MSc Artificial Intelligence
Master Thesis

Towards Single-Stage Self-Supervised Mamba Pretraining on Histopathology Whole Slide Images

by
Zsombor Fülöp
15002861

July 8, 2025

36 ECTS
Period of research: 02.01.2025-08.07.2025

Supervisors:
Prof. dr. Cees G. M. Snoek
Joost van Doorn MSc
dr. Jasper H. J. Linmans

Examiner:
Prof. dr. ir. Clarisa I. Sánchez Gutiérrez

Second reader:
Prof. dr. Cees G.M. Snoek

Universiteit van Amsterdam

kaiko.ai

"Science is admitting what we don't know."

– Interstellar

Acknowledgements

First, I would like to thank my academic supervisor dr. Cees Snoek. His feedback was sometimes negative but always constructive and to the point, consistently challenging me and pushing me toward the end goal. I am grateful for his genuine interest in the topic and for the time he managed to dedicate to our meetings despite his busy schedule.

I would also like to thank Joost van Doorn for his support throughout every stage of this project. Our many in-depth discussions and his technical guidance allowed me to make real progress with the implementation and experimentation. I would also like to thank dr. Jasper Linmans for his helpful feedback on the report and for the many valuable discussions we had.

Although not an official supervisor, I am also very grateful to dr. Fei Tang for joining our weekly meetings and sharing thoughtful suggestions and creative ideas that significantly improved my work.

I would also like to thank Kaiko.ai for the opportunity to conduct this research as part of a paid internship. The provided computational resources were essential to completing this project. I am especially grateful to the team for creating an inspiring environment and for the enjoyable company activities that made me feel truly part of the team.

Finally, I want to thank my partner and friends for their constant encouragement and understanding. Their emotional support has been invaluable, and I could not have completed this journey without it.

Contents

Abstract

Whole slide images (WSIs) in histopathology present an important yet challenging domain for computational pathology due to their gigapixel scale and rich spatial complexity. While most current approaches rely on multi-stage architectures that separately encode patches before aggregating them, this thesis investigates whether a single-stage, end-to-end encoder can be trained in a self-supervised manner. We adapt an efficient masked image modeling framework (LatentMIM), which learns by reconstructing and discriminating latent representations rather than raw pixels. We analyze training dynamics and evaluate embedding quality using linear probing and unsupervised metrics, identifying points of failure such as the predictor network's inability to leverage semantic information and its tendency to collapse. Motivated by DINO's success in the computational pathology domain, we also adapt it to WSI scale and highlight significant inefficiencies and limitations. Due to the quadratic memory requirement of self-attention, ViTs cannot be used for end-to-end WSI representation learning. Therefore we build on the recently proposed Pixel-Mamba architecture—a State Space Model with linear complexity. We reimplement Pixel-Mamba from scratch and validate in supervised settings. Our results show that while Pixel-Mamba is effective in supervised settings, selecting suitable self-supervised objectives for end-to-end WSI modeling remains highly challenging. We suggest that future work should focus on developing a stronger theoretical understanding of Mamba-based models, with particular attention to Pixel-Mamba's unique component, the Region Fusion module, to enable effective self-supervised training strategies for end-to-end WSI modeling.

Chapter 1

Introduction

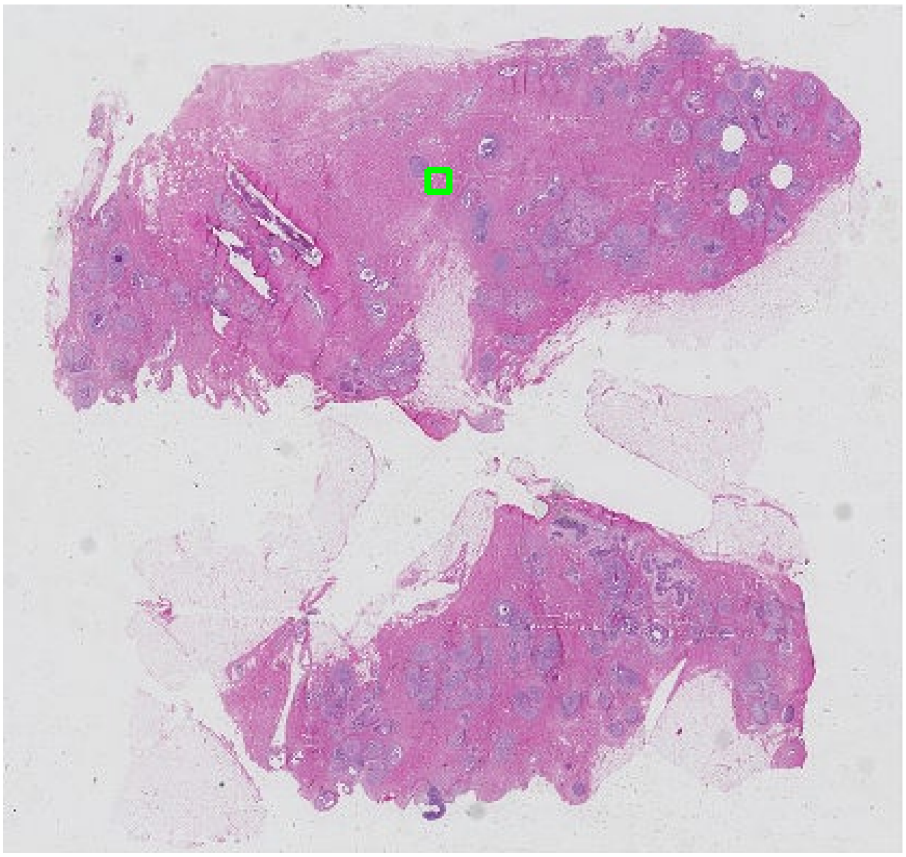
1.1 Whole Slide Imaging in Histopathology

Histopathology whole slide images (WSIs) are high-resolution, digitized scans of tissue samples commonly used in clinical diagnostics and biomedical research. They provide rich morphological details at the cellular level, enabling accurate assessment of disease states and patient conditions. However, the detailed nature of WSIs introduces significant computational and perceptual challenges due to their enormous size. For instance, a standard prostate core biopsy—measuring approximately 15 mm × 3 mm—scanned at the typical clinical magnification of 40x (0.25 microns per pixel) results in an image of about 60,000 × 12,000 pixels. Larger samples, such as those obtained from surgical resections, can easily exceed dimensions of 100,000 × 100,000 pixels, producing gigapixel-scale images. Figure 1.1 illustrates a full WSI and examples of extracted patches at varying magnifications (20×, 10×, 5×), highlighting the change in resolution.

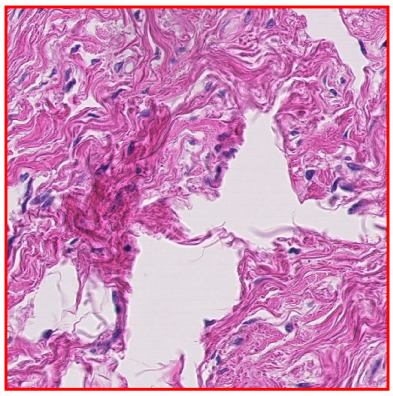
Due to these extreme resolutions, two major challenges arise even for human observation. (1) loading and processing an entire WSI at full resolution is computationally expensive and often infeasible with standard memory and display hardware. (2) Even if such hardware were available, clinicians cannot interpret the full-resolution image in its entirety, unlike lower-resolution modalities such as X-ray, dermatoscopic photos or 2D slices of CT and MRI. To mitigate these issues, clinicians often begin by reviewing lower-magnification overviews and then zoom into regions of interest. This manual approach, however, is labor-intensive and can lead to variability in diagnostic decisions. Studies have shown substantial inter-observer and intra-observer variability among pathologists when evaluating the same WSI, highlighting the subjectivity and difficulty of several pathological tasks [1, 2].

1.2 Challenges and Advances in Deep Learning for WSIs

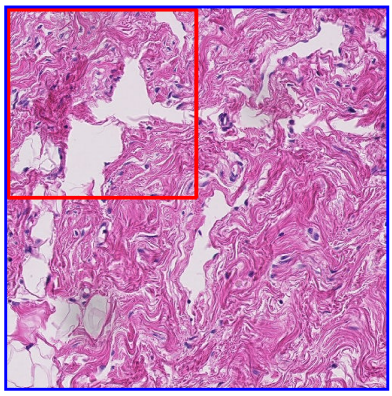
To reduce human labor and improve diagnostic accuracy, substantial research in computational pathology has focused on supervised learning tasks such as cancer grading (assessing tumor aggressiveness), cancer staging (determining disease progression), cancer sub-typing (categorizing based on molecular characteristics) and survival analysis (estimating patient prognosis) among others. However, the size of the images complicates straightforward end-to-end modeling using the full context of the WSI. In fact, WSIs are typically split into smaller patches, which are processed independently by the neural network. The predictions from these patches are then aggregated—either through predefined heuristics or learned mechanisms—to produce slide-level inferences [4, 5, 6]. Next, we discuss key challenges related to data quality, model architecture, and computational efficiency.



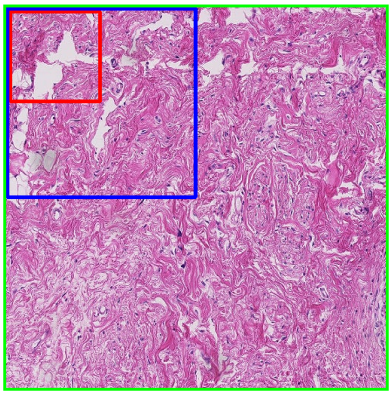
(a) A WSI of size 100,352 by 106,496



(b) A patch at 20x magnification with 0.5 μm/pixel



(c) A patch at 10x magnification with 1 μm/pixel



(d) A patch at 5x magnification with 2 μm/pixel

Figure 1.1: A whole slide image (WSI) **(a)**, and patches extracted at various magnifications from the same WSI **(b–d)**. The colored regions indicate areas with equal resolution. The slide was scanned using a Philips UFS 1.6.1.3 RA scanner, which has a micrometer-per-pixel ratio of 0.25 at 40x magnification. This figure is reused from "Whole-Slide Image Classification in Digital Pathology Using Deep Learning" by P. Pham, 2021 [3].

Annotation Challenges and Data Variability

The high cost of obtaining expert annotations—along with label noise caused by inter- and intra-observer variability—has constrained the scalability of supervised models. Additionally, WSIs often vary significantly across scanners and institutions due to differences in staining protocols, image preprocessing pipelines, and patient populations. Variations in demographics, disease prevalence, and clinical practices can lead to differences in tissue appearance, further hindering the generalizability of supervised models in clinical settings [7].

As a result, there is increasing interest in self-supervised learning (SSL) techniques, which aim to learn robust visual representations without requiring manual labels using large datasets. Recent work has focused on developing general-purpose WSI encoders that can be fine-tuned for various downstream tasks with minimal supervision. These approaches employ transformer-based architectures, which have demonstrated superior scalability with large datasets and SSL objectives compared to traditional CNNs [8]. The most common strategy is multi-stage pre-training: first, a patch-level encoder is trained using SSL, and then a slide encoder is trained to aggregate patch embeddings into slide-level representations using another SSL objective with the patch encoder being frozen in the second stage [9, 10, 11].

Spatial Hierarchy Bias

While multi-stage architectures are computationally efficient—and often necessary due to GPU memory constraints—they impose a strong inductive bias toward spatial hierarchy. This assumes that low-level information should be processed locally before being aggregated globally into high-level features—a structure that may not always align with the nature of histopathology data. Relevant patterns in WSIs can appear at different spatial scales. Important diagnostic signals may also depend on long-range interactions between low-level features, such as the spatial arrangement of cellular structures across distant regions. These dependencies may be lost when using fixed patch-based aggregation. End-to-end slide-level models can avoid this limitation by learning directly from the full input, allowing the model to determine which regions and scales are most relevant.

Supporting this argument, Vision Transformers (ViTs) [12] have shown that the spatial hierarchy bias inherent to CNNs is not universally beneficial—even on natural images. Without hard-coded spatial assumptions, ViTs can incorporate global context from early layers via self-attention. When trained on sufficiently large and diverse datasets, they learn more appropriate inductive biases and often outperform models with fixed hierarchies. Recent work [13] has even shown that removing the locality constraint of standard 14×14 or 16×16 patch embeddings in ViTs can further improve performance. These findings suggest that while spatial hierarchy and locality align with human intuition, enforcing them architecturally may constrain generalization.

Quadratic Memory Complexity

Yet this flexibility comes at a cost: the quadratic memory complexity of standard self-attention with respect to input sequence length. Recent developments in efficient transformer architectures [14] aim to overcome this bottleneck. Approaches such as sparse attention mechanisms [15, 16], linear attention using low-rank approximations [17, 18], and token reduction strategies [19, 20] among others have made it feasible to build attention-based models capable of handling higher-resolution inputs. Another promising direction involves attention-free alternatives, such as State Space Models (SSMs) like Mamba [21, 22], and linear recurrent networks like xLSTM [23], both of which offer linear memory complexity allowing modeling with an order of magnitude larger context. These models have recently shown competitive performance in vision

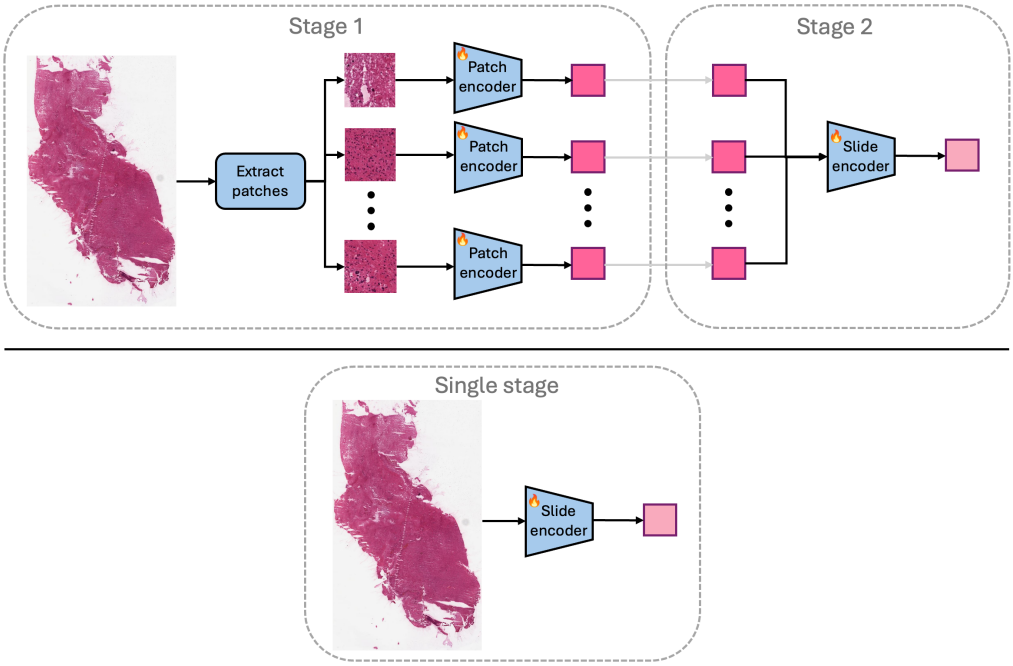


Figure 1.2: Top: Two-stage WSI encoder. In the first stage, patches are extracted from WSIs, and a patch encoder is trained on these patches, treating each patch as an individual sample. In the second stage, the trained patch encoder is used to generate embeddings for the patches of each slide. A slide encoder is then trained on the set of embeddings, where each sample corresponds to the set of patch embeddings of a single slide. **Bottom:** Single-stage WSI encoder. The slide encoder is trained directly on the raw WSIs without intermediate patch-level training. This thesis focuses on the single-stage approach, aiming to develop a robust self-supervised framework for WSIs.

tasks [24, 25, 26], yet remain underexplored in the context of WSI embedding. To the best of our knowledge, only three studies have attempted to process WSIs in a single stage. Pinckaers et al. (2019) [27] define a streaming algorithm for CNNs in which the convolutions are applied only within a window, which is then shifted, accumulating the gradients at each shift. Wang et al. (2023) utilize the LongNet [16] architecture, which uses a sparse attention mechanism, while Qiu et al. (2024) propose a novel architecture based on Mamba, combining token merging and low-magnification inputs to build an efficient end-to-end architecture [28]. The difference of a two-stage and single-stage WSI encoder is showcased in Figure 1.2.

1.3 Thesis Objectives and Contributions

This thesis aims to investigate whether a single-stage generalist encoder for whole slide images can outperform the prevalent multi-stage architectures commonly used in computational pathology. Specifically, the study explores the effectiveness of Pixel-Mamba [28], an attention-free architecture based on State Space Models (SSMs), as a scalable and efficient solution for direct slide-level modeling. Pixel-Mamba incorporates a Region Fusion module, which has shown promise in addressing the high spatial redundancy characteristic of WSIs [29], thereby reducing computational overhead while preserving task-relevant information.

Given the absence of publicly available code, this work includes a complete re-implementation of Pixel-Mamba based solely on the original paper. The model is first pretrained on the ImageNet-1k classification task to validate correctness, and subsequently adapted to histopathol-

ogy data. A core objective of this thesis is to explore efficient self-supervised learning (SSL) techniques for pretraining Pixel-Mamba, as common methods in the natural image domain often do not scale to the resolution of WSIs. Two SSL frameworks are investigated: LatentMIM [30], a latent masked image modeling approach, and DINO [8], a self-distillation-based method.

The main contributions of this work are as follows:

- A full re-implementation of the Pixel-Mamba architecture, validated on ImageNet-1k with a supervised objective. The reproduced model achieves a top-1 accuracy within 0.3% of the result reported in the original paper.
- Extension of prior work by fine-tuning the ImageNet-1k pretrained Pixel-Mamba on the PANDA dataset [31], achieving a QWK score of 0.87, comparable to similarly sized models without extensive hyperparameter tuning or label noise handling.
- Exploration of self-supervised pretraining of Pixel-Mamba using LatentMIM, adapted to WSI-scale and the model's architectural constraints. Includes investigation of design alternatives, qualitative evaluation of embeddings, and discussion of reasons for suboptimal performance. Potential research directions to mitigate these failures are provided.
- Adaptation of the DINO framework to WSI-scale for Pixel-Mamba pretraining, highlighting inefficiencies and limitations of WSI-level DINO pretraining.

The remainder of this thesis is organized as follows. Chapter 2 reviews related work on attention-free architectures, self-supervised visual representation learning, and both multi-stage and single-stage approaches for WSI modeling. Chapter 3 describes the Pixel-Mamba architecture, highlighting design choices made during reimplementation and the adaptation of two SSL methods to WSI modeling: LatentMIM and DINO. Chapter 4 presents the experiments conducted and analyzes the obtained results. Chapter 5 discusses the findings, limitations, future directions, and the resources used in this study. Finally, Appendix A provides supplementary details on the model architecture and additional visualizations.

Chapter 2

Related Work

Below we survey related work from multiple angles. First, we consider recent attention-free models (Section 2.1). Next, we describe self-supervised approaches for learning image representations (Section 2.2). Finally, building on these foundations, we cover influential and relevant multi-stage (Section 2.3) and single-stage (Section 2.4) WSI modeling approaches. This division highlights three key aspects of this study: (1) identifying an attention-free model, (2) selecting an efficient self-supervised learning method, and (3) integrating them into the WSI domain.

2.1 Alternatives for Attention

Transformer architectures have proven highly effective across various sequence modeling tasks in both language and vision among others. However, their core self-attention mechanism suffers from quadratic memory complexity with respect to sequence length, limiting scalability to longer contexts. To address this, a range of linear attention variants has been proposed [14]. While these reduce computational costs, they often sacrifice a key advantage of attention: the ability to flexibly attend to tokens regardless of their position in the sequence.

More recent research seeks to replace attention altogether. One promising direction is Structured State Space Models (S4) [32, 33], which are inspired by classical control theory and the Kalman filter [34]. S4 also bridges ideas from CNNs and RNNs: it allows parallelization during training (like CNNs) and supports efficient autoregressive inference by reusing hidden states (like RNNs), avoiding the need to recompute representations at every step as Transformers do. While S4 achieves linear complexity in sequence length, it struggles with tasks requiring selective focus or forgetting.

To overcome these limitations, Mamba [21] extends S4 by introducing input-dependent parameterization of SSM parameters, analogous to how Transformers compute dynamic queries, keys, and values. This yields the Structured State Space Sequence Model with Selective Scan (S6), which Mamba implements efficiently using hardware-aware techniques. Like Transformers, Mamba uses a block-wise architecture, but replaces self-attention with S6. While preserving linear complexity, Mamba has demonstrated performance comparable to transformer-based large language models, gaining significant traction in the research community. Soon after its initial introduction, Mamba was also successfully applied to vision tasks [24, 25]. More recently, some studies have drawn interesting theoretical connections between self-attention and S6 [22, 35]. For further details on SSMs and Mamba, we refer to a blog post [36] and a survey paper [37].

In parallel, xLSTM [23] proposes a different approach by linearizing and parallelizing LSTM architectures to handle long sequences efficiently. "Were RNNs All We Needed?" [38] evaluates minimal linearized versions of classic RNNs (vanilla RNNs, GRUs, LSTMs) and finds their performance surprisingly strong and similar to each other, xLSTM and Mamba.

In this study, we adopt a Mamba-based architecture for two main reasons: (1) its strong empirical performance in recent literature on WSI modeling [28, 39], and (2) its theoretically elegant design and growing acceptance in the research community—both of which enhance its legitimacy.

Despite these advantages, SSM-based models still face several open challenges. Some studies report performance saturation or suboptimal scaling when applied to extremely long contexts [40, 41]. Others suggest that pure Mamba-based language models struggle with tasks requiring strong dense retrieval capabilities and in-context learning [42, 43] and propose attention-SSM hybrid architectures to get the best of both worlds. Currently, SSMs do not consistently outperform attention-based models across all domains, but show clear promise for long-context perception tasks such as the focus of this study. As a result, experimenting with SSMs remains both exciting and uncertain. The field is rapidly evolving, and a deeper theoretical understanding of these models' capabilities and limitations is still emerging.

2.2 Self-Supervised Learning for Visual Representation

In their comprehensive work, "A Cookbook of Self-Supervised Learning," Balestriero and Ibrahim et al. (2023) [44] categorize SSL for image understanding into four main groups. Of particular relevance to this work are the self-distillation and masked image modeling (MIM) families. These approaches are however often combined, with methods leveraging elements from both.

Self-Distillation

Self-distillation methods feed two different augmented views of the same image to a pair of encoders—referred to as the online (or student) and target (or teacher) networks—and train the student to match the teacher's output. This process allows the model to extract the high-level semantic features common in the views while discarding irrelevant variations. To avoid representation collapse, the pioneering work BYOL (Bootstrap Your Own Latent) [45] introduced an important asymmetry: only the student network is updated through gradient descent, while the teacher network is updated as an exponential moving average (EMA) of the student's weights. This strategy proved effective and was widely adopted in subsequent joint optimization frameworks [8, 46, 47, 30, 48].

DINO (Distillation with No Labels) [8], a method most similar to BYOL, introduced additional mechanisms such as centering and sharpening of feature representations to further stabilize training. DINO pretraining for Vision Transformers (ViTs) led to impressive emergent properties, including zero-shot segmentation—capabilities previously unseen in supervised approaches. iBOT [49] combines the DINO objective with masked reconstruction in the latent space. DINO v2 [46] further refines iBOT with improved regularization strategies and a larger training dataset, representing the current state-of-the-art in SSL for visual representation learning from natural images. These methods typically rely heavily on augmentations such as color jitter, solarization, and blurring, which creates an additional overhead. Nevertheless, due to the demonstrated broad adoption of DINO pretraining, it is shortly investigated in this work as a self-supervised strategy for pretraining Pixel-Mamba on WSIs. We opt for DINO v1 due to its simplicity, despite the stronger performance of v2.

Masked Image Modeling

The Masked Image Modeling (MIM) family is inspired by the success of inpainting autoencoders and Masked Language Modeling (MLM) in NLP architectures like BERT [50]. One of

the most well-known implementations is the Masked Autoencoder (MAE) [51], which adapts MLM to vision by randomly masking a large portion of image patches and training an encoder–decoder architecture to reconstruct the missing content. Specifically, MAE processes only visible patches with an encoder, while a lightweight decoder reconstructs the original image from the encoded patches and mask tokens. MAE shows that reconstructing up to 75% of the masked image provides a challenging and effective pretext task for learning robust and transferable representations.

Despite its initial success, reconstructing raw pixels can be suboptimal—both theoretically and computationally—due to high variance in pixel space and a focus on fine-grained details that may not benefit semantic understanding. As a result, pixel-based MIM methods often exhibit limited robustness to distribution shifts, which is reflected in their lower linear probing performance compared to methods like DINO. I-JEPA [47] addresses these limitations by reconstructing the latent embeddings of masked regions rather than reconstructing pixel values. Its architecture follows the self-distillation setup and includes two encoders and a predictor network. The model learns to infer latent representations of masked areas based on context from visible patches. By applying the loss in embedding space, I-JEPA encourages the learning of higher-level semantic representations and has demonstrated strong performance on downstream tasks such as linear classification, particularly when used with large ViT backbones. The high-level semantic features and the faster convergence compared to pixel-level MIM and the lack of image augmentations as opposed to DINO and iBOT inspired a line of work in this direction.

"Towards Latent Masked Image Modeling for Self-Supervised Visual Representation Learning" by Wei et al. (2024) [30] provides a thorough analysis of the components of latent MIM, including the asymmetry in joint optimization, the choice of the reconstruction objective, decoder design, and optimal masking ratios. Their proposed model, LatentMIM, simplifies and optimizes the I-JEPA framework, achieving competitive performance with smaller backbones and exhibiting clustering behavior in learned features similar to what was observed in DINO. As LatentMIM has been proven high efficiency on natural images without the excessive use of augmentations, effort to adapt the LatentMIM pretraining to WSIs with Pixel-Mamba is carried out in this work.

A notable recent development, "Cluster and Predict Latent Patches for Improved Masked Image Modeling" [48], pushes the boundaries of latent MIM further. This work approaches the performance of DINO v2 while requiring significantly fewer training FLOPs, demonstrating the potential of latent-space MIM approaches. However, since this method was published during the timeframe of this study and its implementation is not yet publicly available, it is not applied in this work.

2.3 Multi-Stage WSI Modeling

Multiple Instance Learning

A common paradigm in multi-stage WSI modeling is Multiple Instance Learning (MIL), where each whole-slide image is treated as a bag of instances—typically patch embeddings—and only a slide-level label is available during training. In standard MIL, the model assumes that at least one patch in a positive bag is positive and uses aggregation methods such as max or average pooling to derive slide-level predictions. Attention-based MIL (ABMIL) [52] improves upon this by introducing a learnable attention mechanism that assigns weights to each patch based on its relevance to the task, allowing the model to focus on the most informative regions. While ABMIL has demonstrated strong performance and remains widely used in the field, it requires supervised labels and cannot be used to learn general-purpose WSI embeddings in a

self-supervised manner.

HIPT

One of the pioneering approaches in multi-stage modeling for WSIs is the Hierarchical Image Pyramid Transformer (HIPT) [53], which employs a hierarchical pretraining scheme. In the first stage, a ViT-16 patch encoder is trained on 256×256 pixel patches using the DINO [8] self-supervised learning framework. This encoder learns to extract cellular-level features and aggregate them into patch-level embeddings. In the second stage, a separate ViT model is pretrained using DINO again, but restricted to crop-based transformations due to the input being embeddings already. This encoder ultimately learns to aggregate the frozen patch embeddings into higher-level regional representations. These regions correspond to larger areas of 4096×4096 pixels (i.e., 16×16 patch grids). Finally, in the third stage, a third ViT model aggregates the regional embeddings into a slide-level representation, optimized directly for the downstream task objective.

This decoupled optimization strategy significantly improves training efficiency, as gradients are only propagated through one transformer at a time with a limited spatial context. HIPT was pretrained on The Cancer Genome Atlas (TCGA) at 20× magnification, using 408,218 extracted regions and 104 million patches. When the third encoder is trained for downstream tasks such as cancer subtyping and survival prediction, HIPT achieved strong performance gains over more traditional weakly supervised narrow patch-encoder models using (attention-based) Multiple Instance Learning for feature aggregation for the slide-level.

Prov-GigaPath

Another influential contribution is Prov-GigaPath [9], a two-stage, purely self-supervised model trained on a significantly larger non-public dataset. In the first stage, a patch-level Vision Transformer (ViT) is pretrained using DINOv2 [46] on 1.3 billion 256×256 patches extracted from 171,189 whole-slide images (WSIs) at 20× magnification—approximately five times the scale of TCGA. In the second stage, LongNet [16] with its dilated self-attention is employed for slide-level embedding, leveraging MAE pretraining on the frozen patch features. This use of MAE pretraining underscores the potential of masked image modeling with patch-level masking in WSI analysis, which is one of the directions explored in this thesis.

Having a self-supervised pretraining objective, Prov-GigaPath can provide good quality general embeddings on various datasets suitable for training lightweight classifiers on top. In linear probing evaluations, Prov-GigaPath outperforms previous full-finetuned models including HIPT as well as other task-specific models, on 25 out of 26 tasks. Furthermore, its effectiveness is demonstrated in a multimodal setting for aligning pathological images with associated reports, achieving strong performance.

UNI, Virchow, Vim4Path

Several popular patch-level foundation models are commonly used in combination with MIL or other neural architectures for slide-level downstream tasks or pretraining. Notable examples include UNI(2) [54] and Virchow(2) [55, 56], which use DINO v2 or a pathology-specific modification of it. A particularly noteworthy contribution is Vim4Path [39], which employs Vision Mamba [24] as the patch encoder. Vim4Path integrates the efficient state-space architecture of Vision Mamba with DINO-based self-supervised pretraining. Evaluations on both patch-level and slide-level classification tasks (the latter using MIL aggregation) show that Vim4Path achieves performance comparable to ViT-based patch encoders, while offering greater efficiency

due to the lightweight Mamba backbone. Motivated by these positive results, this thesis briefly explores the use of DINO pretraining at the WSI level with the Pixel-Mamba architecture.

Multimodal Approaches

More recent studies have investigated training slide-level encoders using multi-modal data, combining textual pathology reports with patch embeddings from vision-only pretrained encoders. Prominent examples of this approach include TITAN [11] and PRISM(2) [10, 57]. However, since this work focuses exclusively on vision-based analysis of whole slide images, we do not further explore the multi-modal line of research.

2.4 Single-Stage WSI Modeling

LongViT

"When an Image is Worth 1,024 × 1,024 Words: A Case Study in Computational Pathology" by Wang et al. (2023) explores a non-hierarchical WSI modeling. The authors divide WSIs into 32×32 pixel patches, which are linearly projected into embeddings and denoted by "words" following the standard procedure in Vision Transformers (ViTs) [12]. For pretraining, they extract 100 random crops of size 1024–1536 × 1024–1536 pixels from each WSI and train a LongNet model on these using the DINO self-supervised learning framework. This pretrained model is referred to as LongViT. The dataset used is TCGA, with WSIs scanned at 20× magnification.

In the fine-tuning stage, a long-context, task-specific training is applied: WSIs are resized to 32,768 × 32,768 pixels, resulting in a 1024 × 1024 grid of words. The authors also conduct an ablation study on input resolution, ranging from 1024 × 1024 and upwards, and find that higher resolutions improve performance in the cancer subtyping tasks they evaluated. However, this trend does not hold in their survival prediction experiments, where larger input sizes do not consistently yield better results. While LongViT often achieves competitive—and in some settings superior—performance compared to HIPT, the benefits of longer context are not uniform across tasks, and the optimal resizing resolution must be carefully selected, as it has a substantial impact on performance. Furthermore, while the heavily pruned dilated attention in LongNet enables linear scaling, it is conceptually limited in its ability to capture fine-grained information from long-range dependencies. Additionally, LongViT cannot function as a generalist encoder in linear probing settings, which significantly limits its applicability compared to multi-stage encoders such as Prov-GigaPath.

Pixel-Mamba

This study builds directly on "From Pixels to Gigapixels: Bridging Local Inductive Bias and Long-Range Dependencies with Pixel-Mamba" by Qiu, Chao and Lin et al. (2024). The authors propose a fundamentally different architecture—Pixel-Mamba—which leverages Mamba Blocks [21] and applies token merging in a manner similar to Bolya et al. (2023) [19], but at the level of 224×224 pixel patches. The selective state space of the Mamba architecture mitigates the quadratic complexity of self-attention with respect to sequence length, while token merging further reduces the linear memory requirements. Despite these efficiencies, training on full-resolution WSIs at 20× magnification remains infeasible with current GPU memory limits, even when token reduction is applied. To address this, the authors use WSIs at a much lower magnification of 2.5×. Unlike ViTs [12] or ViMs [24], which operate on linearly embedded patches (e.g., 16×16), Pixel-Mamba processes raw pixels directly and gradually merges neighboring tokens. This enables richer feature extraction even at low magnification. The design

is reminiscent of CNNs in that the number of pixels represented by a single token increases progressively across layers, but unlike CNNs—whose view is inherently local—Pixel-Mamba maintains global context throughout the network, similar to ViTs. A more detailed explanation of the architecture is provided in Section 3.1.

Pixel-Mamba is first pretrained on ImageNet-1k using a classification objective, with token merging disabled during this stage. The 6.2M parameter variant outperforms both the 7.2M ViM-Ti and the 6M DeiT-Ti models, validating its effectiveness on natural images. The model is then fine-tuned on various tumor staging and survival analysis tasks using five-fold cross-validation, where it consistently outperforms HIPT, LongViT, and multiple MIL-based methods with substantially larger backbones (ranging from 22M to 1.1B parameters).

The authors also compare Pixel-Mamba to Prov-GigaPath and find that it consistently achieves better results—often by a significant margin. However, since Prov-GigaPath was evaluated using a linear probing setup, while Pixel-Mamba was fully fine-tuned, the comparison is not entirely equitable. It is likely that Prov-GigaPath would achieve stronger performance if it were also fully fine-tuned. Still, the findings demonstrate that Pixel-Mamba can outperform much larger models without requiring large-scale pathology-specific pretraining. While most current efforts focus on scaling up datasets to train patch encoders, this study underscores the value of improved architectural design and long-context modeling in driving downstream performance—an insight that directly aligns with the goals of the current study.

As previously mentioned, generalist models pretrained on large unlabeled cohorts tend to be more robust to the variability of WSIs across scanners and institutions, as well as to input and label noise, making them highly promising for clinical applications. Despite its strong results in supervised settings, Pixel-Mamba has not yet been explored as a generalist, pretraining-driven encoder for diverse WSI tasks—a gap this thesis seeks to address.

Chapter 3

Method

The methodology in this chapter is threefold. In Section 3.1, we describe the Pixel-Mamba architecture introduced in [28], along with the assumptions and design choices made during reimplementation. Section 3.2 explains how we adapt LatentMIM [30] self-supervised training to the whole-slide domain, considering the architectural constraints of Pixel-Mamba. Finally, in Section 3.3, we describe the adaptation of the DINO training framework for WSIs, in parallel with LatentMIM.

3.1 Pixel-Mamba Architecture

This section describes the Pixel-Mamba architecture [28] in detail, focusing on its individual modules tailored for WSI modeling. During implementation, a few design choices had to be made due to missing details in the original paper. These decisions are also explained below.

WSI Serialization

Whole slide images often contain a large proportion of background patches, which should be excluded prior to training to improve both computational efficiency and model performance. To address this, a preliminary patch filtering step is applied. Since the original paper does not specify the method used for foreground selection, we reuse the foreground masks previously generated for Kaiko.ai's patch encoder [58]. In that work, a U-Net [59] was trained to segment tissue foregrounds on 512×512 resized TCGA slides using 212 manually labeled masks. The trained model was then applied to all slides at the magnification level closest to a resolution of 1000×1000 to generate foreground masks. Polygons were extracted from these masks and saved. In our setup, we directly use these polygons without modification; they can be easily resized to match other magnification levels. The Pixel-Mamba paper also does not specify the threshold for the minimum foreground ratio per patch, so we default to a value of 0.5.

Following Local-Mamba [60], Pixel-Mamba employs a windowed selective scan—often referred to as a region-based zigzag scan—motivated by the need to preserve local 2D dependencies in a large-scale context. As illustrated in Figure 3.1(a), a whole slide image of size $H \times W \times C$ is composed of n foreground regions (patches) of size $h \times w \times C$; typically defined as $224 \times 224 \times 3$. These n regions are read in row-major order, and the pixels within each region are also flattened in row-major order. Unlike Vision-Mamba [24] and Vision Transformer [12], Pixel-Mamba does not embed pixels into visual tokens ("words") but instead treats raw pixel values as tokens at first. Each region is added a shared learnable CLS token of dimension 3, inserted at the midpoint of the region sequence (position $(224 \times 224 + 1)/2$), resulting in a total of $(224 \times 224 + 1) = 50177$ tokens per region and $H \times W + n = n * 50177$ tokens for the entire slide.

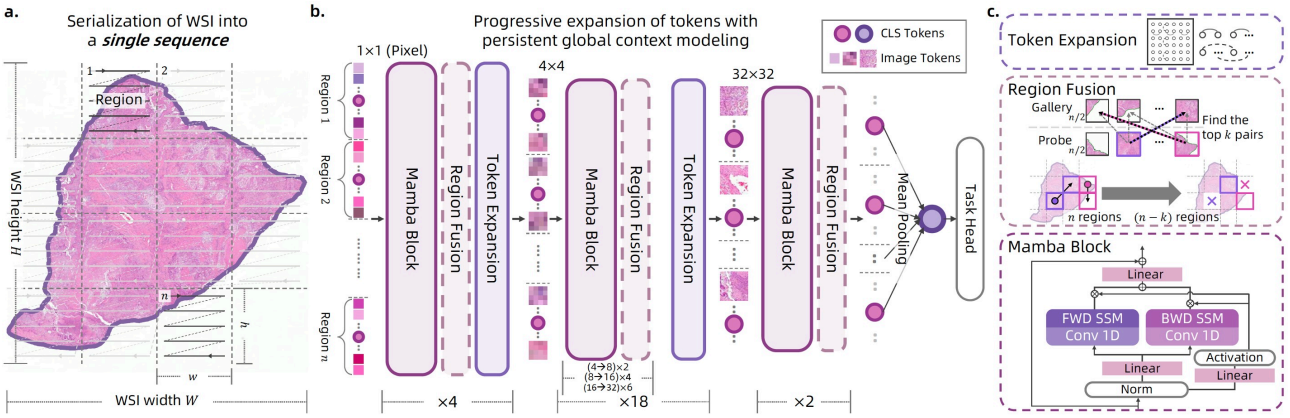


Figure 3.1: Pixel-Mamba Architecture. **(a)** Foreground patches from the WSI are extracted and serialized in per-patch row-major order, with CLS tokens inserted in the center. **(b)** Pixel-Mamba progressively expands the spatial coverage of each token from the input while maintaining global context. Not all layers include all modules; for instance, $(4 \to 8)\times 2$ indicates that two layers use only the Mamba Block and Region Fusion without Token Expansion between token expansions from 4×4 to 8×8 pixels. See A.1 for a detailed architecture overview. **(c)** Illustrations of the Token Expansion, Region Fusion modules, and the Mamba Block. Figure reused from the original Pixel-Mamba paper [28].

Mamba Block

Pixel-Mamba adopts the Mamba Block architecture with a bidirectional state-space model (SSM)—the same design used in Vision Mamba [24]—to capture long-range dependencies (see Fig. 3.1(c), bottom). For an input token sequence t^l at layer l, the forward pass is

$$\begin{aligned} t^{l+1} &= f_y\Big(\mathrm{SSM}_f\big(\psi(x)\big) \otimes z \;+\; \mathrm{SSM}_b\big(\psi(x)\big) \otimes z\Big) + t^l, \\ x &= f_x\big(\mathrm{norm}(t^l)\big), \\ z &= \mathrm{SiLU}\big(f_z\big(\mathrm{norm}(t^l)\big)\big), \end{aligned} \tag{1}$$

where f_* denotes a linear projection; SSM_f and SSM_b are the forward and backward scans of the bidirectional SSM; ψ is a 1D convolution; and $\otimes$ denotes element-wise multiplication.

Token Expansion

Pixel-Mamba progressively increases the number of input pixels each token represents, starting from a single pixel. This approach resembles CNNs, but with the key distinction of preserving global context which is the characteristics of ViTs. The Token Expansion module merges neighboring tokens through a shift merging operation, as illustrated in Figure 3.2. The authors justify this progressive expansion strategy as a way to capture multi-scale representations from the slide, enabling the model to leverage fine-grained, low-level information while inputted a low magnification level WSI such as 2.5x.

Given a region sequence of shape $(h \times w + 1) \times C$ (in the beginning $(224 \times 224 + 1) \times 3$), the CLS token is first removed, and the remaining spatial tokens are reshaped into a 2D grid of size $h \times w$, where each cell is a C-dimensional vector. For vertical expansion, the grid is padded at the top, yielding a $(h + 1) \times w$ array. For horizontal expansion, padding is applied to the left, resulting in a $h \times (w + 1)$ array. A shift-and-merge operation is then performed—shifting down for vertical or right for horizontal—followed by a merge that returns the grid to size $h \times w$. Then, the vertically or horizontally neighboring shift merged tokens can be combined either

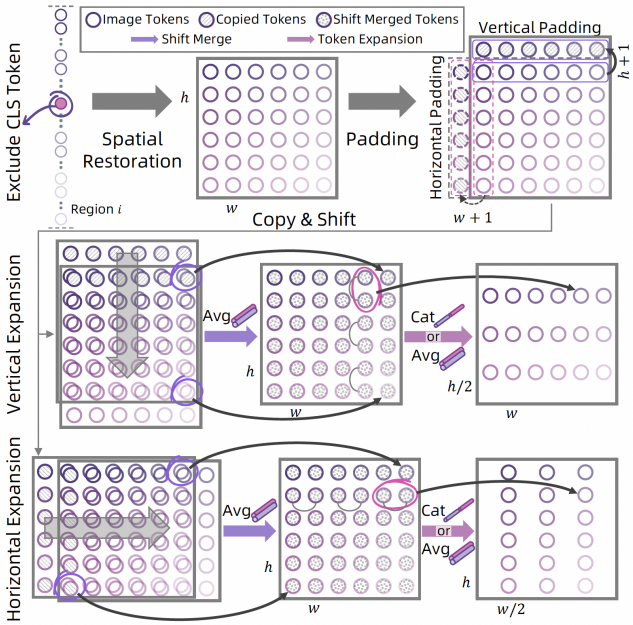


Figure 3.2: Token Expansion module. After temporarily removing the CLS token, each region is reshaped into a 2D grid. Padding is applied along either the horizontal or vertical axis, followed by a corresponding shift-based merging operation. Finally, horizontally or vertically adjacent tokens are combined using either concatenation or averaging. Figure reused from the original Pixel-Mamba paper [28].

by averaging or concatenation. For averaging, the resulting region size becomes $h/2 \times w \times C$ (vertical) or $h \times w/2 \times C$ (horizontal). For concatenation, the size becomes $h/2 \times w \times 2C$ or $h \times w/2 \times 2C$, respectively.

A final step, not specified in the original paper, is the reinsertion of the CLS token. In the case of averaging, we reinsert the CLS token at the new midpoint—either at position $(h/2 \times w + 1)/2$ (vertical) or $(h \times w/2 + 1)/2$ (horizontal). For concatenation, we duplicate the CLS token along the embedding dimension to match the expanded dimensionality of the spatial tokens.

Region Fusion

The Region Fusion module mitigates feature redundancy, shrinking activations and thereby cutting both memory use and runtime. Its design closely resembles the Token Merging approach introduced for ViTs [19], with a key distinction: merging is performed at the regional level rather than at the token level. Given n regions, they are randomly split into two sets—Gallery and

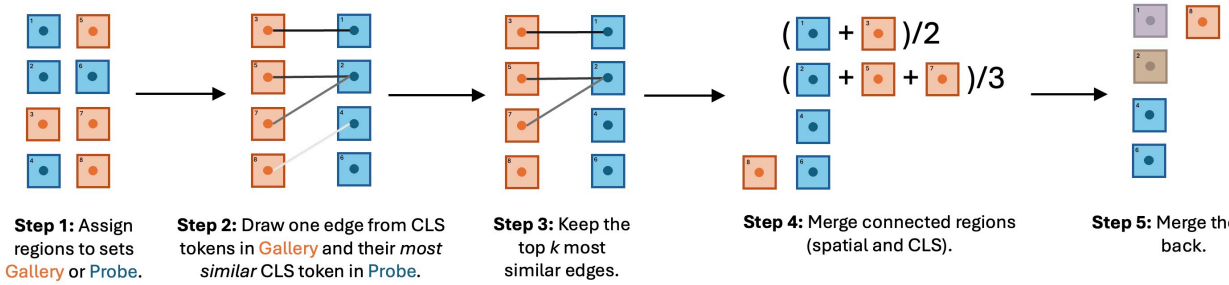


Figure 3.3: Region Fusion module similar to Bipartite Soft Matching from "Token Merging: Your ViT but faster" [19].

Probe—yielding $n/2$ gallery regions and $n/2$ probe regions (or $n/2+1$ if n is odd). For each probe region, the most similar gallery region is identified using cosine similarity between their CLS tokens. The top k most similar pairs are then selected, and the probe regions are fused into their matched galleries, merging both CLS and spatial tokens. The number of fusions, k is defined as $\lceil \alpha * n/L \rceil$ with L the total number of layers in the network and $0 < \alpha < 1$ a hyper-parameter that sets fusion severity. Because k decreases linearly with depth, more regions merge in early layers than in later ones. Following the original ablation study, we use $\alpha = 0.8$.

Notably, a single probe region might be the closest match for multiple gallery regions—a scenario the Pixel-Mamba paper does not discuss. Given the similarity to the Bipartite Soft Matching algorithm in the Token Merging paper, we follow a similar approach and we sum the probe and all affected gallery regions and divide by their count. See Figure 3.3 for a visualization of the Region Fusion module.

Layer Composition and Head

The Pixel-Mamba architecture is composed of stacked Pixel-Mamba layers, each containing a Mamba Block, followed by Region Fusion and, optionally, a Token Expansion module. A detailed layer-wise configuration is provided in Table A.1. At the final stage of the model, all CLS tokens are mean-pooled and passed to a task-specific output head. In the original experiments, this head is typically a linear classifier tailored to the downstream task.

3.2 LatentMIM for WSIs with Pixel-Mamba

LatentMIM [30] extends the Masked Image Modeling paradigm to the latent space, offering more robust and semantically meaningful representations compared to pixel-level reconstruction. This section describes its design rationale and the specific adjustments required for its use with Pixel-Mamba on WSIs.

Region-Level Masking Strategy

The original LatentMIM applies masking to 14×14 pixel patches within 224×224 ImageNet images. However, this patch-level approach is not directly compatible with Pixel-Mamba, due to its Token Expansion and Region Fusion modules, which require consistent and complete regions. Allowing masks to cut through regions would disrupt token merging and necessitate significant modifications to these modules. To avoid this, masking is applied at the region level—where each region is treated as an indivisible unit—after the foreground filtering step (see Paragraph 3.1). This approach aligns with the second-stage encoder training strategy used in GigaPath [9]. In Figure 3.4, we show the masked regions with a blue overlay and the discarded background regions with a black one.

We opt for a 75% masking ratio instead of the 90% used in the original LatentMIM [30]. This decision reflects that much of the redundancy is already eliminated by the Region Fusion module in Pixel-Mamba. In LatentMIM [30], a higher masking ratio was found to be more effective than the 75% typically used in MAE, as latent embeddings are more correlated than raw pixels—making the reconstruction task easier and, therefore, less effective for learning strong features [30]. To maintain training difficulty, LatentMIM increases the masking ratio. However, since Region Fusion already increases information density by merging redundant regions, we lower the masking ratio to avoid making the task overly difficult or unstable in our WSI setting.

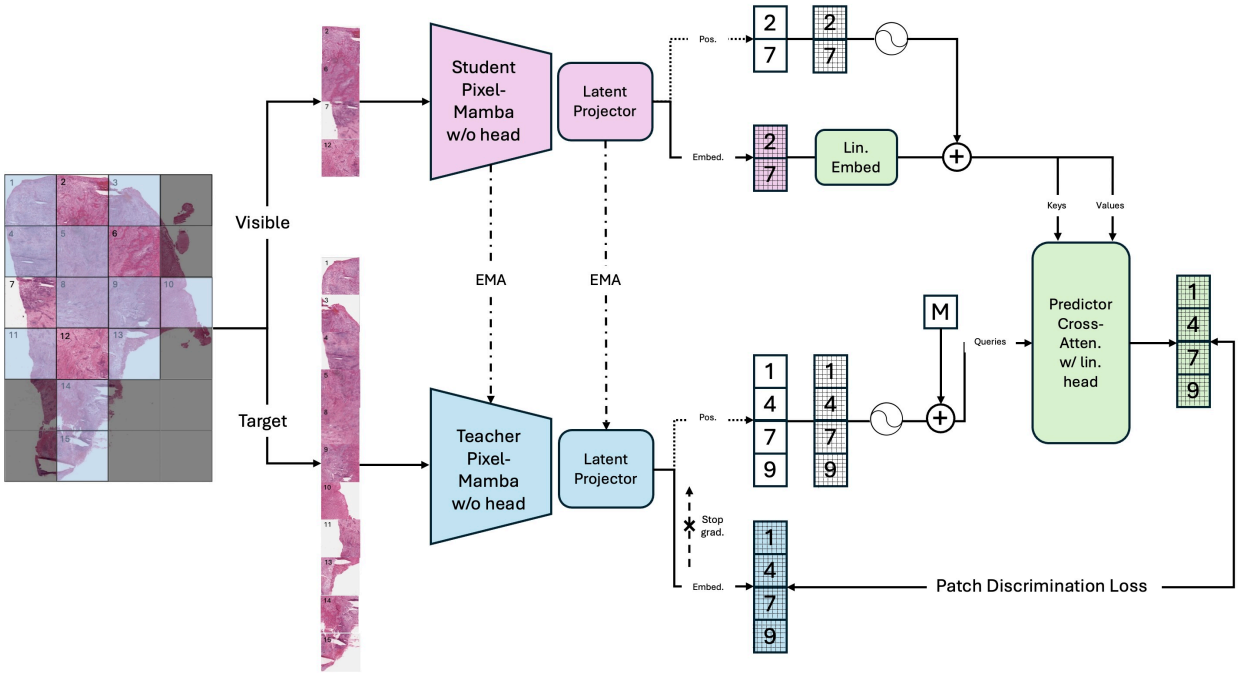


Figure 3.4: Our adaptation of LatentMIM for WSI training with region-level masking using 224×224-pixel regions. Foreground regions are randomly partitioned into *visible* and *target* sets. Target regions are shown with a blue overlay, while visible regions are shown without an overlay. Under a self-distillation setup, the predictor reconstructs target latent tokens from visible latent tokens using spatial context, encouraging the encoder to learn discriminative features.

Student–Teacher Framework, Region Fusion, and Positional Encoding

LatentMIM operates under a student–teacher self-distillation setup. The student encoder embeds only the visible regions, while the teacher—an exponential moving average of the student—embeds the target regions. A three-layer cross-attention predictor (decoder) reconstructs the teacher's outputs based on the student embeddings and the spatial locations of the target tokens, encouraging the encoder to learn semantically meaningful features.

However, Pixel-Mamba applies Region Fusion, which merges multiple spatial regions into a single embedding. This complicates correspondence between the location of the original input regions and the output token sequence. To preserve this mapping, we maintain an index tensor that records which original region IDs are represented in each merged region throughout the forward pass. That means, when a probe region is merged into a gallery region, only the index of the gallery region is retained which is the case for both visible embeddings and the target embeddings.

Based on these region IDs, we can compute token-level positions. Token-level absolute 2D sinusoidal positional embeddings are added to the visible embeddings. The masked (target) tokens' locations are embedded using the same positional encoding scheme, and a randomly initialized CLS token is inserted at the center of each target region. Finally, following the original LatentMIM a zero-initialized shared learnable mask token is added to each target token, forming the input queries to the cross-attention-based predictor.

Beyond the technical adjustments, Region Fusion raises a theoretical concern. While latent prediction from other latents has been shown to work effectively, our setup involves predicting potentially merged latent representations from other potentially merged representations. This deviates from the original LatentMIM setup, where each token corresponds to a fixed image patch. However, given that self-attention also mixes information across tokens based on

similarity, Region Fusion may not introduce a fundamentally different spatial correspondence between latent tokens and image patches. Moreover, the "Token Merging: Your ViT but faster" paper demonstrates that masked autoencoding with token merging remains effective for Vision Transformers, suggesting that merged latent prediction may be effective as well.

Loss Function

LatentMIM leverages an InfoNCE-style contrastive loss. This loss however does not contrast across samples but across patch representations within one sample hence termed as the Patch Discrimination Loss. In the original setup with the ViTs, only one global CLS token is used which is excluded in the loss calculation. In the Pixel-Mamba framework however, we have multiple CLS tokens, one for each region. To encourage rich regional feature representation in these tokens, we experiment with including them in the loss calculation.

Apart from that, we adopt the Patch Discrimination loss as it in LatentMIM:

$$\Delta^{k}_{\text{PatchDisc}} = -\tau \log \frac{\exp\left(-\frac{1}{\tau}, \text{sim}(\hat{z}_k, z_k)\right)}{\sum_{l \in \mathcal{T}} \exp\left(-\frac{1}{\tau}, \text{sim}(\hat{z}_k, z_l)\right)}, \quad \text{sim}(\hat{z}, z) = \frac{\hat{z}^\top z}{||\hat{z}||, ||z||} \tag{3.1}$$

where z_k denotes the teacher's embedding for a target patch k, $\hat{z}_k$ is the predicted latent for the patch k, the set $\mathcal{T}$ contains all target patches and τ is a temperature hyperparameter used with a default value of 0.2. Important to note that while in the original LatentMIM setup that uses ViTs, a final embedded token represents a 14×14 patch from the input, with the Pixel-Mamba, the final tokens represent a 32×32 patch.

Moreover, we omit several components from the original LatentMIM—such as sampling gaps, patch-wise similarity constraint in the loss term, and visual cues—since they only provided marginal gains on ImageNet [30] and do not translate trivially to the WSI domain.

Summary of Key Differences

Relative to the experiment on line 3 of Table 4 in LatentMIM [30], our adaptation introduces the following changes:

- **Different encoder architecture:** Pixel-Mamba-6M is used in place of the ViT-B backbone due to the efficiency of Pixel-Mamba for WSI data.
- **Region-level masking:** We mask 224×224 pixel regions instead of 14×14 patches.
- **Reduced masking ratio:** We use 75% masking instead of 90%, taking into account the redundancy reduction by Region Fusion.

These modifications aim to tailor the strengths of LatentMIM to the characteristics of WSIs and the architectural constraints of Pixel-Mamba.

3.3 DINO Transform for WSIs

To pre-train Pixel-Mamba with DINO at whole-slide scale, the standard DINO data augmentations must be adapted. Two changes are essential; (1) Adapting cropping for background filtered WSI patches and (2) Making sure that other augmentations are applied consistently across patches.

Random Cropping

Original (ImageNet) behaviour. TorchVision's `RandomResizedCrop` proceeds as follows:

1. Sample a target area in the specified scale range (5–40 % for "local" crops, 40–100 % for "global" crops).
2. Sample a random aspect ratio in $\left[\frac{3}{4}, \frac{4}{3}\right]$.
3. Randomly place the crop.
4. Retry up to ten times; otherwise fall back to the entire image.
5. Resize the crop to the required resolution (96×96 for local views, 224×224 for global views).

The operation essentially returns crops that contain a variable amount of semantic content expectedly from different parts of the image with some overlap. The resizing at the end ensures same-sized crops within view types (global or local) allowing for batched processing.

Proposed patch-based algorithm. In WSIs we load only foreground patches for efficiency. If we naively crop, a rectangle covering 40 % of the slide area may include far fewer than 40 % of the foreground patches—especially for needle biopsies, which are long and thin. Moreover, the [3:4, 4:3] aspect-ratio constraint is often incompatible with such elongated tissue regions. We propose the following algorithm:

1. Sample a rectangular window whose area lies in the desired scale range (5–40 % for a local view, 40–100 % for a global view).
2. Draw a random aspect ratio, restricting only that the window fits inside the slide's foreground bounding box.
3. Randomly position the window.
4. Count how many foreground patches fall inside.
   (a) If that count is within the target percentage of foreground patches (5–40 % or 40–100 %, respectively), accept the crop.
   (b) Otherwise, resample from step 1. After 1 000 unsuccessful trials, also relax the rectangular window area range (for local crops only since global views' range is already fully relaxed).
5. If no valid crop is found after 1 000 trials with the enlarged rectangular window, fall back to returning all foreground patches.

This procedure ensures that each view contains the intended proportion of tissue from a consistent rectangular region of the slide, while remaining robust to extreme aspect ratios and sparsely distributed patches. See Figure 3.5 for a visualization. Applying this algorithm to the foreground polygons improves loading efficiency by ensuring that only the regions actually used are loaded.

Unfortunately however, resizing to a given local and global size at the end is not possible with this method. Due to the background filtering, the number of retained regions can vary significantly, and resizing them to obtain a common number of regions is not trivial. Isolated clusters of regions may occur, further complicating the development of a strategy to resize views and enable batching. Moreover, enforcing a specific number of foreground regions in local and global views to achieve a standard size could reduce variability among views.

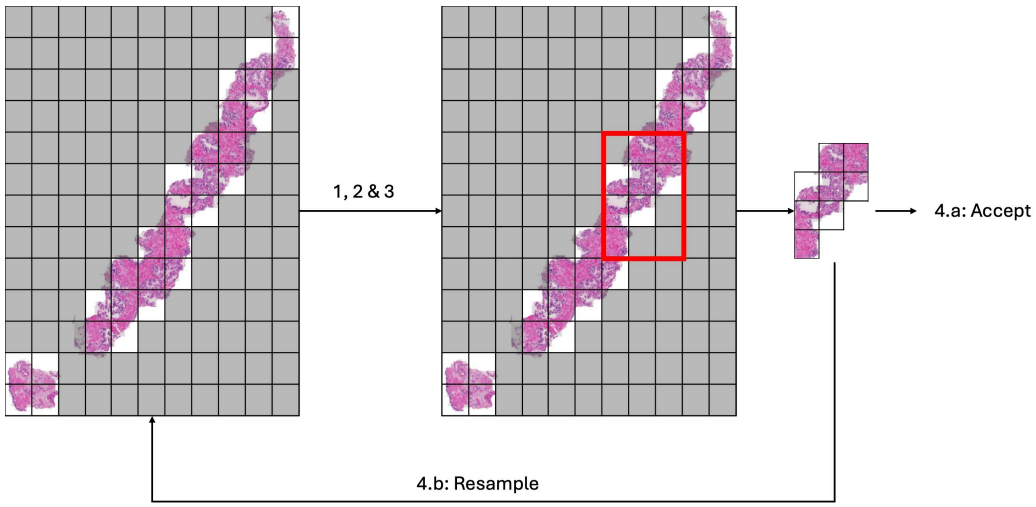


Figure 3.5: Proposed algorithm for cropping background-filtered views from a WSI. A rectangular window is cropped, and if the number of foreground regions falls within the specified scale, it is accepted. Otherwise, the window is resampled and potentially enlarged to increase the likelihood of capturing enough foreground regions.

Other Spatial Transformations and Color Augmentations

To ensure that a given view resembles a globally transformed slide, all color and intensity augmentations—such as color jitter, Gaussian blur, and solarization—must be applied consistently across all patches within that view. This is achieved by sampling the random transformation parameters once per view and applying them uniformly across all its constituent patches.

For spatial augmentations such as horizontal and vertical flips, it is necessary to update the spatial coordinates of each patch accordingly to preserve the correct spatial layout after transformation. In our setup, we use the default DINO hyperparameters for augmentation, with one key modification: we additionally include vertical flipping. This decision is motivated by the fact that pathology slides lack a consistent anatomical orientation—unlike natural images, there is no meaningful 'up' or 'down' in most histological contexts. See visualization of the views of a sample in Figure A.1

Chapter 4

Experiments and Results

This chapter is divided into three parts. Section 4.1 discusses the supervised experiments performed to validate our Pixel-Mamba implementation. Section 4.2 provides details of the LatentMIM experiments, while Section 4.3 covers the DINO pretraining experiments.

4.1 Supervised Experiments

ImageNet-1k Pretraining

To validate our implementation, we follow the Pixel-Mamba paper and pretrain the model on ImageNet-1k using the standard classification objective. This setup does not use Region Fusion because there is only one region of size 224×224. Training is performed on the official training split, and we report top-1 accuracy on the validation set. Since the original Pixel-Mamba paper does not provide detailed training hyperparameters, we adopt the DeiT training recipe [61], which was also used in ViM [24]. For any unspecified settings in the Pixel-Mamba paper, we default to the pretraining configuration of ViM-Ti pretraining. We train on 4 A100s in full precision for approximately 10 days. The training proved to be stable, see Figure 4.1. Pixel-Mamba reaches 77.5% best validation accuracy in epoch 298 which is only 0.3% short of the reported performance.

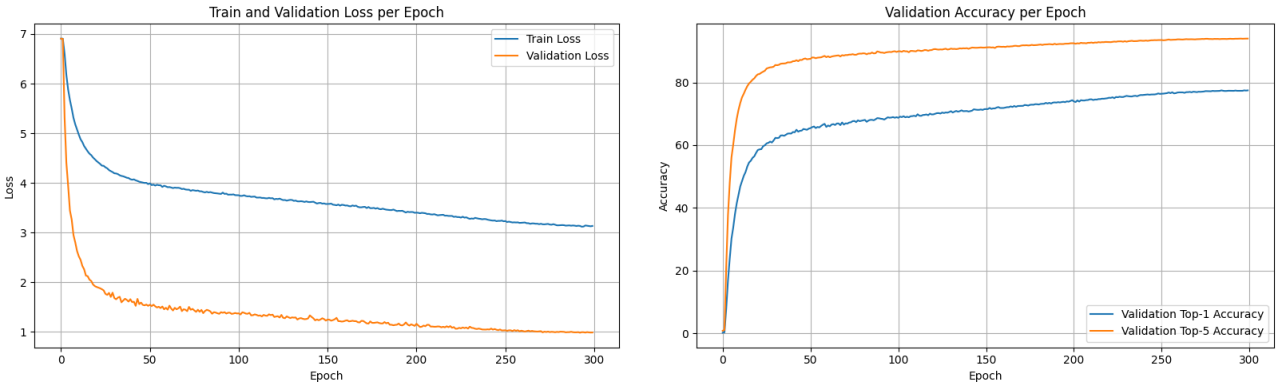


Figure 4.1: Training curves for supervised training of Pixel-Mamba-6M on ImageNet-1k. The model trains stably and reaches a best validation accuracy of 77.5%, only 0.3% lower than as reported in [28].

ISUP Grade	Training Count	Validation Count
0	2,600	288
1	2,367	296
2	1,209	133
3	1,118	124
4	1,116	132
5	1,107	116

Table 4.1: Label distributions in the training and validation splits after filtering samples containing between 4 and 999 foreground regions.

Fine-tuning on PANDA

The Prostate cANcer graDe Assessment (PANDA) dataset is a widely used benchmark for evaluating WSI encoders. The task involves predicting the prostate cancer grade from prostate biopsies, which is derived through the following process: "The grading process consists of finding and classifying cancer tissue into so-called Gleason patterns (3, 4, or 5) based on the architectural growth patterns of the tumor. After the biopsy is assigned a Gleason score, it is converted into an ISUP grade on a 1–5 scale" [31]. By adding class 0 for non-cancerous tissue, the task can be modeled as a six-class classification problem.

To extend the evaluation from the original Pixel-Mamba paper, we fine-tune the model on PANDA for ISUP grade prediction. This setup also utilizes Region Fusion. Similarly as for cancer subtyping in the original study, we replace the ImageNet-1k classification head with a randomly initialized six-class output head and fine-tune the model for 100 epochs.

We use WSIs at 10× magnification (1 μm/pixel) and randomly split the dataset into 90% training and 10% validation sets. A small number of samples were excluded from the dataset: 2 WSIs were overly small containing only 3 regions, and 8 exceeded 1000 regions, making them infeasible to process on an 80GB GPU without activation checkpointing. After filtering, we retain 9,517 training and 1,089 validation samples, with the label distributions shown in Table 4.1. We apply the foreground selection as described in Section 3.1.

Due to the variable size of WSI inputs, we cannot use standard batching. Instead, we apply gradient accumulation, updating model weights every 8 steps. Training is distributed across 8 H100 GPUs, resulting in an effective batch size of 64. We use PyTorch's automatic mixed-

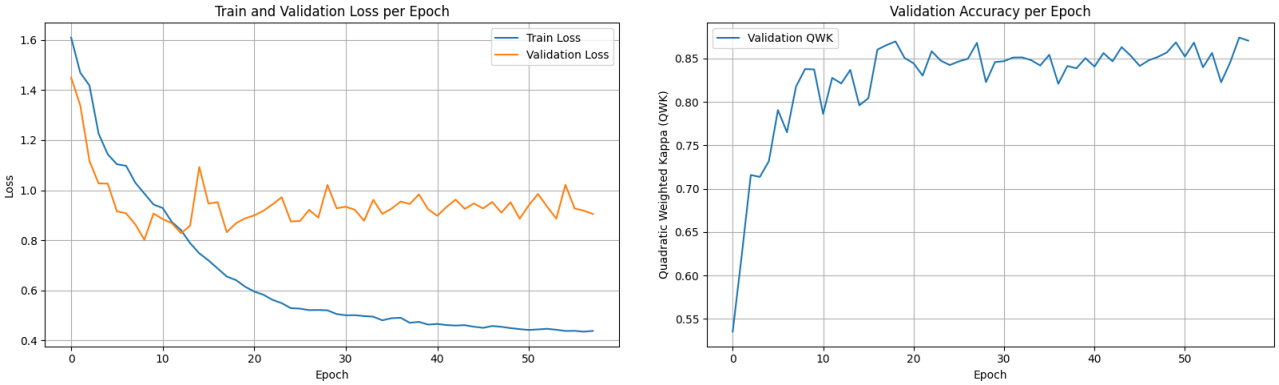


Figure 4.2: Training curves for fine-tuning the ImageNet-1k-pretrained Pixel-Mamba on PANDA for ISUP grade prediction. Despite clear overfitting, the model achieves a competitive QWK validation score of 0.87 and confirms the full implementation with the Region Fusion module.

precision (AMP), gradient clipping with a norm threshold of 3, a learning rate of 4×10^{-4}, and weight decay of 5×10^{-4}. All other hyperparameters follow the ViM-Ti fine-tuning configuration. Training was stopped after 58 epochs (over 2 days and 15 hours) due to stagnating performance.

Training dynamics during fine-tuning is shown in Figure 4.2. Performance is evaluated using the Quadratic Weighted Kappa (QWK) score, a metric well-suited for ordinal classification tasks such as ISUP grade prediction, where the labels have a natural ordering. QWK penalizes larger classification errors more severely. Pixel-Mamba achieves a best QWK validation score of 0.87. However, the training curves indicate clear signs of overfitting. Extensive hyperparameter tuning could mitigate overfitting, but we did not perform it due to computational cost and the broader scope of this study.

Direct comparison to other methods is challenging, as the official PANDA test set is not publicly available and most reported results rely on custom validation splits. Moreover, most multi-stage methods typically use higher magnification levels (the full magnification of PANDA is 20×), providing finer details and potentially benefiting model performance. Nevertheless for reference, top teams in the PANDA 2020 challenge achieved QWK scores around 0.93 on the private test set, most often using extensive label noise correction. Compared to recent foundation models, UNI2 [54]—a ViT-h/14 (630M) model pretrained on patches from 350k WSIs—achieved 0.94 QWK in a linear probing setup with Multiple Instance Learning and cleaned labels.

Overall, these results demonstrate that Pixel-Mamba-6M, when pretrained on ImageNet-1k and fine-tuned for PANDA—a task on which the original paper did not evaluate—it is a competitive and effective model in the WSI scale. This further validates our implementation and allows for exploring self-supervised pretraining while serving as a baseline.

4.2 Self-Supervised Experiments — LatentMIM

Next we perform our self-supervised experiments with LatentMIM. We train Pixel-Mamba using the setup described in Section 3.2 on the PANDA dataset with the same data split used in the supervised setup (Section 4.1). PANDA offers the advantage of thin needle biopsies that are 5–10× smaller than tissue resections in the popular TCGA dataset, enabling quicker experimentation. Training is conducted on 8 to 32 H100 GPUs, depending on resource availability, with a fixed effective batch size of 256 achieved via gradient accumulation. We use 10 warmup epochs and PyTorch's automatic mixed precision (AMP). All other hyperparameters are consistent with the original LatentMIM implementation. Average training times per epoch are as follows: 32 GPUs — 12 minutes, 16 GPUs — 27 minutes, 8 GPUs — 59 minutes. To evaluate embedding quality, we use a combination of unsupervised metrics, 5-fold linear probing, and k-nearest neighbor (kNN) classification on the validation set.

Unsupervised metrics: During training, we monitor the average pairwise cosine similarity between visible, target, and predicted tokens, which are defined according to the procedure described in Section 3.2. These similarities are expected to decrease as the model learns to distinguish semantically different parts of the input. If the similarity converges to 1, it suggests model collapse, where all tokens are mapped to nearly identical embeddings.

During validation, we track embedding diversity using both cosine similarity and the RankMe score [62], computed at three levels. RankMe is a matrix-based metric that estimates the effective rank (or intrinsic dimensionality) of the embeddings, shown to highly correlate with downstream performance.

- Token-level (spatial tokens within a slide): We compute the average pairwise cosine similarity and RankMe score of spatial tokens within each slide, averaging over the validation

split. The spatial tokens correspond to 32×32 regions in the input. Lower similarity and higher RankMe indicate that the model is capturing finer-grained distinctions within a region and may capture cellular-scale features.

- Region-level (CLS tokens within a slide): We calculate the average pairwise cosine similarity and RankMe score of CLS tokens within each slide, then average these values across the validation set. These CLS tokens summarize tissue-level regions of the slide, each corresponding to a 224×224 area in the input. Decreasing similarity and increasing RankMe suggest the model is learning to differentiate between regions within a slide and encode richer representations possibly tissue-level structures or patterns.

- Slide-level (mean pooled CLS tokens across slides): We measure the average similarity and RankMe score of the mean CLS tokens across different slides. Improvements in these metrics indicate that the model is better at distinguishing whole-slide images, potentially capturing cancer grade or other high-level features.

Experiment Excluding CLS Tokens from Loss

The original LatentMIM approach trains a ViT with a single global CLS token [30]. This CLS token is included in the predictor network but is excluded from the loss calculation, similar to how Masked Autoencoders pretrain [51]. Although excluded from the loss calculation, it still interacts with spatial tokens and therefore receives gradient updates. Following this approach, we include Pixel-Mamba's region-level CLS tokens in the prediction but exclude them from the loss calculation.

The unsupervised embedding quality metrics on the validation set (Figure 4.3) generally exhibit the desired trends. The similarity among spatial tokens decreases over training while their RankMe scores also rise, suggesting improved discriminative representations. Region-level and slide-level CLS tokens similarly show increasing RankMe scores, though their cosine similarities plateau after an initial decline.

Improvement in embedding quality is also reflected in downstream performance (Figure 4.4). Both linear probe and kNN classification scores increase steadily during training. However, the best achieved performance (∼0.5 quadratic weighted kappa score in the linear probe setting) remains substantially below the supervised baseline (0.87).

After approximately 100 epochs, we observe representational collapse in the decoder outputs, while the encoder representations remain stable. Figure 4.5 shows the average pairwise cosine similarities of the visible and predicted tokens during training. Similarities for the target (masked) tokens are not shown, as the target encoder is an EMA of the student encoder and its similarities closely mirror those of the visible tokens. The expected behavior is for the plotted similarities to gradually decrease, which is indeed observed initially. Over time, however, the decoder learns to map all tokens to nearly identical outputs. While this is an alarming sign, the downstream performance plateaus much earlier than the collapse occurs; therefore, there is likely another reason why the model lags behind the supervised baseline. Nevertheless, we further explore this collapse later in this section.

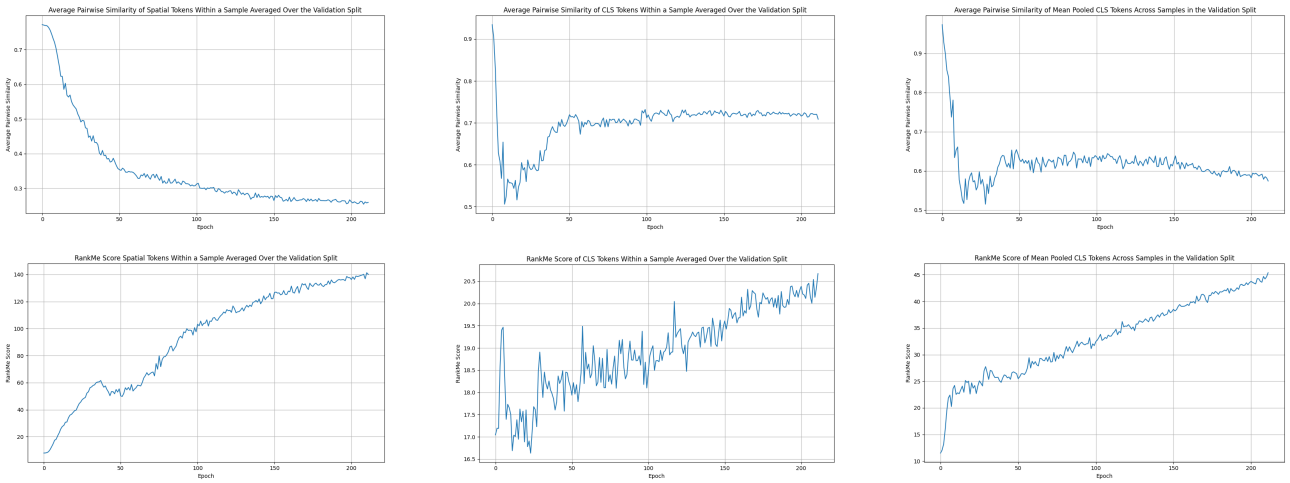


Figure 4.3: Unsupervised embedding quality measure on the validation set during training with the LatentMIM setup excluding CLS tokens from the loss calculation. **Top:** average pairwise cosine similarities on three levels (token, regional and slide-level). **Bottom:** RankMe scores on the same three levels. In general, similarities decrease, while RankMe scores increase at all three levels, indicating increasingly discriminative features.

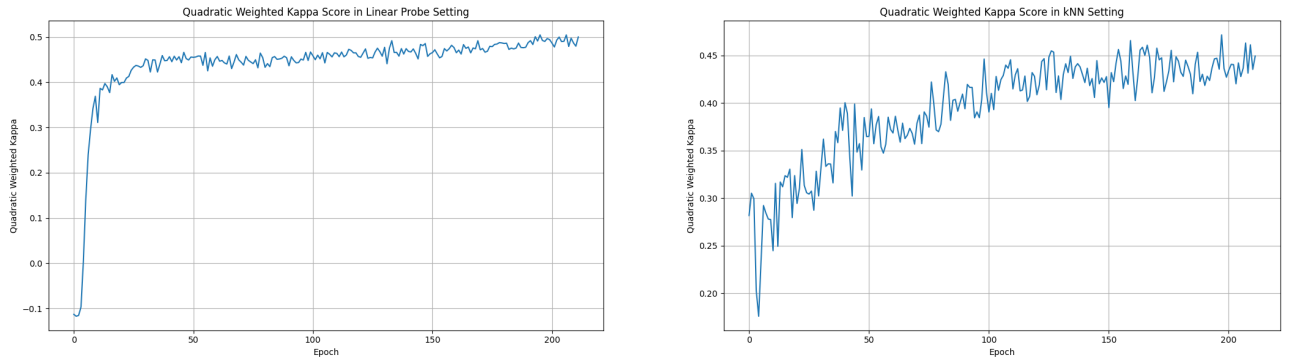


Figure 4.4: Quadratic Weighted Kappa (QWK) scores on the validation set for 5-fold linear probing **(left)** and kNN classification **(right)** during training with the LatentMIM setup excluding CLS tokens from the loss calculation. Downstream performance improves over training but saturates early, remaining well below the supervised baseline (QWK: 0.87).

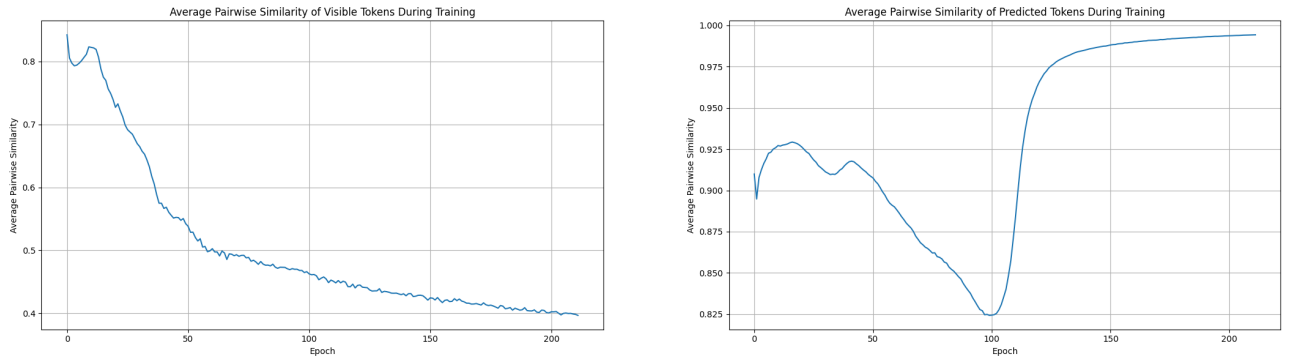


Figure 4.5: Average pairwise cosine similarities of visible **(left)** and predicted **(right)** tokens during training with the LatentMIM setup excluding CLS tokens from the loss calculation. While both visible and predicted token similarities decrease for approximately 100 epochs, the predicted representations then collapse suddenly, while the encoder features remain stable.

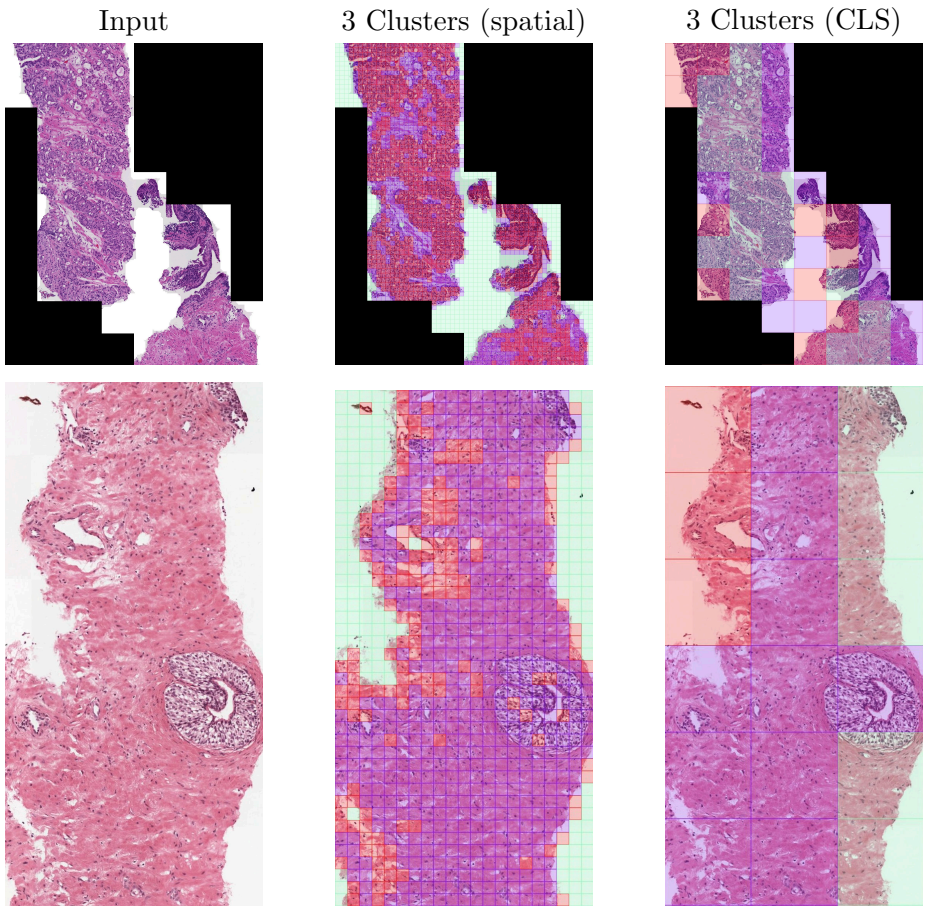


Figure 4.6: Qualitative analysis of embeddings of selected parts of two validation samples. Input **(left)**, spatial tokens clustered into 3 clusters and overlaid on the input **(middle)**, CLS tokens clustered into 3 clusters and overlaid on the input **(right)**. **Top sample:** nuclei-dense regions. **Bottom sample:** prostatic gland and its surroundings. Spatial tokens can distinguish nuclei-dense regions from stroma (connective tissue) but fail to capture the semantically relevant prostatic gland. CLS tokens do not appear to exhibit such clustering behavior. Zoom in for finer details.

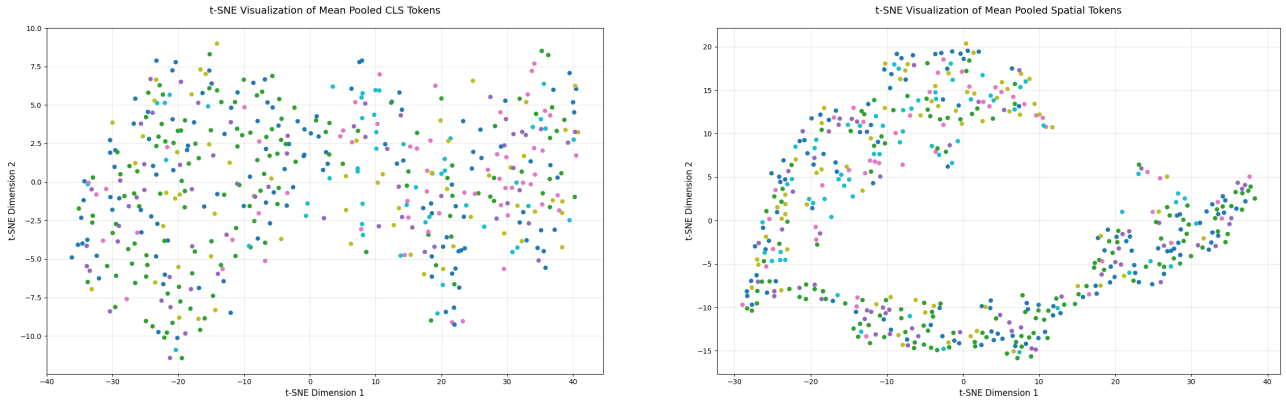


Figure 4.7: 2D t-SNE projection of mean pooled CLS tokens **(left)** and mean pooled spatial tokens **(right)**. Each datapoint represents a validation WSI, with colors indicating the ground truth ISUP grade of the WSI. Neither mean pooled CLS nor spatial tokens cluster based on the ISUP grade which explains the poor quantitative performance.

Qualitative Analysis of Embeddings

Motivated by the Token Merging paper [19], which shown that models trained with token merging could be evaluated without it and still achieve comparable performance, Region Fusion is disabled to qualitatively assess the learned embeddings. Embeddings are generated on validation samples using the checkpoint with the best linear probe performance, and the spatial tokens are clustered using K-Means. The resulting clusters are visualized in a manner similar to LatentMIM [30], as shown in Figure 4.6 middle (see Figure A.2 for the full WSIs).

The results indicate that the model has learned to distinguish the foreground from the background. It also appears to capture nuclei-dense areas—potentially cancerous regions—to some extent, as seen in the first example. However, certain relevant features are missed, such as the prostatic gland shown in the second example. Additionally, clustering into four or more groups (Figure A.2) does not reveal finer-grained structures, suggesting limited feature differentiation.

To further evaluate representation quality, CLS tokens were clustered and visualized in a similar manner in Figure 4.6 right (see Figure A.3 for full WSIs). These tokens, however, appear to encode less meaningful information—such as the relative position of background tissue—rather than task-relevant features. Motivated by these observations we evaluated quantitative performance using the mean spatial token embeddings instead of the mean pooled CLS tokens. Contrary to expectations, this approach did not result in better quadratic weighted kappa score but performed slightly worse. We hypothesize that this may be due to Region Fusion blending semantically different spatial tokens. We elaborate on this later in the section.

We also evaluated the mean-pooled CLS and spatial tokens by generating embeddings for 500 evaluation samples and visualizing their 2D t-SNE projections in Figure 4.7. As shown, the embeddings do not form distinct clusters based on ISUP grade, which aligns with the weak downstream performance observed in the quantitative evaluations (4.4).

Experiment Including CLS Tokens in Loss

Based on the qualitative assessment, we identified that the feature quality of CLS tokens is not as expressive as spatial tokens'. To reinforce rich representation in CLS tokens we experimented with including the CLS token in the loss. Specifically, in Equation 3.1, k and l can also be CLS tokens, so that the loss explicitly considers how well the CLS tokens are predicted. However, this approach proved highly unstable, as shown in Figure 4.8, and downstream performance did not improve during training (Figure 4.9).

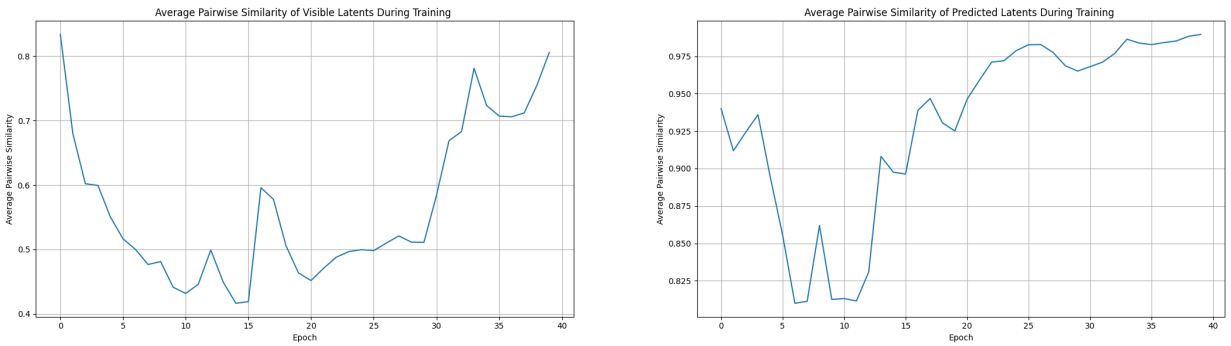


Figure 4.8: Average pairwise cosine similarities of visible **(left)** and predicted **(right)** tokens during training with the LatentMIM setup including CLS tokens in the loss calculation. Both similarities start increasing early, suggesting the development of a representational collapse.

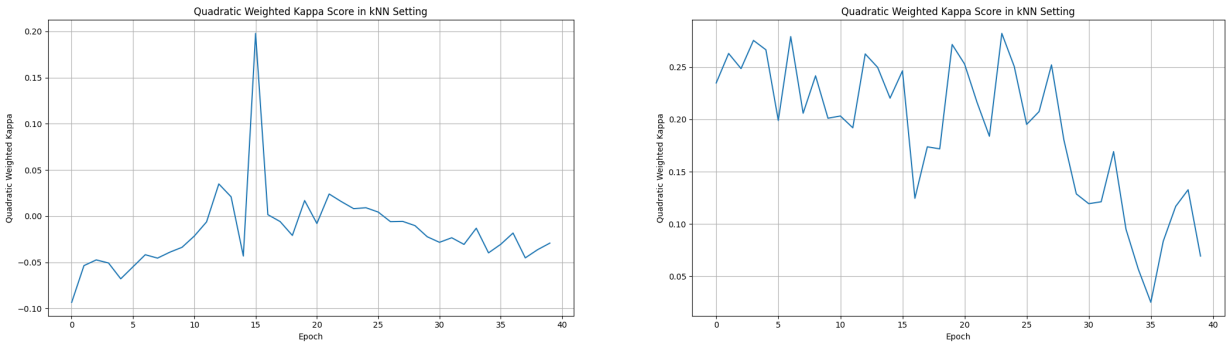


Figure 4.9: Quadratic Weighted Kappa (QWK) scores on the validation set for 5-fold linear probing **(left)** and kNN classification **(right)** during training with the LatentMIM setup including CLS tokens in the loss calculation. Downstream performance fails to increase, remaining at random levels.

Role of Region Fusion

Previously, we assumed that Region Fusion would not disrupt learning of spatial correspondences, reasoning that self-attention also merges tokens in a similar manner, and that the Token Merging paper [19] demonstrated successful masked image modeling with merged tokens. However, this reasoning overlooks a key difference.

In the Token Merging approach, merging is performed by directly comparing spatial tokens to one another and selecting the top k pairs to merge. By contrast, Region Fusion operates by comparing only the CLS tokens, and then merges all tokens within the selected regions based on those CLS token similarities. This difference leads to the scenario illustrated in Figure 4.10.

We can argue that the CLS tokens of these two regions in the figure should be similar, as both appear visually alike, containing many nuclei and a small area of background on the right. As a result, these two regions should be merged by averaging the tokens on the same relative positions. However, the highlighted green sections within these regions are substantially different. Merging based solely on the CLS similarity would therefore blend spatial tokens representing distinct content, disrupting the intended spatial correspondence of the token to the input it encodes. If this occurs systematically, it can undermine the model's ability to learn representations from spatial correspondences.

To avoid this issue, we modified the algorithm to include only the CLS tokens in the prediction and loss calculation. Therefore, in this setup the model would only learn from the spatial relationships of CLS tokens which are directly compared and merged eliminating pre-

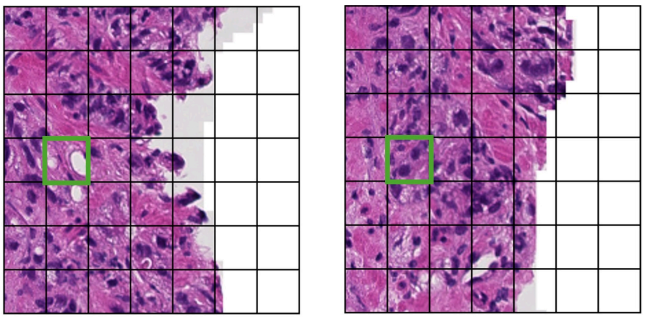


Figure 4.10: Visualization of the potential issue of training LatentMIM with Region Fusion. Shown regions are in general similar so they might merge. However, individual sections (shown as 32×32-pixel grids) might differ significantly such as the ones in the highlighted grids undermining learning from spatial relationships.

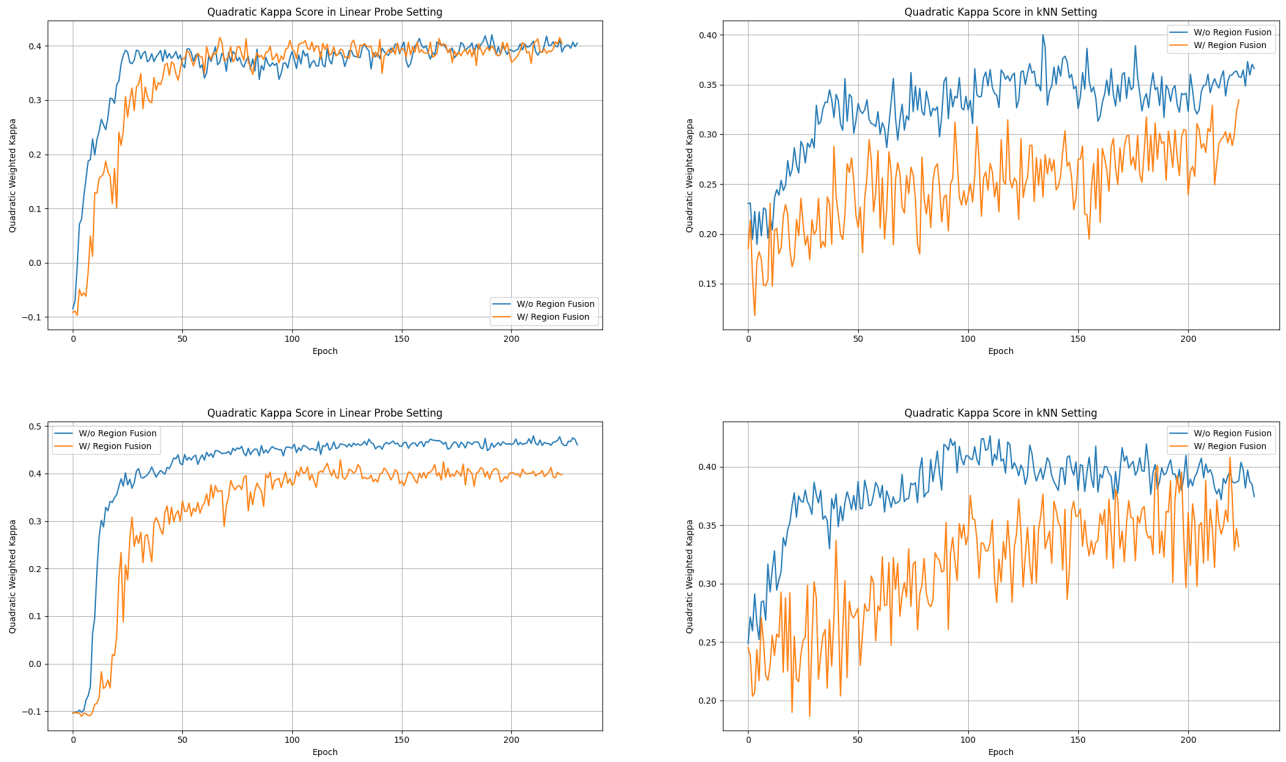


Figure 4.11: Quadratic Weighted Kappa (QWK) scores on the validation set for 5-fold linear probing **(left)** and kNN classification **(right)** during training with and without Region Fusion. **Top:** evaluation using mean-pooled CLS tokens; **bottom:** evaluation using mean-pooled spatial tokens. The performance gap from using Region Fusion is smaller than the gap to the supervised baseline (QWK: 0.87), suggesting it is not the main reason for poor performance.

dicting from the blended spatial tokens. However, this variant quickly led both the encoder and decoder to collapse, resulting in essentially random downstream performance similarly to the experiment shown in Figure 4.9.

Given this clear failure, we designed an ablation in which we retained the original setup but disabled Region Fusion entirely. Disabling Region Fusion significantly increases memory requirements, so we reduced the input size accordingly. Our experiments were therefore conducted on PANDA slides at 5× magnification. We visualize the results in Figure 4.11. While evaluation on the mean spatial tokens clearly suggests a gap between performance, evaluating with linear probe on the mean pooled CLS tokens (the setup most similar to the original Pixel-Mamba) yields essentially same performance.

These mixed results suggest that Region Fusion does indeed hurt performance of LatentMIM training to some degree but the gap is rather small and it is not the primary reason why the linear probe evaluation falls short of the supervised baseline.

Decoder Collapse

To better understand decoder collapse, we qualitatively analyzed the decoder outputs from the experiment depicted in Figure 4.5. We disabled Region Fusion and generated decoder outputs on training samples using the weights of epoch 99. The predicted tokens were clustered using DBSCAN and visualized after 2D t-SNE reduction in Figure 4.12. The results show that the predictor network largely ignores the semantic information in the encoded tokens and instead emphasizes positional embeddings even before collapsing. Tokens cluster based on spatial proximity as seen by homogeneous cluster labels per region and the neighboring regions forming clusters. As collapse progresses, the predicted tokens move even closer together forming

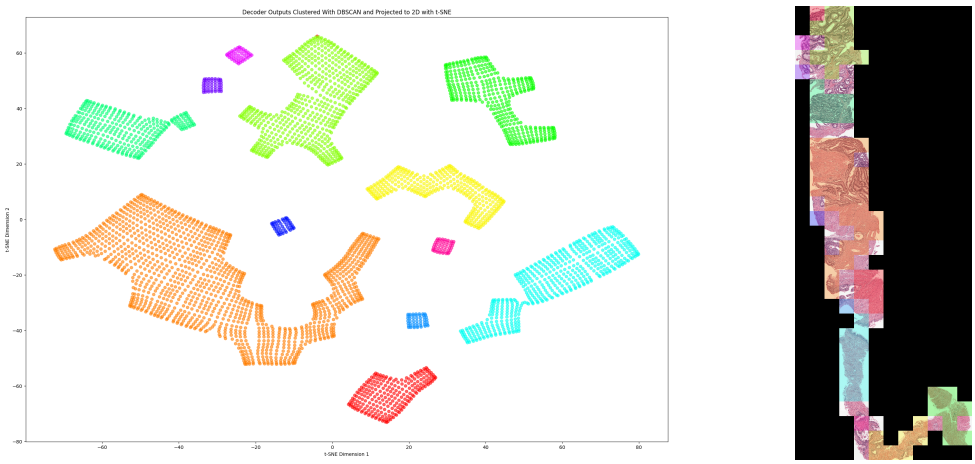


Figure 4.12: Decoder output tokens clustered with DBSCAN and shown after a 2D t-SNE projection **(left)** and overlaid on the input **(right)**. Embeddings were generated before the predictor collapse using the weights of epoch 99. Zoom in on the overlaid image for finer details. Tokens from nearby regions form clusters, with each region belonging homogeneously to a single cluster (all its tokens assigned to the same cluster).

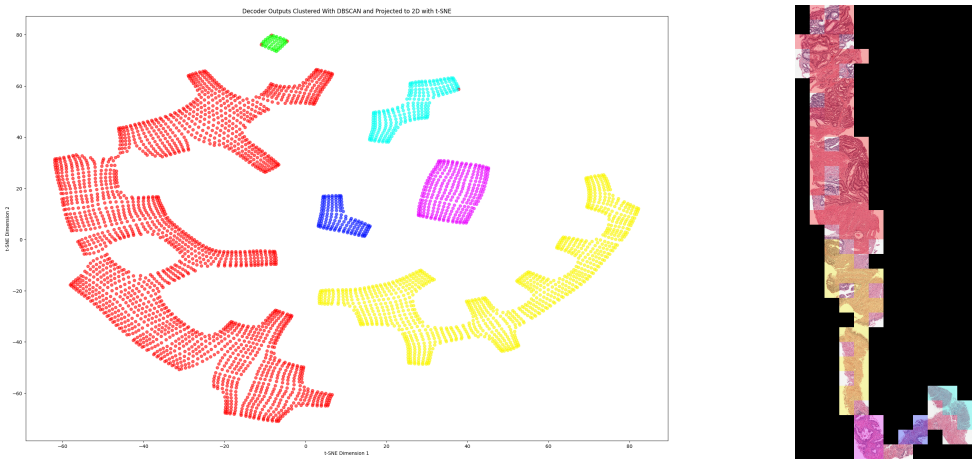


Figure 4.13: Decoder output tokens after collapse clustered with DBSCAN and shown after a 2D t-SNE projection **(left)** and overlaid on the input **(right)**. Embeddings were generated after the predictor collapse using the weights of epoch 211. Zoom in on the overlaid image for finer details. Tokens from nearby regions form clusters, with each region belonging homogeneously to a single cluster (all its tokens assigned to the same cluster). As the collapse progresses, tokens are pushed closer together, forming larger and larger clusters.

larger and fewer clusters as shown from the outputs using the weights of epoch 211 in Figure 4.13.

Based on these observations, we can conclude that the predictor is struggling to leverage semantic features even before the collapse, which can result in poor training signals and the observed early saturation of downstream performance (Figure 4.4). This is likely the primary reason for the performance gap with the supervised baseline and warrants further investigation to understand why it occurs and how to potentially mitigate it.

Configuration	Num Patches	Time (s)	Peak Memory (GB)
Without activation checkpointing	110	8.71	76.41
Checkpointing 1st layer	135	11.50	74.26
Checkpointing first 2 layers	155	14.44	72.22
Checkpointing first 4 layers	200	18.13	77.38
Checkpointing first 16 layers	400	37.71	75.00

Table 4.2: Impact of activation checkpointing on number of patches, runtime, and peak memory usage. Benchmarked on a single 80GB A100 GPU using forward and backward passes on a sample of size Num patches $\times$ 224 $\times$ 224. DINO Transform applies random cropping, so the scale was fixed to the worst-case scenario (local views: 40% of total patches, global views: 100% of total patches). Checkpointing layers allows using larger context at constant memory usage, but increases runtime significantly.

4.3 Self-Supervised Experiments — DINO

Below, we experiment with DINO training at the whole-slide level using the modified DINO transform described in Section 3.3. This is intended as an exploratory probe into applying the DINO approach in this context and assessing its advantages and limitations. This setup has the advantage that only the slide-level mean-pooled CLS tokens are used in the DINOLoss calculation, eliminating the need to handle merged spatial tokens as required in the LatentMIM approach. Using only the mean-pooled CLS tokens for distillation also more closely resembles the supervised setup that has been shown to work well.

Challenges of Inefficiency

The main limitation of the WSI adapted DINO Transform is that the generated views cannot be batched due to variable sizes. This significantly slows down training and greatly increases memory usage, as the computation graphs for each individual forward pass must be retained until the loss is computed. For comparison, using the LatentMIM setup, we were able to handle WSIs of up to 50MP (equivalent to 1000 patches of size 224 $\times$ 224) during training using an 80GB GPU. In contrast, with the DINO setup, we are limited to only 5.5MP (approximately 110 patches).

Activation checkpointing can reduce memory consumption and thus allow for longer context lengths. However, this comes at the cost of increased runtime. Table 4.2 shows this trade-off for Pixel-Mamba with the proposed DINO Transform. Checkpointing more layers allows for larger context sizes (more regions) while maintaining similar peak memory usage. However, runtime also increases significantly, limiting the feasibility of this approach.

Padding and masking are standard techniques in transformer architectures to support batching of variable-sized inputs. However, applying them to Mamba is non-trivial due to its recurrent structure. Even if implemented, a considerable amount of resources would be wasted on padded tokens—an issue that is especially problematic given the already limited memory and runtime budget.

To enable faster experimentation with larger contexts without wasting resources, we evaluated PackMamba [63], an extension of Mamba with a custom CUDA kernel that supports variable-length batching. A single Block with PackMamba achieved a $\sim$3$\times$ speedup with large batch sizes (e.g., 512), compared to repeated calls of the original Mamba kernel. However, with small batch sizes (e.g., 8, as used in DINO), it was $\sim$2.5$\times$ slower. We also integrated PackMamba into Pixel-Mamba and benchmarked it on worst-case DINO views using a 42-patch input (the median for PANDA at 5$\times$ magnification). In this case, runtime was $\sim$2$\times$ slower

ISUP Grade	Training Count	Validation Count
0	2,590	286
1	2,355	296
2	1,202	132
3	1,114	124
4	1,116	131
5	1,106	116

Table 4.3: Label distributions in training and validation splits after filtering.

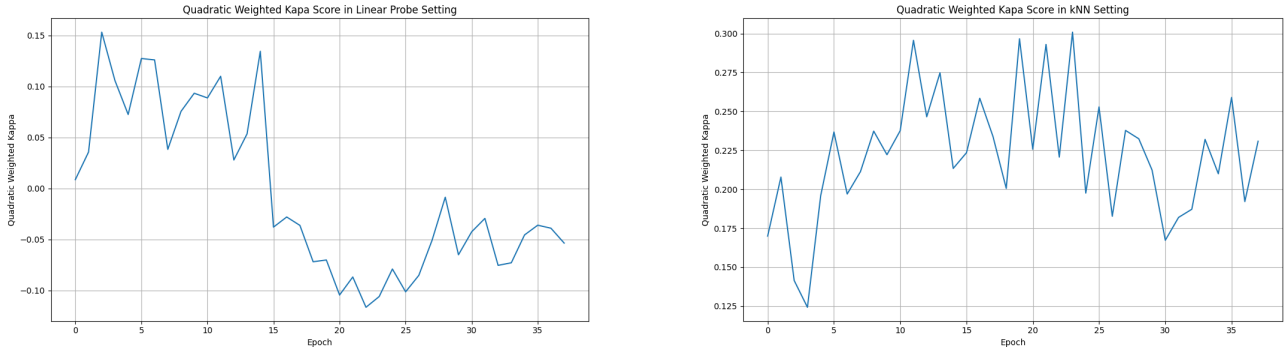


Figure 4.14: Validation curves of DINO training on PANDA. Quadratic Weighted Kappa (QWK) scores on the validation set for 5-fold linear probing **(left)** and kNN classification **(right)**. Downstream performance fails to increase, remaining at random levels.

than using separate forward passes. Based on these results, we proceeded with the original kernel and multiple forward passes, avoiding checkpointing for runtime efficiency.

PANDA Experiment

We train on PANDA at 5× magnification (2 μm/pixel) with the splits previously described and discarded 38 additional samples exceeding 110 regions, the maximum count that can fit safely on the 80GB GPU during training. After filtering, we retain 9,483 training and 1,085 validation samples with the label distributions shown in Table 4.3.

We train for 200 epochs using a base learning rate of 5e-4, a cosine learning rate scheduler with a final value of 0.002, and 10 warmup epochs. Weight decay is set to 0.04, also scheduled with a cosine decay ending at 0.4. Training is performed on 16 H100 GPUs with gradient accumulation, reaching an effective batch size of 256. All remaining hyperparameters, including the DINO loss and head configurations, match the original DINO implementation.

We evaluate using the same supervised and unsupervised metrics as in the LatentMIM experiment in Section 4.2. Even though DINO uses samples four times smaller than LatentMIM (due to the decreased magnification), DINO requires 47 minutes per epoch on 16 H100s—compared to just 27 minutes for LatentMIM. Training was eventually suspended after 39 epochs (∼17 hours) due to the lack of improvement as seen in Figure 4.14.

TCGA Experiment

Because PANDA samples are long and narrow needle biopsies, the resulting views can become extremely thin, which may negatively affect training potentially leading to the observed divergence. To test this hypothesis, we also train on the TCGA dataset, which contains larger tissue samples rather than thin needle biopsies. We use a split of 6,339 training and 1,585 validation

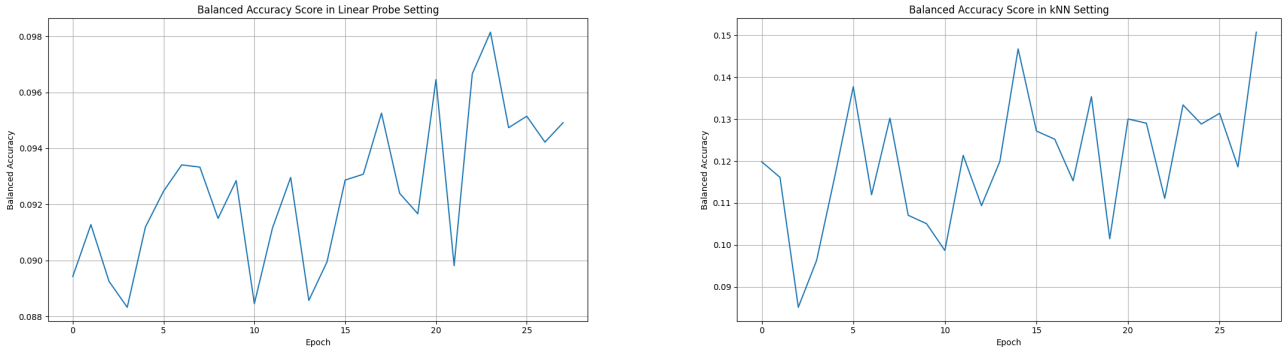


Figure 4.15: Validation curves of DINO training on TCGA. Balanced Accuracy scores on the validation set for 5-fold linear probing **(left)** and kNN classification **(right)**. Increase in performance is very slow.

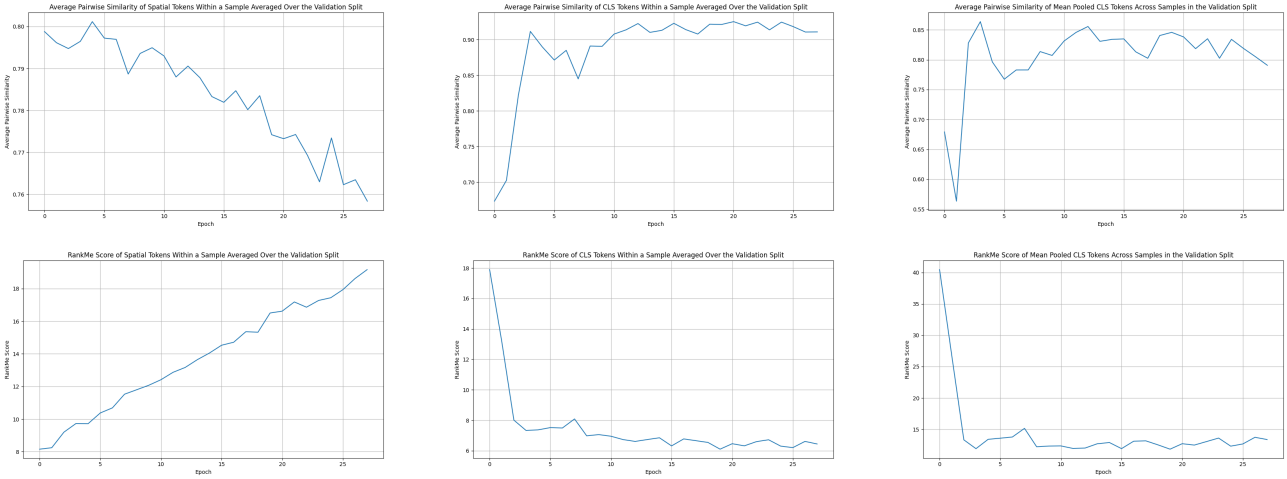


Figure 4.16: Unsupervised evaluation of embedding quality during DINO training on TCGA on the validation split. Spatial token similarities decrease while their RankMe score increases, albeit still very low. CLS token similarities and RankMe scores remain stagnant, raising concerns about their representational quality.

samples. For validation, we use ICD-O-2 labels as a classification target. These labels define both tumor site (topography) and type (morphology), resulting in a 14-class classification task.

We use TCGA at a magnification of 2.5× due to its significantly larger number of foreground regions. Even with the 2.5× magnification, the majority of the samples have more than 110 regions which are too many to discard. To mitigate this, we apply another level of cropping in the beginning to make sure that the crop before the DINO Transform is not larger than 110 regions. This works in an online way ensuring maximum diversity. We first sample a random region and then iteratively add its neighboring regions until we obtain a crop with the largest number of foreground regions that is still fewer than 110.

While validation results show a slight upward trend (Figure 4.15), a steeper increase was expected during the early epochs. Given this and the high computational requirements, we suspended training after 29 epochs (19 hours).

Figure 4.16 shows embedding quality based on token similarity and RankMe scores as described in Section 4.2. Spatial token similarity decreases over training, while the RankMe score increases—although remains significantly lower than in LatentMIM experiments. Notably, CLS token similarities remain high while RankMe scores remain low and even decrease during the initial epochs. This suggests that the encoder struggles to extract higher-level features, which is consistent with the poor and nearly stagnant downstream performance (Figure 4.15).

The obtained results and the discussed limitations suggest that while DINO offers strong representation learning capabilities on region-level [9, 55, 39], its inefficiencies make it less practical (if not completely unsuitable) for high-resolution WSI modeling as compared to masked image modeling like LatentMIM. Based on the limited experimental results, it appears that DINO pretraining of Pixel-Mamba struggles to extract slide-level features.

Chapter 5

Discussion and Limitations

This thesis set out to evaluate whether a single-stage, attention-free architecture — Pixel-Mamba — could be trained on WSIs using self-supervised learning in an end-to-end fashion. Although the study did not achieve competitive performance compared to multi-stage methods, the successful reproduction of Pixel-Mamba and the exploration of SSL strategies such as LatentMIM and DINO provide valuable insights for the research community.

Pixel-Mamba was successfully re-implemented and validated through supervised pretraining on ImageNet-1k. The reproduced model closely matched the performance reported in the original paper, demonstrating its reproducibility. When fine-tuned on the PANDA dataset for ISUP grading—a benchmark not explored in the original work—Pixel-Mamba achieved a competitive QWK score of 0.87, highlighting its potential for cancer grade classification in pathology. However, the results underline that end-to-end slide-level modeling remains highly computationally demanding, even with linear, attention-free architectures. Training a relatively small 6M-parameter model on WSIs in an end-to-end manner poses practical challenges (memory and runtime), especially when combined with complex SSL objectives and dynamic views, as observed in the case of DINO.

Masked image modeling with latent reconstruction proved far more computationally feasible than DINO for pretraining. Despite this advantage, the learned representations fell short of expectations. While some semantically meaningful patterns appeared in the embeddings, their overall quality remained significantly below that of the supervised baseline. The study explored several design alternatives for handling CLS tokens and examined whether the Region Fusion module might degrade LatentMIM training.

A brief qualitative review of predictor outputs was conducted to investigate potential causes of collapse. This analysis revealed that the predictor outputs cluster primarily based on spatial proximity, indicating that the predictor relies too heavily on positional embeddings rather than semantic features. This likely results in poor training signals to the encoder, eventually causing it to fail to extract meaningful features. To close the gap with the supervised baseline, future work should investigate the dynamics of the predictor in greater depth. For example, increasing the number of predictor layers may help it capture more complex semantic features. Alternatively, downweighting the positional embeddings rather than simply adding them to the latents may be worth exploring. Another potential approach could be to introduce a regularizer that penalizes the predictor if its outputs too closely resemble the positional embeddings.

There are also important limitations to this study. No ablations were performed on key hyperparameters, most critically on the masking ratio in LatentMIM, due to the significant computational demands. Additionally, there were no evaluations on out-of-distribution data, limiting claims about the generalizability of the learned representations. The absence of large-scale pretraining further constrains the conclusions that can be drawn. Training and validating on the PANDA dataset was primarily a practical choice, as its thin needle biopsies are 5–10×

smaller than tissue resections in the popular TCGA dataset.

Looking forward, training end-to-end WSI encoders remains a worthwhile but challenging goal. Augmentation based methods like DINO impose high computational costs and remain unproven at whole-slide scale, while reconstruction-based objectives such as LatentMIM offer better efficiency but have yet to deliver strong representations. Adapting self-supervised learning frameworks to fully leverage the rich spatial information in WSIs may require thoughtful redesign, as well as a deeper theoretical understanding of model components like Region Fusion. Specifically, qualitatively studying Region Fusion in a trained supervised model could be an informative first step.

Overall, Pixel-Mamba provides a promising foundation for end-to-end WSI modeling but requires further research to support effective self-supervised learning. We suggest that future work should prioritize developing a deeper theoretical understanding of Mamba models in general—and in particular Pixel-Mamba and the role of the Region Fusion module. Such understanding could support the design of more effective self-supervised frameworks. Further ablations with the LatentMIM setup might also be relevant to understand the behavior of the decoder and potentially find a more performant setup.

GPU Hours and CO_2 Emissions

The total GPU usage for this study amounted to approximately 15,000 hours. Experiments were conducted on A100 GPUs via Microsoft Azure and H100 GPUs on private infrastructure.

To estimate the carbon footprint, we used the Machine Learning Impact calculator introduced by [64]. For simplicity, we assumed all 15,000 hours were run on A100 GPUs in the Azure West Europe region. Based on this estimate, the total emissions are approximately 2,137.5 kg CO_2—roughly equivalent to driving an average internal combustion engine (ICE) car for 8,640 kilometers, or the carbon sequestered by 35.6 tree seedlings grown for ten years.

Use of AI

Generative AI tools were used during the development of this thesis in the following ways:

- **Coding support:** GitHub Copilot and Cursor were used for code completion and data visualization. They provided suggestions for debugging, optimizing Python code, and generated documentation in some cases. All final implementations and documentation were reviewed and validated by the author to ensure correctness.
- **Writing assistance:** Language models (specifically, OpenAI's ChatGPT-4o) were used to improve grammar, clarity, flow, and academic tone in selected paragraphs. All text was reviewed and edited by the author to ensure accuracy and originality.
- **Literature support:** Perplexity was occasionally used as a search tool to identify relevant literature. All retrieved and cited papers were independently reviewed by the author to ensure relevance and reliability.

At no point did AI tools generate original content or ideas without critical review. All scientific contributions, analyses, and conclusions presented in this thesis are the author's own work.

Bibliography

[1] Ivana Kholová, Giovanni Negri, Maria Nasioutziki, Laura Ventura, Arrigo Capitanio, Massimo Bongiovanni, Paul A Cross, Claire Bourgain, Henrik Edvardsson, Rosario Granados, et al. Inter-and intraobserver agreement in whole-slide digital thinprep samples of low-grade squamous lesions of the cervix uteri with known high-risk hpv status: a multicentric international study. Cancer Cytopathology, 130(12):939–948, 2022.

[2] Rossana CN Melo, Maximilian WD Raas, Cinthia Palazzi, Vitor H Neves, Kássia K Malta, and Thiago P Silva. Whole slide imaging and its applications to histopathological studies of liver disorders. Frontiers in medicine, 6:310, 2020.

[3] P. Pham. Whole-slide image classification in digital pathology using deep learning. Master's thesis, Eindhoven Universiry of Technology, 3 Dec 2021.

[4] Jakob Nikolas Kather, Johannes Krisam, Pornpimol Charoentong, Tom Luedde, Esther Herpel, Cleo-Aron Weis, Timo Gaiser, Alexander Marx, Nektarios A Valous, Dyke Ferber, et al. Predicting survival from colorectal cancer histology slides using deep learning: A retrospective multicenter study. PLoS medicine, 16(1):e1002730, 2019.

[5] Pegah Khosravi, Ehsan Kazemi, Marcin Imielinski, Olivier Elemento, and Iman Hajirasouliha. Deep convolutional neural networks enable discrimination of heterogeneous digital pathology images. EBioMedicine, 27:317–328, 2018.

[6] Xinliang Zhu, Jiawen Yao, Feiyun Zhu, and Junzhou Huang. Wsisa: Making survival prediction from whole slide histopathological images. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 7234–7242, 2017.

[7] Daisuke Komura, Mieko Ochi, and Shumpei Ishikawa. Machine learning methods for histopathological image analysis: Updates in 2024. Computational and Structural Biotechnology Journal, 2024.

[8] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. CoRR, abs/2104.14294, 2021.

[9] 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(8015):181–188, 2024.

[10] George Shaikovski, Adam Casson, Kristen Severson, Eric Zimmermann, Yi Kan Wang, Jeremy D Kunz, Juan A Retamero, Gerard Oakley, David Klimstra, Christopher Kanan, et al. Prism: A multi-modal generative foundation model for slide-level histopathology. arXiv preprint arXiv:2405.10254, 2024.

[11] Tong Ding, Sophia J Wagner, Andrew H Song, Richard J Chen, Ming Y Lu, Andrew Zhang, Anurag J Vaidya, Guillaume Jaume, Muhammad Shaban, Ahrong Kim, et al. Multimodal whole slide foundation model for pathology. arXiv preprint arXiv:2411.19666, 2024.

[12] 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. CoRR, abs/2010.11929, 2020.

[13] Duy-Kien Nguyen, Mahmoud Assran, Unnat Jain, Martin R. Oswald, Cees G. M. Snoek, and Xinlei Chen. An image is worth more than 16x16 patches: Exploring transformers on individual pixels, 2025.

[14] Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey, 2022.

[15] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer. CoRR, abs/2004.05150, 2020.

[16] Jiayu Ding, Shuming Ma, Li Dong, Xingxing Zhang, Shaohan Huang, Wenhui Wang, Nanning Zheng, and Furu Wei. Longnet: Scaling transformers to 1,000,000,000 tokens, 2023.

[17] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. Rethinking attention with performers, 2022.

[18] Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity, 2020.

[19] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your vit but faster, 2023.

[20] Youwei Liang, Chongjian Ge, Zhan Tong, Yibing Song, Jue Wang, and Pengtao Xie. Not all patches are what you need: Expediting vision transformers via token reorganizations, 2022.

[21] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces, 2024.

[22] Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality, 2024.

[23] Maximilian Beck, Korbinian Pöppel, Markus Spanring, Andreas Auer, Oleksandra Prudnikova, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. xlstm: Extended long short-term memory, 2024.

[24] Lianghui Zhu, Bencheng Liao, Qian Zhang, Xinlong Wang, Wenyu Liu, and Xinggang Wang. Vision mamba: Efficient visual representation learning with bidirectional state space model, 2024.

[25] Yue Liu, Yunjie Tian, Yuzhong Zhao, Hongtian Yu, Lingxi Xie, Yaowei Wang, Qixiang Ye, Jianbin Jiao, and Yunfan Liu. Vmamba: Visual state space model, 2024.

[26] Benedikt Alkin, Maximilian Beck, Korbinian Pöppel, Sepp Hochreiter, and Johannes Brandstetter. Vision-lstm: xlstm as generic vision backbone, 2025.

[27] 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, March 2022.

[28] Zhongwei Qiu, Hanqing Chao, Tiancheng Lin, Wanxing Chang, Zijiang Yang, Wenpei Jiao, Yixuan Shen, Yunshuo Zhang, Yelin Yang, Wenbin Liu, et al. From pixels to gigapixels: Bridging local inductive bias and long-range dependencies with pixel-mamba. arXiv preprint arXiv:2412.16711, 2024.

[29] Peyman Nejat, Areej Alsaafin, Ghazal Alabtah, Nneka I Comfere, Aaron R Mangold, Dennis H Murphree, Patricija Zot, Saba Yasir, Joaquin J Garcia, and Hamid R Tizhoosh. Creating an atlas of normal tissue for pruning wsi patching through anomaly detection. Scientific reports, 14(1):3932, 2024.

[30] Yibing Wei, Abhinav Gupta, and Pedro Morgado. Towards latent masked image modeling for self-supervised visual representation learning, 2024.

[31] 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.

[32] Albert Gu, Isys Johnson, Karan Goel, Khaled Saab, Tri Dao, Atri Rudra, and Christopher Ré. Combining recurrent, convolutional, and continuous-time models with linear state-space layers, 2021.

[33] Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces, 2022.

[34] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. 1960.

[35] Sukjun Hwang, Aakash Lahoti, Ratish Puduppully, Tri Dao, and Albert Gu. Hydra: Bidirectional state space models through generalized matrix mixers. Advances in Neural Information Processing Systems, 37:110876–110908, 2024.

[36] Maarten Grootendorst. A visual guide to mamba and state space models. https://newsletter.maartengrootendorst.com/p/a-visual-guide-to-mamba-and-state, 2024. Accessed: 23-06-2025.

[37] Xingtai Lv, Youbang Sun, Kaiyan Zhang, Shang Qu, Xuekai Zhu, Yuchen Fan, Yi Wu, Ermo Hua, Xinwei Long, Ning Ding, and Bowen Zhou. Technologies on effectiveness and efficiency: A survey of state spaces models, 2025.

[38] Leo Feng, Frederick Tung, Mohamed Osama Ahmed, Yoshua Bengio, and Hossein Hajimirsadeghi. Were rnns all we needed?, 2024.

[39] Ali Nasiri-Sarvi, Vincent Quoc-Huy Trinh, Hassan Rivaz, and Mahdi S. Hosseini. Vim4path: Self-supervised vision mamba for histopathology images, 2024.

[40] Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, Tatsunori Hashimoto, and Carlos Guestrin. Learning to (learn at test time): Rnns with expressive hidden states, 2025.

[41] Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time, 2024.

[42] Roger Waleffe, Wonmin Byeon, Duncan Riach, Brandon Norick, Vijay Korthikanti, Tri Dao, Albert Gu, Ali Hatamizadeh, Sudhakar Singh, Deepak Narayanan, Garvit Kulshreshtha, Vartika Singh, Jared Casper, Jan Kautz, Mohammad Shoeybi, and Bryan Catanzaro. An empirical study of mamba-based language models, 2024.

[43] Georgios Pantazopoulos, Malvina Nikandrou, Alessandro Suglia, Oliver Lemon, and Arash Eshghi. Shaking up vlms: Comparing transformers and structured state space models for vision language modeling, 2024.

[44] Randall Balestriero, Mark Ibrahim, Vlad Sobal, Ari Morcos, Shashank Shekhar, Tom Goldstein, Florian Bordes, Adrien Bardes, Gregoire Mialon, Yuandong Tian, Avi Schwarzschild, Andrew Gordon Wilson, Jonas Geiping, Quentin Garrido, Pierre Fernandez, Amir Bar, Hamed Pirsiavash, Yann LeCun, and Micah Goldblum. A cookbook of self-supervised learning, 2023.

[45] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning, 2020.

[46] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2024.

[47] Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture, 2023.

[48] Timothée Darcet, Federico Baldassarre, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Cluster and predict latent patches for improved masked image modeling, 2025.

[49] Jinghao Zhou, Chen Wei, Huiyu Wang, Wei Shen, Cihang Xie, Alan Yuille, and Tao Kong. ibot: Image bert pre-training with online tokenizer, 2022.

[50] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[51] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners, 2021.

[52] Maximilian Ilse, Jakub M. Tomczak, and Max Welling. Attention-based deep multiple instance learning, 2018.

[53] Richard J. Chen, Chengkuan Chen, Yicong Li, Tiffany Y. Chen, Andrew D. Trister, Rahul G. Krishnan, and Faisal Mahmood. Scaling vision transformers to gigapixel images via hierarchical self-supervised learning, 2022.

[54] Richard J. Chen, Tong Ding, Ming Y. Lu, Drew F. K. Williamson, Guillaume Jaume, Bowen Chen, Andrew Zhang, Daniel Shao, Andrew H. Song, Muhammad Shaban, Mane Williams, Anurag Vaidya, Sharifa Sahai, Lukas Oldenburg, Luca L. Weishaupt, Judy J. Wang, Walt Williams, Long Phi Le, Georg Gerber, and Faisal Mahmood. A general-purpose self-supervised model for computational pathology, 2023.

[55] Eugene Vorontsov, Alican Bozkurt, Adam Casson, George Shaikovski, Michal Zelechowski, Siqi Liu, Kristen Severson, Eric Zimmermann, James Hall, Neil Tenenholtz, Nicolo Fusi, Philippe Mathieu, Alexander van Eck, Donghun Lee, Julian Viret, Eric Robert, Yi Kan Wang, Jeremy D. Kunz, Matthew C. H. Lee, Jan Bernhard, Ran A. Godrich, Gerard Oakley, Ewan Millar, Matthew Hanna, Juan Retamero, William A. Moye, Razik Yousfi, Christopher Kanan, David Klimstra, Brandon Rothrock, and Thomas J. Fuchs. Virchow: A million-slide digital pathology foundation model, 2024.

[56] Eric Zimmermann, Eugene Vorontsov, Julian Viret, Adam Casson, Michal Zelechowski, George Shaikovski, Neil Tenenholtz, James Hall, David Klimstra, Razik Yousfi, Thomas Fuchs, Nicolo Fusi, Siqi Liu, and Kristen Severson. Virchow2: Scaling self-supervised mixed magnification models in pathology, 2024.

[57] George Shaikovski, Eugene Vorontsov, Adam Casson, Julian Viret, Eric Zimmermann, Neil Tenenholtz, Yi Kan Wang, Jan H. Bernhard, Ran A. Godrich, Juan A. Retamero, Razik Yousfi, Nicolo Fusi, Thomas J. Fuchs, Kristen Severson, and Siqi Liu. Prism2: Unlocking multi-modal general pathology ai with clinical dialogue, 2025.

[58] Nanne Aben, Edwin D de Jong, Ioannis Gatopoulos, Nicolas Känzig, Mikhail Karasikov, Axel Lagré, Roman Moser, Joost van Doorn, Fei Tang, et al. Towards large-scale training of pathology foundation models. arXiv preprint arXiv:2404.15217, 2024.

[59] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[60] Tao Huang, Xiaohuan Pei, Shan You, Fei Wang, Chen Qian, and Chang Xu. Localmamba: Visual state space model with windowed selective scan, 2024.

[61] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers distillation through attention, 2021.

[62] Quentin Garrido, Randall Balestriero, Laurent Najman, and Yann Lecun. Rankme: Assessing the downstream performance of pretrained self-supervised representations by their rank, 2023.

[63] Haoran Xu, Ziqian Liu, Rong Fu, Zhongling Su, Zerui Wang, Zheng Cai, Zhilin Pei, and Xingcheng Zhang. Packmamba: Efficient processing of variable-length sequences in mamba training, 2024.

[64] Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. Quantifying the carbon emissions of machine learning. arXiv preprint arXiv:1910.09700, 2019.

Appendix A

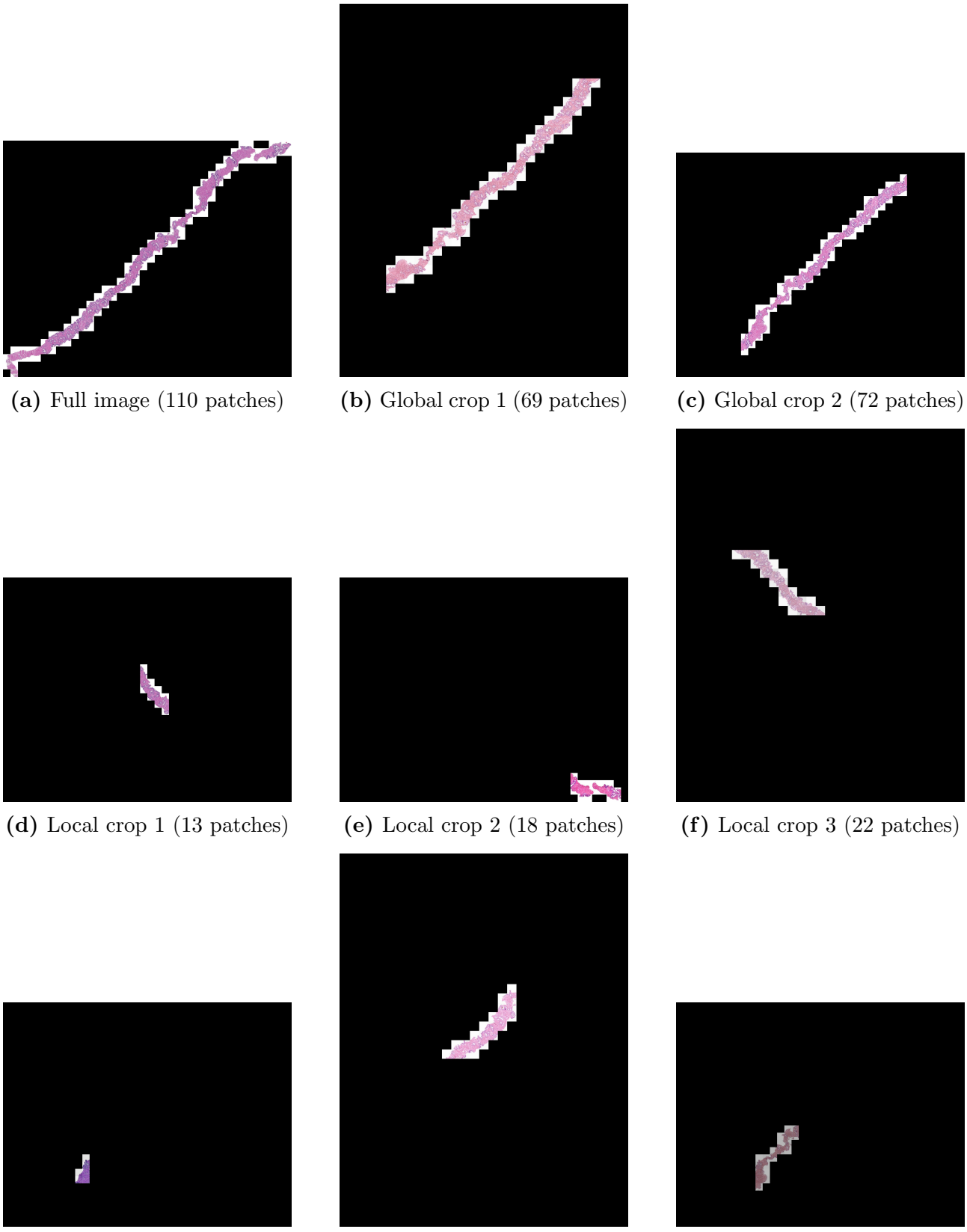
Appendix

A.1 Layerwise Network Architecture

Table A.1: Overview of Pixel-Mamba-6M architecture. RF corresponds to Region Fusion, TE denotes Token Expansion either horizontally or vertically with Cat and Avg describing Concatenate and Average as the expansion method. Table reused from Pixel-Mamba paper [28].

Pixel-Mamba-6M			
Layer id	**Token Size**	**Channels**	**Pixel-Mamba Layers**
1	1×1	3	Mamba + RF + Horizontal TE-Cat
2	1×2	6	Mamba + RF + Vertical TE-Cat
3	2×2	12	Mamba + RF + Horizontal TE-Cat
4	2×4	24	Mamba + RF + Vertical TE-Cat
5	4×4	48	Mamba + RF
6	4×4	48	Mamba + RF + Horizontal TE-Cat
7	4×8	96	Mamba + RF
8	4×8	96	Mamba + RF + Vertical TE-Cat
9	8×8	192	Mamba + RF
10	8×8	192	Mamba + RF
11	8×8	192	Mamba + RF + Horizontal TE-Avg
12	8×16	192	Mamba + RF
13	8×16	192	Mamba + RF
14	8×16	192	Mamba + RF + Vertical TE-Avg
15	16×16	192	Mamba + RF
16	16×16	192	Mamba + RF
17	16×16	192	Mamba + RF
18	16×16	192	Mamba + RF
19	16×16	192	Mamba + RF
20	16×16	192	Mamba + RF + Horizontal TE-Avg
21	16×32	192	Mamba + RF
22	16×32	192	Mamba + RF + Vertical TE-Cat
23	32×32	384	Mamba + RF
24	32×32	384	Mamba + RF

A.2 Example WSI DINO Views



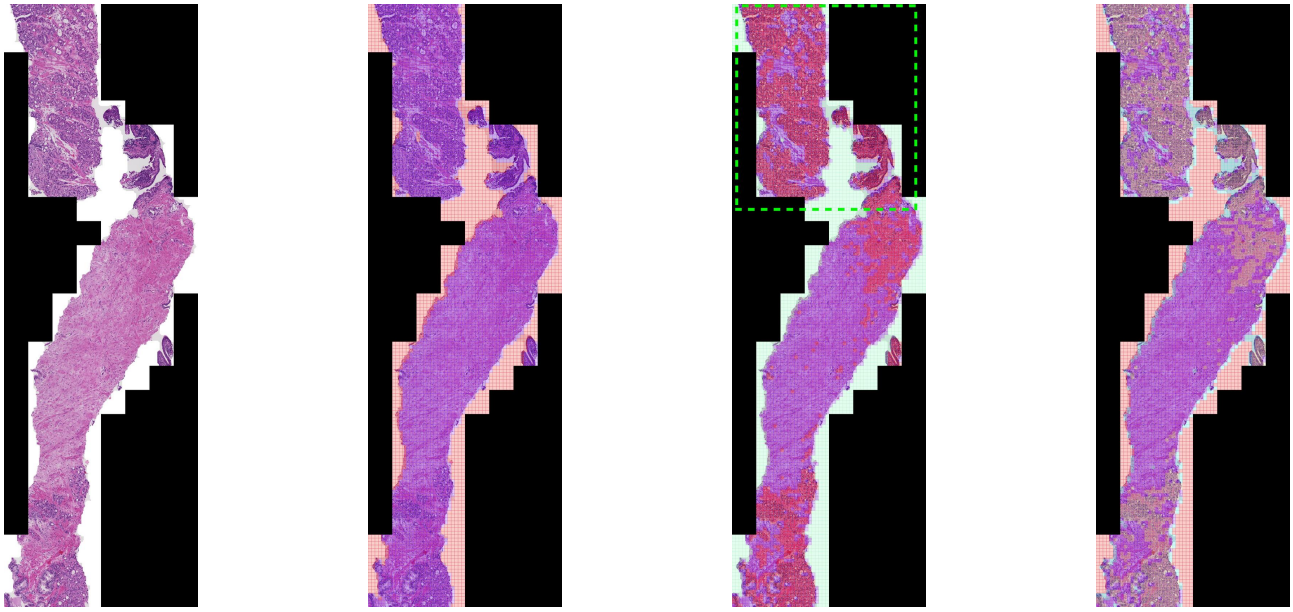
(a) Full image (110 patches) **(b)** Global crop 1 (69 patches) **(c)** Global crop 2 (72 patches)

(d) Local crop 1 (13 patches) **(e)** Local crop 2 (18 patches) **(f)** Local crop 3 (22 patches)

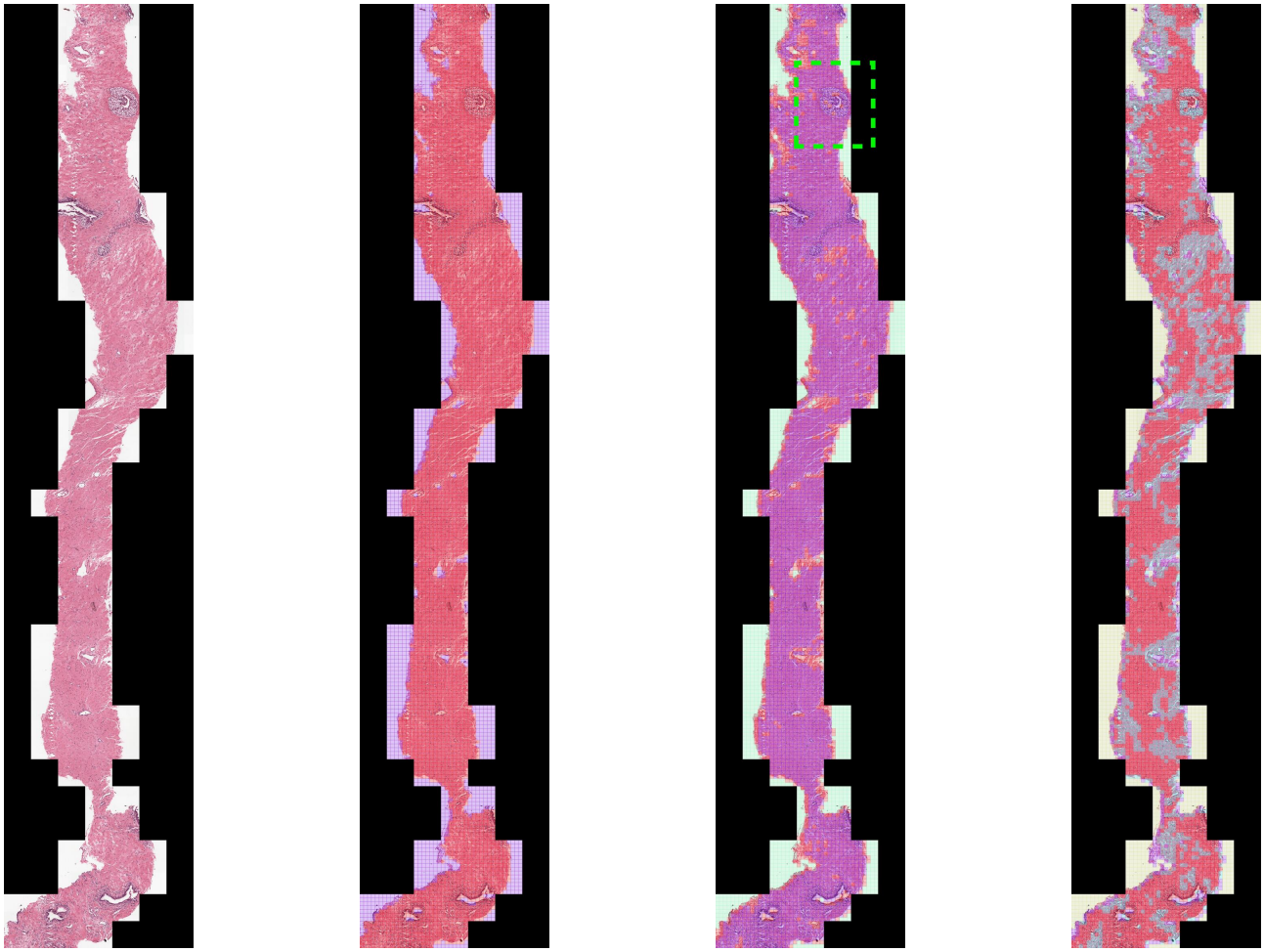
(g) Local crop 4 (6 patches) **(h)** Local crop 5 (22 patches) **(i)** Local crop 6 (20 patches)

Figure A.1: Examples of global and local views for DINO pretraining on PANDA. Each view is sampled from the same WSI. Full image: all foreground patches. Global views cover 40–100% of the foreground area. Local views cover 5–40%. Zoom in for finer details.

A.3 Qualitative Examples of Embeddings with Latent-MIM



Original | 2 clusters | 3 clusters | 4 clusters



Original | 2 clusters | 3 clusters | 4 clusters

Figure A.2: Unsupervised segmentation maps on two validation samples obtained by clustering spatial token embeddings. One final token corresponds to 32×32 pixels on the input image. Highlighted are the sections shown in the main paper. Zoom in for finer details.

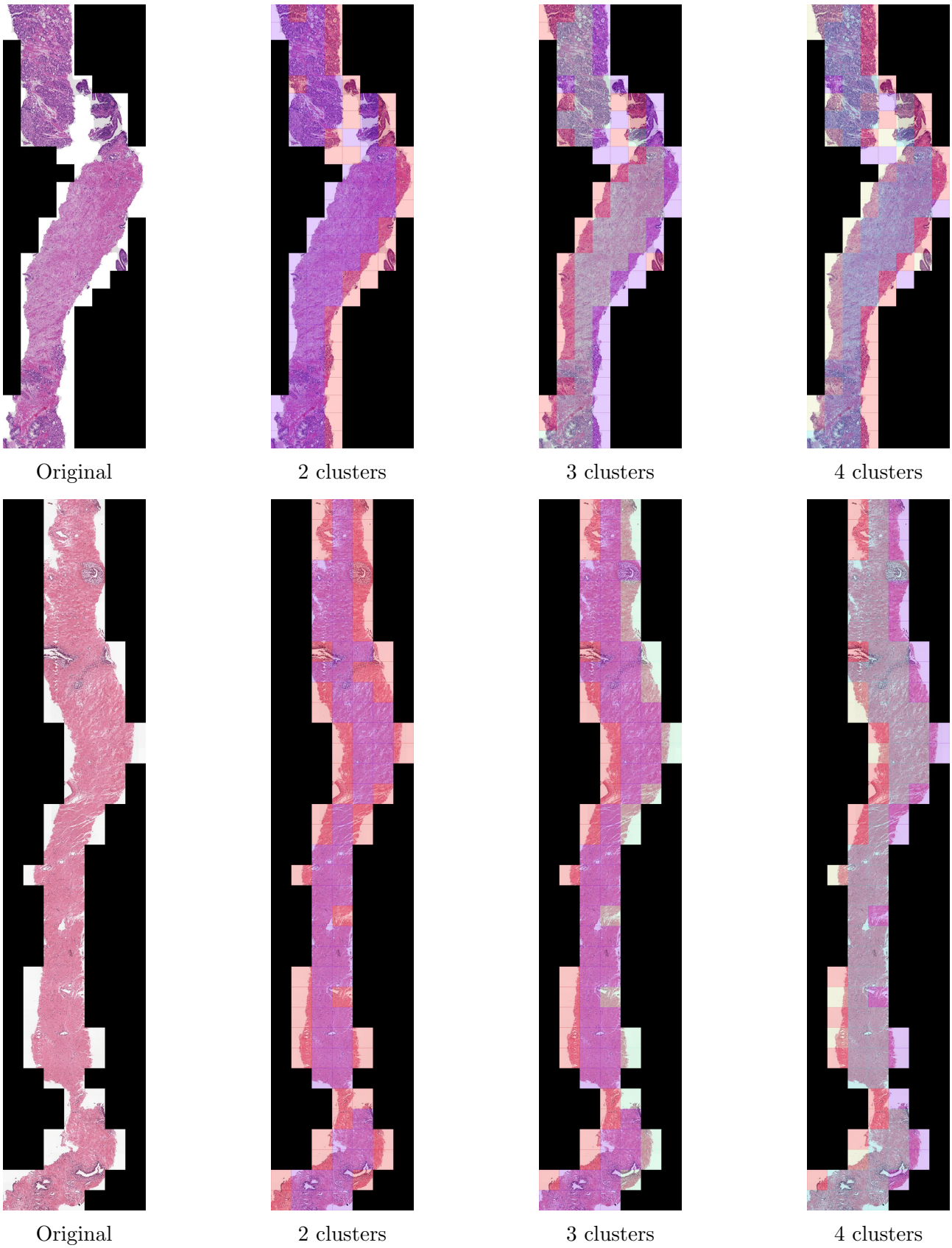


Figure A.3: Unsupervised segmentation maps on two validation samples obtained by clustering CLS token embeddings. One CLS token corresponds to a region of 224×224 pixels. Zoom in for finer details.