To Freeze or Not to Freeze? Memory-Constrained End-to-End Training for Whole-Slide Image Classification in Histopathology

Anonymous authors
Paper under double-blind review

Abstract

Training deep neural networks on gigapixel Whole Slide Images (WSIs) poses significant GPU memory challenges. Multiple Instance Learning (MIL) circumvents this by processing a bag of patches per WSI; however, memory limits typically force the feature encoder—whether a Convolutional Neural Network (CNN) or a foundation model—to remain frozen. In this work, we investigate whether end-to-end (E2E) training improves MIL performance and out-of-domain generalisation. To overcome the memory bottleneck, we introduce a memory-efficient strategy that divides computations into manageable chunks. For CNNs, this involves dynamically offloading intermediate activations to CPU RAM. For ViT-based MIL encoders, we process patches in micro-batches and recompute each micro-batch during backpropagation, recovering the full-bag encoder gradient without retaining all encoder activations simultaneously. Across PANDA/TCGA-PRAD and CAMELYON17/16, we find that E2E training is useful but not uniformly beneficial: its external effect depends on the adaptation recipe and evaluation setting. In particular, combining full E2E ResNet-18 (RN18) training with WSI-level augmentation gives the strongest RN18 transfer to TCGA-PRAD, and E2E + Learning without Forgetting allows H0-mini to match or exceed much larger frozen foundation models there. On CAMELYON16, however, frozen foundation models remain stronger. These results suggest that E2E training is a conditional but important tool for WSI model adaptation rather than a universal replacement for frozen encoders. [1]

1 Introduction

Advances in deep learning have revolutionised histopathology (Unger & Kather, 2024; Van der Laak et al., 2021), but some challenges in handling Whole Slide Images (WSIs) remain. Their enormous size, which can be up to a few gigapixels, prevents the direct use of standard machine learning techniques, which generally require much smaller inputs. For classification tasks, a common approach to handle such large images is to use Multiple Instance Learning (MIL) (Dietterich et al., 1997; Maron & Lozano-Pérez, 1997) in which some patches are extracted from the WSI, and it is assumed that a subset of those patches corresponds to the desired label. MIL is a form of weakly supervised learning. It typically involves two main stages: first, an encoder (e.g., a CNN) converts each patch into an embedding; second, an MIL head—comprising an aggregator, a classifier, and any intermediate projection layers—pools these embeddings to predict a label.

Historically, GPU memory limits and the limited size of annotated histopathology datasets have made end-to-end training of these pipelines practically infeasible (Campanella et al., 2019). To contextualise the severity of this bottleneck, consider that while a standard ResNet-18 contains only ∼11 million parameters (occupying less than 50 MB), the intermediate activations generated by processing a typical WSI bag (e.g., thousands of 256×256 patches) easily exceed hundreds of gigabytes. This overwhelming activation footprint, rather than parameter storage, is the primary hardware barrier to end-to-end WSI training. Consequently, researchers conventionally freeze the pretrained encoder and train only the MIL head (Campanella et al., 2019; Song et al., 2023).

[1] We used ChatGPT for language editing. The authors verified all content.

Foundation models—large, heavily pretrained encoders learned from massive histopathology image corpora—have become increasingly popular, prompting a rapid proliferation of models developed for a wide range of computational pathology tasks. Despite their rapid adoption, accumulating evidence shows these models can be brittle under realistic distribution shifts (de Jong et al., 2025). Recent evaluations show that pathology foundation models often encode non-biological technical signals—such as medical-centre, scanner and staining signatures—that undermine their robustness under distribution shift (Kömen et al., 2024; de Jong et al., 2025; Gustafsson & Rantalainen, 2024). Linear-probe analyses further reveal that these site- and batch-specific factors are readily recoverable from foundation-model embeddings and can dominate biologically relevant variation, leading to systematic errors when models are deployed across institutions (Kömen et al., 2024; de Jong et al., 2025; Kömen et al., 2025). Moreover, standard mitigation strategies, including stain normalisation, distillation and larger-scale pretraining, only partially reduce these vulnerabilities (Kömen et al., 2024; 2025; Gustafsson & Rantalainen, 2024; Filiot et al., 2025). These findings underscore the need for rigorous robustness assessment and task-aware adaptation when applying foundation encoders in heterogeneous clinical environments (Henriksen et al., 2026).

In this work, we study whether the frozen-encoder paradigm is holding back WSI-level generalisation. We do not assume that E2E training is always preferable. Instead, we ask three questions: (i) can standard CNN and ViT-based WSI encoders be trained end-to-end without changing the MIL objective or dropping architecture components; (ii) under what data and augmentation regimes does E2E training improve in-domain and external performance; and (iii) can a computationally accessible fine-tuned foundation model compete with much larger frozen encoders?

To answer these questions, we develop a partition-and-compute strategy for full E2E training under single-GPU activation-memory constraints. For CNNs, we partition computations inside memory-heavy layers and explicitly support global-statistic layers such as BatchNorm (Ioffe & Szegedy, 2015). For ViT-based MIL encoders, where WSI patches are independent before the MIL head, we process the patches through the full encoder in micro-batches and recompute each micro-batch during backpropagation, recovering the full-bag gradient without retaining all encoder activations simultaneously. We further use Learning without Forgetting (LwF) (Li & Hoiem, 2017) as a regulariser for full foundation-model fine-tuning.

Our experiments show a conditional picture. E2E training consistently improves the performance of ResNet-18 (RN18) (He et al., 2016) on the PANDA in-domain test set, whereas external generalisation depends strongly on augmentation. With WSI-level augmentation, training RN18 end-to-end substantially improves TCGA-PRAD performance. Fine-tuning H0-mini end-to-end on PANDA also improves external TCGA-PRAD performance and can match or exceed larger frozen foundation models. On CAMELYON17/16, where the training set is smaller and the external gap appears less severe, frozen foundation models remain stronger on CAMELYON16. These findings suggest that E2E training is most valuable when researchers have enough training data and use realistic augmentation, a setting likely to become increasingly relevant as annotated WSI cohorts continue to grow.

Definition of full E2E training. We use *full end-to-end training* to mean that the supervised WSI-level loss updates the MIL head and all trainable layers of the patch encoder used during inference. Methods that freeze or remove architecture components, update only selected encoder layers, or compute encoder gradients for only a subset of patches address related but distinct settings. This distinction matters for standard CNNs: omitting layers such as BatchNorm, or replacing them with non-equivalent alternatives, does not train the original architecture fully E2E. It also matters for foundation models, where the dominant bottleneck is storing activations for thousands of patches rather than the parameter or optimiser state alone. Throughout, we target a single-GPU setting; multi-GPU systems are outside the scope of this study.

2 Related Work

Memory- and I/O-aware techniques for training on large inputs have followed several broad paradigms. We first review representative approaches to the activation-memory bottleneck and then discuss foundation models in histopathology. Table 1 summarises the specific single-GPU, full E2E scope considered here.

Table 1: Representative memory-efficient paradigms under the single-GPU, full E2E scope considered here.

Approach	Solves Activation Bottleneck	Exact Full-Bag Gradients	CNN Global Stats	ViT / FM Encoders	Native PyTorch/ TF2 API
CUDA Unified Memory	✓	✓	✓	✓	× (Custom)
Streaming CNNs	✓	✓	×	×	✓
Partial Backward	✓	× (Approx)	✓	✓	✓
State/Optim Offload	× (State only)	✓	✓	✓	✓
This work	✓	✓	✓	✓	✓

2.1 Memory-efficient training for large inputs

Unified / runtime-managed memory Chen et al. (2021) leverage unified memory in TensorFlow to train whole-slide models end-to-end. However, this is not directly applicable in PyTorch due to the lack of an API for using unified memory. In practice, the explicit high-level unified-memory knobs used in older TensorFlow releases are not exposed as stable TensorFlow version 2 public APIs. Furthermore, relying on runtime-managed unified memory often yields suboptimal transfer scheduling compared to methods that explicitly manage data transfers between GPU and CPU (Landaverde et al., 2014; Jarząbek & Czarnul, 2017; Alawneh et al., 2018). Our method does not depend on CUDA unified memory; instead, we perform layer-aware transfers and precisely control when and how tensors move between CPU and GPU.

Streaming / tiled convolutional training Pinckaers et al. (2020) split the input into spatial tiles and execute convolutions tile-by-tile, stitching intermediate feature maps. While they are able to partially train CNNs, their scope differs from ours. They stream convolutional feature maps and avoid storing all intermediate activations, but operations that require global statistics, such as BatchNorm, cannot be handled in a streaming region. Therefore, for standard CNN architectures containing such layers, streaming does not provide full-architecture E2E training as defined above. Our CNN implementation instead recomputes or offloads at the layer level and explicitly computes global statistics over the full tensor before normalisation.

Partial backward / selective gradient updates Skrede et al. (2020) reduce memory consumption by computing gradients for the encoder only for selected patches. Although this reduces computational load, it produces approximate gradients that can harm representational learning when end-to-end fidelity is crucial. Conversely, our strategy computes exact gradients for every tile within the input bag.

Optimizer/state offload (ZeRO-offload) Methods like ZeRO-Offload (Ren et al., 2021) reduce GPU memory pressure by offloading model parameters and optimiser states to the CPU. However, these techniques solve a fundamentally different problem: they target scenarios where the model itself is too large for the GPU, whereas the input remains relatively small (e.g., Large Language Models). In histopathology, the inverse is true; the models are relatively small (e.g., ResNet-18 or ViT-Base), but the input (a massive bag of WSI patches) overwhelmingly exceeds memory limits. Our approach directly addresses this large-input bottleneck, though it remains entirely complementary to model-state offloading techniques.

2.2 Foundation Models

The field of computational pathology is increasingly adopting foundation models—such as UNI (Chen et al., 2024), Prov-GigaPath (Xu et al., 2024), and the H-optimus (Bioptimus, 2025) family including H0-mini (Filiot et al., 2025). Pre-trained via self-supervised learning on vast cohorts of unannotated tissue, these models yield rich, generic representations. However, their deployment drastically exacerbates the GPU memory bottleneck. Because these encoders contain hundreds of millions or even billions of parameters, fine-tuning them end-to-end on gigapixel WSI inputs is computationally prohibitive on standard hardware.

Consequently, the prevailing paradigm strictly uses a frozen-encoder setup during downstream MIL tasks. Yet, as noted in section 1, relying on frozen features from foundation models leaves downstream models

vulnerable to site-specific distribution shifts (Kömen et al., 2024; de Jong et al., 2025; Gustafsson & Rantalainen, 2024). Unfreezing and adapting these massive models may improve out-of-distribution generalisation, but doing so introduces a secondary challenge: catastrophic forgetting (McCloskey & Cohen, 1989; French, 1999). When researchers attempt to fine-tune heavily pretrained models on smaller, task-specific datasets, the models frequently overwrite and lose their valuable pretrained generalisability (Kumar et al., 2022; Bafghi et al., 2024). Therefore, successfully adapting foundation models in histopathology requires addressing both the extreme VRAM constraints of end-to-end WSI training and the preservation of pretrained knowledge.

3 Approach for memory-efficient full E2E training

3.1 Overview

To investigate the impact of end-to-end training, we require a memory-efficient strategy that can process layers whose inputs and outputs exceed GPU capacity. We introduce a layer-aware partition-and-compute approach that dynamically adapts to the network's architectural dependencies.

Our method relies on four key strategies:

- **Partitioning:** We divide large input tensors into sub-tensors that safely fit within GPU memory.
- **Selective Offloading or Recomputation:** For layers requiring global statistics, we store intermediate activations in CPU RAM and transfer them back to the GPU only when needed. For architectures that do not compute intra-batch global statistics, we substitute CPU offloading with gradient checkpointing to avoid PCIe transfer overhead.
- **Layer-Specific Execution:** We execute compute-intensive layers (e.g., convolutions, batch normalisation, self-attention) on the GPU using our partitioned strategies, while computationally cheap layers (e.g., ReLU, MaxPool) run directly on the CPU.
- **Efficient Backpropagation:** We apply identical data partitioning and memory-management techniques to gradient computations, ensuring an exact and memory-efficient backward pass.

Depending on the network architecture, this layer-aware strategy manifests in distinct ways. For standard CNNs, which aggressively downsample inputs but rely on cross-instance global statistics (e.g., BatchNorm), our method utilises dynamic CPU-offloading for the initial high-memory layers. For Vision Transformers (ViTs) (Dosovitskiy, 2020) in a Multiple Instance Learning (MIL) setup with no intra-batch interactions in the encoder, the strategy naturally morphs into micro-batching coupled with gradient checkpointing. Algorithm 1 outlines the general flow of our method, and fig. 1 illustrates its CPU-offloading manifestation for a ResNet-18 architecture.

3.2 Partitioning and CPU-offloading for CNNs

The core of our method for handling computationally heavy layers is to partition the input tensor into sub-tensors and incrementally compute the output using those sub-tensors. First, we divide the input tensor into (potentially overlapping) sub-tensors. Then, we transfer sub-tensors to the GPU one by one and perform the layer-specific computation with the sub-tensor present on the GPU. Some layers, like normalisation layers, require repeating the previous step to get the final output. For example, for BatchNorm, we first compute the mean and standard deviation of the input using Welford's algorithm in the first sequential transfer of sub-tensors, followed by a second transfer to normalise the input using these calculated statistics. For the convolution layer, we transfer the sub-tensors only once, but the sub-tensors might have an overlap depending on the stride used in the layer.

Since the concrete implementation of our generic method differs from layer to layer, we detail the implementation of the forward pass for BatchNorm in algorithm 2 and illustrate it in fig. 1, demonstrating how to use the generic method for a layer that requires global statistics. We provide a similar implementation of the Convolution layer in Appendix C.

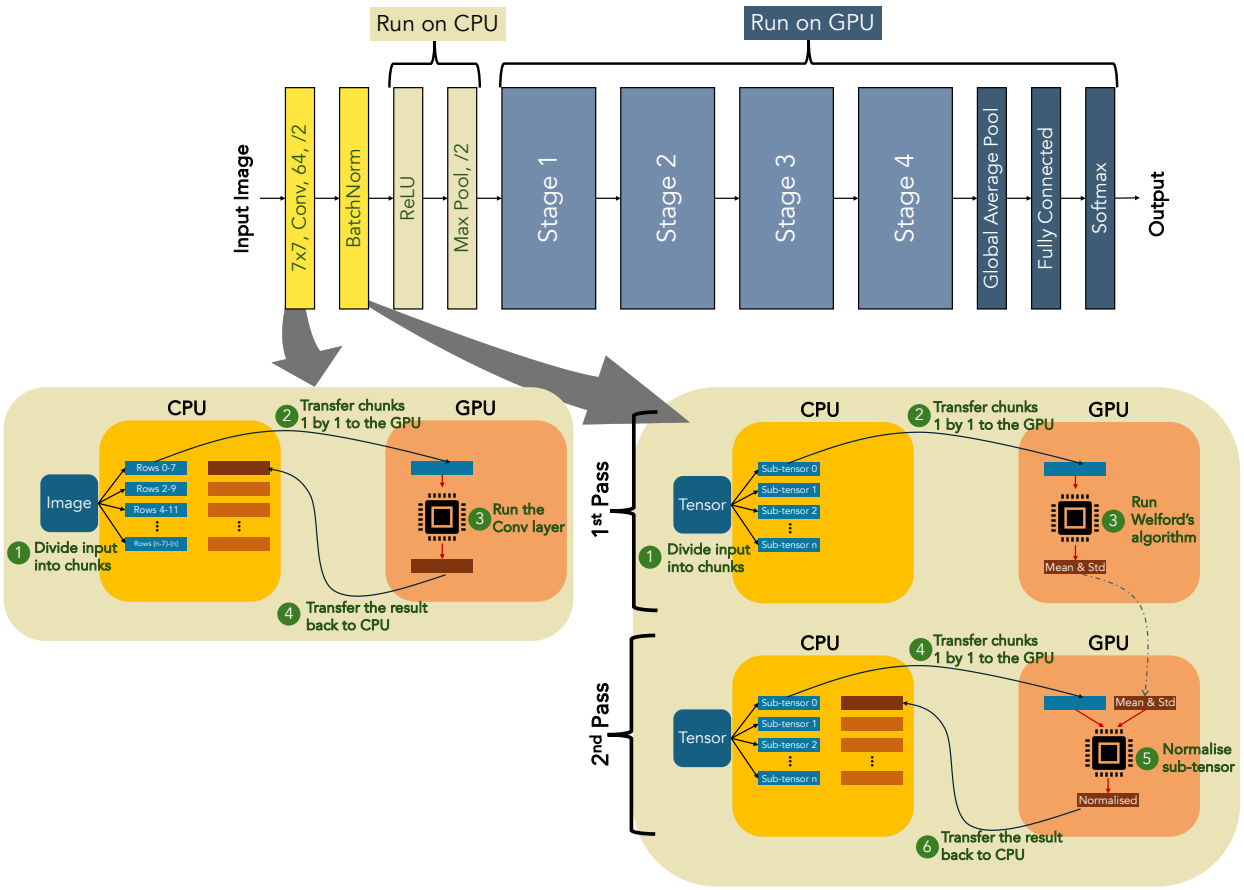

Figure 1: Layer-level illustration of partition-and-compute execution in ResNet-18. A spatially large input is shown for clarity: convolution uses overlapping row chunks, whereas BatchNorm first computes full-tensor statistics and then normalises each chunk. In the MIL experiments, these custom layers process the sampled patch-bag tensor, and the final classifier is replaced by the MIL head.

3.3 Micro-batch recomputation for Vision Transformer encoders

For ViT-based MIL encoders, we handle the activation-memory bottleneck by processing patches in micro-batches rather than partitioning individual layers. In the standard MIL formulation, the patch encoder processes each WSI patch independently before the MIL head. Let a WSI bag be $X = \{x_i\}_{i=1}^{N}$, let F_θ be the encoder, and let g_ϕ be the MIL head. The full-bag objective is

$$\mathcal{L}\big(g_\phi(F_\theta(x_1), \ldots, F_\theta(x_N)), y\big).$$

Because there are no cross-patch interactions inside F_θ, the embeddings $F_\theta(x_i)$ can be computed in any partitioning of the bag. We therefore compute embeddings one micro-batch at a time, evaluate the MIL loss on the concatenated embeddings, and then recompute each encoder micro-batch during the backward pass using the corresponding gradient $\partial\mathcal{L}/\partial F_\theta(x_i)$. By the chain rule, the accumulated encoder gradient is identical to the gradient obtained by processing the full bag at once, up to the usual floating-point effects.

For completeness, Appendix D describes a separate CPU-offloaded blockwise self-attention extension for continuous WSI token sequences. This extension preserves the self-attention computation up to floating-point ordering effects, avoids materialising the full attention matrix on the GPU, remains quadratic in sequence length, and is not used in the experiments reported here.

Algorithm 1 Layer-wise partition-and-compute execution

Require: Input tensor X, layers $\{L_i\}_{i=1}^{N}$
Ensure: Output tensor Y

```
procedure EfficientForward(X, {L_i}_{i=1}^N)
    for i = 1, ..., N do
        if the remaining computation fits in GPU memory then
            Transfer X to GPU
            X ← (L_N ∘ ··· ∘ L_i)(X)
            break
        else if L_i has a partitioned GPU implementation then
            Choose layer-specific sub-tensors {X_j} and initialise output Y
            for each pass required by L_i do
                for each sub-tensor X_j do
                    Transfer X_j to GPU and update the local output or global accumulators
                end for
            end for
            Assemble or finalise the full output Y
            X ← Y
        else
            X ← L_i(X) on CPU                                   ▷ e.g., ReLU or MaxPool
        end if
    end for
    Y ← X
    return Y
end procedure
```

Algorithm 2 Memory-efficient Batch Normalisation (forward pass)

Require: Input X, affine parameters γ, β, running statistics r_μ, r_{σ^2}, transfer budget max_N, ϵ
Ensure: Normalised output Y and updated running statistics

```
procedure PartitionedBatchNorm(X, γ, β, r_μ, r_σ², max_N, ε)
    Partition X into sub-tensors {X_i} with at most max_N elements
    if training then
        Initialise the Welford count, mean, and second-moment accumulators
        for each X_i do
            Transfer X_i to GPU and update the Welford accumulators
        end for
        Obtain full-tensor μ, σ² and update r_μ, r_σ²
    else
        μ ← r_μ and σ² ← r_σ²
    end if
    for each X_i do
        Transfer X_i to GPU
        Y_i ← γ ⊙ (X_i − μ) / √(σ² + ε) + β
        Write Y_i into the CPU-resident output Y
    end for
    return Y, r_μ, r_σ²
end procedure
```

3.4 Foundation Model Fine-Tuning via Learning without Forgetting

While our strategy (algorithm 1) resolves the VRAM bottleneck, fine-tuning large foundation models introduces the risk of catastrophic forgetting (McCloskey & Cohen, 1989; French, 1999). Because foundation

models are pretrained on vast, diverse datasets, aggressively updating their weights on a smaller, task-specific dataset can overwrite their valuable generic representations (Kumar et al., 2022; Bafghi et al., 2024).

To preserve this pretrained knowledge during end-to-end training, we use an embedding-matching regulariser inspired by Learning without Forgetting (LwF) (Li & Hoiem, 2017). We precompute embeddings of the unaugmented patches with the frozen encoder. During E2E training, the encoder processes augmented versions of the same patches, and we penalise the distance between the current embeddings (E_{aug}) and the frozen representations (E_{frozen}). We define the total loss as:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{MIL}} + \lambda \mathcal{L}_{\text{MSE}}(E_{\text{aug}}, E_{\text{frozen}}) \tag{1}$$

We use $\mathcal{L}_{\text{MIL}}$ for the main cross-entropy loss plus any method-specific auxiliary cross-entropy losses; DTFD includes its helper predictions in this term. λ controls the regularisation strength. The penalty encourages task adaptation while limiting drift from the pretrained representation. We use this penalty only for H0-mini fine-tuning; it does not form part of the memory-saving method. Comparisons involving H0-mini therefore evaluate the complete E2E + LwF adaptation recipe rather than E2E fine-tuning in isolation.

4 Experiments

4.1 Datasets and Preparation

We evaluate our approach on 4 datasets. We extract non-overlapping 256×256 patches from all WSIs.

PANDA and TCGA-PRAD: PANDA (Bulten et al., 2022) consists of 10,616 H&E-stained prostate biopsy WSIs (20×) with a 6-class ISUP grade. We use the 80:10:10 split from Song et al. (2024) (7,647/954/954 WSIs). For external testing, we use TCGA-PRAD (Abeshouse et al., 2015). After a pathologist's review, we removed 5 WSIs lacking tumour and 23 that could not be opened. The remaining 421 WSIs represent 394 unique patients; we randomly selected one WSI per patient. We retain patches with ≥60% foreground (greyscale intensities 3–230).

CAMELYON17 and 16: CAMELYON17 (Litjens et al., 2018) comprises 1,000 WSIs (40×) for lymph node metastasis classification. We use 472 WSIs for training and 50 for validation, and evaluate on the official test set. We use CAMELYON16 (Bejnordi et al., 2017) (399 WSIs) as the external test set. We formulate both datasets as binary metastasis classification, tile the WSIs at 20× using Otsu thresholding (Otsu et al., 1975; Zhang et al., 2022), and use the labels proposed by Ling et al. (2025).

4.2 Experimental Setup

Evaluation axes. We design our experiments around four axes. First, we compare frozen and E2E CNN encoders to test whether updating the encoder helps when the architecture can be trained fully E2E. Second, we evaluate large frozen pathology foundation models to represent the current practical default in WSI MIL. Third, we fine-tune H0-mini E2E as a computationally feasible foundation model. Fourth, we include augmentation and bag size ablations to distinguish three effects: encoder adaptation, domain generalisation, and train–test bag size discrepancy.

Methods and Encoders. We evaluate three MIL methods: ABMIL (Ilse et al., 2018), DTFD (Zhang et al., 2022), and TransMIL (Shao et al., 2021). We compare a frozen ResNet-18 (RN18) encoder pretrained on ImageNet-1K (Deng et al., 2009) with full E2E RN18 trained using the memory-efficient implementation described above. We additionally evaluate four frozen foundation encoders: H-optimus-1 (Bioptimus, 2025), H0-mini (Filiot et al., 2025), UNI2-h (Chen et al., 2024), and Prov-GigaPath (Xu et al., 2024), using the same MIL-head hyperparameters as the frozen RN18 setting. Finally, we evaluate H0-mini with full E2E training and LwF regularisation.

We fine-tune H0-mini, rather than the largest foundation models, because our goal is to study full E2E fine-tuning under realistic single-GPU memory constraints. The larger models are included as frozen baselines because this is the setting in which they are commonly used in downstream WSI MIL pipelines.

Augmentation ablation. For RN18 experiments, we train with and without uniform WSI-level augmentations. We include the non-augmented setting as a diagnostic ablation to test whether E2E training alone improves external generalisation. The augmented setting is the intended OOD adaptation setting because prior work and standard practice in histopathology indicate that photometric augmentations are important for robustness to domain shifts. Rather than augmenting each patch independently, we sample one set of transformation parameters per WSI and apply it to all patches in the bag.

Hyperparameters and Tile Selection. During training, we sample 256 tiles per WSI for PANDA. On CAMELYON17, full E2E models use 512 tiles, and frozen models use 1024 tiles. During testing, we evaluate on all foreground tiles and additionally report 256-tile (TCGA-PRAD) or 2048-tile (CAMELYON) subsets to isolate length-shift effects. For PANDA RN18 models, we use Adam (1e−4 weight decay), cosine annealing for 20 epochs (batch size 2, 16 accumulation steps), and learning rates in $\{1\text{e}{-4}, 5\text{e}{-5}, 1\text{e}{-5}\}$. We fine-tune H0-mini for 20 epochs (lr=5e−5, 32 accumulation steps, $\lambda = 1$). For CAMELYON17, frozen RN18 models train for 120 epochs (lr=1e−4), whereas full E2E RN18 models train for 90 epochs (lr=5e−5) with DEMON momentum (Chen et al., 2022) for the small batch size of 2. We tuned these configurations separately under their respective memory and optimisation constraints; the comparison therefore concerns the complete frozen and full E2E training regimes rather than a one-factor optimisation ablation. We fine-tune H0-mini for 20 epochs (lr=5e−5, 2 accumulation steps) and use $\lambda = 10$ to more strongly limit representation drift on the smaller CAMELYON17 cohort. We select the learning rate using the source-domain validation set and run each configuration with three random seeds; the tables report mean ± standard deviation across these seeds.

5 Results and discussion

5.1 PANDA and TCGA-PRAD

Encoder updates and augmentation work together. For RN18, E2E training consistently improves PANDA, but the strongest and most consistent transfer to TCGA-PRAD comes from combining encoder updates with WSI-level augmentation.

Adaptability can outweigh scale. Frozen H0-mini is already competitive with much larger frozen foundation encoders. After E2E + LwF on PANDA, it is competitive with or stronger than the best larger frozen encoder across the all-tile TCGA-PRAD evaluations despite its much smaller size.

Table 2 shows two distinct effects for RN18. On the in-domain PANDA test set, E2E training consistently improves QWK across all MIL heads, with gains of 7–15 percentage points over frozen RN18 baselines. On the external TCGA-PRAD cohort, however, E2E training without augmentation is not sufficient: ABMIL and TransMIL can degrade when evaluated on all tiles. This negative result is informative because it shows that E2E optimisation alone is not a domain-generalisation method.

When we add WSI-level augmentation, the external picture changes. Augmented E2E RN18 improves TCGA-PRAD performance for ABMIL and DTFD by large margins, and improves TransMIL when evaluated under the 256-tile setting that matches the training bag size. Thus, our results suggest that E2E training is useful for OOD generalisation because it enables the encoder to learn from augmented input.

TransMIL exhibits high variance when tested on all tiles. We hypothesise that these inferior results stem from a training-testing mismatch. Specifically, all models train on bags with 256 tiles per WSI. While the PANDA test set averages 400 patches per WSI (maximum 1400), TCGA-PRAD averages 12800 (maximum 41000). Because TransMIL employs self-attention that directly processes inter-tile interactions, it likely performs suboptimally on WSIs with approximately 50 times as many tiles as during training.

We evaluate this hypothesis by testing on TCGA-PRAD using 256 randomly selected tiles per WSI, matching the training bag size. To account for variability arising from the random selection of tiles, we run the evaluation 100 times and use the mean QWK as the final QWK for each model. Under this controlled setting, TransMIL performance improves substantially, indicating that some of the poor all-tile results reflect bag-size shift rather than encoder quality alone.

Table 2: QWK (%) on PANDA and TCGA-PRAD. Values are mean±SD over three trained models. For TCGA-PRAD-256, we evaluate each model on 100 random 256-tile samples and average the resulting QWK values. "Aug." denotes WSI-level augmentation; shaded rows update the encoder. Bold and underlined entries mark the best and second-best means within each encoder group for a given MIL head. Rows without "Aug." are augmentation ablations and should not be interpreted as the recommended OOD configuration.

Agg.	Encoder	Method	PANDA	TCGA-PRAD-All	TCGA-PRAD-256
ABMIL	Foundation Model	H-optimus-1	**94.04 ± 0.30**	68.11 ± 2.21	66.60 ± 1.47
		Prov-GigaPath	93.91 ± 0.25	67.53 ± 1.10	64.71 ± 1.28
		UNI2-h	93.61 ± 0.12	71.10 ± 0.72	68.65 ± 0.63
		H0-mini (Frozen)	93.80 ± 0.15	68.63 ± 0.93	66.80 ± 0.66
		H0-mini (E2E + LwF)	93.55 ± 0.40	**73.38 ± 0.41**	**70.61 ± 0.54**
	ResNet-18	Frozen	76.74 ± 0.43	56.34 ± 0.85	47.31 ± 1.18
		Full E2E	84.60 ± 1.15	53.53 ± 2.35	49.61 ± 1.62
		Frozen + Aug.	77.89 ± 0.50	52.83 ± 0.87	51.29 ± 0.69
		Full E2E + Aug.	**86.91 ± 1.31**	**65.86 ± 1.73**	**63.46 ± 0.89**
DTFD	Foundation Model	H-optimus-1	93.18 ± 0.44	59.28 ± 1.77	66.34 ± 0.49
		Prov-GigaPath	92.90 ± 0.29	61.94 ± 1.31	64.10 ± 0.42
		UNI2-h	93.20 ± 0.15	**69.98 ± 0.73**	**67.55 ± 1.04**
		H0-mini (Frozen)	92.86 ± 0.06	65.84 ± 2.74	67.42 ± 0.60
		H0-mini (E2E + LwF)	**94.12 ± 0.54**	69.88 ± 0.60	66.72 ± 1.09
	ResNet-18	Frozen	73.64 ± 1.48	55.26 ± 1.87	47.83 ± 0.74
		Full E2E	**87.13 ± 0.63**	64.54 ± 2.97	56.04 ± 2.42
		Frozen + Aug.	71.53 ± 1.18	56.79 ± 4.21	49.21 ± 2.44
		Full E2E + Aug.	86.59 ± 0.40	**69.47 ± 0.76**	**62.65 ± 0.54**
TransMIL	Foundation Model	H-optimus-1	**95.56 ± 0.39**	46.19 ± 10.48	54.10 ± 4.56
		Prov-GigaPath	94.14 ± 0.43	52.30 ± 5.19	53.89 ± 2.89
		UNI2-h	94.35 ± 0.38	55.62 ± 5.98	58.43 ± 4.27
		H0-mini (Frozen)	94.72 ± 0.46	54.75 ± 5.84	59.18 ± 2.78
		H0-mini (E2E + LwF)	94.35 ± 0.51	**70.23 ± 1.82**	**70.00 ± 1.36**
	ResNet-18	Frozen	81.87 ± 1.13	44.63 ± 3.16	46.86 ± 1.70
		Full E2E	**89.60 ± 0.52**	29.96 ± 9.12	52.49 ± 1.67
		Frozen + Aug.	78.06 ± 1.06	36.25 ± 0.78	44.69 ± 2.54
		Full E2E + Aug.	89.39 ± 0.85	**45.56 ± 7.15**	**65.33 ± 1.14**

For foundation models, frozen encoders yield strong in-domain PANDA performance, as expected. The key question is whether a smaller foundation model can gain external robustness through E2E adaptation. On TCGA-PRAD, fine-tuned H0-mini improves over frozen H0-mini and can match or exceed much larger frozen foundation models. For example, with ABMIL, fine-tuned H0-mini reaches 73.38 QWK, compared with 68.63 for frozen H0-mini and 71.10 for the best frozen larger model. With TransMIL, fine-tuned H0-mini reaches 70.23 QWK, compared with 54.75 for frozen H0-mini and 55.62 for the best frozen larger model. H0-mini was distilled from TCGA-derived tiles (Filiot et al., 2025), so TCGA-PRAD may have contributed to its pretraining corpus. This exposure is shared by the frozen and E2E + LwF conditions and therefore does not explain their difference, although it may affect absolute cross-encoder comparisons.

Table 3: Accuracy (%) on CAMELYON17 and CAMELYON16. Values are mean±SD over three trained models. "All" uses all foreground tiles; "2K" averages 10 evaluations with up to 2,048 randomly sampled tiles per WSI. "Aug." denotes WSI-level augmentation; shaded rows update the encoder. Bold and underlined entries mark the best and second-best means within each encoder group for a given MIL head. Rows without "Aug." are augmentation ablations and should not be interpreted as the recommended OOD configuration.

Agg.	Encoder	Method	CAM17(All)	CAM17(2K)	CAM16(All)	CAM16(2K)
ABMIL	Foundation Model	H-optimus-1	88.64 ± 0.92	90.52 ± 0.40	<u>96.84 ± 0.34</u>	<u>95.15 ± 0.20</u>
		Prov-GigaPath	<u>89.92 ± 0.69</u>	<u>91.43 ± 0.27</u>	96.48 ± 1.42	94.21 ± 0.48
		UNI2-h	87.63 ± 0.90	89.60 ± 0.54	93.11 ± 2.10	93.29 ± 1.16
		H0-mini (Frozen)	89.07 ± 0.86	91.13 ± 0.23	**97.67 ± 0.26**	**95.79 ± 0.16**
		H0-mini (E2E + LwF)	**91.74 ± 1.52**	**92.09 ± 0.97**	96.11 ± 1.55	93.68 ± 1.45
	ResNet-18	Frozen	88.49 ± 1.22	87.97 ± 1.04	81.26 ± 2.39	79.36 ± 1.72
		Full E2E	<u>89.12 ± 0.88</u>	<u>90.58 ± 0.68</u>	**87.05 ± 3.05**	<u>82.52 ± 2.33</u>
		Frozen + Aug.	87.71 ± 0.92	88.52 ± 0.31	81.35 ± 1.19	79.70 ± 0.22
		Full E2E + Aug.	**90.54 ± 0.88**	**91.14 ± 1.18**	<u>86.53 ± 0.69</u>	**82.69 ± 0.66**
DTFD	Foundation Model	H-optimus-1	87.75 ± 1.45	90.26 ± 0.72	<u>94.35 ± 2.00</u>	93.98 ± 0.65
		Prov-GigaPath	87.63 ± 0.94	90.76 ± 0.46	94.09 ± 0.84	92.31 ± 0.45
		UNI2-h	<u>88.81 ± 0.56</u>	90.83 ± 0.40	93.78 ± 3.16	<u>94.22 ± 1.44</u>
		H0-mini (Frozen)	87.92 ± 1.88	<u>90.87 ± 0.64</u>	**96.11 ± 0.66**	**95.42 ± 0.34**
		H0-mini (E2E + LwF)	**92.16 ± 1.05**	**92.37 ± 0.79**	93.68 ± 1.06	92.83 ± 0.61
	ResNet-18	Frozen	87.50 ± 1.12	88.52 ± 0.68	81.69 ± 1.73	80.77 ± 0.62
		Full E2E	**89.46 ± 2.35**	**90.12 ± 1.97**	<u>86.27 ± 3.48</u>	<u>84.29 ± 1.63</u>
		Frozen + Aug.	<u>87.64 ± 0.44</u>	88.27 ± 0.09	82.21 ± 0.98	80.37 ± 0.17
		Full E2E + Aug.	85.95 ± 2.44	<u>88.63 ± 1.58</u>	**87.56 ± 2.59**	**86.04 ± 2.07**
TransMIL	Foundation Model	H-optimus-1	87.12 ± 1.49	89.11 ± 0.73	93.16 ± 4.06	92.52 ± 3.23
		Prov-GigaPath	<u>89.36 ± 1.32</u>	91.01 ± 0.93	<u>95.28 ± 1.12</u>	<u>93.65 ± 0.66</u>
		UNI2-h	87.46 ± 1.12	89.86 ± 0.42	91.97 ± 4.15	92.40 ± 2.55
		H0-mini (Frozen)	89.07 ± 0.65	<u>91.14 ± 0.48</u>	**97.41 ± 0.37**	**95.51 ± 0.20**
		H0-mini (E2E + LwF)	**91.91 ± 1.29**	**92.06 ± 1.09**	93.42 ± 2.41	93.23 ± 1.44
	ResNet-18	Frozen	<u>88.44 ± 2.27</u>	87.48 ± 2.19	<u>79.90 ± 2.66</u>	78.26 ± 1.48
		Full E2E	**88.77 ± 2.81**	<u>90.12 ± 1.53</u>	78.11 ± 2.84	80.79 ± 1.70
		Frozen + Aug.	87.29 ± 0.56	86.61 ± 0.47	**81.52 ± 0.15**	<u>80.81 ± 0.83</u>
		Full E2E + Aug.	**88.77 ± 2.97**	**91.20 ± 0.39**	79.71 ± 4.38	**83.05 ± 1.09**

5.2 CAMELYON

In-domain gains do not guarantee external robustness. E2E + LwF improves H0-mini on CAMELYON17 across all three MIL heads, yet frozen H0-mini remains better on CAMELYON16. Researchers should therefore judge encoder adaptation on external cohorts, not only on the source cohort. The AUC results in appendix A show the same pattern.

Table 3 reports accuracy on the CAMELYON17 test subset and the CAMELYON16 whole set. For the CNN architectures, end-to-end training increases accuracy by up to 6 percentage points. Our end-to-end RN18 models generalise effectively, demonstrating particularly large performance improvements on the external CAMELYON16 test set compared to the internal CAMELYON17 test subset.

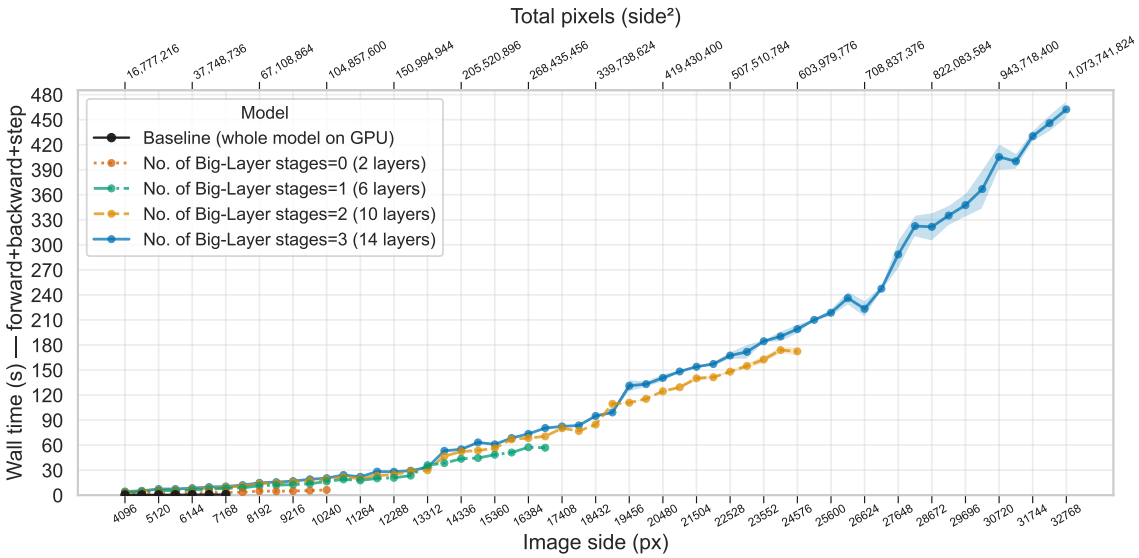

Figure 2: Incremental-stage benchmark for ResNet-18: mean wall time per training iteration (forward + backward + optimiser step) versus input side length. Shaded bands denote ±1 standard deviation. Each curve extends the partitioned implementation through a different number of early network stages. We call each standard layer that uses this implementation a *Big-Layer*. Benchmarks were run on an NVIDIA RTX 3090 (24 GB); we report scaling fits in Appendix B.

Many whole-slide images in both CAMELYON datasets contain substantially more tiles—up to an 80× increase—than the 512-tile bags we use for training; accordingly, we observe high evaluation variance for both DTFD and TransMIL when we evaluate on all tiles. To address this training–testing mismatch, we evaluate every model on two setups: (1) all foreground tiles per WSI and (2) 2048 randomly sampled tiles per WSI (the same number we use in the validation set for checkpoint selection). We repeat the 2048-tile evaluation 10 times and use the mean accuracy as the final score for a given model. On the CAMELYON17 test set, our models generally achieve higher accuracy in the 2048-tile setting, whereas the baseline models show comparable performance across both setups. In contrast, on CAMELYON16, most methods perform better when evaluated on all tiles, with the sole exception of our TransMIL models. This pattern is consistent with PANDA results, in which TransMIL exhibits improved performance on external datasets when evaluated with a limited number of tiles.

The CAMELYON foundation-model results show a different regime. The domain gap between CAMELYON17 and CAMELYON16 appears smaller than the PANDA-to-TCGA-PRAD gap, and the train subset is also much smaller. In this setting, fine-tuned H0-mini improves internal CAMELYON17 accuracy over frozen H0-mini, but it does not improve external CAMELYON16 accuracy. Frozen foundation models, especially frozen H0-mini, remain very strong on CAMELYON16. We observe similar trends for the AUC metrics in Appendix A.

We interpret this as an important boundary condition rather than a contradiction. E2E fine-tuning can specialise the encoder to the training distribution when the dataset is small, and LwF does not guarantee improved external performance in every regime. Therefore, CAMELYON supports a conditional conclusion: E2E adaptation is most compelling when the training cohort is large enough, and the external domain shift is substantial enough for adaptation to outweigh the risk of over-specialisation.

5.3 Speed

We report per-epoch wall-clock time on PANDA (NVIDIA RTX 3090, 24 GB): precomputed-embeddings (train only MIL head) takes ≈ 20 s/epoch; baseline on-the-fly (compute embeddings each iteration, frozen

encoder) takes ≈ 45 min/epoch; end-to-end RN18 (encoder backpropagation using our method) takes ≈ 200 min/epoch; end-to-end H0-mini (encoder backpropagation using our method) takes ≈ 255 min/epoch.

Furthermore, we characterise the scaling of per-iteration training time with input resolution using ResNet-18 (fig. 2). As expected, progressively enabling more *Big-Layer* stages increases the maximum feasible input size but introduces variable runtime scaling with depth and the number of Big-Layers. We provide detailed ResNet-50 benchmarks in fig. 3 and full scaling fit statistics in appendix B.

5.4 Discussion and future work

Our investigation shows that E2E training is a useful but conditional mechanism for improving WSI generalisation. It is not, by itself, a domain-generalisation method: the non-augmentation ablations show that external performance can remain mixed or even degrade. Its value is that it allows the model to benefit from task-specific adaptation, and domain-generalisation techniques that require encoder updates. It also allows exploring other domain-generalisation techniques (Huang et al., 2020; Wang et al., 2021; Zhang et al., 2024; Tomar et al., 2024) and spatial augmentations (Marini et al., 2023; Shen et al., 2022) that require encoder updates. We leave the integration of these techniques with fine-tuned models to future work. As annotated WSI cohorts continue to grow, the regime in which E2E adaptation is both feasible and beneficial is likely to become increasingly important. While we focus on histopathology, our memory-efficient approach is broadly applicable to other large-image domains, such as remote sensing (Huang et al., 2018; Li et al., 2022).

6 Limitations and scope

Our methodology involves partitioning the input tensor into sub-tensors and executing computations incrementally using these sub-tensors. Although all commonly used layers can be computed in this manner, any computationally intensive layer whose computation cannot be decomposed in this way will present a significant constraint as it will necessitate computation on the CPU, which can be prohibitively expensive.

We fine-tune only H0-mini among the foundation models. Full E2E fine-tuning of substantially larger encoders such as UNI2-h or H-optimus models would require considerably more compute and memory. Additionally, our method trades memory for runtime. The ability to train models E2E comes at the cost of additional CPU–GPU transfers and recomputation. This trade-off is appropriate when exact E2E training would otherwise be impossible, but it is not a replacement for standard GPU training when the full computation already fits in memory.

7 Conclusion

Our investigation reveals a conditional picture. E2E training consistently improves in-domain RN18 performance and, when paired with WSI-level augmentation, can substantially improve external generalisation. For foundation models, fine-tuning a computationally accessible H0-mini encoder on the large PANDA cohort can match or exceed much larger frozen encoders on TCGA-PRAD. However, CAMELYON shows that E2E adaptation is not universally beneficial: in smaller-data or lower-shift regimes, frozen foundation models can remain the stronger choice. Overall, our results argue against a fixed rule of always freezing or always fine-tuning. Instead, they show that exact E2E training is an important capability for studying and exploiting encoder adaptation as WSI datasets continue to grow.

References

Adam Abeshouse, Jaeil Ahn, Rehan Akbani, Adrian Ally, Samirkumar Amin, Christopher D Andry, Matti Annala, Armen Aprikian, Joshua Armenia, Arshi Arora, et al. The molecular taxonomy of primary prostate cancer. *Cell*, 163(4):1011–1025, 2015. doi: 10.1016/j.cell.2015.10.025.

Luay Alawneh, Emad Rawashdeh, Mahmoud Al-Ayyoub, and Yaser Jararweh. Gpu parallelization of sequence segmentation using information theoretic models. *Simulation Modelling Practice and Theory*, 86: 11–24, 2018. doi: 10.1016/j.simpat.2018.04.007.

Reza Akbarian Bafghi, Nidhin Harilal, Claire Monteleoni, and Maziar Raissi. Parameter efficient fine-tuning of self-supervised vits without catastrophic forgetting. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 7864–7869. IEEE, 2024. doi: 10.1109/cvprw63382.2024.00371.

Babak Ehteshami Bejnordi, Mitko Veta, Paul Johannes Van Diest, Bram Van Ginneken, Nico Karssemeijer, Geert Litjens, Jeroen AWM Van Der Laak, Meyke Hermsen, Quirine F Manson, Maschenka Balkenhol, et al. Diagnostic assessment of deep learning algorithms for detection of lymph node metastases in women with breast cancer. *Jama*, 318(22):2199–2210, 2017. doi: 10.1001/jama.2017.14585.

Bioptimus. H-optimus-1. `https://huggingface.co/bioptimus/H-optimus-1`, 2025. Accessed: 2025-11-15.

Wouter Bulten, Kimmo Kartasalo, Po-Hsuan Cameron Chen, Peter Ström, Hans Pinckaers, Kunal Nagpal, Yuannan Cai, David F Steiner, Hester Van Boven, Robert Vink, et al. Artificial intelligence for diagnosis and gleason grading of prostate cancer: the panda challenge. *Nature medicine*, 28(1):154–163, 2022. doi: 10.1038/s41591-021-01620-2.

Gabriele Campanella, Matthew G Hanna, Luke Geneslaw, Allen Miraflor, Vitor Werneck Krauss Silva, Klaus J Busam, Edi Brogi, Victor E Reuter, David S Klimstra, and Thomas J Fuchs. Clinical-grade computational pathology using weakly supervised deep learning on whole slide images. *Nature medicine*, 25(8):1301–1309, 2019. doi: 10.1038/s41591-019-0508-1.

Chi-Long Chen, Chi-Chung Chen, Wei-Hsiang Yu, Szu-Hua Chen, Yu-Chan Chang, Tai-I Hsu, Michael Hsiao, Chao-Yuan Yeh, and Cheng-Yu Chen. An annotation-free whole-slide training approach to pathological classification of lung cancer types using deep learning. *Nature communications*, 12(1):1193, 2021. doi: 10.1038/s41467-021-21467-y.

John Chen, Cameron Wolfe, Zhao Li, and Anastasios Kyrillidis. Demon: improved neural network training with momentum decay. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3958–3962. IEEE, 2022. doi: 10.1109/icassp43922.2022.9746839.

Richard J Chen, Tong Ding, Ming Y Lu, Drew FK 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, 2024. doi: 10.1038/s41591-024-02857-3.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. *Advances in neural information processing systems*, 35:16344–16359, 2022. doi: 10.52202/068431-1189.

Edwin D. de Jong, Eric Marcus, and Jonas Teuwen. Current pathology foundation models are unrobust to medical center differences. *arXiv preprint arXiv:2501.18055*, 2025. URL `https://arxiv.org/abs/2501.18055`.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009. doi: 10.1109/cvpr.2009.5206848.

Thomas G Dietterich, Richard H Lathrop, and Tomás Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial intelligence*, 89(1-2):31–71, 1997. doi: 10.1016/s0004-3702(96)00034-3.

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

Alexandre Filiot, Nicolas Dop, Oussama Tchita, Auriane Riou, Rémy Dubois, Thomas Peeters, Daria Valter, Marin Scalbert, Charlie Saillard, Geneviève Robin, et al. Distilling foundation models for robust and efficient models in digital pathology. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 162–172. Springer, 2025. doi: 10.1007/978-3-032-04981-0_16.

Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4): 128–135, 1999. doi: 10.1016/S1364-6613(99)01294-2.

Fredrik K Gustafsson and Mattias Rantalainen. Evaluating computational pathology foundation models for prostate cancer grading under distribution shifts. *arXiv preprint arXiv:2410.06723*, 2024.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016. doi: 10.1109/cvpr.2016.90.

Audun L Henriksen, Ole-Johan Skrede, Lisa van der Schee, Enric Domingo, Sepp De Raedt, Ilyá Kostolomov, Jennifer Hay, Karolina Cyll, Wanja Kildal, Joakim Kalsnes, et al. Enabling clinical use of foundation models in histopathology. *arXiv preprint arXiv:2602.22347*, 2026.

Bohao Huang, Daniel Reichman, Leslie M Collins, Kyle Bradbury, and Jordan M Malof. Tiling and stitching segmentation output for remote sensing: Basic challenges and recommendations. *arXiv preprint arXiv:1805.12219*, 2018.

Zeyi Huang, Haohan Wang, Eric P Xing, and Dong Huang. Self-challenging improves cross-domain generalization. In *European conference on computer vision*, pp. 124–140. Springer, 2020. doi: 10.1007/978-3-030-58536-5_8.

Maximilian Ilse, Jakub Tomczak, and Max Welling. Attention-based deep multiple instance learning. In *International conference on machine learning*, pp. 2127–2136. PMLR, 2018.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. pmlr, 2015.

Łukasz Jarząbek and Paweł Czarnul. Performance evaluation of unified memory and dynamic parallelism for selected parallel cuda applications. *The Journal of Supercomputing*, 73:5378–5401, 2017. doi: 10.1007/s11227-017-2091-x.

Jonah Kömen, Hannah Marienwald, Jonas Dippel, and Julius Hense. Do histopathological foundation models eliminate batch effects? a comparative study. *arXiv preprint arXiv:2411.05489*, 2024.

Jonah Kömen, Edwin D de Jong, Julius Hense, Hannah Marienwald, Jonas Dippel, Philip Naumann, Eric Marcus, Lukas Ruff, Maximilian Alber, Jonas Teuwen, et al. Towards robust foundation models for digital pathology. *arXiv preprint arXiv:2507.17845*, 2025.

Ananya Kumar, Aditi Raghunathan, Robbie Jones, Tengyu Ma, and Percy Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. In *International Conference on Learning Representations*, 2022.

Raphael Landaverde, Tiansheng Zhang, Ayse K Coskun, and Martin Herbordt. An investigation of unified memory access performance in cuda. In *2014 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–6. IEEE, 2014. doi: 10.1109/hpec.2014.7040988.

Zheng Li, Yongcheng Wang, Ning Zhang, Yuxi Zhang, Zhikang Zhao, Dongdong Xu, Guangli Ben, and Yunxiao Gao. Deep learning-based object detection techniques for remote sensing images: A survey. *Remote Sensing*, 14(10):2385, 2022. doi: 10.3390/rs14102385.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

Xitong Ling, Yuanyuan Lei, Jiawen Li, Junru Cheng, Wenting Huang, Tian Guan, Jian Guan, and Yonghong He. Comprehensive benchmark dataset for pathological lymph node metastasis in breast cancer sections. *Scientific Data*, 12(1):1381, 2025. doi: 10.1038/s41597-025-05586-5.

Geert Litjens, Peter Bandi, Babak Ehteshami Bejnordi, Oscar Geessink, Maschenka Balkenhol, Peter Bult, Altuna Halilovic, Meyke Hermsen, Rob Van de Loo, Rob Vogels, et al. 1399 h&e-stained sentinel lymph node sections of breast cancer patients: the camelyon dataset. *GigaScience*, 7(6):giy065, 2018. doi: 10.1093/gigascience/giy065.

Niccolò Marini, Sebastian Otalora, Marek Wodzinski, Selene Tomassini, Aldo Franco Dragoni, Stephane Marchand-Maillet, Juan Pedro Dominguez Morales, Lourdes Duran-Lopez, Simona Vatrano, Henning Müller, et al. Data-driven color augmentation for h&e stained images in computational pathology. *Journal of Pathology Informatics*, 14:100183, 2023. doi: 10.1016/j.jpi.2022.100183.

Oded Maron and Tomás Lozano-Pérez. A framework for multiple-instance learning. *Advances in neural information processing systems*, 10, 1997.

Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pp. 109–165. Elsevier, 1989. doi: 10.1016/s0079-7421(08)60536-8.

Maxim Milakov and Natalia Gimelshein. Online normalizer calculation for softmax. *arXiv preprint arXiv:1805.02867*, 2018.

Nobuyuki Otsu et al. A threshold selection method from gray-level histograms. *Automatica*, 11(285-296): 23–27, 1975. doi: 10.1109/tsmc.1979.4310076.

Hans Pinckaers, Bram Van Ginneken, and Geert Litjens. Streaming convolutional neural networks for end-to-end learning with multi-megapixel images. *IEEE transactions on pattern analysis and machine intelligence*, 44(3):1581–1590, 2020. doi: 10.1109/tpami.2020.3019563.

Jie Ren, Samyam Rajbhandari, Reza Yazdani Aminabadi, Olatunji Ruwase, Shuangyan Yang, Minjia Zhang, Dong Li, and Yuxiong He. {Zero-offload}: Democratizing {billion-scale} model training. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*, pp. 551–564, 2021.

Zhuchen Shao, Hao Bian, Yang Chen, Yifeng Wang, Jian Zhang, Xiangyang Ji, et al. Transmil: Transformer based correlated multiple instance learning for whole slide image classification. *Advances in neural information processing systems*, 34:2136–2147, 2021.

Yiqing Shen, Yulin Luo, Dinggang Shen, and Jing Ke. Randstainna: Learning stain-agnostic features from histology slides by bridging stain augmentation and normalization. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 212–221. Springer, 2022. doi: 10.1007/978-3-031-16434-7_21.

Ole-Johan Skrede, Sepp De Raedt, Andreas Kleppe, Tarjei S Hveem, Knut Liestøl, John Maddison, Hanne A Askautrud, Manohar Pradhan, John Arne Nesheim, Fritz Albregtsen, et al. Deep learning for prediction of colorectal cancer outcome: a discovery and validation study. *The Lancet*, 395(10221):350–360, 2020. doi: 10.1016/s0140-6736(19)32998-8.

Andrew H Song, Guillaume Jaume, Drew FK Williamson, Ming Y Lu, Anurag Vaidya, Tiffany R Miller, and Faisal Mahmood. Artificial intelligence for digital and computational pathology. *Nature Reviews Bioengineering*, 1(12):930–949, 2023.

Andrew H Song, Richard J Chen, Tong Ding, Drew FK Williamson, Guillaume Jaume, and Faisal Mahmood. Morphological prototyping for unsupervised slide representation learning in computational pathology. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11566–11578, 2024. doi: 10.1109/cvpr52733.2024.01099.

Dhananjay Tomar, Alexander Binder, and Andreas Kleppe. Are nuclear masks all you need for improved out-of-domain generalisation? a closer look at cancer classification in histopathology. *Advances in Neural Information Processing Systems*, 37:43499–43532, 2024. doi: 10.52202/079017-1378.

Michaela Unger and Jakob Nikolas Kather. Deep learning in cancer genomics and histopathology. *Genome medicine*, 16(1):44, 2024. doi: 10.1186/s13073-024-01315-6.

Jeroen Van der Laak, Geert Litjens, and Francesco Ciompi. Deep learning in histopathology: the path to the clinic. *Nature medicine*, 27(5):775–784, 2021. doi: 10.1038/s41591-021-01343-4.

Zijian Wang, Yadan Luo, Ruihong Qiu, Zi Huang, and Mahsa Baktashmotlagh. Learning to diversify for single domain generalization. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 834–843, 2021. doi: 10.1109/iccv48922.2021.00087.

Hanwen Xu, Naoto Usuyama, Jaspreet Bagga, Sheng Zhang, Rajesh Rao, Tristan Naumann, Cliff Wong, Zelalem Gero, Javier González, Yu Gu, Yanbo Xu, Mu Wei, Wenhui Wang, Shuming Ma, Furu Wei, Jianwei Yang, Chunyuan Li, Jianfeng Gao, Jaylen Rosemon, Tucker Bower, Soohee Lee, Roshanthi Weerasinghe, Bill J. Wright, Ari Robicsek, Brian Piening, Carlo Bifulco, Sheng Wang, and Hoifung Poon. A whole-slide foundation model for digital pathology from real-world data. *Nature*, 2024. doi: 10.1038/s41586-024-07441-w.

Hongrun Zhang, Yanda Meng, Yitian Zhao, Yihong Qiao, Xiaoyun Yang, Sarah E Coupland, and Yalin Zheng. Dtfd-mil: Double-tier feature distillation multiple instance learning for histopathology whole slide image classification. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 18802–18812, 2022. doi: 10.1109/cvpr52688.2022.01824.

Zheyuan Zhang, Bin Wang, Debesh Jha, Ugur Demir, and Ulas Bagci. Domain generalization with correlated style uncertainty. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pp. 2000–2009, 2024. doi: 10.1109/wacv57701.2024.00200.

Appendix

A AUC results for CAMELYON 17/16

We report AUCs for the CAMELYON datasets in table 4.

Table 4: AUC (%) on CAMELYON17 and CAMELYON16. Values are mean±SD over three trained models. "All" uses all foreground tiles; "2K" averages 10 evaluations with up to 2,048 randomly sampled tiles per WSI. "Aug." denotes WSI-level augmentation; shaded rows update the encoder. Bold and underlined entries mark the best and second-best means within each encoder group for a given MIL head. Rows without "Aug." are augmentation ablations and should not be interpreted as the recommended OOD configuration.

Agg.	Encoder	Method	CAM17(All)	CAM17(2K)	CAM16(All)	CAM16(2K)
ABMIL	Foundation Model	H-optimus-1	95.64 ± 0.52	96.00 ± 0.41	97.98 ± 0.22	95.76 ± 0.42
		Prov-GigaPath	95.81 ± 0.69	95.43 ± 0.24	97.66 ± 0.14	94.29 ± 0.23
		UNI2-h	96.36 ± 0.48	**96.70 ± 0.21**	**98.60 ± 0.04**	**96.16 ± 0.18**
		H0-mini (Frozen)	95.90 ± 0.45	95.66 ± 0.24	98.21 ± 0.20	95.91 ± 0.11
		H0-mini (Fine-tuned)	**96.79 ± 0.51**	96.17 ± 0.38	97.63 ± 0.85	94.27 ± 1.05
	ResNet-18	Frozen	89.72 ± 1.94	88.10 ± 1.94	81.96 ± 1.60	79.46 ± 1.71
		Full E2E	**94.76 ± 0.60**	**92.36 ± 1.06**	**87.80 ± 2.66**	80.68 ± 3.54
		Frozen + Aug.	91.77 ± 0.37	90.41 ± 0.56	80.82 ± 0.48	78.00 ± 0.40
		Full E2E + Aug.	92.20 ± 1.00	91.06 ± 0.43	85.96 ± 3.03	**81.66 ± 1.17**
DTFD	Foundation Model	H-optimus-1	95.10 ± 1.30	95.76 ± 0.69	97.93 ± 0.15	95.76 ± 0.29
		Prov-GigaPath	95.61 ± 0.18	95.26 ± 0.21	97.68 ± 0.36	93.61 ± 0.32
		UNI2-h	95.95 ± 0.94	96.11 ± 0.69	98.53 ± 0.27	**96.06 ± 0.43**
		H0-mini (Frozen)	95.54 ± 1.72	95.82 ± 1.03	**98.64 ± 0.46**	95.94 ± 0.57
		H0-mini (Fine-tuned)	**97.88 ± 0.30**	**96.78 ± 0.22**	97.53 ± 0.43	94.96 ± 0.27
	ResNet-18	Frozen	92.10 ± 0.28	89.93 ± 0.36	83.02 ± 1.16	79.06 ± 1.02
		Full E2E	94.62 ± 1.36	**93.81 ± 0.45**	**89.72 ± 3.61**	87.11 ± 0.96
		Frozen + Aug.	92.38 ± 0.36	90.37 ± 0.26	85.48 ± 1.22	82.29 ± 0.87
		Full E2E + Aug.	**94.75 ± 1.18**	92.83 ± 1.17	87.56 ± 2.59	**89.30 ± 2.13**
TransMIL	Foundation Model	H-optimus-1	95.71 ± 1.91	**96.67 ± 0.36**	98.22 ± 0.32	95.89 ± 0.58
		Prov-GigaPath	95.90 ± 1.39	96.29 ± 0.52	97.77 ± 0.27	95.20 ± 0.75
		UNI2-h	93.50 ± 1.83	96.09 ± 1.45	**98.50 ± 0.13**	**96.35 ± 0.46**
		H0-mini (Frozen)	95.41 ± 1.60	96.37 ± 0.58	98.30 ± 0.27	96.18 ± 0.31
		H0-mini (Fine-tuned)	**96.64 ± 0.53**	94.12 ± 0.55	96.27 ± 0.88	93.72 ± 1.44
	ResNet-18	Frozen	89.70 ± 1.82	88.82 ± 1.68	76.59 ± 2.49	74.12 ± 2.90
		Full E2E	93.65 ± 1.73	92.92 ± 1.63	79.62 ± 4.20	79.34 ± 3.50
		Frozen + Aug.	90.96 ± 0.50	89.10 ± 0.53	81.45 ± 2.53	80.09 ± 2.41
		Full E2E + Aug.	**94.06 ± 1.25**	**93.31 ± 0.86**	**85.01 ± 4.38**	**83.79 ± 1.86**

B Runtime scaling analysis

To analyse how per-iteration runtime t scales with input resolution, we fit the model

$$t(A) \;=\; a\,A^{b} + c,$$

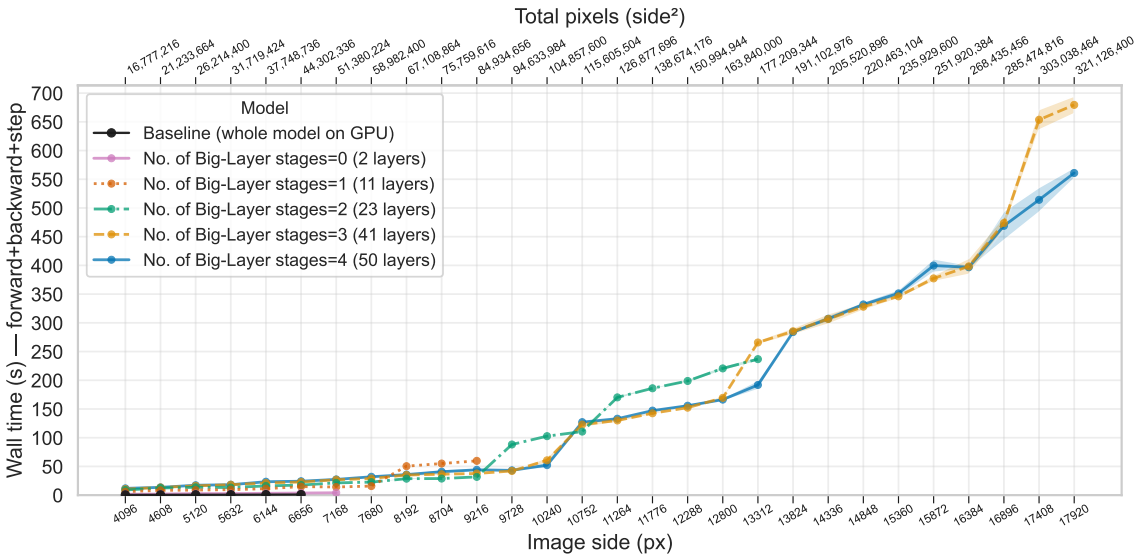

Figure 3: Incremental-stage benchmark for ResNet50: mean wall time per training iteration (forward + backward + optimiser step) versus input side length. Shaded bands denote ±1 standard deviation across repeated iterations. The top axis reports image area (side^2). Each curve corresponds to a model configuration where the number in brackets indicates the number of Big-Layers the model uses. Benchmarks were run on an NVIDIA A100 (80 GB).

where A denotes image area (pixels), $a \geq 0$ is a scale factor, $b \geq 0$ is the power-law exponent and $c \geq 0$ is an additive overhead to capture constant per-iteration costs.

For each model configuration we report:

- the fitted parameters a, b, c;
- the coefficient of determination R^2;
- the *doubling factor* computed at the median tested area,

$$F_{\text{dbl}} = \frac{a(2A_{\text{med}})^b + c}{a(A_{\text{med}})^b + c},$$

  which reports how many times slower an iteration becomes when the image area doubles;
- the number of measured points n used in the fit.

We report the results of the analysis in table 5.

C Partitioned convolution

algorithm 3 and algorithm 4 detail the implementation of the Convolution layer using our method.

Note: patches include halo (overlap) rows to guarantee correct convolution outputs at patch boundaries; during the backward pass overlapping gradient contributions to ∇X are accumulated (reduced) when stitching. Gradient accumulation for ∇W is performed incrementally on the chosen accumulation device (GPU or CPU) to bound peak memory usage.

Table 5: Power-law-plus-offset fit results for per-iteration wall time, $t(A) = a\,A^b + c$. Columns list the model configuration, number of measured points n, fitted parameters a, b and c, coefficient of determination R^2 (original scale), and doubling factor F_{dbl} (time multiplier when area doubles at the median tested area). See Section B for benchmarking details.

Model	n	a	b	c	R^2	F_{dbl}
ResNet-18						
Baseline (whole model on GPU)	7	7.43×10^{-9}	1.03	1.69×10^{-2}	0.99	2.00
No. of Big-Layer stages = 0 (2 Big-Layers)	13	1.23×10^{-6}	0.83	1.54×10^{-4}	0.98	1.78
No. of Big-Layer stages = 1 (6 Big-Layers)	26	1.67×10^{-11}	1.48	2.99	0.98	2.48
No. of Big-Layer stages = 2 (10 Big-Layers)	41	2.40×10^{-9}	1.24	3.21×10^{-15}	0.99	2.36
No. of Big-Layer stages = 3 (14 Big-Layers)	57	7.96×10^{-11}	1.41	5.37	0.99	2.57
ResNet-50						
Baseline (whole model on GPU)	6	2.24×10^{-8}	1.00	7.86×10^{-3}	0.99	1.98
No. of Big-Layer stages = 0 (2 Big-Layers)	7	1.80×10^{-6}	0.83	3.21×10^{-4}	0.97	1.77
No. of Big-Layer stages = 1 (11 Big-Layers)	11	2.64×10^{-19}	2.57	2.63	0.93	5.04
No. of Big-Layer stages = 2 (23 Big-Layers)	19	5.24×10^{-14}	1.90	7.07×10^{-4}	0.97	3.74
No. of Big-Layer stages = 3 (41 Big-Layers)	29	5.00×10^{-15}	2.01	9.30	0.97	3.77
No. of Big-Layer stages = 4 (50 Big-Layers)	28	3.13×10^{-12}	1.68	4.14×10^{-14}	0.99	3.20

Algorithm 3 Memory-Efficient Convolution (Forward Pass)

procedure BigConv2DForward(X, W, stride, padding, dilation, max_N)
  Determine maximal patch height from max_N and model dimensions.
  for each batch slice of X **do**
    for each (height) patch of X (include halo rows for kernel support) **do**
      Transfer the patch to GPU (non-blocking).
      Compute local convolution:

$$Y_{\text{patch}} \leftarrow \text{Conv2D}(X_{\text{patch}}, W, \text{stride}, \text{padding}, \text{dilation})$$

      Transfer Y_{patch} back to CPU and place it into its position in Y.
    end for
  end for
  return Concatenated output Y (stitching overlaps / halos if present).
end procedure

Algorithm 4 Memory-Efficient Convolution (Backward Pass)

procedure BigConv2DBackward($X, W, \nabla Y, \text{stride}, \text{padding}, \text{dilation}, \text{max_N}$)
  Determine maximal patch height from max_N and model dimensions.
  Initialize $\nabla W \leftarrow 0$ (accumulator on device) and $\nabla X \leftarrow 0$ (host or preallocated buffer).
  for each batch slice of X **do**
    for each (height) patch of X (use same partitioning/halo policy as forward) **do**
      Extract corresponding slice ∇Y_{patch} (output gradient) for this patch.
      Transfer X_{patch} and ∇Y_{patch} to GPU.
      Compute partial gradients on GPU:

$$\Delta W \leftarrow \text{Conv2D_weight_grad}(X_{\text{patch}}, \nabla Y_{\text{patch}})$$

$$\nabla X_{\text{patch}} \leftarrow \text{Conv2D_input_grad}(\nabla Y_{\text{patch}}, W)$$

      Transfer ΔW to the accumulator location and update ∇W (reduce/accumulate).
      Transfer ∇X_{patch} to host and add into the corresponding slice of ∇X (accumulate where patches overlap).
    end for
  end for
  return $\nabla X, \nabla W$ (and ∇b if bias is present).
end procedure

D Blockwise self-attention for continuous WSI token sequences

While our core method enables the training of individual layers with inputs exceeding GPU memory, applying this to Vision Transformers (ViT) on Whole Slide Images (WSI) introduces distinct challenges: the quadratic memory complexity of Self-Attention with respect to sequence length (L), and the depth of the network which accumulates activations across many blocks. We address these with two specialised strategies that extend our tiling approach.

Memory-Efficient Self-Attention for Long Sequences. Standard Self-Attention requires instantiating the full attention matrix ($L \times L$) or at least holding the full Q, K, V tensors in VRAM, which is infeasible for WSIs where L can reach hundreds of thousands of patches. To resolve this, we propose **Big-FlashAttention** (algorithms 5 and 6), which combines FlashAttention (Dao et al., 2022) with our CPU-offloading strategy. Unlike standard FlashAttention, which assumes inputs fit in GPU memory, algorithms 5 and 6 keep the input sequence X resident in CPU memory. We iterate through the sequence in tiles, loading only small chunks of the input X to the GPU on demand to compute the corresponding Query (Q_i) and Key/Value (K_j, V_j) blocks. By employing the Online Softmax technique Milakov & Gimelshein (2018); Dao et al. (2022), we compute the exact attention output and global normalisation statistics (M_I, Z_I) without ever materialising the full attention matrix or the full input tensor on the GPU. This avoids the quadratic attention matrix and supports much longer sequences, although runtime remains quadratic and the saved log-normalisers grow linearly with sequence length.

Hybrid Gradient Checkpointing and LwF for Foundation Models. While algorithms 5 and 6 address the quadratic self-attention bottleneck for a continuous WSI token sequence, fine-tuning a deep foundation model (e.g., H0-mini) end-to-end on WSIs introduces a second bottleneck: storing intermediate activations for thousands of patches in a bag. Standard gradient checkpointing assumes that the entire input bag fits in GPU memory—a condition that fails for WSI-scale inputs.

To overcome this, we leverage the instance-wise independence of the encoder to construct a **Micro-Batch Whole-Model Checkpointing** approach combined with Learning without Forgetting (LwF) (Algorithm 7). During the gradient-free embedding pass (Phase 1), we process the large input bag of augmented patches in smaller micro-batches using `no_grad` mode, accumulating the final patch embeddings (E_{aug}) on the GPU. We then compute the primary MIL classification loss, as well as the MSE loss between these embeddings and the precomputed embeddings of the unaugmented patches (E_{frozen}) residing on the GPU (Phase 2). Finally, during the backward phase (Phase 3), we sequentially re-run the forward pass for each micro-batch, inject the corresponding precomputed gradients, and backpropagate to accumulate parameter gradients. This whole-model micro-batching significantly reduces PCIe transfer overheads compared to block-wise tiling, while enforcing strict LwF regularisation without memory overflow.

Algorithm 5 CPU-offloaded blockwise self-attention (forward pass)

Require: CPU-resident $X \in \mathbb{R}^{B \times N \times D}$; projections (W_Q, b_Q), (W_K, b_K), (W_V, b_V), (W_O, b_O); heads H; block size C

Ensure: CPU-resident output Y and saved log-normalisers Λ

$d \leftarrow D/H$; $\alpha \leftarrow 1/\sqrt{d}$; initialise Y and Λ
for all query index blocks I of size at most C **do**
  Transfer X_I to GPU
  $Q_I \leftarrow \text{heads}(X_I W_Q + b_Q)$
  $M_I \leftarrow -\infty$; $Z_I \leftarrow 0$; $U_I \leftarrow 0$
  for all key/value index blocks J of size at most C **do**
    Transfer X_J to GPU
    $K_J \leftarrow \text{heads}(X_J W_K + b_K)$
    $V_J \leftarrow \text{heads}(X_J W_V + b_V)$
    $S_{IJ} \leftarrow \alpha Q_I K_J^\top$
    $\widehat{M}_I \leftarrow \max(M_I, \text{rowmax}(S_{IJ}))$
    $R_I \leftarrow \exp(M_I - \widehat{M}_I)$
    $P_{IJ} \leftarrow \exp(S_{IJ} - \widehat{M}_I)$
    $U_I \leftarrow R_I \odot U_I + P_{IJ} V_J$
    $Z_I \leftarrow R_I \odot Z_I + \text{rowsum}(P_{IJ})$
    $M_I \leftarrow \widehat{M}_I$
  end for
  $A_I \leftarrow U_I / Z_I$
  $\Lambda_I \leftarrow M_I + \log Z_I$
  $Y_I \leftarrow \text{merge}(A_I) W_O + b_O$
  Write Y_I to the CPU output and retain Λ_I
end for
return Y, Λ

Algorithm 6 CPU-offloaded blockwise self-attention (backward pass)

Require: CPU-resident X; saved Λ; upstream gradient $G = \nabla_Y \mathcal{L}$; projections; heads H; block size C
Ensure: $\nabla_X \mathcal{L}$ and gradients of all projection parameters

$d \leftarrow D/H$; $\alpha \leftarrow 1/\sqrt{d}$
Initialise all gradient accumulators to zero
for all query index blocks I **do**
    Transfer X_I and G_I to GPU
    $Q_I \leftarrow \text{heads}(X_I W_Q + b_Q)$
    $A_I \leftarrow 0$
    for all key/value index blocks J **do**
        Transfer X_J to GPU
        Recompute K_J, V_J, and $S_{IJ} = \alpha Q_I K_J^\top$
        $P_{IJ} \leftarrow \exp(S_{IJ} - \Lambda_I)$
        $A_I \leftarrow A_I + P_{IJ} V_J$
    end for
    $\bar{A}_I \leftarrow \text{heads}(G_I W_O^\top)$
    $\nabla W_O \leftarrow \nabla W_O + \sum_b \text{merge}(A_I^{(b)})^\top G_I^{(b)}$
    $\nabla b_O \leftarrow \nabla b_O + \sum_{b,i \in I} G_{b,i}$
    $\delta_I \leftarrow \text{rowsum}(\bar{A}_I \odot A_I)$
    $\bar{Q}_I \leftarrow 0$
    for all key/value index blocks J **do**
        Transfer X_J to GPU
        Recompute K_J, V_J, S_{IJ}, and $P_{IJ} = \exp(S_{IJ} - \Lambda_I)$
        $\bar{V}_J \leftarrow P_{IJ}^\top \bar{A}_I$
        $\bar{S}_{IJ} \leftarrow P_{IJ} \odot (\bar{A}_I V_J^\top - \delta_I)$
        $\bar{Q}_I \leftarrow \bar{Q}_I + \alpha \bar{S}_{IJ} K_J$ ▷ Accumulate every key/value block
        $\bar{K}_J \leftarrow \alpha \bar{S}_{IJ}^\top Q_I$
        $K'_J \leftarrow \text{merge}(\bar{K}_J)$
        $V'_J \leftarrow \text{merge}(\bar{V}_J)$
        $\nabla X_J \leftarrow \nabla X_J + K'_J W_K^\top + V'_J W_V^\top$
        $\nabla W_K \leftarrow \nabla W_K + \sum_b (X_J^{(b)})^\top K_J'^{(b)}$
        $\nabla W_V \leftarrow \nabla W_V + \sum_b (X_J^{(b)})^\top V_J'^{(b)}$
        $\nabla b_K \leftarrow \nabla b_K + \sum_{b,j \in J} K'_{b,j}$
        $\nabla b_V \leftarrow \nabla b_V + \sum_{b,j \in J} V'_{b,j}$
    end for
    $Q'_I \leftarrow \text{merge}(\bar{Q}_I)$
    $\nabla X_I \leftarrow \nabla X_I + Q'_I W_Q^\top$
    $\nabla W_Q \leftarrow \nabla W_Q + \sum_b (X_I^{(b)})^\top Q_I'^{(b)}$
    $\nabla b_Q \leftarrow \nabla b_Q + \sum_{b,i \in I} Q'_{b,i}$
end for
return all accumulated gradients

Algorithm 7 Full-bag E2E training by micro-batch recomputation with LwF

Require: Augmented bag $X_{\text{aug}} \in \mathbb{R}^{N \times C \times H \times W}$
Require: Frozen embeddings $E_{\text{frozen}} \in \mathbb{R}^{N \times D}$
Require: Encoder F_θ, MIL head g_ϕ, label y, LwF weight λ, micro-batch size B_μ
Ensure: Gradients $\nabla\theta$ and $\nabla\phi$

Phase 1: Gradient-free embedding pass

Initialise $E_{\text{list}} \leftarrow []$
for $i = 0$ to N step B_μ **do**
  $x_\mu \leftarrow X_{\text{aug}}[i : i + B_\mu]$
  $e_\mu \leftarrow F_\theta(x_\mu)$ in `no_grad` mode
  Append e_μ to E_{list}
end for
$\widetilde{E}_{\text{aug}} \leftarrow \text{stopgrad}(\text{Cat}(E_{\text{list}}))$
Enable gradients on the leaf tensor $\widetilde{E}_{\text{aug}}$

Phase 2: MIL-head and embedding backward pass

$\ell_{\text{MIL}} \leftarrow \mathcal{L}_{\text{MIL}}(g_\phi(\widetilde{E}_{\text{aug}}), y)$
$\ell_{\text{total}} \leftarrow \ell_{\text{MIL}} + \lambda\,\text{MSE}(\widetilde{E}_{\text{aug}}, E_{\text{frozen}})$
Backpropagate once to obtain $\nabla\phi$ and $\Delta \leftarrow \partial\ell_{\text{total}}/\partial\widetilde{E}_{\text{aug}}$

Phase 3: Encoder recomputation and gradient accumulation

Initialise $\nabla\theta \leftarrow 0$
for $i = 0$ to N step B_μ **do**
  $x_\mu \leftarrow X_{\text{aug}}[i : i + B_\mu]$
  $\delta_\mu \leftarrow \Delta[i : i + B_\mu]$
  Recompute $\widehat{e}_\mu \leftarrow F_\theta(x_\mu)$ with gradients enabled
  $\widehat{e}_\mu.\,\text{backward}(\delta_\mu)$ ▷ Accumulates $\nabla\theta$
  Delete $\widehat{e}_\mu$, x_μ, and δ_μ
end for
return $\nabla\theta, \nabla\phi$