N+LoRA-TAPFM** (Section 4.5) — last N of 40 blocks with LoRA rank r=8; 1000 tiles/WSI/epoch.

Metrics. We report (i) the maximum tiles/WSI/epoch achievable before out-of-memory, (ii) peak GPU memory at that tile budget, (iii) downstream task performance (AUC) on two institutional and external TCGA test sets.

6 Results

6.1 Layer-wise representation drift and the principled N-selection diagnostic

Figure 2 shows the measured diagnostic on GigaPath/BLCA: $\tilde{s}_\ell$ is small and flat for early blocks and rises smoothly over the last $\sim$15, giving $N^\star = 11$ at $\tau = 0.90$ and $N^\star = 14$ at $\tau = 0.95$ (Table 2). The corresponding diagnostic for GigaPath/LUAD is qualitatively similar and is shown in Figure 3 (Appendix D).

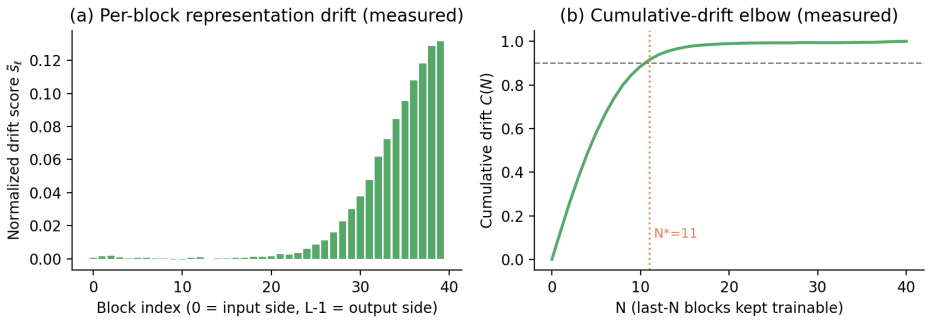

Figure 2: Measured representation-drift diagnostic for GigaPath/BLCA: (a) per-block drift $\tilde{s}_\ell$, (b) cumulative elbow, $N^\star$=11 at τ=0.90.

Table 2: $N^\star(\tau)$ from the representation-drift diagnostic vs. brute-force N_b, by cohort. Bold text indicates the values of N used for rest of the experiments in this paper.

	$N^\star$ at $\tau =$					
Cohort	0.70	0.80	0.85	0.90	0.95	N_b (brute-force)
BLCA-FGFR3	7	9	10	**11**	14	11
LUAD-EGFR	6	7	8	**9**	11	10

Table 2 reports the measured representation-drift diagnostic (Algorithm 2) for GigaPath on both cohorts, alongside each cohort's brute-force-optimal N_b. In both cases the diagnostic's recommendation tracks the brute-force optimum closely across the full τ-sweep, and $N^\star = 11$ vs. $N_\mathrm{b} = 11$ for BLCA-FGFR3 and is within one block of N_b at $\tau = 0.90$: $N^\star = 9$ vs. $N_\mathrm{best} = 10$ for LUAD-EGFR. We fix $\tau = 0.90$ for all remaining experiments in this paper, yielding $N^\star = 11$ for BLCA-FGFR3 and $N^\star = 9$ for LUAD-EGFR. For BLCA-FGFR3 this matches N_b exactly, so there is no AUC gap; for LUAD-EGFR, the one-block shortfall ($N^\star = 9$ vs. $N_\mathrm{b} = 10$) costs 0.0037 AUC, a negligible difference in validation AUC, a small price for replacing a full brute-force sweep with a single calibration probe at a fraction of cost. For external evaluation, we process all available tissue tiles extracted from each WSI.

6.2 Tile budget and memory

Table 3 reports the resource profile of each configuration at the tile budget used for downstream training. On the H200 GPU, full fine-tuning at $K_\mathrm{train} = 100$ used 73.43/75.64 GiB of peak allocated/reserved memory. All-block LoRA at $K_\mathrm{train} = 300$ used 126.27/130.55 GiB, while dense Freeze-N at $K_\mathrm{train} = 500$ used 73.47/76.23 GiB. Freeze-N+LoRA supported full-dataset training at $K_\mathrm{train} = 1000$ with 109.10/119.71 GiB of peak allocated/reserved memory.

At a matched budget of $K = 100$, all-block LoRA reduced peak allocated memory from 73.43 to 47.78 GiB (34.9%). At $K = 200$, it reduced allocated memory from 112.88 to 87.07 GiB (22.9%). At a matched $K = 500$, Freeze-N+LoRA reduced allocated memory from 73.47 to 57.20 GiB (22.2%) relative to Freeze-N, while reducing the number of trainable encoder parameters by 99.83%.

6.3 Downstream mutation-prediction performance

Table 4 shows that Freeze-N TAPFM provides the strongest overall predictive performance, while the LoRA-based configurations retain competitive performance with substantially fewer trainable parameters. All-block LoRA generalizes well across the completed external evaluations. The hybrid configuration does not consistently improve AUC, despite supporting the largest tile budget and the smallest trainable parameter count.

These results reveal a task-dependent trade-off between predictive performance and resource efficiency. Increasing the feasible tile budget is useful up to a saturation point, but maximizing the

Table 3: Resource profile of GigaPath configurations on a single H200 GPU. K_{train} denotes the tile budget used in the downstream experiments. Peak allocated/reserved GPU memory is reported at the largest capacity-test budget evaluated for each configuration. $N = 10$ in this table.

Configuration	Blocks trainable	LoRA	K_{train}	Peak alloc./reserv. (GiB)	Trainable params
TAPFM (full FT)	40/40	—	100	73.43 / 75.64	1,134,953,984 (100%)
LoRA-TAPFM	40/40 (adapters only)	✓	300	126.27 / 130.55	1,966,080 (0.173%)
Freeze-N-TAPFM	10/40	—	500	73.47 / 76.23	283,369,472 (24.97%)
Freeze-N+LoRA-TAPFM	10/40 (adapters only)	✓	**1000**	109.10 / 119.71	494,592 (0.0436%)

Table 4: Mutation-prediction AUC for GigaPath under each configuration.

Configuration	BLCA FGFR3			LUAD EGFR		
	Institute 1	TCGA	Institute 2	Institute 1	TCGA	Institute 2
GigaPath + ABMIL (fixed encoder, from Kumar & Vanderbilt, 2025)	0.8294	0.8712	0.7028	0.7711	0.8205	0.8130
GigaPath (TAPFM, full FT, 100 tiles, from Kumar & Vanderbilt, 2025)	0.8624	0.8881	0.7165	0.8376	0.8482	0.8384
GigaPath (LoRA-TAPFM, 300 tiles)	0.8389	**0.9333**	0.8752	0.8203	0.8329	0.8471
GigaPath (Freeze-N-TAPFM, 500 tiles)	**0.9128**	0.9290	**0.8831**	**0.8496**	**0.8417**	**0.8548**
GigaPath (Freeze-N+LoRA-TAPFM, 1000 tiles)	0.8691	0.9098	0.8522	0.8108	0.8364	0.8406

number of processed tiles is not itself the optimal objective. Instead, the adaptation strategy and tile budget should be selected jointly according to the downstream task and available compute.

7 Conclusion and Limitations

Conclusion. We extended TAPFM's single-GPU joint adaptation of PFM and MIL aggregators by identifying and exploiting a memory bottleneck that TAPFM's own detached dual-gradient formulation does not address: activation memory scales with the number of transformer blocks on the backward path, not with the number of trainable parameters. By formalizing this as a decomposition into parameter/optimizer memory (addressed by LoRA) and activation memory (addressed only by removing frozen blocks from the autograd graph), and by combining layer-selective freezing with partial LoRA restricted to the unfrozen blocks, we raised GigaPath's achievable tile budget on a single 141GB H200 to 1000 tiles/WSI while training only 494,592 encoder parameters (0.0436% of the encoder). Our downstream results show that the memory savings can be converted into substantially larger tile budgets while retaining strong mutation-prediction performance, with the most effective operating point depending on the task. We recommend Freeze-N when prioritizing predictive performance and Freeze-N+LoRA when prioritizing tile capacity and parameter efficiency. We believe this activation-memory-first perspective is broadly applicable beyond TAPFM to any setting that combines a large pretrained ViT encoder with a downstream aggregator or head trained end-to-end under a tight single-GPU memory budget.

Limitations and future work. External cohorts (TCGA, Institute 2) give stronger evidence of cross-institutional generalization than cross-validation alone (Section 5), but broader validation is needed across additional PFMs (e.g., UNI, Virchow2, H-Optimus-0), institutions, and tasks beyond mutation prediction, such as subtyping, prognosis, and treatment-response prediction. The representation-drift diagnostic (Section 4.4) was validated on two cohorts sharing one PFM; whether a single calibration probe transfers across PFMs without recomputation, and whether $\tau = 0.90$ generalizes as a default rather than requiring per-task tuning, should be tested. The complexity analysis (Appendix B) treats each block's cost as a uniform primitive C_{block}; validating the predicted time/memory trade-offs (Table 5) against wall-clock profiling, accounting for kernel-launch and I/O overhead, should be considered. LoRA is restricted to fixed-rank query/key/value adapters; combining freezing with adaptive-rank PEFT (AdaLoRA, Zhang et al., 2023b; DoRA, Liu et al., 2024) or orthogonal techniques such as gradient checkpointing (Chen et al., 2016) and multi-GPU sharding (Rajbhandari et al., 2020) may extend the achievable tile budget further. Finally, whether the activation-memory savings translate into clinical deployment benefit, including prospective validation analogous to EAGLE's silent trial (Campanella et al., 2025), remains an open question.

Reproducibility Statement

The code to reproduce all experiments reported in this paper is available at https://anonymous.4open.science/r/tapfm_freezN_LORA.

Institutional BLCA and LUAD patient data cannot be publicly released. TCGA-BLCA and TCGA-LUAD slide images are available through the NCI Genomic Data Commons. Hyperparameters and departures from the original TAPFM setup are specified in Section 5 and the released configurations.

Ethics Statement

This work uses de-identified institutional whole-slide images and the publicly available TCGA-BLCA/TCGA-LUAD cohorts, and the required institutional review board approvals are obtained wherever needed.

AI Disclosure

We used generative AI tools to improve the clarity of the manuscript.

References

Elad Ben-Zaken, Shauli Ravfogel, and Yoav Goldberg. BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Association for Computational Linguistics (ACL)*, 2022. arXiv:2106.10199.

Bioptimus. H-optimus-0: A foundational model for histopathology. https://github.com/bioptimus/releases/tree/main/models/h-optimus/v0, 2024.

Gabriele Campanella, Eugene Fluder, Jennifer Zeng, Chad Vanderbilt, and Thomas J. Fuchs. Beyond multiple instance learning: Full resolution all-in-memory end-to-end pathology slide modeling. *arXiv preprint arXiv:2403.04865*, 2024.

Gabriele Campanella, Neeraj Kumar, Swaraj Nanda, Siddharth Singi, Eugene Fluder, Ricky Kwan, Silke Muehlstedt, Nicole Pfarr, Peter J. Schüffler, Ida Häggström, et al. Real-world deployment of a fine-tuned pathology foundation model for lung cancer biomarker detection. *Nature Medicine*, 31:3002–3010, 2025.

Richard J. Chen, Tong Ding, Ming Y. Lu, Drew F. K. Williamson, Guillaume Jaume, Andrew H. Song, Bowen Chen, Andrew Zhang, Daniel Shao, Muhammad Shaban, et al. Towards a general-purpose foundation model for computational pathology. *Nature Medicine*, 30(3):850–862, 2024a.

Shengjia Chen, Gabriele Campanella, Abdulkadir Elmas, Aryeh Stock, Jennifer Zeng, Alexandros D. Polydorides, Adam J. Schoenfeld, Kuan-lin Huang, Jane Houldsworth, Chad Vanderbilt, and Thomas J. Fuchs. Benchmarking embedding aggregation methods in computational pathology: A clinical data perspective. In *MICCAI Workshop on Computational Pathology with Multimodal Data (COMPAYL)*, 2024b.

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

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. arXiv:2305.14314.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021. arXiv:2010.11929.

Zhengrui Guo, Zhengyu Zhang, Jiabo Ma, Yihui Wang, Fengtao Zhou, Yingxue Xu, Ling Liang, Chenglong Zhao, Qi Xie, Jinbang Li, Shujing Guo, Fangyi Han, Zhijian Cen, Ziyi Liu, Cheng Jin, Junlin Hou, Zhixuan Chen, Yu Cai, Lijuan Qu, Shifu Chen, Yueping Liu, Zhe Wang, Xiuming Zhang, Muyan Cai, Li Liang, and Hao Chen. A clinically validated foundation model for comprehensive lung pathology interpretation. *arXiv preprint arXiv:2605.25878*, 2026.

Xuehai He, Chunyuan Li, Pengchuan Zhang, Jianwei Yang, and Xin Eric Wang. Parameter-efficient model adaptation for vision transformers, 2023. arXiv:2203.16329.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022. arXiv:2106.09685.

Maximilian Ilse, Jakub Tomczak, and Max Welling. Attention-based deep multiple instance learning. *International Conference on Machine Learning (ICML)*, 2018.

Hassan Keshvarikhojasteh, Josien P. W. Pluim, and Mitko Veta. Multiple instance learning with random sampling for whole slide image classification. *SPIE Medical Imaging*, 2024. arXiv:2403.05351.

Dawid J. Kopiczko, Tijmen Blankevoort, and Yuki M. Asano. VeRA: Vector-based random matrix adaptation. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2310.11454.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International Conference on Machine Learning (ICML)*, 2019. arXiv:1905.00414.

Neeraj Kumar and Chad Vanderbilt. Single GPU task adaptation of pathology foundation models for whole slide image analysis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. arXiv:2506.05184.

Yoonho Lee, Annie S. Chen, Fahim Tajwar, Ananya Kumar, Huaxiu Yao, Percy Liang, and Chelsea Finn. Surgical fine-tuning improves adaptation to distribution shifts. In *International Conference on Learning Representations (ICLR)*, 2023. arXiv:2210.11466.

Honglin Li, Chenglu Zhu, Yunlong Zhang, Yuxuan Sun, Zhongyi Shui, Wenwei Kuang, Sunyi Zheng, and Lin Yang. Task-specific fine-tuning via variational information bottleneck for weakly-supervised pathology whole slide image classification. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

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 *International Conference on Machine Learning (ICML)*, 2024. arXiv:2402.09353.

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. *International Conference on Learning Representations (ICLR)*, 2017.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *International Conference on Learning Representations (ICLR)*, 2019.

Kilian Pfeiffer, Mohamed Aboelenien Ahmed, Ramin Khalili, and Jörg Henkel. Efficient federated finetuning of tiny transformers with resource-constrained devices. *arXiv preprint arXiv:2411.07826*, 2024.

Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. SVCCA: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. arXiv:1706.05806.

Maithra Raghu, Thomas Unterthiner, Simon Kornblith, Chiyuan Zhang, and Alexey Dosovitskiy. Do vision transformers see like convolutional neural networks? *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. arXiv:2108.08810.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. ZeRO: Memory optimizations toward training trillion parameter models. In *International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, 2020. arXiv:1910.02054.

Eugene Vorontsov, Alican Bozkurt, Adam Casson, George Shaikovski, Michal Zelechowski, Kristen Severson, Eric Zimmermann, James Hall, Neil Tenenholtz, Nicolo Fusi, et al. A foundation model for clinical-grade computational pathology and rare cancers detection. *Nature medicine*, 30(10): 2924–2935, 2024.

Hanwen Xu, Naoto Usuyama, Jaspreet Bagga, Sheng Zhang, Rajesh Rao, Tristan Naumann, Cliff Wong, Zelalem Gero, Javier González, Yu Gu, et al. A whole-slide foundation model for digital pathology from real-world data. *Nature*, 630:181–188, 2024.

Longteng Zhang, Lin Zhang, Shaohuai Shi, Xiaowen Chu, and Bo Li. LoRA-FA: Memory-efficient low-rank adaptation for large language models fine-tuning. *arXiv preprint arXiv:2308.03303*, 2023a.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. AdaLoRA: Adaptive budget allocation for parameter-efficient fine-tuning. In *International Conference on Learning Representations (ICLR)*, 2023b. arXiv:2303.10512.

Zikai Zhang, Rui Hu, Ping Liu, and Jiahao Xu. Fed-pilot: Optimizing LoRA allocation for efficient federated fine-tuning with heterogeneous clients. *arXiv preprint arXiv:2410.10200*, 2024.

Zihan Zhao, Fengtao Zhou, Ronggang Li, Bing Chu, Xinke Zhang, Xueyi Zheng, Ke Zheng, Xiaobo Wen, Jiabo Ma, Yihui Wang, Jiewei Chen, Chengyou Zheng, Jiangyu Zhang, Yongqin Wen, Jiajia Meng, Ziqi Zeng, Xiaoqing Li, Jing Li, Dan Xie, Yaping Ye, Yu Wang, Hao Chen, and Muyan Cai. A clinical-grade universal foundation model for intraoperative pathology, 2025. URL https://arxiv.org/abs/2510.04861.

Samir Arora Zhu and Yuchen Xu. SPAFIT: Stratified progressive adaptation fine-tuning for pre-trained large language models. *arXiv preprint arXiv:2405.00201*, 2024.

Acknowledgments

Will be provided in the final version to respect the double-blind review policy of ICLR.

A Additional Implementation Details

A.1 LoRA injection details

We implemented LoRA using PEFT. GigaPath contains 40 transformer blocks with hidden dimension $d = 1536$. In each selected block, we applied LoRA exclusively to the fused `qkv` linear projection, $W_{\text{qkv}} \in \mathbb{R}^{3d\times d}$; the attention output projection (`proj`) and the MLP layers (`fc1` and `fc2`) were left unchanged. For each targeted projection, the frozen pretrained weight was augmented as

$$W'_{\text{qkv}} = W_{\text{qkv}} + \frac{\alpha}{r} BA,$$

where $A \in \mathbb{R}^{r\times d}$ and $B \in \mathbb{R}^{3d\times r}$. We used rank $r = 8$, LoRA scaling parameter $\alpha = 16$ (i.e., $\alpha/r = 2$), no LoRA dropout, and no trainable bias parameters in all experiments. All original GigaPath parameters remained frozen unless explicitly stated otherwise.

A.2 Freeze-N implementation notes

As implemented (Algorithm 1), freezing proceeds in three steps: (1) set `requires_grad=False` on every parameter of the tile encoder, including `patch_embed`, class token, and positional embedding; (2) re-enable `requires_grad=True` only on blocks $[L-N, L-1]$ and the final `norm`; (3) construct the optimizer only over parameters with `requires_grad=True`. Step (1) applied

to the *entire* model, not only `tile_model.blocks`, is necessary for the activation-memory saving described in Section 4.1: leaving upstream modules trainable while freezing only block-level parameters would still require materializing the backward graph (and activations) for the nominally frozen early blocks, because their outputs remain inputs to a graph that must backpropagate into the (trainable) upstream modules.

A.3 Hyperparameters

Unless otherwise stated, all configurations use mixed-precision AdamW optimization with encoder and MIL-aggregator learning rates of 10^{-6} and 10^{-5}, respectively. The implementation applies AdamW's default weight decay of 10^{-2} to matrix-valued parameters and no decay to biases or normalization parameters. Both optimizers use cosine annealing with warm restarts; the first-cycle length is set to the configured training horizon, with $T_{\text{mult}} = 1$ for the encoder and $T_{\text{mult}} = 2$ for the aggregator. Each update processes one WSI, with its K sampled tiles evaluated in two sequential tile chunks. Training augmentation consists of random horizontal flipping, random rotations by multiples of $90°$, Gaussian blur, and ImageNet normalization; validation uses only normalization. LoRA uses rank $r = 8$, scaling $\alpha = 16$, zero dropout, and targets the attention QKV projections; the hybrid applies these adapters only to the last 10 blocks and trains the final normalization layer. All runs use seed 1634. Because several runs were limited by wall-clock availability, we report the best checkpoint among completed epochs rather than claiming convergence.

B Training Computational Complexity: Time and Space

We derive per-tile time (FLOP) and space (memory) complexity for the four configurations of Section 5, extending TAPFM's own Theorem 2 (time) and Theorem 3 (space). Notation: L blocks, N trainable last blocks ($j_{\min} = L - N$), hidden dimension D, T tokens per tile (we use T rather than TAPFM's original N for token count, to avoid clashing with our N for trainable-block count), H attention heads, K tiles per training step, LoRA rank $r \ll D$, and $P = |\theta_{PFM}|$ total encoder parameters.

B.1 Per-block cost primitive

A standard ViT block's forward FLOPs, per tile, are dominated by the QKV/output projections and MLP ($O(TD^2)$) and self-attention ($O(T^2D)$):

$$C_{\text{block}} \;=\; O(TD^2 + T^2D). \tag{6}$$

B.2 Time complexity

Forward pass is identical across all four configurations. Every tile is propagated through all L blocks regardless of which parameters are trainable (Algorithm 1 only toggles `requires_grad`; it does not skip forward computation), so forward cost is $O(K \cdot L \cdot C_{\text{block}})$ in every configuration.

Backward pass differs by configuration. Using the standard estimate that a full backward pass costs $\approx 2\times$ its forward pass:

- **TAPFM (full FT):** all L blocks are on the backward path: $O(K \cdot L \cdot C_{\text{block}})$.
- **LoRA-TAPFM:** every block carries an adapter, so the input-gradient must still be computed at every block down to block 0 (computing $\partial\mathcal{L}/\partial x$ at a LoRA-adapted layer requires multiplying by the *frozen* $W_0^\top$, the same $O(TD^2)$ matmul as full fine-tuning); LoRA only shrinks the *weight*-gradient term ($O(TrD) \ll O(TD^2)$). Backward cost is therefore still $O(K \cdot L \cdot C_{\text{block}})$ — same order as full FT.
- **Freeze-N-TAPFM:** only blocks $j_{\min}, \ldots, L-1$ are on the backward path: $O(K \cdot N \cdot C_{\text{block}})$.
- **Freeze-N+LoRA-TAPFM:** same backward-path length as Freeze-N (N blocks); LoRA again only shrinks the lower-order weight-gradient term within those N blocks: $O(K \cdot N \cdot C_{\text{block}})$, i.e. the same order as Freeze-N alone.

Total per-tile time, forward + backward:

$$\text{TAPFM, LoRA-TAPFM: } O(K{\cdot}2L{\cdot}C_{\text{block}}), \qquad \text{Freeze-}N\text{, Freeze-}N\text{+LoRA: } O(K{\cdot}(L{+}N){\cdot}C_{\text{block}}). \tag{7}$$

The freezing-derived time-savings ratio is $\frac{L+N}{2L}$ — e.g. for $N{=}10, L{=}40$: $\frac{50}{80} = 0.625$, a $\approx 37.5\%$ time reduction, *substantially smaller* than the corresponding activation-memory reduction (Section B.3 below reduces activation memory to $N/L{=}25\%$ of its full-FT value, a 75% reduction).

B.3 Space complexity

Reusing the decomposition of Section 4.1, $S_{\text{tile}} = S_{\text{param}} + S_{\text{opt+grad}}(\theta_{\text{train}}) + S_{\text{act}}(N_{\text{onward}})$, with $S_{\text{act}}(N) \approx \frac{N}{L} S_{\text{act}}(L) + S_{\text{act}}^{\text{res}}$ (Equation 9 of the original derivation):

Table 5: Per-tile time and space complexity, all four configurations. $S_{\text{act}}(L)$ is full-model per-tile activation memory; $S_{\text{act}}^{\text{res}}$ is the non-linear residual (Section 4.1).

Configuration	Time (per tile)	$S_{\text{opt+grad}}(\theta_{\text{train}})$	S_{act}
TAPFM (full FT)	$O(2L \cdot C_{\text{block}})$	$O(P)$	$S_{\text{act}}(L)$
LoRA-TAPFM	$O(2L \cdot C_{\text{block}})$	$O(LrD)$	$S_{\text{act}}(L)$
Freeze-N-TAPFM	$O((L{+}N) \cdot C_{\text{block}})$	$O(\frac{N}{L}P)$	$\frac{N}{L} S_{\text{act}}(L) + S_{\text{act}}^{\text{res}}$
Freeze-N+LoRA	$O((L{+}N) \cdot C_{\text{block}})$	$O(NrD)$	$\frac{N}{L} S_{\text{act}}(L) + S_{\text{act}}^{\text{res}}$

Main finding: memory and time savings are decoupled, and LoRA's marginal benefit over Freeze-N is memory-only. Two consequences follow directly from Table 5. First, **freezing trades a large, near-linear activation-memory reduction (N/L) for a much smaller time reduction (bounded by $\frac{1}{2}$, since forward cost is irreducible)** — the tile-budget gains reported in Section 5 come from memory headroom, not from each tile being cheaper to process. Second, **going from Freeze-N to Freeze-N+LoRA changes only $S_{\text{opt+grad}}$**, from $O(\frac{N}{L}P)$ to $O(NrD)$ — a reduction by a factor of $\frac{LrD}{P}$, which is small since $r \ll D$ — while S_{act} and per-tile time are *unchanged*. This exactly predicts the asymmetry in our own measurements (Table 3): at matched $K{=}500$, adding LoRA to Freeze-N cuts trainable encoder parameters by 99.83% but peak memory by only 22.2% (73.47→57.20 GiB), because the dominant remaining term, S_{act}, is untouched by LoRA. The additional tile headroom Freeze-N+LoRA provides over Freeze-N alone (1000 vs. 500 tiles/WSI) is therefore memory reallocated from freed optimizer state into a larger batch, not a faster per-tile forward/backward pass.

C Representation-Drift Diagnostic: Algorithm and Theory

Algorithm 2 Selecting N via representation drift

1: Unfreeze all θ_{PFM}; hook every block to record pooled outputs
2: Compute pre-probe representations X_ℓ on held-out tiles $\mathcal{D}_{\text{probe}}$
3: **for** $s = 1$ to S **do**
4: sample WSI from $\mathcal{D}_{\text{cal}}$; TAL forward/backward; AdamW step
5: **end for**
6: Compute post-probe representations Y_ℓ on $\mathcal{D}_{\text{probe}}$
7: $D_\ell \leftarrow 1 - \text{CKA}(X_\ell, Y_\ell)$; normalize; $N^\star \leftarrow \min\{N : C(N) \geq \tau\}$

Proposition 1 (Invariances of linear CKA; Kornblith et al., 2019)**.** *For any orthogonal U and scalars $\alpha, \beta > 0$: CKA(X, Y) = CKA(XU, Y) = CKA$(\alpha X, \beta Y)$.*

D_ℓ is therefore insensitive to rotation/rescaling of a block's output that carries no functional meaning, while remaining sensitive to genuine geometric change. Raghu et al. (2017) show networks converge bottom-up (early layers stabilize first); Raghu et al. (2021) show CKA-measured change concentrates in the last third of ViT blocks under adaptation — consistent with treating large D_ℓ as evidence of active task-specific repurposing, and small D_ℓ as evidence a block is safe to freeze. This is an empirical regularity, not a theorem, and is the reason we prefer it to instantaneous gradient

magnitude, which we found less robust to per-example loss-weight outliers under class imbalance (Section 4.4).

Algorithm 2's cost is one short real probe (S AdamW steps, same memory regime as full fine-tuning) plus two forward-only passes over held-out tiles — far cheaper than the brute-force sweep, which trains to convergence at every candidate N.

D Additional Results

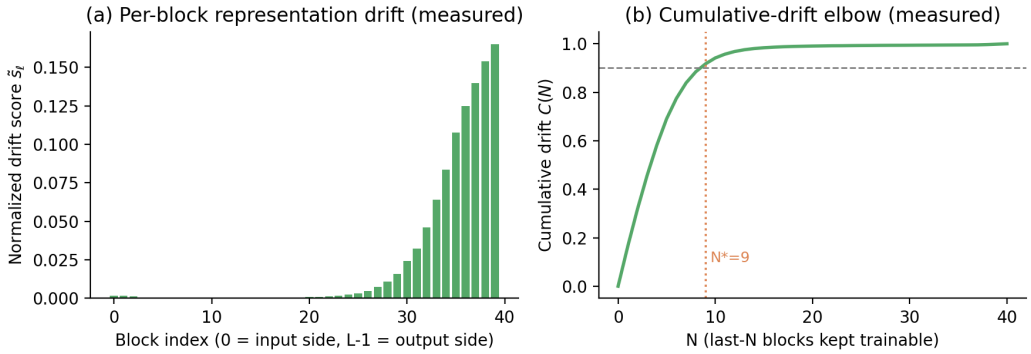

Figure 3: Measured representation-drift diagnostic for GigaPath/LUAD: (a) per-block drift $\tilde{s}_\ell$, (b) cumulative elbow, $N^\star{=}9$ at $\tau{=}0.90$.