MUVIT: Multi-Resolution Vision Transformers for Learning Across Scales in Microscopy

Supplementary Material

A. Related work and novelty

Method	Hierarchical Stages	Physical Multi-Res Input	Joint Attention	World Coordinate Alignment
MultiMAE	✗	✗	✓	✗
Scale-MAE	✗	✗	✗	✗
PVT / Swin	✓	✗	✗	✗
HIPT	✓	✓	✗	✗
MUVIT (Ours)	✗	✓	✓	✓

Table S1. Conceptual comparison of relevant related work.

B. Training details and hyperparameters

This section provides comprehensive implementation details for reproducibility. Note that the full codebase (including experiments code) and datasets will be released upon publication.

B.1. Hardware specifications

All models were trained on the *Capella* HPC cluster at TU Dresden. Its nodes consist of 2x AMD EPYC 7282 CPUs (64 cores). Runs booked up to 256 GB RAM (DDR-4800). For MAE pre-training, models were trained on a single node in a multi-GPU setting (4x Nvidia H100, 96GB). All semantic segmentation models (including the baselines) were trained on a single node using a single GPU (1x Nvidia H100, 96 GB)

B.2. MAE pre-training

Architecture. The encoder consists of 12 transformer layers with embedding dimension $D = 512$ and patch size $P = 8$. Each resolution level has independent patch embedding layers followed by learnable level embeddings. The decoder uses L lightweight networks (one per level), each with 2 transformer layers and dimension 256. The first decoder layer incorporates cross-attention to the full set of visible encoder outputs with coordinate-based RoPE. All transformer layers (including those in the decoder) use the coordinate-based RoPE embeddings described in the main text.

Masking strategy. We use a masking ratio of $\rho = 0.75$ with Dirichlet-weighted sampling across resolution levels. The proportions of masked tokens per level $(\lambda_1, \ldots, \lambda_L)$ are sampled from $\mathrm{Dir}(\alpha)$ with $\alpha = 0.5$, encouraging the model to learn cross-scale relationships by varying the relative masking ratio per level.

Data sampling. Due to the large-scale nature of the data, we employ a sampling-based strategy in which a nested multi-resolution crop is randomly sampled from each image (see Section 4). We therefore use the term *training sample* to denote such a multi-resolution crop extracted from a training image. We define an *epoch* as a pass of 10000 training samples to the network during training. Sampled crops are normalized with percentile-based normalization ($p_m, p_M = 1, 99.8$) for MOUSE and by dividing by 255 for KPIS.

Multi-resolution configuration. For KPIS, we use resolution levels $[1, 8, 32, 64]$ during MAE pre-training. For MOUSE, we use levels $[1, 8, 32]$ for both MAE pre-training. All levels use the same pixel-space patch size (256×256 pixels), with lower resolution levels capturing progressively larger spatial context.

Optimization. MAE pre-training uses a batch size of 48/64 for 2000/1000 epochs for MOUSE/KPIS epochs with learning rate 6×10^{-5}. We use the AdamW optimizer with gradient clipping at 0.5 and deterministic cosine annealing learning rate schedule. Early stopping is not used.

Loss function. The loss function for MAE pre-training is the mean squared error (MSE) (computed on masked patches only) averaged across all resolution levels.

Data augmentation. Used augmentations include random horizontal and vertical flips as well as intensity jittering and bounding box perturbations (random shifts and scaling). Note that the horizontal and vertical flips affect the whole nested crop (*i.e.* they are not applied independently to each resolution level) and that bounding boxes are augmented accordingly.

B.3. Segmentation fine-tuning

Architecture. We use the same pre-trained encoder architecture (12 transformer layers, token embedding dimension $D = 512$) and couple it with either a UNETR-style decoder or Mask2Former decoder as described in Section 3.

Data sampling. We use the same sampling strategy as for MAE pre-training. Note that the target semantic segmentation mask is only sampled at the highest resolution level. For KPIS we ensure that at least half the training samples contain at least a positive class pixel. We define an *epoch* as a pass of 4096 training samples to the network during training for the semantic segmentation task. Sampled crops are normalized with percentile-based normalization ($p_m, p_M = 1, 99.8$) for MOUSE and by dividing by 255 for KPIS.

Multi-resolution configuration. For KPIS, we extract only levels $[1, 8]$ from the MAE pretrained encoder for segmentation fine-tuning (as shown in Tab. 4). For MOUSE,

we use the full encoder. All levels use the same pixel-space patch size (256×256 pixels for MOUSE, 512×512 for KPIS differing from the pre-training patch size).

Optimization. Semantic segmentation training uses a batch size of 16 for MOUSE (except MUVIT+UNETR variants which used a batch size of 8 due to memory limitations) and a batch size of 8 for KPIS. All our models were trained for 100 epochs. We employ asymmetric learning rates: decoder learning rate 1×10^{-4}, encoder learning rate 1×10^{-5} ($0.1\times$ decoder rate). We use the AdamW optimizer with gradient clipping at 0.5 and cosine annealing schedule.

Loss function. We use a combined cross-entropy and Dice loss: $\mathcal{L} = \lambda_{\text{CE}} \cdot \mathcal{L}_{\text{CE}}(\tilde{y}, y) + \lambda_{\text{Dice}} \cdot \mathcal{L}_{\text{Dice}}(\tilde{y}, y)$ with $\lambda_{\text{CE}} = 1.0$ and $\lambda_{\text{Dice}} = 0.1$. For KPIS, we double the weight to the foreground class to aid with the imbalance between foreground and background. The Dice loss excludes the background class (class 0) and is computed separately per resolution level before averaging.

Data augmentation. We use the same augmentation set as in the MAE pre-training case.

C. MUVIT components

C.1. Number of parameters

Table S2. Number of parameters for MUVIT components and variants.

Module	Parameters
MUVIT$_{[1]}$ (encoder)	25.24M
MUVIT$_{[1,8,32]}$ (encoder)	25.31M
MUVIT$_{[1]}$+MAE decoder	26.44M
MUVIT$_{[1,8,32]}$+MAE decoder	28.88M
MUVIT$_{[1]}$+Mask2Former decoder	33.44M
MUVIT$_{[1,8,32]}$+Mask2Former decoder	36.80M
MUVIT$_{[1]}$+UNETR decoder	31.66M
MUVIT$_{[1,8,32]}$+UNETR decoder	35.02M

C.2. Semantic Segmentation Decoder

UNETR decoder The *UNETR* decoder uses skip connections from multiple intermediate encoder layers to preserve fine-grained spatial information during upsampling, as introduced in the original UNETR work. The decoder consists of a progressive upsampling pathway that combines transformer-encoded features with skip connections at multiple resolution levels. We extract intermediate features from arbitrarily selected encoder layers (by default, evenly spaced across the encoder depth) and project them to a D-dimensional vector ($D = 256$). The decoder operates through multiple upsampling stages, where each stage performs bilinear upsampling followed by residual convolutional blocks that fuse the upsampled features with corresponding skip connections from the intermediate encoder outputs. The last skip connection involves directly processing the original input image at the highest resolution level instead of the encoder outputs. Each upsampling stage consists of bilinear interpolation followed by two regular convolutional blocks (3×3 convolutional layers with batch normalization and a ReLU activation). Skip connections are incorporated through element-wise addition after using 1×1 convolutions to match the channel axes. The final output layer produces segmentation predictions at full input resolution through a 1×1 convolution that maps from the embedding dimension D to the target number of output channels/classes.

Mask2Former decoder The *Mask2Former* decoder follows the query-based segmentation paradigm introduced in the original work. The decoder consists of three main components: a pixel decoder that progressively upsamples encoder features to full resolution, a set of learnable mask queries that represent potential objects, and a transformer decoder that refines these queries. For the pixel decoder we use a series of bilinear upsampling layers followed by regular convolutional blocks (as described above) in order to create dense mask features at the original image resolution. We initialize $|M_q|$ learnable embeddings of dimension $D = 256$ that serve as mask queries, where the number of queries is set to be at least equal to the number of output classes. The transformer decoder refines the mask queries using multiple transformer layers. Each layer applies self-attention among the queries to model relationships between different potential masks, followed by cross-attention between queries and the encoder features. We use RoPE for the spatial features obtained from the encoder to maintain spatial awareness during cross-attention, while queries remain position-agnostic so that they can be used as learnable representations. We then feed the refined queries to a layer normalization module followed by an MLP to produce the mask embeddings. The final segmentation masks are generated by computing the dot product between these mask embeddings and the pixel decoder's mask features, followed by a refinement convolution that maps from $|M_q|$ to the desired number of output channels.

D. Semantic segmentation results

D.1. MOUSE

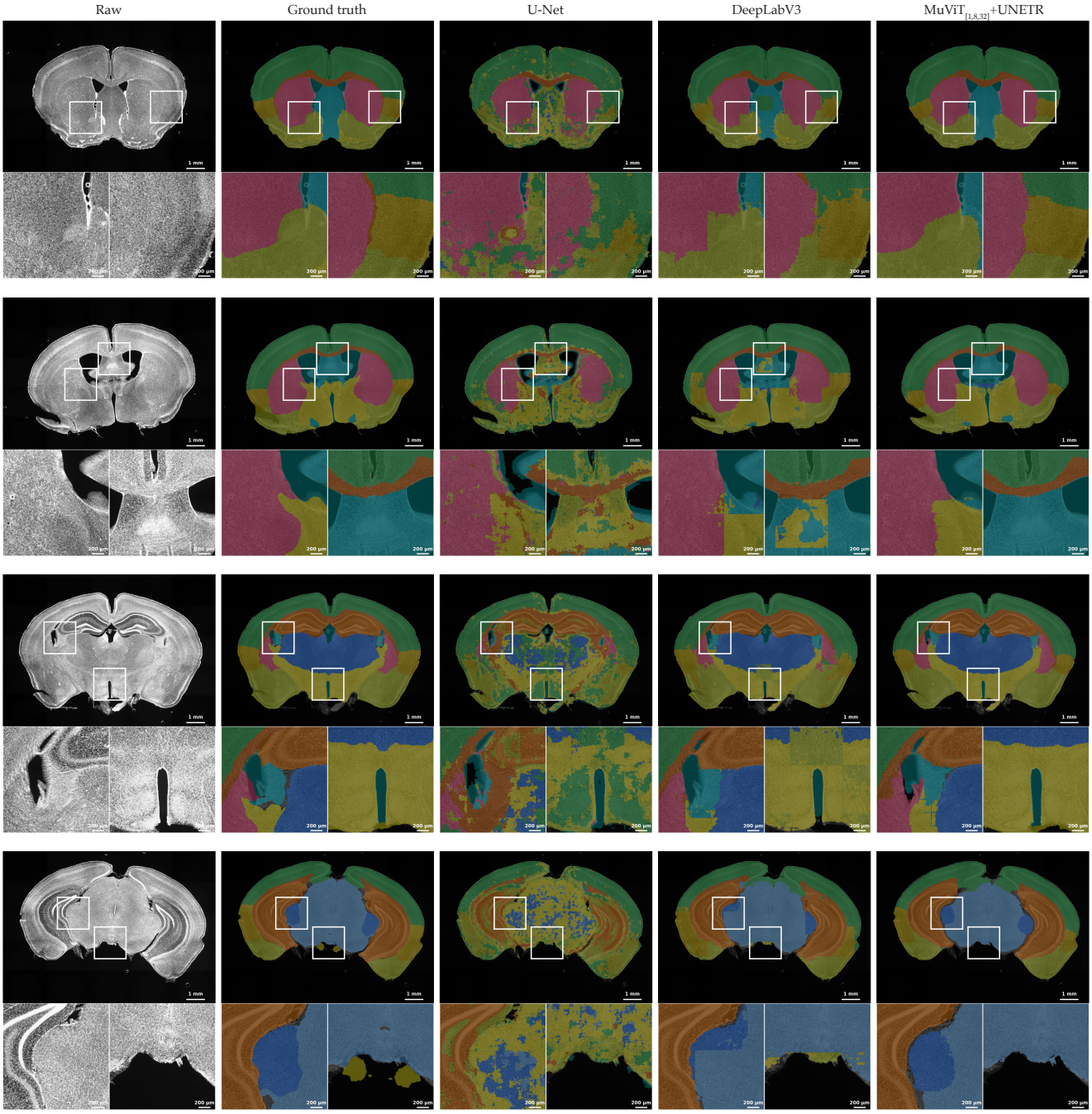

Figure S1. Semantic segmentation results on four test images of MOUSE. The images correspond to different brain sections of the same specimen, exhibiting anatomical changes and, thus, showcasing different semantic classes predictions. Columns show (left to right): input image, ground truth, U-Net, DeepLabV3, and MUVIT$_{[1,8,32]}$+UNETR semantic predictions. The input sizes of the baselines are 1024×1024 while the input for MUVIT$_{[1,8,32]}$+UNETR is $3 \times 256 \times 256$. Magnified regions (white boxes) show detailed comparison.

D.2. KPIS

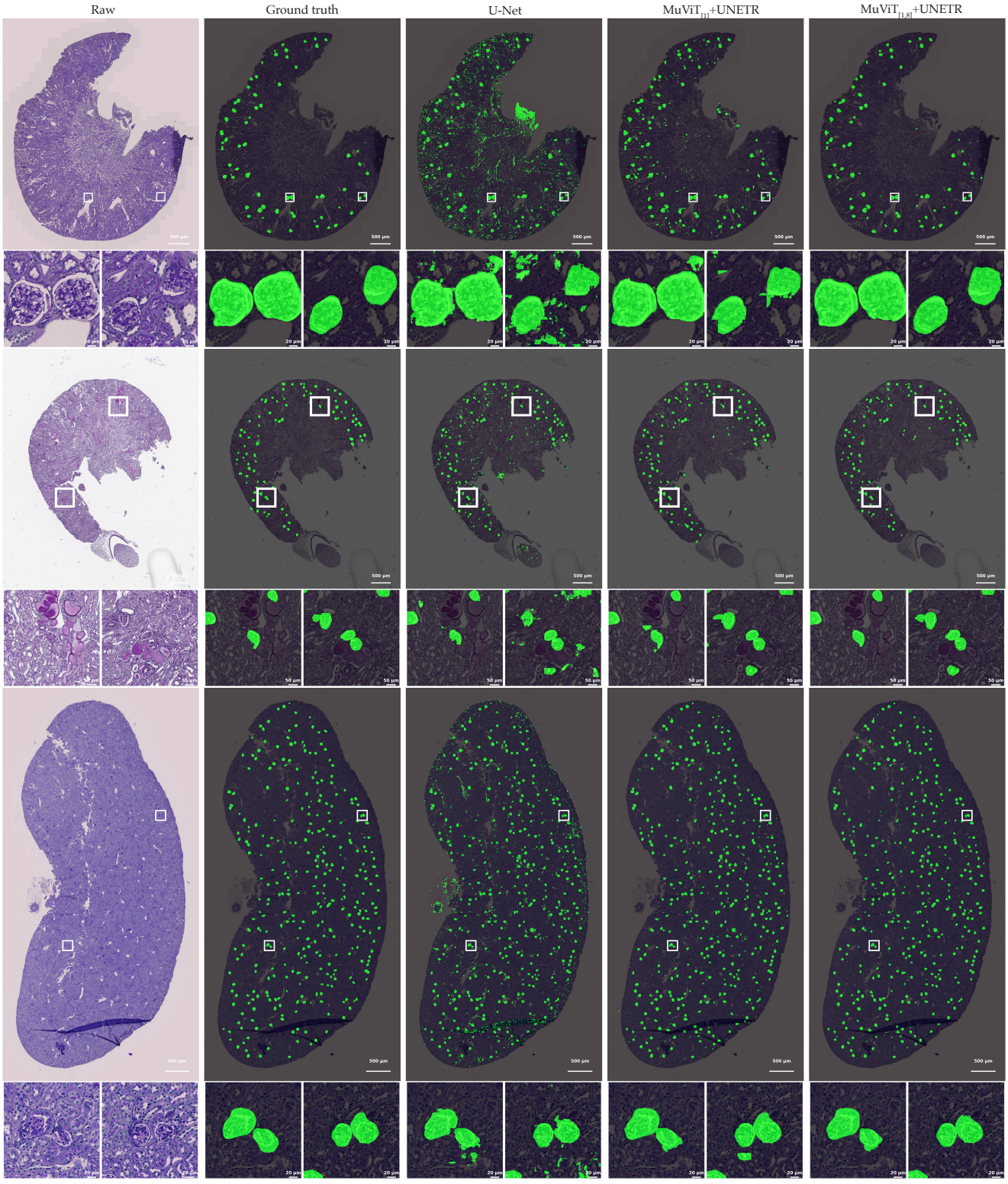

Figure S2. Semantic segmentation results on three test whole slide images (WSIs) of KPIS. The images correspond to three different conditions. Columns show (left to right): input image, ground truth, U-Net, MuViT$_{[1,8]}$+UNETR, and MuViT$_{[1,8]}$+UNETR semantic predictions. The spatial input size of all models is 512×512. Magnified regions (white boxes) show detailed comparison.

D.3. MAE

Table S3. MOUSE training progress results. Validation mDSC (11 classes) is reported at different epochs during training.

Method	Input Size	Epoch			
		E1	E10	E25	E50
U-Net	1024×1024	0.086	0.110	0.288	0.412
DeepLabV3	1024×1024	0.234	0.582	0.634	0.704
SegFormer	1024×1024	0.045	0.159	0.266	0.332
SwinUNETRV2	1024×1024	0.049	0.232	0.305	0.388
MUVIT [1] +Mask2Former	256×256	0.033	0.142	0.266	0.310
MUVIT [1] +UNETR	256×256	0.105	0.086	0.261	0.281
MUVIT [1,8,32] +Mask2Former	3×256×256	**0.619**	**0.843**	**0.880**	**0.878**
MUVIT [1,8,32] +UNETR	3×256×256	0.297	0.748	0.852	0.864

E. Coordinate sensitivity

Despite world-coordinate alignment across multiple scales being essential for performance as highlighted by the *naive* experiments (Tabs. 2 to 4), we further test whether MUVIT models exhibit robustness to coordinate noise during inference. Note that we include an augmentation which adds scaling and shift to the samples bounding box coordinates during training.

We thus performed such an analysis by measuring the segmentation performance of the best performing MUVIT models for each dataset while adding Gaussian noise with standard deviation σ to the inference bounding box coordinates. We observe that MUVIT remains robust to such noise, with only minor degradation up to $\approx$ 32px on all datasets (Figure S3).

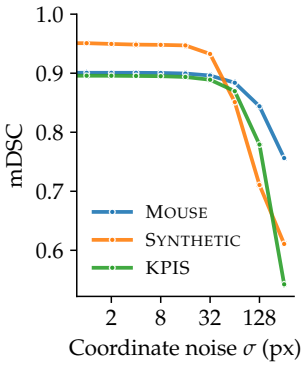

Figure S3. Effect of adding Gaussian noise of varying standard deviation σ on MUVIT segmentation performance across different datasets.

F. Scaling behaviour

We also benchmarked runtime and peak memory on arbitrarily-sized inputs using a fixed patch size $p = 8$, as used throughout the paper, as well as an increasing number of resolution levels $l \in \{1, 8, 32\}$. Figure S4 expectedly showcases that MUVIT incurs computational overhead in exchange for a substantially larger effective context.

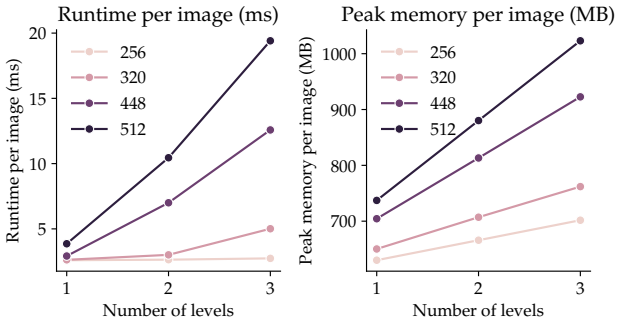

Figure S4. Runtime and memory scaling of MUVIT. Hue denotes input spatial dimensions.