2DMamba: Efficient State Space Model for Image Representation with Applications on Giga-Pixel Whole Slide Image Classification

Supplementary Material

A. Implementation Details

SSM. For a fair comparison, we use a single SSM-based block with a 128-dimensional SSM and set the state dimension to 16 for all Mamba-based methods.

Feature extractor. We use UNI [6], a well-known and current SOTA foundation model for feature extraction, which is a ViT-L/16 pretrained on more than 100 million pathology patches from from over 100,000 H&E-stained WSIs across 20 major tissue types. UNI is pretrained in a self-supervised manner using DINOv2 [35].

Aggregator. The aggregator in our 2DMamba follows [17, 43] using attention pooling [17] and two linear layers (128 intermediate dimensions). This module produce the attention scores of each patch embedding, then aggregate them using the weighted summation to produce the WSI embedding. Mathematically, let $H = \{h_1, \ldots, h_K\}$ be a bag of K embeddings, the attention pooling is:

$$z = \sum_{k=1}^{K} a_k h_k, \tag{9}$$

where:

$$a_k = \frac{\exp\{w^\top \tanh\left(V h_k^\top\right)\}}{\sum_{j=1}^{K} \exp\{w^\top \tanh\left(V h_j^\top\right)\}}, \tag{10}$$

where $w \in \mathbb{R}^{128\times 1}$ and $V \in \mathbb{R}^{128\times M}$ are parameters (M is the embedding dimension). For fair comparisons, we use the same feature extractor and aggregator as other MILs.

WSI pre-processing. We extract patches from WSIs at 20x magnification with no overlapping. The patch size is set to 512x512 pixels. We used the preprocessing tool in CLAM [29] to segment and extract tissue regions.

Training. We use AdamW [28] to optimize the models for 20 epochs of training with batch size being 1. The initial learning rate is set to 0.0001 and is adjusted with a cosine annealing scheduler. All pathology and natural image experiments are trained using one NVIDIA V100 GPU and eight NVIDIA A100 GPUs, respectively.

B. Details of the hardware-aware 2D selective scan operator

In this section, we detail the hardware-aware 2D selective scan operator in 2DMamba introduced in section 3.4. The forward pass of the 2D selective scan, implemented as a fused kernel with 2D tiling, is formulated in the algorithm below. The backward pass of 2DMamba follows a similar structure but involves four 2D selective scans: one horizontal and one vertical scan to reconstruct intermediate variables from the forward pass, and one horizontal reverse and one vertical reverse scan to propagate gradients.

Algorithm 2D selective scan (fused kernel with tiling)

```
Require: 2D input feature $x : (H, W)$; Time step $\Delta : (H, W)$;
Require: State dimension $N$. Tile size $T$.
SSM parameters: Input independent $A : (N)$, $D$ and $bias$;
SSM parameters: Input dependent $B : (H, W)$;
SSM parameters: Input dependent $C : (H, W)$.
Return: 2D aggregated result $y$.
  $K_H = \lceil H/T \rceil$
  $K_W = \lceil W/T \rceil$
  # loop for $K_H * K_W$ tiles
  for $k_h = 1$ to $K_H$ and $k_w = 1$ to $K_W$ do
      $x_{k_h,k_w} : (T, T)$ = Read from HBM.
      $\Delta_{k_h,k_w} : (T, T)$ Read from HBM.
      $\Delta_x = softplus(\Delta * x_{k_h,k_w} + bias)$
      $y_{k_h,k_w} = 0$
      # loop for $N$ state dimensions
      for d = 1 to $N$ do
          $A^d$ = Read from HBM.
          $B^d_{k_h,k_w}$ = Read from HBM.
          $C^d_{k_h,k_w}$ = Read from HBM.
          $B^d_\Delta x = B^d * \Delta * x_{k_h,k_w}$
          $A^d_\Delta = A^d * \Delta_x$
          # Initialize horizontal and vertical prefix
          $P^h$ = Read $P^h_{k_h,k_w-1}$ from HBM.
          $P^v$ = Read $P^h_{k_h-1,k_w}$ from HBM.
          Initialize $y = 0$
          $h^{hor,d} = parallel\_horizontal\_scan(A^d_\Delta, B^d_\Delta x, P^h)$
          Write last column of $h^{hor,d}$ as $P^h_{k_h,k_w}$ to HBM.
          $h^d = parallel\_vertical\_scan(A^d_\Delta, h^{hor,d}, P^v)$
          Write last row of $h^d$ as $P^v_{k_h,k_w}$ to HBM.
          $y_{k_h,k_w} = y_{k_h,k_w} + C^d * h^d$
      end for                                          ▷ End $N$ states
      $y_{k_h,k_w} = y_{k_h,k_w} + D * x_{k_h,k_w}$
      Write $y_{k_h,k_w}$ to HBM.
  end for                                              ▷ End tiles
```

Memory access complexity. In the algorithm, we use T to denote the height and width of a tile, and K_H, K_W to represent the number of tiles along height and width dimensions. Thus, we have $H = K_h \times T$ and $W = K_w \times T$. We also assume $N = \mathcal{O}(H + W)$. We further use green color to highlight the extra HBM transactions of 2DMamba compared to the vanilla Mamba. Specifically, for each tile, an extra row and column for every state dimension must be read and written to concatenate tiles. This adds an extra memory access complexity of $\mathcal{O}(NT)$ per tile. The total extra memory access complexity is therefore $K_h \times K_w \times \mathcal{O}(NT) = \mathcal{O}(N(H+W))$. Since we assume $N = \mathcal{O}(H+W)$, this simplifies to $\mathcal{O}(H \times W) = \mathcal{O}(L)$, where L the the sequence length. Thus, the extra memory access complexity matches that of vanilla Mamba, ensuring the total memory access complexity remains $\mathcal{O}(L)$.

Correctness. The 2D scan is decomposed into a horizontal scan and a vertical scan, which are performed sequentially. That is, we first conduct the horizontal scan, and after it's done, we then conduct the vertical scan. However, each scan itself is conducted by a GPU scanner in parallel, ensuring the overall efficiency. This sequential process guarantees the correctness of the decomposition. Meanwhile, the correctness of the 1D parallel scan algorithm is elaborated by the vanilla Mamba.

Tiling and edge cases. We choose two grid tile sizes: 16×16 and 32×32. Feature maps smaller than 32×32 are processed directly without tiling. Larger feature maps are tiled into 32×32 blocks. This tile size is chosen manually to balance between runtime efficiency and hardware constraints: While a larger tile size brings in higher parallelism, it comes with the cost of increased register and SRAM consumption. An excessive grid size will result in register spills, which will severely penalize performance. Following vanilla Mamba [15]'s practice, our current choice is the trade-off between parallelism and register spills. The tiles are processed sequentially, from top left to bottom right. This process is similar to conducting a 2D convolution over the input feature map, with kernel size equal to our tile size, and stride equal to 16 or 32. For input sizes that are not perfect multiples of our tile size, we pad the "spilling" areas with naive values $\bar{A} = 1$ and $x = 0$, which is also the strategy of the vanilla Mamba scanner.

Thread granularity and load balancing. Each feature map is processed with 64 threads, which properly respects the 32-thread granularity. For tile size 16×16, each thread processes a 2×2 subregion; for 32×32, each thread processes a 4×4 subregion. Accordingly, all threads process the same amount of data so *load imbalances* are not an issue.

Difference with other hardware-optimized methods. To the best of our knowledge, all currently available hardware-optimized Mamba-based methods rely on the vanilla Mamba scanner and its CUDA pipeline, which conduct 1D scans (2D input must be flattened into 1D sequence to be processed). In contrast, 2DMamba conducts 2D scans (without the need to flatten the input) and make novel modifications to the CUDA part for the best efficiency.

C. Mathematical derivations of 2DMamba

We formulate 2D scanning in a manner similar to Mamba [15] for efficient parallelism. To achieve the spatial continuity in Eq. (7), we first scan row-wise in Eq. (5) and get

$$h_{i,j}^{hor} = \sum_{j' \le j} \bar{A}^{(j-j')} \bar{B} x_{i,j'} \text{ (The vanilla Mamba)} \quad (11)$$

We then scan column-wise as Eq.(6):

$$\begin{aligned} h_{i,j} &= \bar{A} h_{i-1,j} + x_{i,j}^{hor} && (12)\\ &= \bar{A} h_{i-1,j} + \bar{B}' x_{i,j}^{hor} \; (\bar{B}' = I) && (13)\\ &= \sum_{i' \le i} \bar{A}^{(i-i')} \bar{B}' h_{i',j}^{hor} \text{ (The vanilla Mamba)} && (14)\\ &= \sum_{i' \le i} \bar{A}^{(i-i')} \sum_{j' \le j} \bar{A}^{(j-j')} \bar{B} x_{i',j'} \; (\bar{B}' = I) && (15)\\ &= \sum_{i' \le i} \sum_{j' \le j} \bar{A}^{(i-i'+j-j')} \bar{B} x_{i',j'} \text{ (Eq.(7))} && (16) \end{aligned}$$

D. Details of WSI datasets

Breast invasive carcinoma subtyping on BRACS and TCGA-BRCA. BRACS [3] includes 547 H&E breast carcinoma WSIs collected from 187 patients. There are 3 classes: *benign tumor* (265 slides), *atypical tumor* (89 slides), or *malignant tumor* (193 slides). We used the official train–validation–test split with a ratio of 395:65:87 slides. TCGA-BRCA comprise 1033 H&E WSIs with 2 subtypes: *invasive ductal carcinoma* (822 slides) and *invasive lobular carcinoma* (211 slides). We follow [5] to get the train–validation–test folds with the ratio of 841:96:96 slides.

Prostate cancer grading based on PANDA. The dataset [4] consists of 10,614 digitized prostate cancer biopsies. There are 6 categories: *grade 0* (2890 slides), *grade 1* (2666 slides),*grade 2* (1343 slides), *grade 3* (1242 slides), *grade 4* (1249 slides), or *grade 5* (1224 slides). We label-stratified PANDA into 80:10:10 train–validation–test sets (8491:1061:1062 slides).

Renal cell carcinoma subtyping based on DHMC. The dataset [50] include 563 H&E WSIs collected from 485 resections and 78 biopsies. The label includes 5 types: *clear cell renal cell carcinoma* (344 slides), *papillary renal cell carcinoma* (101 slides) and *chromophobe renal cell carcinoma* (23 slides), *renal oncocytoma* (66 slides) and *benign* (29 slides). We split the dataset into 393:23:147 slides for train, validation, and test sets.

Non-small cell lung carcinoma subtyping on TCGA-NSCLC. The dataset include 957 H&E breast carcinoma

	Feature size	14 × 14			56 × 56			200 × 200		
Scope	Method	FLOPs	Thro.	GPU Mem.	FLOPs	Thro.	GPU Mem.	FLOPs	Thro.	GPU Mem.
MIL framework	Mamba (CUDA)	63M	115	30 MB	1.0G	100	78 MB	12.9G	56	804 MB
	2DMamba (Python)	72M	53	70 MB	1.2G	40	670 MB	14.7G	5	9110 MB
	2DMamba (CUDA)	72M	110	30 MB	1.2G	88	102 MB	14.7G	49	912 MB

Table S1. Comparison of floating-point operations (FLOPs), throughput (Thro., feature maps per second), and GPU memory consumption during training. MIL frameworks are measured using 128 dimensional feature input and the state dimension is set to 16 for all experiments.

WSIs, including 2 subtypes: *lung adenocarcinoma* (490 slides) and *lung squamous cell carcinoma* (468 slides). We follow [5] to split the dataset into train–validation–test folds with the ratio of 785:86:87 slides.

Survival prediction on TCGA-KIRC, TCGA-KIRP, TCGA-LUAD, TCGA-STAD, and TCGA-UCEC. For TCGA-KIRC (*kidney renal clear cell carcinoma*), the dataset includes 498 slides, with 329 censored and 169 uncensored samples, of which 300 WSIs are used for training and 100 for validation. The TCGA-KIRP dataset (*kidney renal papillary cell carcinoma*) consists of 261 slides (220 censored, 41 uncensored), with 208 WSIs for training and 53 for validation. In TCGA-LUAD (*lung adenocarcinoma*), there are 455 slides (159 censored, 296 uncensored), split into 364 WSIs for training and 91 for validation. The TCGA-STAD dataset (*stomach adenocarcinoma*) includes 363 slides, of which 145 are censored and 218 uncensored, divided into 290 WSIs for training and 73 for validation. Finally, the TCGA-UCEC dataset (*uterine corpus endometrial carcinoma*) contains 539 slides (460 censored, 79 uncensored), with 431 WSIs allocated to training and 108 for validation. The 5-fold cross-validation splits for each dataset were derived from [46].

E. Evaluation of speed and GPU memory efficiency during training

In Tab. 3, we analyze speed and GPU memory efficiency during inference; here, we evaluate the same metrics during training, the floating-point operations (FLOPs), throughput, and GPU memory consumption. Since the CUDA operators are identical for both training and inference, we focus on comparing the CUDA implementation of Mamba, the Python implementation of 2DMamba, and the CUDA implementation of 2DMambawithin the MIL framework, across the three input feature sizes, 14×14, 56×56, and 200×200. As shown in Table S1, for all three sizes, with 10% to 20% increases of FLOPs, our method achieves throughput at approximately 90% of vanilla Mamba while significantly outperforming the Python implementation of 2DMamba. In terms of GPU memory consumption, our approach incurs only a slight increase compared to vanilla Mamba while reducing memory usage by 57% to 90% compared to the Python implementation of 2DMamba. These results demonstrate that our hardware-aware 2D selective scan operator remains both fast and GPU memory-efficient during training.

F. Additional results on natural image classification

In Tab. 4, we apply 2DMamba to the SOTA Mamba-based method on natural images VMamba [26] and name it as 2DVMamba. Our results showed that 2DVMamba-T outperforms all SOTA methods. In this section, we scale 2DVMamba to its small version: 2DVMamba-S. As shown in Table S2, similar to the improvements seen in the tiny version, 2DVMamba-S surpasses VMamba-S by 0.2% with a negligible increase in FLOPs (0.1G). It also outperforms all current SOTA methods, demonstrating that our approach scales effectively to larger models.

Method	#Param	FLOPs	Top-1 Acc%
DeiT-S	22M	4.6G	79.8
Swin-T	28M	4.5G	81.3
Vim-S	26M	-	80.3
EfficientVMamba-B	33M	4.0G	81.8
LocalVMamba-T	26M	5.7G	82.7
VMamba-T	30M	4.91G	82.6
2DVMamba-T	30M	4.94G	**82.8**
DeiT-B	86M	17.5G	81.8
Swin-S	50M	8.7G	83.0
LocalVMamba-S	50M	11.4G	83.7
VMamba-S	50M	8.7G	83.6
2DVMamba-S	50M	8.8G	**83.8**

Table S2. The top-1 accuracy (%) of our 2DVMamba on the ImageNet-1K dataset. All images are of size 224×224.

G. Additional ablation studies

Parameter $\bar{A}$ reuse. In 2DMamba, $\bar{A}$ is reused for both horizontal and vertical scan to maintain the same number of parameters as the vanilla Mamba [15]. Although we lack theoretical guarantees of optimality, our ablation studies in Tab. S3 demonstrate that 2DMambaMIL with inde-

Setting	BRACS			DHMC			PANDA			TCGA-NSCLC			TCGA-BRCA		
	Acc	$F1$	AUC	Acc	$F1$	AUC	Acc	$F1$	AUC	Acc	$F1$	AUC	Acc	$F1$	AUC
Independent $\bar{A}$	0.7379	0.6786	0.8857	**0.8935**	**0.8122**	0.9410	0.4917	0.4466	0.8131	**0.8851**	**0.8851**	0.9577	0.9333	0.9124	0.9759
Reused $\bar{A}$	**0.7517**	**0.7045**	**0.8964**	0.8926	0.8027	**0.9468**	**0.5075**	**0.4562**	**0.8184**	**0.8851**	0.8850	**0.9618**	**0.9458**	**0.9156**	**0.9782**

Table S3. Comparison of using independent and reused parameter $\bar{A}$ in the 2DMambaMIL. They achieve comparable performance.

pendent $\bar{A}$ achieves comparable performance (within $\pm 1\%$) to 2DMambaMIL with reused $\bar{A}$ across five datasets.

Positional embeddings (PE). We investigate the impact of PE in Mamba-based MIL. We compare MambaMIL, SRMambaMIL, and 2DMambaMILwith and without PE on the PANDA and TCGA-BRCA datasets. Due to the large size of WSIs, absolute PE, as in [12], results in an excessive number of parameters for MIL models. Instead, we adopt a linear projection to map the 2D coordinates of each patch into a PE and added to the patch embeddings to integrate positional information. As shown in Table S4, incorporating PE generally improves the performance of 1D Mamba-based methods, indicating the additional spatial information helps mitigate spatial discrepancies. In contrast, adding PE to our 2DMambaMIL reduces its performance. This decline occurs because our 2D formulation effectively integrates spatial information, making the additional PE redundant.

Method	PANDA		TCGA-BRCA	
	Acc	AUC	Acc	AUC
MambaMIL	0.4679	0.7781	0.9333	0.9657
MambaMIL-PE	**0.4887**	**0.7976**	**0.9292**	**0.9705**
SRMambaMIL	0.4711	**0.7776**	0.9313	0.9657
SRMambaMIL-PE	**0.4774**	0.7705	**0.9333**	**0.9703**
2D-MambaMIL	**0.5075**	**0.8184**	**0.9458**	**0.9782**
2D-MambaMIL-PE	0.4971	0.8083	0.9290	0.9765

Table S4. The native 2D formulation of 2D-MambaMIL obtains higher performance than integrating the positional embedding (PE) into 1D-based models. Moreover, adding PE into 2D-MambaMIL consistently decreases the performance.

Comparison with a naive 2D method. A naive 2D approach involves applying 1D Mamba independently to all rows and then to all columns. We compare the performance of our formulation with this naive 2D method. As shown in Table S5, the naive approach is 1%-2% less accurate in Accuracy and AUC on the PANDA and the TCGA-BRCA datasets. Additionally, the naive method is 50% more computationally expensive due to the padding of 14×14 tiles to 14×32 or 32×14, as discussed in Section 3.4.

2D scan	PANDA		TCGA-BRCA	
	Acc	AUC	Acc	AUC
Naive	0.4856	0.8077	0.9333	0.9760
Ours	**0.5075**	**0.8184**	**0.9458**	**0.9782**

Table S5. The comparison of naive 2D approach and our 2DMambaMIL. The naive approach applies 1D Mamba independently to all row and then all columns. Our 2DMambaMIL surpasses the naive 2D approach.

2D scan order. We ablate the 2D scan order of our 2DMamba by comparing two different orders: Horizontal-Vertical and Vertical-Horizontal. The results in Table S6 show that the two scan orders of 2DMamba achieve comparable performance on the PANDA and the TCGA-BRCA datasets, with the average differences of 0.5% in accuracy and 0.6% in AUC. This ablation shows that the scan order does not have a large influence on the performance of 2DMamba.

2D scan order	PANDA		TCGA-BRCA	
	Acc	AUC	Acc	AUC
Horizontal-Vertical	**0.5075**	**0.8184**	**0.9458**	**0.9782**
Vertical-Horizontal	0.5001	0.8141	0.9427	0.9707

Table S6. Ablation on the 2D scan order on the PANDA and TCGA-BRCA dataset. The scan order does not have a large influence on the performance.

Number of blocks. We ablate using one, two and three 2DMamba blocks. Table S7 shows that employing one 2DMamba block achieves the overall best performance. A single layer yields the highest performance in both accuracy and AUC on the TCGA-BRCA dataset and yields the best AUC with a slightly lower accuracy on the PANDA dataset.

Number of blocks U	PANDA		TCGA-BRCA	
	Acc	AUC	Acc	AUC
1	0.5075	**0.8184**	**0.9458**	**0.9782**
2	**0.5134**	0.8153	0.9427	0.9778
3	0.5045	0.8178	0.9340	0.9558

Table S7. Ablation study on the number of 2DMamba blocks U on the PANDA and TCGA-BRCA dataset.

Model dimension. We ablate the model dimensions of 2DMamba. Table S8 depicts that, a dimension of 128 generally provides the best performance on both the PANDA and TCGA-BRCA datasets. Specifically, for the PANDA dataset, increasing the dimension to 256 or 512 slightly improves accuracy but results in a significant drop in AUC. For the TCGA-BRCA dataset, the 128-dimension model achieves the highest performance, with improvements of at least 0.9% in accuracy and 0.7% in AUC.

Model dimension	PANDA		TCGA-BRCA	
	Acc	AUC	Acc	AUC
32	0.4916	0.8073	0.9250	0.9745
64	0.4987	0.8066	0.9375	0.9713
128	0.5075	**0.8184**	**0.9458**	**0.9782**
256	0.5132	0.8072	0.9292	0.9671
512	**0.5194**	0.8067	0.9271	0.9678

Table S8. Ablation study on the model dimensions on the PANDA and TCGA-BRCA dataset.

State dimension. We ablate the state dimension N of 2DMamba. Table S9 shows that using $N = 16$ provides overall the highest performance in the PANDA and TCGA-BRCA dataset. Particularly, setting $N = 16$ obtains the highest AUC and the highest accuracy on TCGA-BRCA datase. On the PANDA dataset, $N = 16$ obtains the highest AUC and a slightly lower accuracy, compared with $N = 32$. Thus, we set $N = 16$ for all our experiments.

State dimension N	PANDA		TCGA-BRCA	
	Acc	AUC	Acc	AUC
4	0.4999	0.8105	0.9271	0.9712
8	0.4970	0.8114	0.9354	0.9754
16	0.5075	**0.8184**	**0.9458**	**0.9782**
32	**0.5121**	0.8174	0.9271	0.9705
64	0.5040	0.8096	0.9375	0.9773

Table S9. Ablation study on the state dimension N on the PANDA and TCGA-BRCA dataset.

H. Standard derivation

The metrics reported in Tab. 1 represent the means of five runs conducted with different random seeds. The standard derivations of these metrics are provided in Tab. S10. The standard deviations of 2DMamba are generally comparable to or smaller than those of other methods, demonstrating the stability of the proposed approach.

I. Additional qualitative evaluation

In addition to the heatmaps of the TCGA-KIRC sample shown in Fig. 4, we also analyze other samples for comprehensive qualitative evaluation. As shown in Fig. S1 to Fig. S6, overall, these generated heatmaps show that 2DMambaMIL consistently generates heatmaps that are more logical than the other models. 2DMambaMIL highlights tumor features based on the task while others seem to use features from both tumor and non-tumor, showing that the model is non-specific or is tagging onto features that are not truly biologically relevant. For classification purposes, 2DMambaMIL heatmaps consistently highlight tumor-area pixels for classification. MambaMIL and CLAM are slightly less specific, with pixels from non-tumor areas being more often used by the model. These three models generate heatmaps that are more tumor-specific than AB-MIL and SRMambaMIL, which also highlight non-tumor features during the tumor classification task. For survival prediction purposes, 2DMambaMIL also consistently used tumor areas for survival prediction while also using some pixels from the immediate tumor-adjacent areas. Interestingly, the signal detection with 2DMambaMIL was heterogeneous within the tumor, with highly- and low-attended areas of the tumor being highlighted, a feature less present with MambaMIL and CLAM, and not achieved by AB-MIL and SRMambaMIL. In addition, we also analyze the attention heatmap of 2DMambaMIL in high resolution patches. As shown in Fig. S7, within tumor regions, our model distinguishes fine-grained regions of high and low mortality that correspond to high-grade and low-grade tumors.

Method	**BRACS**			**DHMC**			**PANDA**			**TCGA-NSCLC**			**TCGA-BRCA**		
	Acc	*F1*	*AUC*	*Acc*	*F1*	*AUC*	*Acc*	*F1*	*AUC*	*Acc*	*F1*	*AUC*	*Acc*	*F1*	*AUC*
AB-MIL	0.0197	0.0178	0.0199	0.0231	**0.0219**	0.0273	0.0116	0.0139	0.0157	0.0104	0.0103	0.0208	0.0123	0.0145	0.0210
DSMIL	0.0185	**0.0165**	0.0210	0.0248	0.0220	0.0255	0.0118	0.0136	0.0161	0.0111	0.0111	0.0211	0.0132	0.0134	0.0201
CLAM	0.0229	0.0189	0.0221	0.0241	0.0258	0.0295	0.0141	0.0166	0.0184	0.0130	0.0129	0.0249	0.0131	0.0160	0.0268
DTFD-MIL	0.0254	0.0225	0.0255	0.0294	0.0274	0.0345	0.0148	0.0179	0.0197	0.0133	0.0132	0.0269	0.0158	0.0187	0.0267
TransMIL	0.0262	0.0236	0.0263	0.0310	0.0287	0.0367	0.0155	0.0181	0.0208	0.0138	0.0139	0.0274	0.0162	0.0190	0.0281
S4-MIL	0.0214	0.0209	0.0194	0.0223	0.0240	0.0246	0.0131	0.0139	0.0172	0.0116	0.0116	**0.0189**	0.0145	0.0134	0.0216
MambaMIL	0.0179	0.0193	0.0217	0.0229	0.0261	0.0328	0.0133	0.0144	**0.0146**	0.0104	**0.0103**	0.0249	0.0140	**0.0132**	0.0199
SRMambaMIL	0.0189	0.0168	0.0203	0.0232	0.0258	0.0232	**0.0107**	0.0147	0.0147	0.0117	0.0115	0.0192	0.0121	0.0139	0.0179
2DMambaMIL	**0.0175**	0.0184	**0.0173**	**0.0209**	0.0225	**0.0227**	**0.0107**	**0.0132**	0.0162	**0.0102**	0.0124	0.0205	**0.0113**	0.0139	**0.0175**

Table S10. The standard deviation of accuracy (Acc), F1 and AUC on five WSI classification datasets. We conducted each experiment five times using five different random seeds and reported their standard deviations. The lowest values are marked as **bold**.

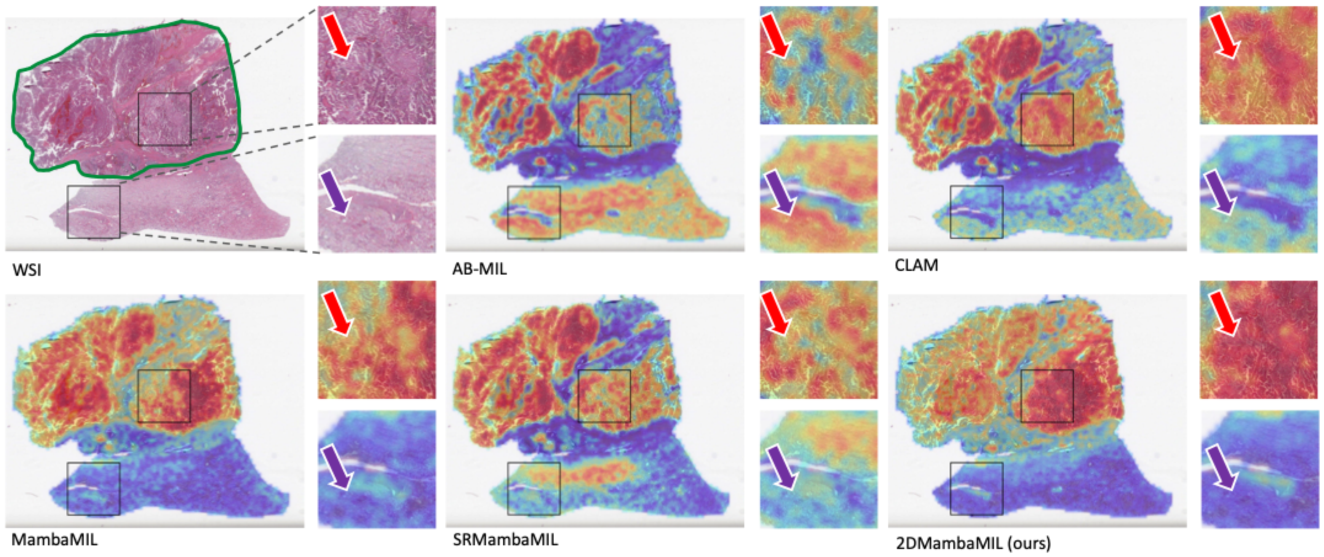

Figure S1. The attention visualization of 2DMambaMIL and four other methods on a TCGA-KIRP sample for **survival** analysis. Tumor regions are outlined in green. Red arrows and violet arrows point to the highly survival relevant areas and non-tumor areas, respectively. 2DMambaMIL and MambaMIL use tumoral and peritumoral pixels to drive the model, consistent with the heterogeneous feature of the distribution of high mortality predicting areas. By contrast, AB-MIL, CLAM, and SRMambaMIL are less specific, with high probability areas being located randomly or in insignificant structures in the non-tumoral tissue. 2DMambaMIL slightly outperforms MambaMIL in the heatmap distribution.

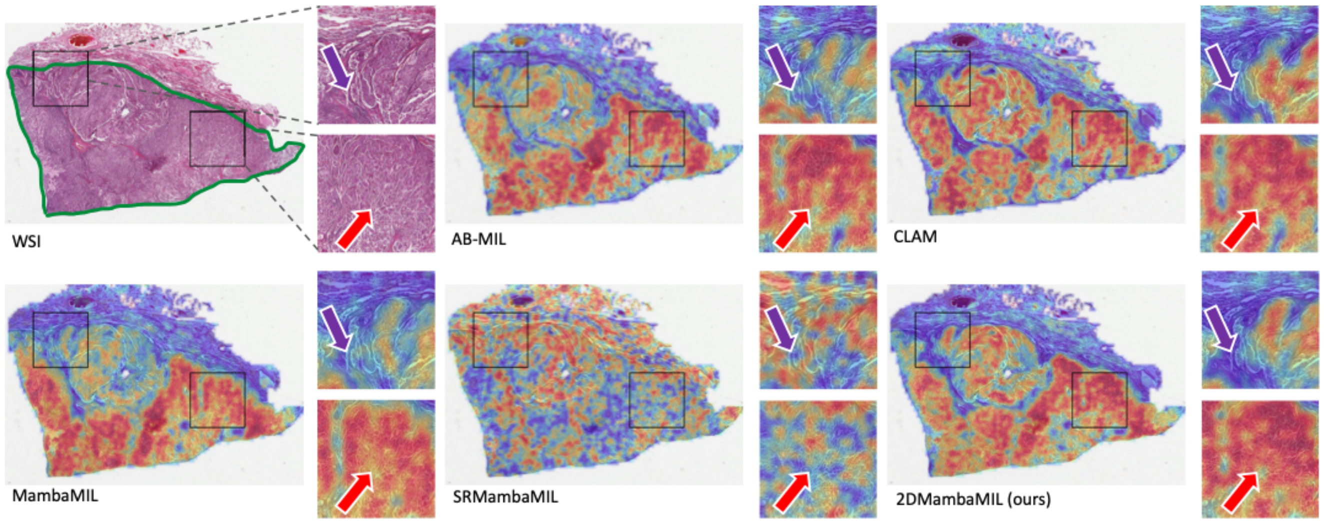

Figure S2. The attention visualization of 2DMambaMIL and four other methods on a TCGA-LUAD sample for **survival** analysis. Tumor regions are outlined in green. Red arrows and violet arrows point to the highly survival relevant areas and survival irrelevant areas, respectively. 2DMambaMIL and MambaMIL use tumoral and peritumoral pixels to drive the model, consistent with the heterogeneous feature of the distribution of high mortality predicting areas. By contrast, AB-MIL, CLAM, and SRMambaMIL are less specific, with high probability areas being located randomly or in insignificant structures in the non-tumoral tissue.

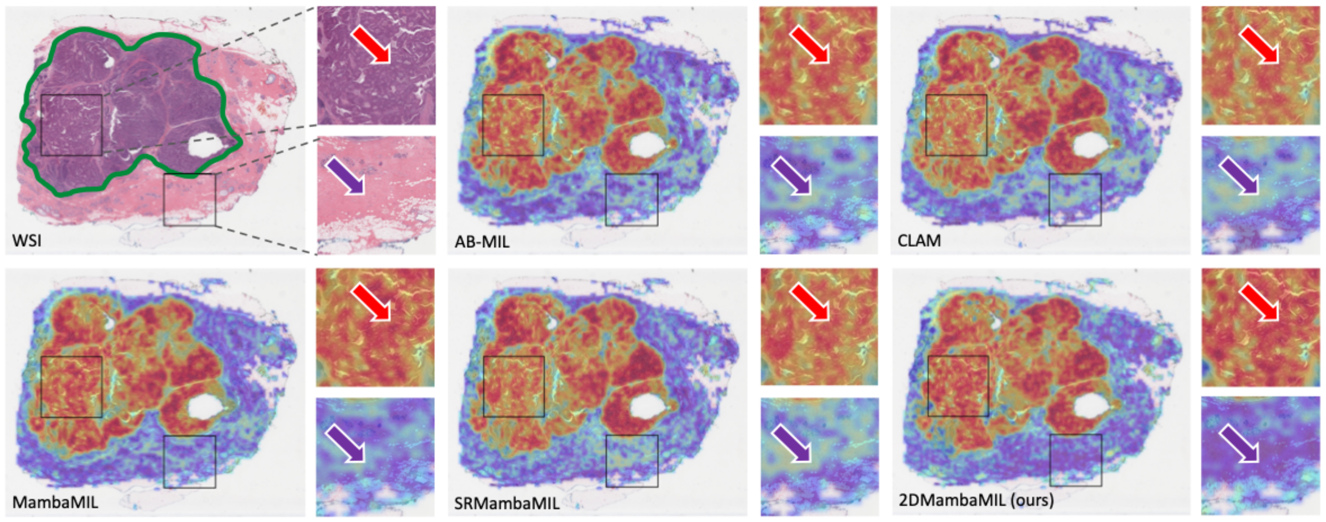

Figure S3. The attention visualization of 2DMambaMIL and four other methods on an IDC sample for TCGA-BRCA **sub-typing**. Tumor regions are outlined in green. Red arrows and violet arrows point to the highly task-relevant areas and non-tumor areas, respectively. 2DMambaMIL and MambaMIL outperform the other models in qualitative specificity, as the background non-tumor tissue contains less high-probability pixels. 2DMambaMIL slightly outperforms MambaMIL in the heatmap distribution.

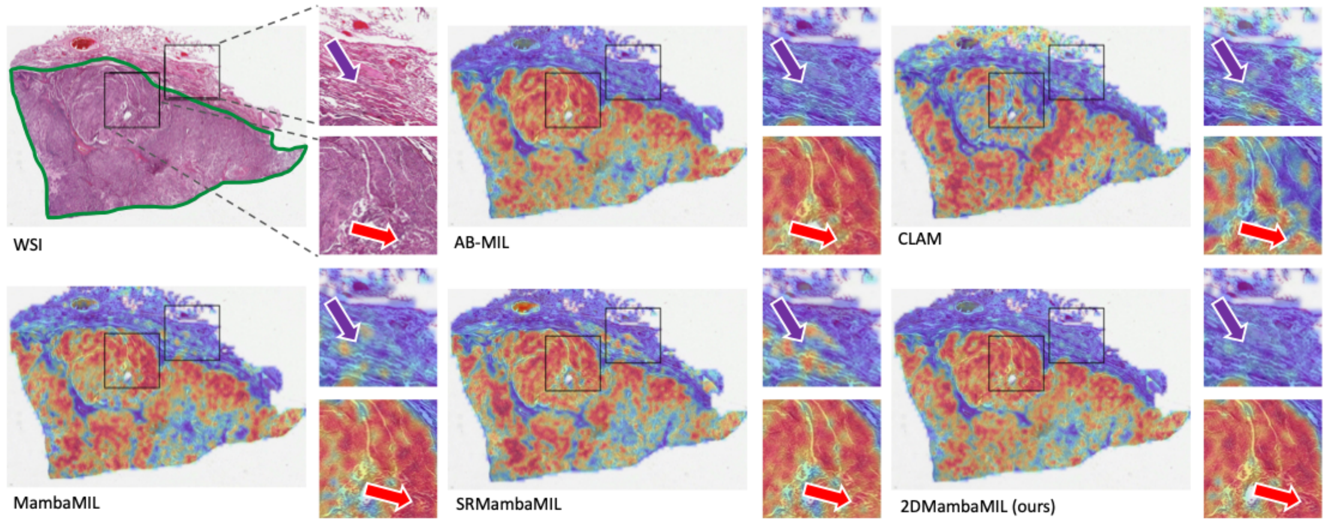

Figure S4. The attention visualization of 2DMambaMIL and four other methods on a LUAD sample for TCGA-NSCLC **sub-typing**. Tumor regions are outlined in green. Red arrows and violet arrows point to the highly task-relevant areas and non-tumor areas, respectively. 2DMambaMIL and MambaMIL outperform the other models in qualitative specificity, as the background non-tumor tissue contains fewer high-probability pixels. 2DMambaMIL slightly outperforms MambaMIL in the heatmap distribution.

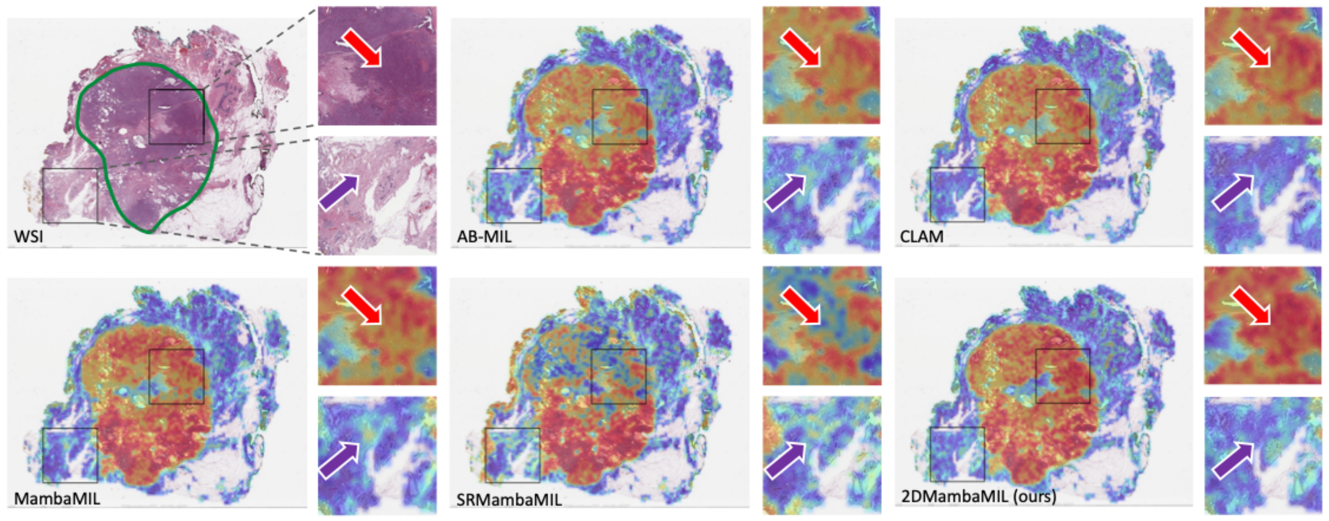

Figure S5. The attention visualization of 2DMambaMIL and four other methods on an ILC sample for TCGA-BRCA **sub-typing**. Tumor regions are outlined in green. Red arrows and violet arrows point to the highly task-relevant areas and non-tumor areas, respectively. 2DMambaMIL and MambaMIL outperform the other models in qualitative specificity, as the background non-tumor tissue contains fewer high-probability pixels.

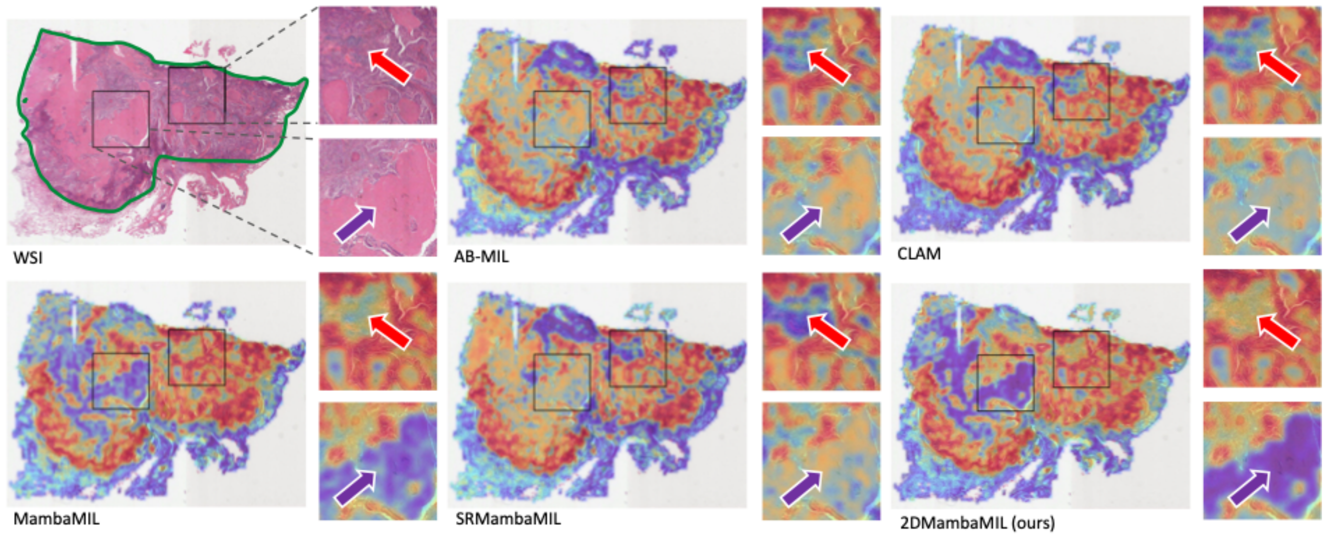

Figure S6. The attention visualization of 2DMambaMIL and four other methods on a LUSC sample for TCGA-NSCLC **sub-typing**. Tumor regions are outlined in green. Red arrows and violet arrows point to the highly task-relevant areas and less task-relevant areas, respectively. 2DMambaMIL and MambaMIL outperform the other models in qualitative specificity, as the background non-tumor tissue contains fewer high-probability pixels.

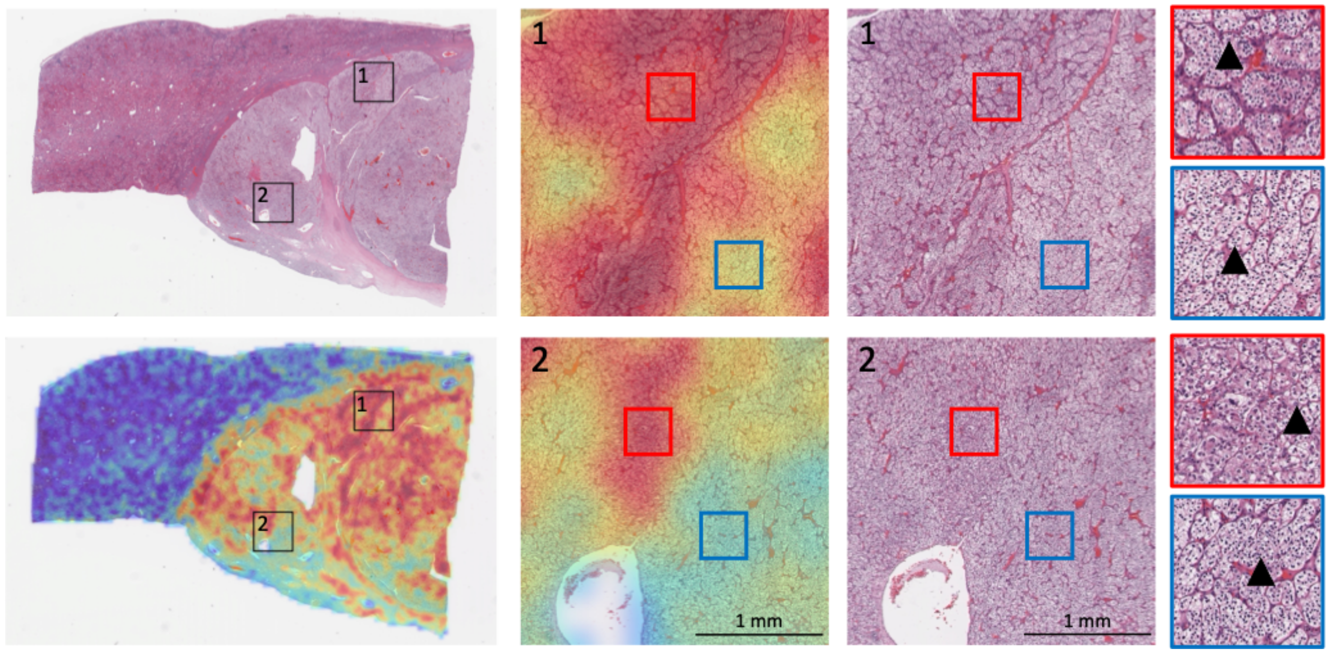

Figure S7. Two critical patches (1 and 2) of a kidney cell clear cell carcinoma sample from the TCGA-KIRC overlaid with attention heatmaps of 2DMambaMIL. The heatmaps of 2DMambaMIL heterogeneously show areas driving higher mortality and areas driving lower mortality. Specifically, 2DMambaMIL focuses more on the red squares that are directly related to mortality in survival analysis and focuses less on the blue squares that are less related to mortality. Areas in the red squares show features of high-grade disease (grade 2-3 pointed by black arrowheads), notably areas of tumor cells with inconspicuous nucleoli. Areas in the blue squares show low-grade cytological features (grade 1 pointed by black arrowheads).