MFTrans: A Multi-Resolution Fusion Transformer for Robust Tumor Segmentation in Whole Slide Images - Supplementary

S1. Computational complexity of GT-MSA.

The overall computational complexity for the GT-MSA and G-MSA (from GLAM-transformer [35]) operations for given input global tokens $G \in \mathbb{R}^{N_g \times N_w \times C}$ are given by:

$$\Omega(\text{GT-MSA}) = N_g \times N_w \times C \times (4C + 2N_w),$$
$$\Omega(\text{G-MSA}) = N_g \times N_w \times C \times (4C + 2N_w \times N_g),$$

where N_g is the number of global tokens per window, N_w is the number of windows, and C is the patch embedding dimension.

Specifically, the computational complexity of GT-MSA consists of two main parts: 1) the computation of the query, key, and value (Ω_{qkv}), and 2) the computation of self-attention ($\Omega(q \times k), \Omega(k \times v)$, and $\Omega(linear)$). First, we compute the query, key, and value for each global tokens (N_g) and windows (N_w) through linear transformations with dimension of C. The computational complexity for this operation is:

$$\Omega(qkv) = 3 \times N_g \times N_w \times C^2,$$

Second, for the self-attention, given query($Q \in \mathbb{R}^{N_w \times C}$), key($K \in \mathbb{R}^{N_w \times C}$), and value($V \in \mathbb{R}^{N_w \times C}$), we compute the dot product between the query and key, followed by multiplying the result with the value for all global tokens independently. The output global tokens are then calculated using a linear transformation. The computational complexity for these operations are:

$$\Omega(q \times k) = N_g \times N_w^2 \times C,$$
$$\Omega(k \times v) = N_g \times N_w^2 \times C,$$
$$\Omega(linear) = N_g \times N_w \times C^2$$

Threrfore, the computational complexity of GT-MSA is lower than that of G-MSA, as GT-MSA computes self-attention independently for each global token.Our method computes the global tokens at the same positions within each window independently, while G-MSA considers all global tokens across the entire window simultaneously. Despite its lower complexity, GT-MSA outperforms G-MSA in segmentation performance.

S2. Detailed classifier block

In our model, we add an auxiliary network, denoted as the classifier block, to enhance discrimination in the lower stages of our model. Specifically, the global feature from the last stage, $G_4 \in \mathbb{R}^{N_g \times 8C}$, is passed through a linear layer to reduce its dimensionality. Following this, we employ an attention-based MLP to classify whether each window originates from the tumor or normal tissue, as shown in fig. S1. The attention weights are computed by a linear layer followed by a softmax operation, which assigns higher importance to relevant tokens. For classification, we use binary cross-entropy loss, where a label of 1 indicates the tumor patch, and 0 indicates the normal patch.

To assess the effect of classification loss in the total loss, Eq. (5), we compare the segmentation performance by varying ω using the Camelyon16 dataset. As shown in table S1, ω = 0.2 shows the best segmentation performance, with a Dice score of 0.818, Jaccard index of 0.692, and accuracy of 0.938. Notably, the performance significantly dropped, Dice score of 0.759, Jaccard index of 0.612, and accuracy of 0.911, when the classification loss is not used (i.e., ω = 0). This result demonstrate the importance of the classifier

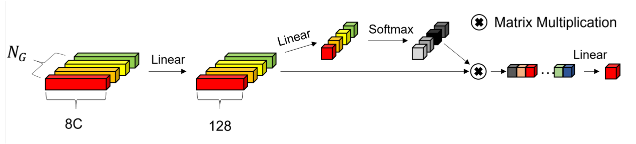

Figure S1. **Architecture of the classifier block.** An attention-based MLP is used for tumor versus normal tissue classification.

	ω = 0	ω =0.2	ω =0.4
Dice	0.759	**0.818**	0.806
Jaccard	0.612	**0.692**	0.675
Accuracy	0.911	**0.938**	0.935

Table S1. **Effect of classification loss.** Segmentation performance is compared by varying ω in the total loss on the Camelyon16 dataset. Performance is reported in terms of Dice score, Jaccard index, and accuracy, with top results in **bold**.

block in helping the model capture global context more effectively.

S3. Detailed Image Decoder

To effectively upsample image features, we modify the architecture of the image decoder across different stages. The detailed architecture is provided in table S2. In the first stage, The height and width are each increased by 4 times with the same channel dimension. This is accomplished using four kernels of varying sizes: 4×4, 8×8, 16×16, and 32×32, with different the channel dimensions: C/2, C/4, C/8, and C/8, respectively. The outputs are concatenated to maintain the same channel with input. In the subsequent stages, the height and width are doubled while the channel dimension is halved. Two kernels, 2×2 and 4×4, are employed, and each output feature maps with a reduced channel dimension of C/4. The output feature maps are concatenated and passed forward as input to the “Fusion stage” in the next stage, as shown in fig. 3-(b), and (c).

To evaluate the effectiveness of the image decoder, we compare it with bilinear upsampling on the Camelyon16 dataset. As shown in table S3, we observe that the image decoder outperformed bilinear upsampling, achieving a Dice score of 0.818, a Jaccard index of 0.692, and an accuracy of 0.938. This result demonstrate that the image decoder provides superior performance compared to bilinear upsampling method.

Stage	Kernel	Stride	dim
1	4×4	4×4	C / 2
	8×8	4×4	C / 4
	16×16	4×4	C / 8
	32×32	4×4	C / 8
$2, 3, 4$	2×2	2×2	C / 4
	4×4	2×2	C / 4

Table S2. **Detailed architecture of the image decoder across different stages.**

	Image Decoder	Upsample
Dice	**0.818**	0.813
Jaccard	**0.692**	0.685
Accuracy	**0.938**	0.939

Table S3. **Effect of image decoder.** Segmentation performance is compared between the image decoder versus bilinear upsampling on the Camelyon16 dataset. Performance is reported in terms of Dice score, Jaccard index, and accuracy, with top results in **bold**.

S4. Effect of global token number on PAIP.

We conduct a comparative study to assess the effect of the number of global tokens on our method for datasets from different organs (i.e., PAIP2019). Table 5 and table S5 present the segmentation performance comparison results for the Camelyon16 and PAIP2019 datasets, respectively [3,21]. Similar to the results on the Camelyon16 dataset, MFTrans with nine tokens demonstrates the best performance on the PAIP2019 dataset, with performance dropping when fewer or more than 9 tokens are used. Despite the datasets being from different organs, the similar outcomes suggest that our model is robust and generalizable across various organ types.

	7	8	**9**	10	11	12
Dice	0.773	0.790	**0.801**	0.788	0.781	0.774
Jaccard	0.648	0.671	**0.688**	0.669	0.657	0.647

Table S4. **Effect of number of Global token on PAIP2019 dataset.** We compare segmentation performance with varying numbers of Global tokens on the Camelyon 16 dataset. The tumor segmentation performance is compared in terms of Dice score, Jaccard index, and accuracy, with top results in **bold**.

	7	8	**9**	10	11	12
Dice	0.899	0.880	0.859	0.895	0.902	0.892
Jaccard	0.825	0.825	0.778	0.821	0.829	0.814

Table S5. Comparison with segmentation performance with varing number of Global token.

S5. Experimental setting

S5.1. Data preparation

We evaluate our proposed model on three different datasets: Camelyon16, Catholic Uijeongbu St. Mary’s hospital, and PAIP2019. For the Camelyon16 dataset, a total of 400 H&E-stained WSIs are used, with 270 WSIs for training and 130 WSIs for testing. In the Catholic Uijeongbu St. Mary’s hospital dataset, we collect a total of 83 WSIs of lymph nodes from 36 patients, with 43 WSIs used for training, 29 WSIs for validation, and 10 WSIs for testing. For the PAIP2019 dataset, 60 H&E-stained WSIs from liver biopsies are used, with 36 WSIs for training, 12 WSIs for validation, and 12 WSIs for testing. During training, we randomly select to balance the proportion of tumor and normal patches to avoid the class imbalance problem. All WSIs undergo pre-processing, as described in section S5.2, before further analysis.

S5.2. Data pre-processing

We conduct several pre-processing steps to avoid batch effect. Briefly, we first segment the tissue area from background area with Otsu algorithm. Then, we divide WSI into non-overlapping patch of 224×224 pixel, which is used for input of training and testing phase. The patches containing less than 10% of the tumor area are excluded.

S5.3. Evaluate strategy

To evaluate performance, we employ several commonly used metrics for segmentation tasks, including Dice score, Jaccard Index, and Accuracy. These metrics are calculated based on the counts of True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN). The metrics are defined as follows:

$$\text{Dice } = \frac{2 \times \text{TP}}{2 \times \text{TP} + \text{FP} + \text{FN}} \tag{1}$$

$$\text{Jaccard Index} = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}} \tag{2}$$

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{3}$$

In our implementation, we compute the metrics at the WSI level by merging non-overlapping segmented patches from the model into a complete WSI. However, for the Camelyon16 dataset, the metrics are calculated using patches because the tumor regions in the patches are too small compared to the normal areas, making it difficult to compute a reliable metric.