西安交通大學
XI'AN JIAOTONG UNIVERSITY

目录

5.1 PPMA: Motivation (Current Issue)

Question: Have SOTA Mamba-based models outperform SOTA ViTs in computer vision domain, especially on high-level vision tasks?

Not yet, experimental results speak louder than words

Table 5: Image classification performance on the ImageNet-1K.

Model	Arch.	#Param. (M)	FLOPs (G)	Top-1 (%)
ConvNeXt-T [32]	CNN	29	4.5	82.1
EffNet-B4 [41]	CNN	19	4.2	82.9
VMamba-T [30]	SSM	30	4.9	82.6
GrootVL-T [48]	SSM	30	4.8	83.4
Spatial-Mamba-T [46]	SSM	27	4.5	83.5
MLLA-T [15]	SSM	25	4.2	83.5
Swin-T [31]	Trans.	29	4.5	82.1
NAT-T [16]	Trans.	28	4.3	83.2
BiFormer-S [56]	Trans.	26	4.5	83.8
RMT-S [10]	Trans.	27	4.5	84.0
ConvNeXt-S [32]	CNN	50	8.7	83.1
EffNet-B5 [41]	CNN	30	9.9	83.6
VMamba-S [30]	SSM	50	8.7	83.6
GrootVL-S [48]	SSM	51	8.5	84.2
MLLA-S [15]	SSM	43	7.3	84.4
Spatial-Mamba-S [46]	SSM	43	7.1	84.6
Swin-S [31]	Trans.	50	8.7	83.0
NAT-S [16]	Trans.	51	7.8	83.7
BiFormer-B [56]	Trans.	57	9.8	84.3
iFormer-B [37]	Trans.	48	9.4	84.6
RMT-B [10]	Trans.	54	9.7	84.9

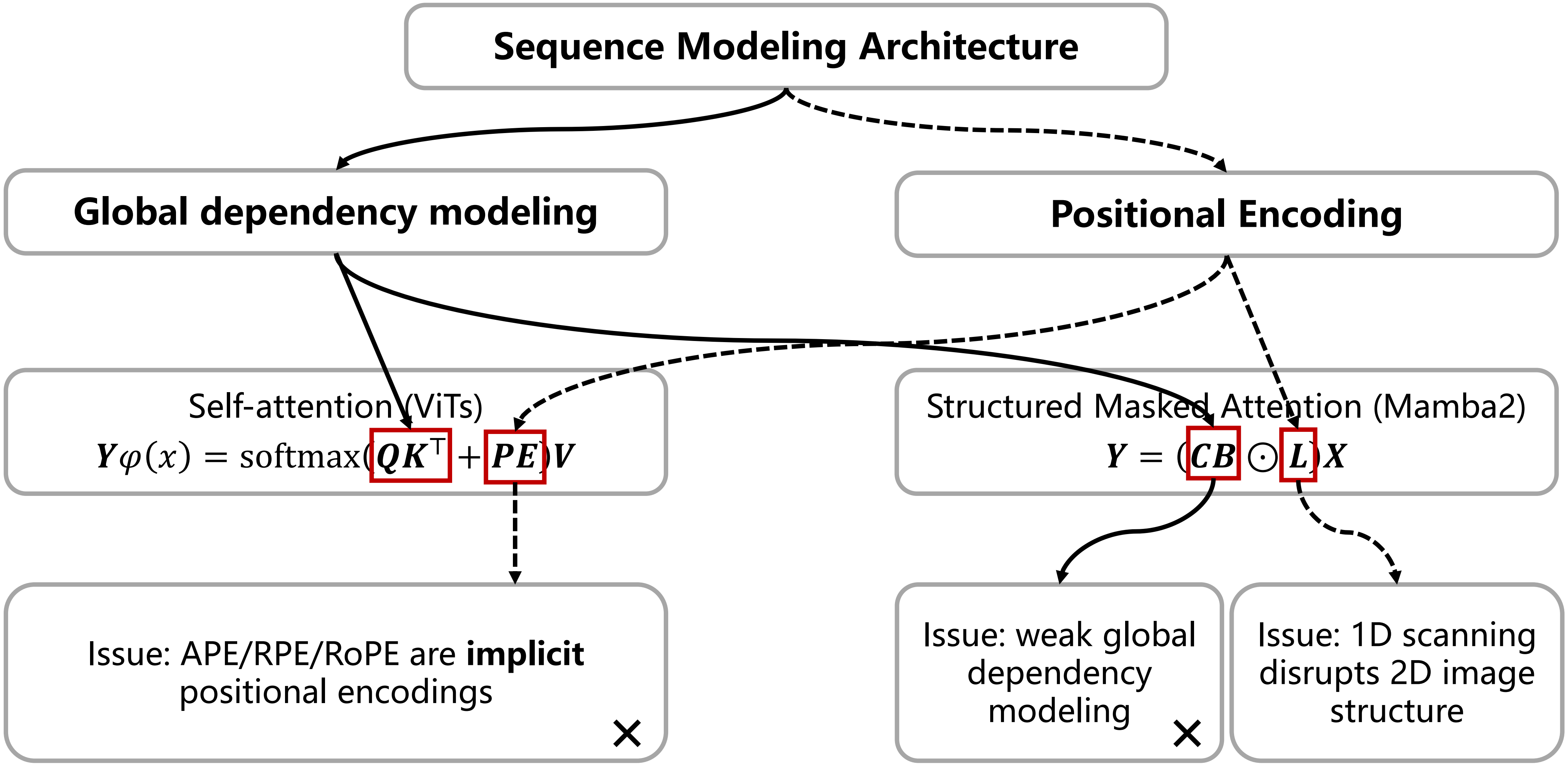
Table 6: Object detection and instance segmentation performance on COCO.

Backbone	Arch.	#Param. (M)	FLOPs (G)	AP^b	AP^m
ResNet-50 [19]	CNN	44	260	38.2	34.7
ConvNeXt-T [32]	CNN	48	262	44.2	40.1
MLLA-T [15]	SSM	44	255	46.8	42.1
GrootVL-T [48]	SSM	49	265	47.0	42.7
VMamba-T [30]	SSM	50	271	47.3	42.7
Spatial-Mamba-T [46]	SSM	46	216	47.6	42.9
Swin-T [31]	Trans.	48	267	43.7	39.8
CSWin-T [8]	Trans.	42	279	46.7	42.2
BiFormer-S [56]	Trans.	–	–	47.8	43.2
RMT-S [10]	Trans.	46	262	48.8	43.6
ResNet-101 [19]	CNN	63	336	40.4	36.4
ConvNeXt-S [32]	CNN	70	348	45.4	41.8
GrootVL-S [48]	SSM	70	341	48.6	43.6
VMamba-S [30]	SSM	70	349	48.7	43.7
Spatial-Mamba-S [46]	SSM	63	315	49.2	44.0
MLLA-S [15]	SSM	63	319	49.2	44.2
Swin-S [31]	Trans.	69	359	45.7	41.1
CSWin-S [8]	Trans.	54	342	47.9	43.2
BiFormer-B [56]	Trans.	–	–	48.6	43.7
RMT-B [10]	Trans.	73	373	50.7	45.1

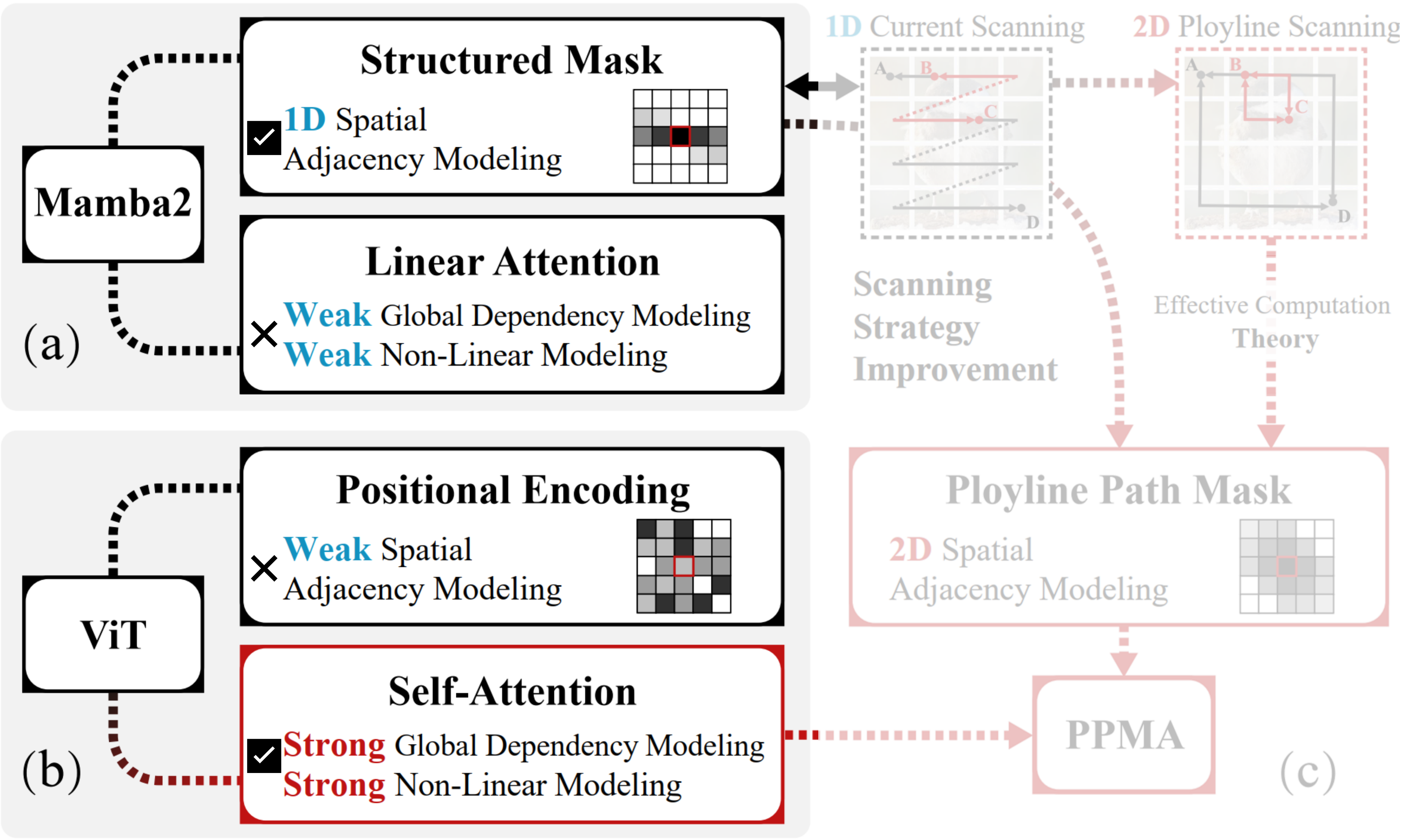
Table 7: Semantic segmentation performance on ADE20K.

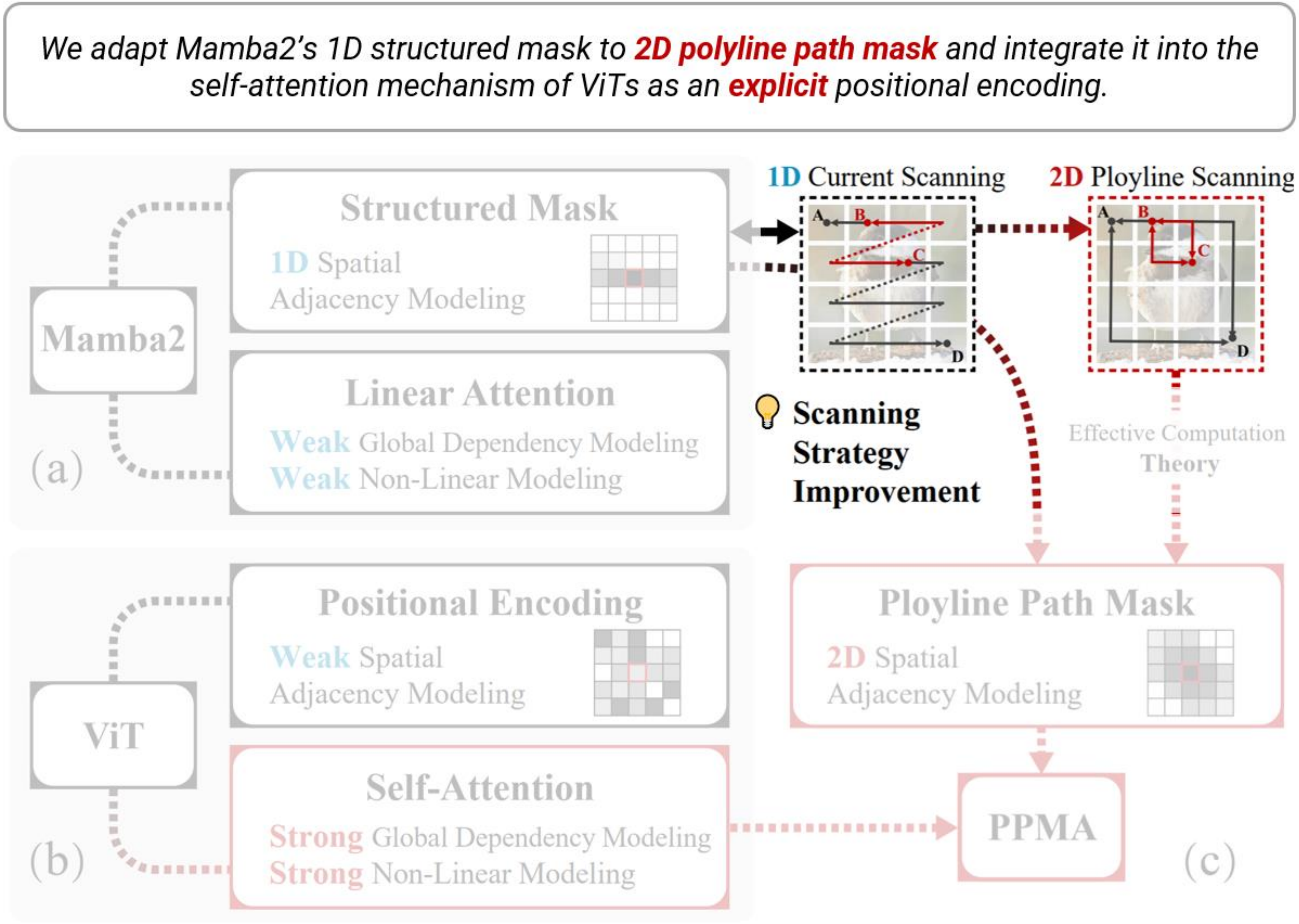
Backbone	Arch.	#Param. (M)	FLOPs (G)	mIoU(%) SS	mIoU(%) MS
ResNet-50 [19]	CNN	67	953	42.1	42.8
ConvNeXt-T [32]	CNN	60	939	46.0	46.7
VMamba-T [30]	SSM	62	949	48.0	48.8
2DMamba-T [52]	SSM	62	950	48.6	49.3
GrootVL-T [48]	SSM	60	941	48.5	49.4
Spatial-Mamba-S [46]	SSM	57	936	48.6	49.4
Swin-T [31]	Trans.	60	945	44.4	45.8
NAT-T [16]	Trans.	58	934	47.1	48.4
BiFormer-S [56]	Trans.	–	–	49.8	50.8
RMT-S [10]	Trans.	56	937	49.8	49.7
ResNet-101 [19]	CNN	85	1030	42.9	44.0
ConvNeXt-S [32]	CNN	82	1027	48.7	49.6
VMamba-S [30]	SSM	82	1028	50.6	51.2
Spatial-Mamba-S [46]	SSM	73	992	50.6	51.4
GrootVL-S [48]	SSM	82	1019	50.7	51.7
Swin-S [31]	Trans.	81	1039	47.6	49.5
NAT-S [16]	Trans.	82	1010	48.0	49.5
BiFormer-B [56]	Trans.	–	–	51.0	51.7
RMT-B [10]	Trans.	83	1051	52.0	52.1

5.1 PPMA: Motivation (Current Issue)

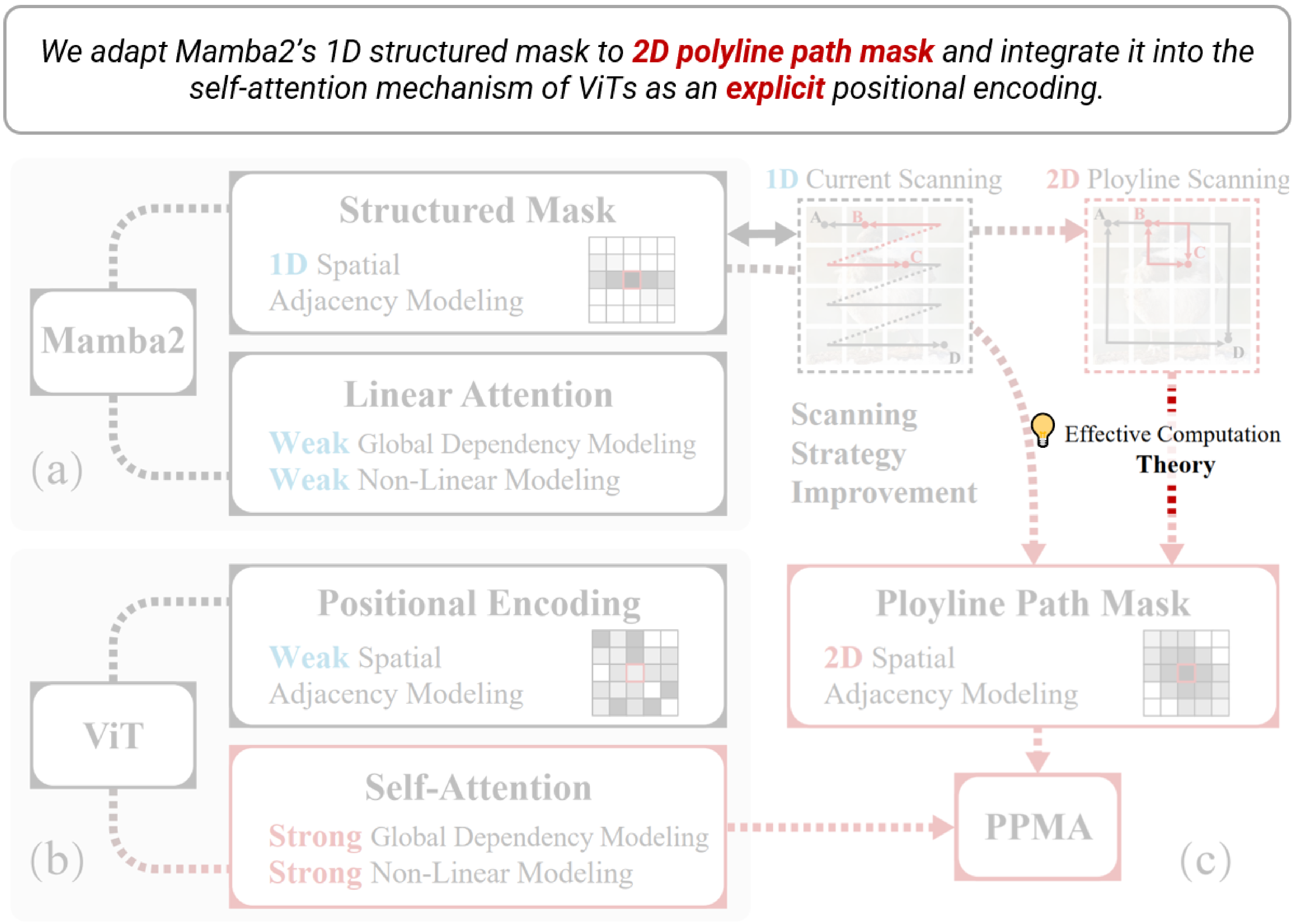


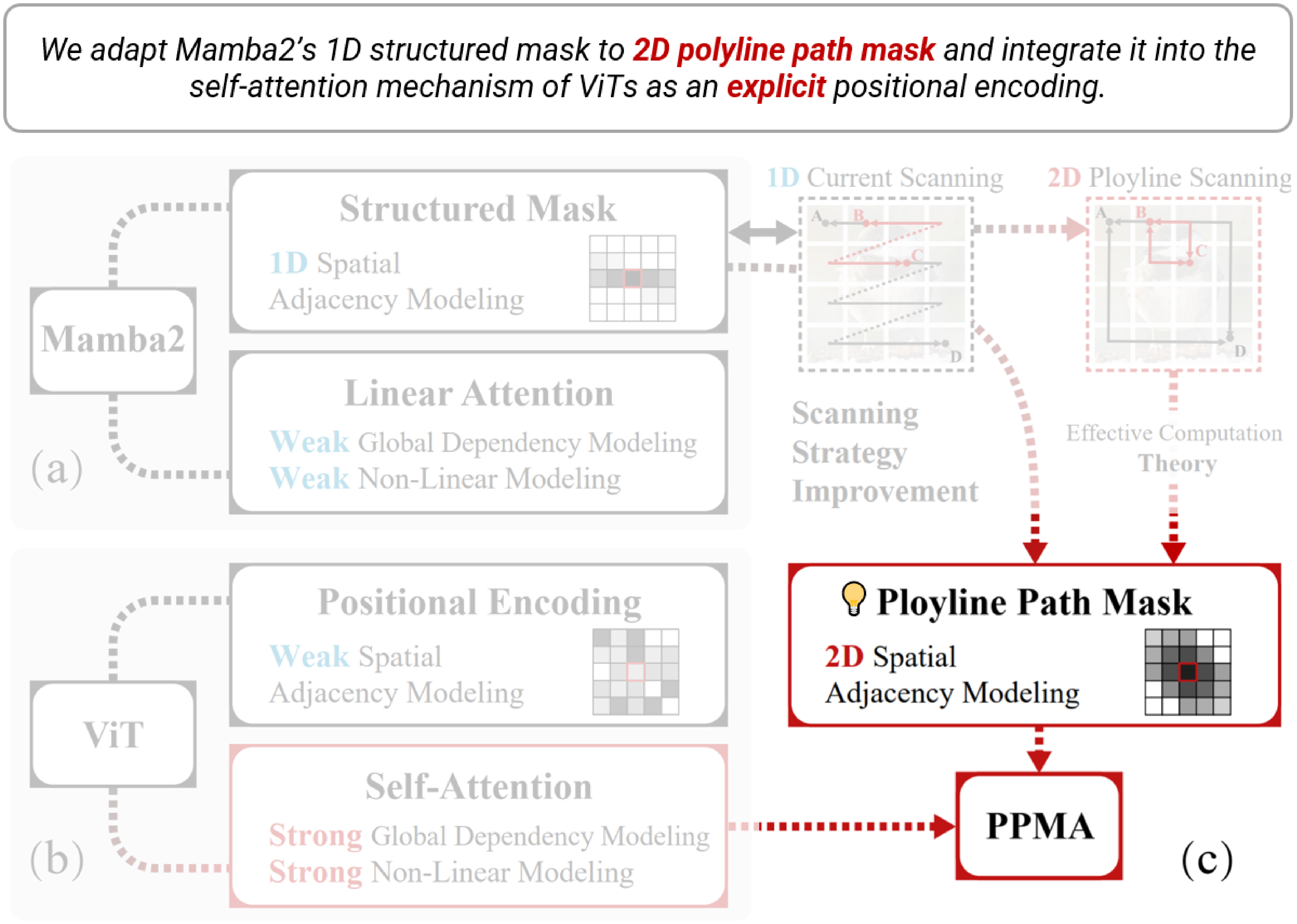
We adapt Mamba2's 1D structured mask to ***2D polyline path mask*** *and integrate it into the self-attention mechanism of ViTs as an* ***explicit*** *positional encoding.*





We adapt Mamba2's 1D structured mask to 2D polyline path mask and integrate it into the self-attention mechanism of ViTs as an explicit positional encoding.
Structured Mask
1D Spatial Adjacency Modeling
Mamba2
Linear Attention
Weak Global Dependency Modeling
Weak Non-Linear Modeling
(a)
1D Current Scanning
2D Ployline Scanning
A
B
C
D
Scanning Strategy Improvement
Effective Computation Theory
Positional Encoding
Weak Spatial Adjacency Modeling
ViT
Self-Attention
Strong Global Dependency Modeling
Strong Non-Linear Modeling
(b)
Ployline Path Mask
2D Spatial Adjacency Modeling
PPMA
(c)

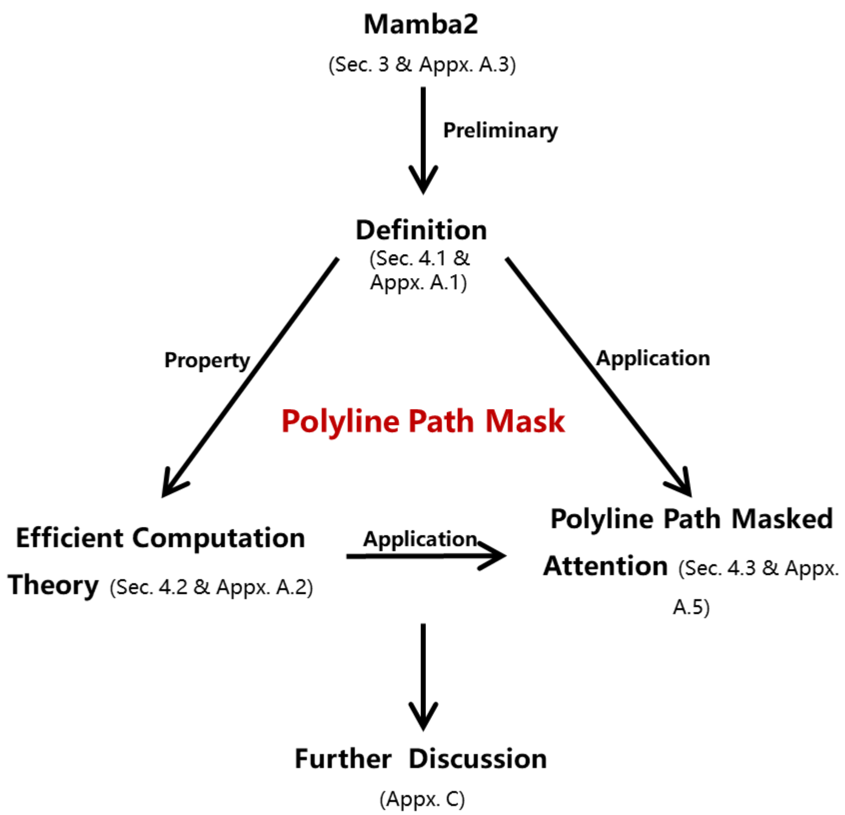

We adapt Mamba2's 1D structured mask to 2D polyline path mask and integrate it into the self-attention mechanism of ViTs as an explicit positional encoding.
Mamba2
Structured Mask
1D Spatial Adjacency Modeling
Linear Attention
Weak Global Dependency Modeling
Weak Non-Linear Modeling
(a)
1D Current Scanning
2D Ployline Scanning
Scanning Strategy Improvement
Effective Computation Theory
ViT
Positional Encoding
Weak Spatial Adjacency Modeling
Self-Attention
Strong Global Dependency Modeling
Strong Non-Linear Modeling
(b)
Ployline Path Mask
2D Spatial Adjacency Modeling
PPMA
(c)


We adapt Mamba2's 1D structured mask to 2D polyline path mask and integrate it into the self-attention mechanism of ViTs as an explicit positional encoding.
Mamba2
Structured Mask
1D Spatial Adjacency Modeling
Linear Attention
Weak Global Dependency Modeling
Weak Non-Linear Modeling
(a)
ViT
Positional Encoding
Weak Spatial Adjacency Modeling
Self-Attention
Strong Global Dependency Modeling
Strong Non-Linear Modeling
(b)
1D Current Scanning
2D Ployline Scanning
Scanning Strategy Improvement
Effective Computation Theory
Ployline Path Mask
2D Spatial Adjacency Modeling
PPMA
(c)

5.1 PPMA: Insight & Contribution

- Our insight: Mamba2 core mechanism is **structured mask**
  - **Explicit positional encoding** through the recursive propagation mechanism
  - **Semantic continuity awareness** in sequences through the **selective** mechanism

- Our contribution
  - A novel **2D polyline path structured mask**
  - An **efficient algorithm** for the calculation of the polyline path mask
  - **Polyline Path Masked (Sparse) Attention**
  - **SOTA performance** on image classification, object detection, and segmentation tasks compared to SSM-based models and ViTs



5.2 PPMA: Method (Definition of Polyline Path Mask)

- 2D polyline path scanning
  - Current issue: fail to preserve the distance relationship of 2D tokens
  - Our solution: scan the 2D tokens along multiple paths

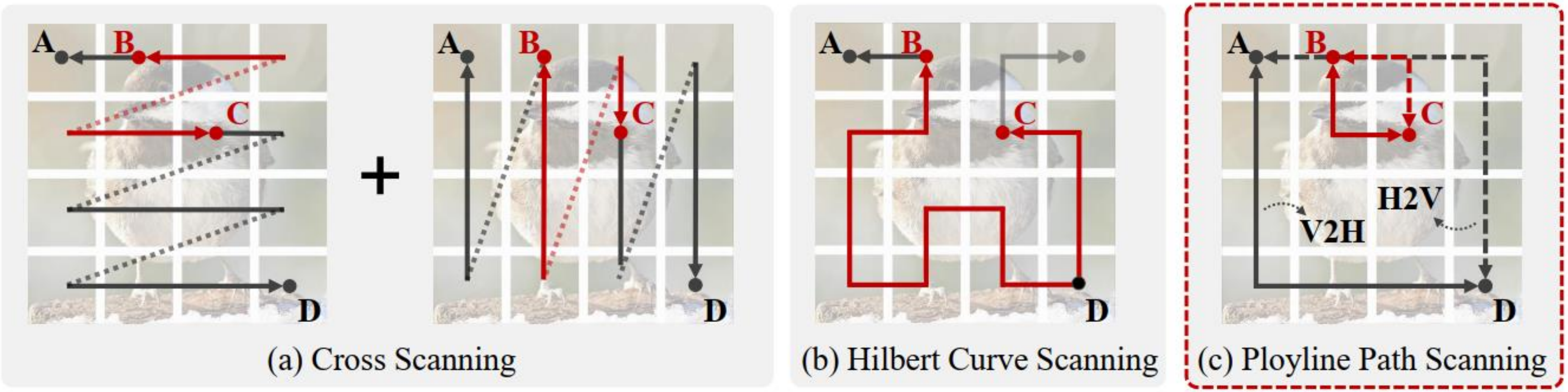


Figure 2: Compared to existing scanning strategies (a) and (b), which flatten 2D tokens into a 1D sequence, our polyline path scanning (c) better preserves the adjacency of 2D tokens.

5.2 PPMA: Method (Definition of Polyline Path Mask)

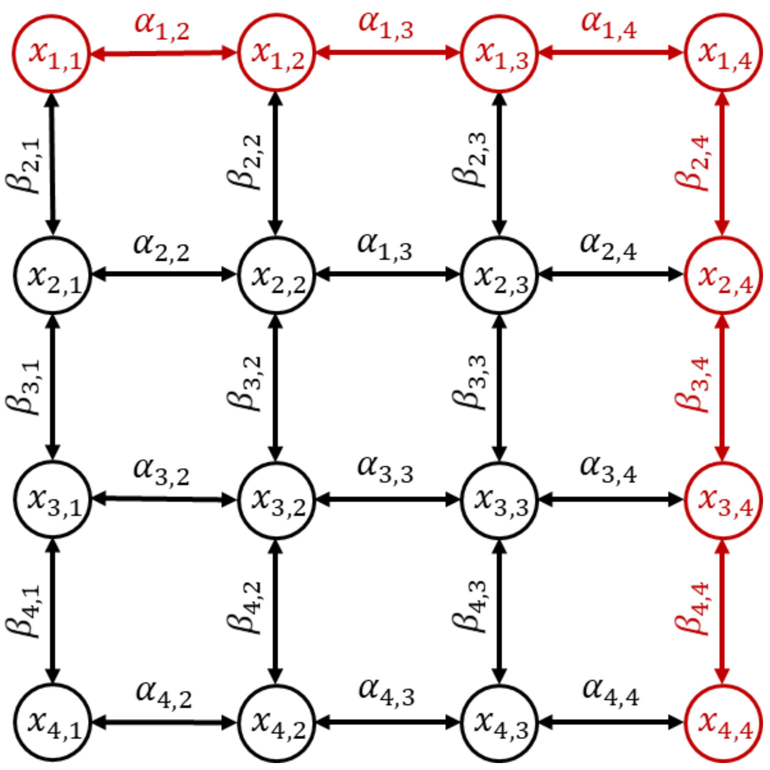
- 2D polyline path mask
  - Learn the **horizontal** factor and **vertical** decay factors

$$\begin{cases} \alpha_{i,j} = \exp(-\mathrm{Softplus}(\mathrm{MLP}_{\alpha}(\boldsymbol{x}_{i,j}))) \in \mathbb{R}^1\ (\mathbf{0}\sim\mathbf{1}) \\ \beta_{i,j} = \exp(-\mathrm{Softplus}(\mathrm{MLP}_{\alpha}(\boldsymbol{x}_{i,j}))) \in \mathbb{R}^1\ (\mathbf{0}\sim\mathbf{1}) \end{cases}$$

  - Calculate the decay weight of each polyline path, i.e., for path from $\boldsymbol{x}_{k,l}$ to $\boldsymbol{x}_{i,j}$:

$$\boldsymbol{\mathcal{L}}_{i,j,k,l} = \alpha_{i,j:l}\beta_{i:k,l}$$

$$\alpha_{i,j:l} = \begin{cases} \alpha_{i,j+1} \times \cdots \times \alpha_{i,l} & if\ j < l \\ 1 & if\ j = l \\ \alpha_{i,l+1} \times \cdots \times \alpha_{i,j} & if\ j > l \end{cases}, \quad \beta_{i:k,l} = \begin{cases} \beta_{i+1,l} \times \cdots \times \beta_{k,l} & if\ i < k \\ 1 & if\ i = k \\ \beta_{k+1,l} \times \cdots \times \beta_{i,l} & if\ i > k \end{cases}$$



$$\boldsymbol{\mathcal{L}}_{1,1,4,4} = \alpha_{1,2}\alpha_{1,3}\alpha_{1,4}\beta_{2,4}\beta_{3,4}\ \beta_{4,4}$$

5.2 PPMA: Method (Definition of Polyline Path Mask)

- 2D polyline path mask
  - Learn the horizontal factor and vertical decay factors
  - Calculate the decay weight of each polyline path, i.e., for path from $\boldsymbol{x}_{k,l}$ to $\boldsymbol{x}_{i,j}$:

$$\boldsymbol{\mathcal{L}}_{i,j,k,l} = \alpha_{i,j:l}\beta_{i:k,l}$$

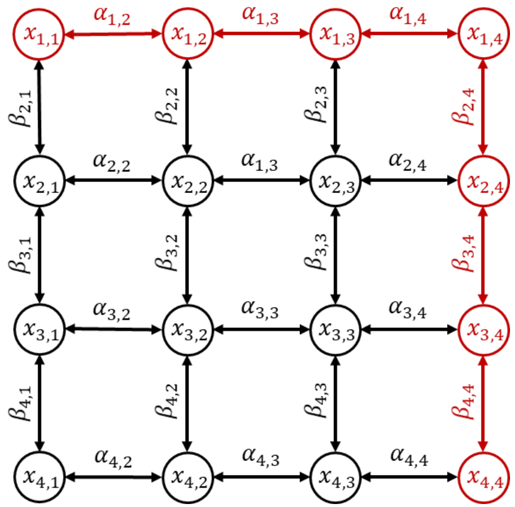
  - Combine bidirectional vertical-then-horizontal path and horizontal-then-vertical path:

$$\boldsymbol{\mathcal{L}}^{2D} = \boldsymbol{\mathcal{L}} + \widetilde{\boldsymbol{\mathcal{L}}}, \quad \widetilde{\boldsymbol{\mathcal{L}}}_{i,j,k,l} = \alpha_{k,j:l}\beta_{i:k,j} = \boldsymbol{\mathcal{L}}_{k,l,i,j}$$

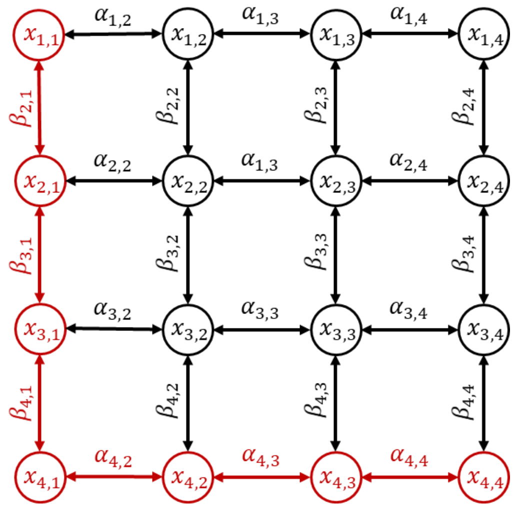
  - Unfold 4D tensors $\boldsymbol{\mathcal{L}}^{2D} \in \mathbb{R}^{H\times W\times H\times W}$ to 2D matrix $\boldsymbol{L}^{2D} \in \mathbb{R}^{HW\times HW}$:

$$\boldsymbol{L}^{2D} = \mathrm{unfold}(\boldsymbol{\mathcal{L}}^{2D}), \quad \boldsymbol{L}^{2D}_{(i-1)\times W+j,(k-1)\times W+l,} = \boldsymbol{\mathcal{L}}^{2D}_{i,j,k,l}$$

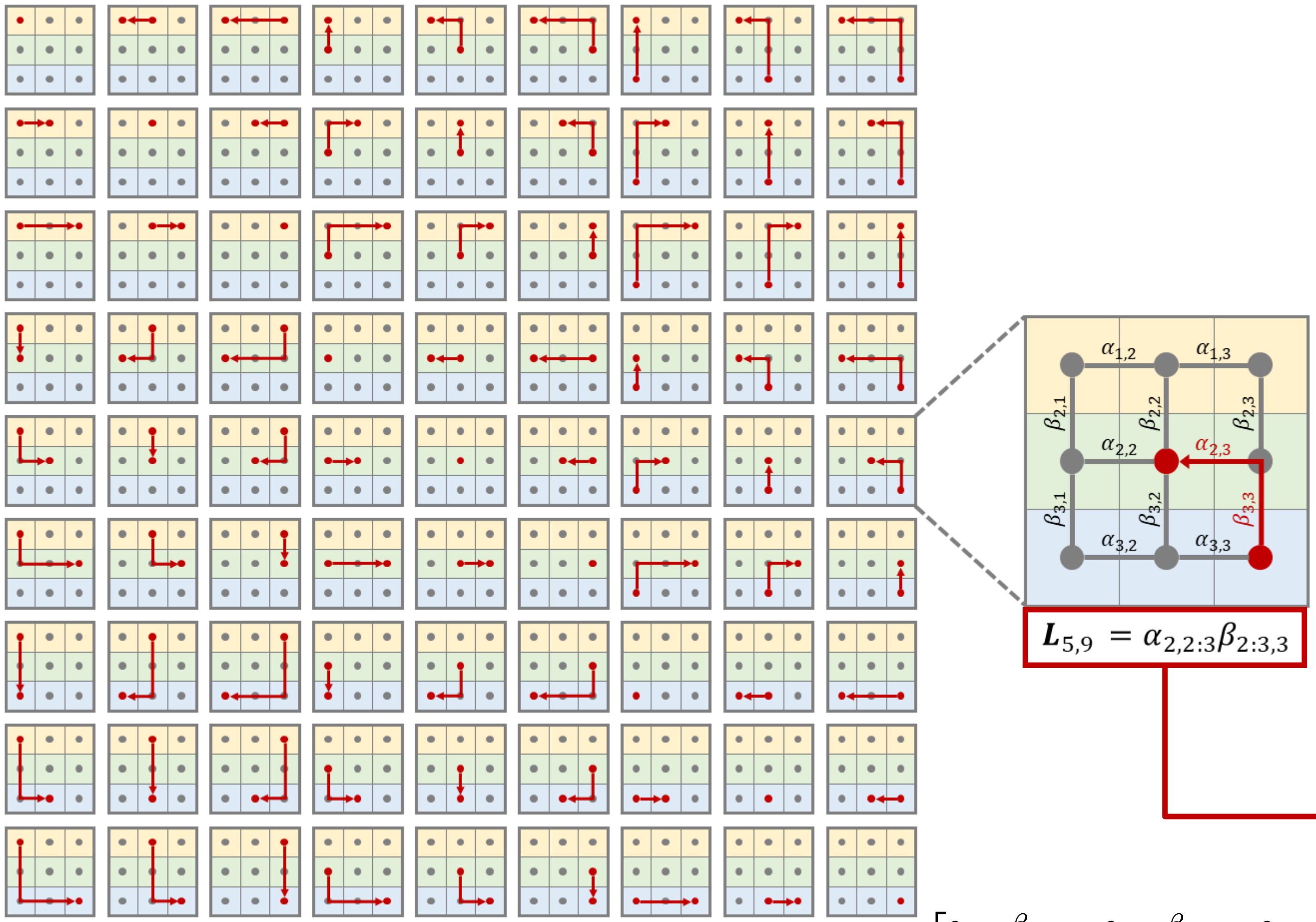
$$\boldsymbol{L} = \mathrm{unfold}(\boldsymbol{\mathcal{L}})$$



(a) V2H polyline path $\boldsymbol{\mathcal{L}}_{1,1,4,4}$



(b) H2V polyline path $\widetilde{\boldsymbol{\mathcal{L}}}_{1,1,4,4}$



(c) An illustration of the V2H polyline path mask on a 3×3 grid (with a total of 9 tokens).

$$L = \begin{bmatrix}
\alpha_{1,1:1}\beta_{1:1,1} & \alpha_{1,1:2}\beta_{1:1,2} & \alpha_{1,1:3}\beta_{1:1,3} & \alpha_{1,1:1}\beta_{1:2,1} & \alpha_{1,1:2}\beta_{1:2,2} & \alpha_{1,1:3}\beta_{1:2,3} & \alpha_{1,1:1}\beta_{1:3,1} & \alpha_{1,1:2}\beta_{1:3,2} & \alpha_{1,1:3}\beta_{1:3,3} \\
\alpha_{1,2:1}\beta_{1:1,1} & \alpha_{1,2:2}\beta_{1:1,2} & \alpha_{1,2:3}\beta_{1:1,3} & \alpha_{1,2:1}\beta_{1:2,1} & \alpha_{1,2:2}\beta_{1:2,2} & \alpha_{1,2:3}\beta_{1:2,3} & \alpha_{1,2:1}\beta_{1:3,1} & \alpha_{1,2:2}\beta_{1:3,2} & \alpha_{1,2:3}\beta_{1:3,3} \\
\alpha_{1,3:1}\beta_{1:1,1} & \alpha_{1,3:2}\beta_{1:1,2} & \alpha_{1,3:3}\beta_{1:1,3} & \alpha_{1,3:1}\beta_{1:2,1} & \alpha_{1,3:2}\beta_{1:2,2} & \alpha_{1,3:3}\beta_{1:2,3} & \alpha_{1,3:1}\beta_{1:3,1} & \alpha_{1,3:2}\beta_{1:3,2} & \alpha_{1,3:3}\beta_{1:3,3} \\
\alpha_{2,1:1}\beta_{2:1,1} & \alpha_{2,1:2}\beta_{2:1,2} & \alpha_{2,1:3}\beta_{2:1,3} & \alpha_{2,1:1}\beta_{2:2,1} & \alpha_{2,1:2}\beta_{2:2,2} & \alpha_{2,1:3}\beta_{2:2,3} & \alpha_{2,1:1}\beta_{2:3,1} & \alpha_{2,1:2}\beta_{2:3,2} & \alpha_{2,1:3}\beta_{2:3,3} \\
\alpha_{2,2:1}\beta_{2:1,1} & \alpha_{2,2:2}\beta_{2:1,2} & \alpha_{2,2:3}\beta_{2:1,3} & \alpha_{2,2:1}\beta_{2:2,1} & \alpha_{2,2:2}\beta_{2:2,2} & \alpha_{2,2:3}\beta_{2:2,3} & \alpha_{2,2:1}\beta_{2:3,1} & \alpha_{2,2:2}\beta_{2:3,2} & \alpha_{2,2:3}\beta_{2:3,3} \\
\alpha_{2,3:1}\beta_{2:1,1} & \alpha_{2,3:2}\beta_{2:1,2} & \alpha_{2,3:3}\beta_{2:1,3} & \alpha_{2,3:1}\beta_{2:2,1} & \alpha_{2,3:2}\beta_{2:2,2} & \alpha_{2,3:3}\beta_{2:2,3} & \alpha_{2,3:1}\beta_{2:3,1} & \alpha_{2,3:2}\beta_{2:3,2} & \alpha_{2,3:3}\beta_{2:3,3} \\
\alpha_{3,1:1}\beta_{3:1,1} & \alpha_{3,1:2}\beta_{3:1,2} & \alpha_{3,1:3}\beta_{3:1,3} & \alpha_{3,1:1}\beta_{3:2,1} & \alpha_{3,1:2}\beta_{3:2,2} & \alpha_{3,1:3}\beta_{3:2,3} & \alpha_{3,1:1}\beta_{3:3,1} & \alpha_{3,1:2}\beta_{3:3,2} & \alpha_{3,1:3}\beta_{3:3,3} \\
\alpha_{3,2:1}\beta_{3:1,1} & \alpha_{3,2:2}\beta_{3:1,2} & \alpha_{3,2:3}\beta_{3:1,3} & \alpha_{3,2:1}\beta_{3:2,1} & \alpha_{3,2:2}\beta_{3:2,2} & \alpha_{3,2:3}\beta_{3:2,3} & \alpha_{3,2:1}\beta_{3:3,1} & \alpha_{3,2:2}\beta_{3:3,2} & \alpha_{3,2:3}\beta_{3:3,3} \\
\alpha_{3,3:1}\beta_{3:1,1} & \alpha_{3,3:2}\beta_{3:1,2} & \alpha_{3,3:3}\beta_{3:1,3} & \alpha_{3,3:1}\beta_{3:2,1} & \alpha_{3,3:2}\beta_{3:2,2} & \alpha_{3,3:3}\beta_{3:2,3} & \alpha_{3,3:1}\beta_{3:3,1} & \alpha_{3,3:2}\beta_{3:3,2} & \alpha_{3,3:3}\beta_{3:3,3}
\end{bmatrix}$$

5.2 PPMA: Method (Efficient Computation Theory)

- Efficient Computation of Polyline Path Mask
  - **Naive Computation**: the polyline mask $\boldsymbol{L} \in \mathbb{R}^{N \times N}$ is large in size and each element $\mathcal{L}_{i,j,k,l} = \alpha_{i,j:l}\beta_{i:k,l}$ require numerous multiplications, resulting in a total complexity of $\mathcal{O}(N^{\frac{5}{2}})$

$$\mathcal{L}_{i,j,k,l} = \alpha_{i,j:l}\beta_{i:k,l}, \quad where\ i,k = 0,1,\dots,\mathrm{H}-1; \quad k,l = 0,1,\dots,\mathrm{W}-1$$

$$L = \underbrace{\begin{bmatrix}
\alpha_{1,1:1}\beta_{1:1,1} & \alpha_{1,1:2}\beta_{1:1,2} & \alpha_{1,1:3}\beta_{1:1,3} & \alpha_{1,1:1}\beta_{1:2,1} & \alpha_{1,1:2}\beta_{1:2,2} & \alpha_{1,1:3}\beta_{1:2,3} & \alpha_{1,1:1}\beta_{1:3,1} & \alpha_{1,1:2}\beta_{1:3,2} & \alpha_{1,1:3}\beta_{1:3,3} \\
\alpha_{1,2:1}\beta_{1:1,1} & \alpha_{1,2:2}\beta_{1:1,2} & \alpha_{1,2:3}\beta_{1:1,3} & \alpha_{1,2:1}\beta_{1:2,1} & \alpha_{1,2:2}\beta_{1:2,2} & \alpha_{1,2:3}\beta_{1:2,3} & \alpha_{1,2:1}\beta_{1:3,1} & \alpha_{1,2:2}\beta_{1:3,2} & \alpha_{1,2:3}\beta_{1:3,3} \\
\alpha_{1,3:1}\beta_{1:1,1} & \alpha_{1,3:2}\beta_{1:1,2} & \alpha_{1,3:3}\beta_{1:1,3} & \alpha_{1,3:1}\beta_{1:2,1} & \alpha_{1,3:2}\beta_{1:2,2} & \alpha_{1,3:3}\beta_{1:2,3} & \alpha_{1,3:1}\beta_{1:3,1} & \alpha_{1,3:2}\beta_{1:3,2} & \alpha_{1,3:3}\beta_{1:3,3} \\
\alpha_{2,1:1}\beta_{2:1,1} & \alpha_{2,1:2}\beta_{2:1,2} & \alpha_{2,1:3}\beta_{2:1,3} & \alpha_{2,1:1}\beta_{2:2,1} & \alpha_{2,1:2}\beta_{2:2,2} & \alpha_{2,1:3}\beta_{2:2,3} & \alpha_{2,1:1}\beta_{2:3,1} & \alpha_{2,1:2}\beta_{2:3,2} & \alpha_{2,1:3}\beta_{2:3,3} \\
\alpha_{2,2:1}\beta_{2:1,1} & \alpha_{2,2:2}\beta_{2:1,2} & \alpha_{2,2:3}\beta_{2:1,3} & \alpha_{2,2:1}\beta_{2:2,1} & \alpha_{2,2:2}\beta_{2:2,2} & \alpha_{2,2:3}\beta_{2:2,3} & \alpha_{2,2:1}\beta_{2:3,1} & \alpha_{2,2:2}\beta_{2:3,2} & \alpha_{2,2:3}\beta_{2:3,3} \\
\alpha_{2,3:1}\beta_{2:1,1} & \alpha_{2,3:2}\beta_{2:1,2} & \alpha_{2,3:3}\beta_{2:1,3} & \alpha_{2,3:1}\beta_{2:2,1} & \alpha_{2,3:2}\beta_{2:2,2} & \alpha_{2,3:3}\beta_{2:2,3} & \alpha_{2,3:1}\beta_{2:3,1} & \alpha_{2,3:2}\beta_{2:3,2} & \alpha_{2,3:3}\beta_{2:3,3} \\
\alpha_{3,1:1}\beta_{3:1,1} & \alpha_{3,1:2}\beta_{3:1,2} & \alpha_{3,1:3}\beta_{3:1,3} & \alpha_{3,1:1}\beta_{3:2,1} & \alpha_{3,1:2}\beta_{3:2,2} & \alpha_{3,1:3}\beta_{3:2,3} & \alpha_{3,1:1}\beta_{3:3,1} & \alpha_{3,1:2}\beta_{3:3,2} & \alpha_{3,1:3}\beta_{3:3,3} \\
\alpha_{3,2:1}\beta_{3:1,1} & \alpha_{3,2:2}\beta_{3:1,2} & \alpha_{3,2:3}\beta_{3:1,3} & \alpha_{3,2:1}\beta_{3:2,1} & \alpha_{3,2:2}\beta_{3:2,2} & \alpha_{3,2:3}\beta_{3:2,3} & \alpha_{3,2:1}\beta_{3:3,1} & \alpha_{3,2:2}\beta_{3:3,2} & \alpha_{3,2:3}\beta_{3:3,3} \\
\alpha_{3,3:1}\beta_{3:1,1} & \alpha_{3,3:2}\beta_{3:1,2} & \alpha_{3,3:3}\beta_{3:1,3} & \alpha_{3,3:1}\beta_{3:2,1} & \alpha_{3,3:2}\beta_{3:2,2} & \alpha_{3,3:3}\beta_{3:2,3} & \alpha_{3,3:1}\beta_{3:3,1} & \alpha_{3,3:2}\beta_{3:3,2} & \alpha_{3,3:3}\beta_{3:3,3}
\end{bmatrix}}_{\mathbb{R}^{\mathrm{N}\times\mathrm{N}}}$$

5.2 PPMA: Method (Efficient Computation Theory)

- Efficient Computation of Polyline Path Mask
  - **Naive Computation**: the polyline mask $\boldsymbol{L} \in \mathbb{R}^{N\times N}$ is large in size and each element $\mathcal{L}_{i,j,k,l} = \alpha_{i,j:l}\beta_{i:k,l}$ require numerous multiplications, resulting in a total complexity of $\mathcal{O}(N^{\frac{5}{2}})$
  - **Efficient Computation**: $\boldsymbol{L}$ can be decomposed as the multiplication of two **sparse matrices**: $\boldsymbol{L} = \boldsymbol{L}^H \times \boldsymbol{L}^V = \hat{\boldsymbol{L}}^H \odot \hat{\boldsymbol{L}}^V$

Theorem 1 (Matrix Decomposition). *For any matrix $\boldsymbol{M} \in \mathbb{R}^{HW\times HW}$ and $\boldsymbol{\mathcal{M}} = \text{fold}(\boldsymbol{M})$, if for $\forall i,j,k,l$, $\exists \boldsymbol{A}^i \in \mathbb{R}^{W\times W}$ and $\boldsymbol{B}^l \in \mathbb{R}^{H\times H}$, s.t., $\boldsymbol{\mathcal{M}}_{i,j,k,l} = \left[\boldsymbol{A}^i\right]_{j,l} \times \left[\boldsymbol{B}^l\right]_{i,k}$, then $\boldsymbol{M}$ can be decomposed as:*

$$\boldsymbol{M} = \boldsymbol{M}^A \times \boldsymbol{M}^B = \hat{\boldsymbol{M}}^A \odot \hat{\boldsymbol{M}}^B, \tag{6}$$

where $\boldsymbol{M}^A, \boldsymbol{M}^B, \hat{\boldsymbol{M}}^A, \hat{\boldsymbol{M}}^B \in \mathbb{R}^{HW\times HW}$, which satisfy

$$\boldsymbol{M}^A = \text{unfold}(\boldsymbol{\mathcal{M}}^A), \boldsymbol{M}^B = \text{unfold}(\boldsymbol{\mathcal{M}}^B), s.t., \boldsymbol{\mathcal{M}}^A_{i,:,k,:} = \begin{cases} \boldsymbol{A}^i & k=i \\ 0 & k\neq i \end{cases}, \boldsymbol{\mathcal{M}}^B_{:,j,:,l} = \begin{cases} \boldsymbol{B}^l & j=l \\ 0 & j\neq l \end{cases}, \tag{7}$$

$$\hat{\boldsymbol{M}}^A = \text{unfold}(\hat{\boldsymbol{\mathcal{M}}}^A), \ \hat{\boldsymbol{M}}^B = \text{unfold}(\hat{\boldsymbol{\mathcal{M}}}^B), \ \ s.t., \ \ \hat{\boldsymbol{\mathcal{M}}}^A_{i,:,k,:} = \boldsymbol{A}^i, \ \hat{\boldsymbol{\mathcal{M}}}^B_{:,j,:,l} = \boldsymbol{B}^l. \tag{8}$$

Corollary 1 (Mask Complexity). *The complexity of directly computing polyline path mask $\boldsymbol{L}$ is $\mathcal{O}(N^{\frac{5}{2}})$, which can be reduced to $\mathcal{O}(N^2)$ by applying Theorem 1, where $N = H\times W$.*

Theorem 1 (Matrix Decomposition). *For any matrix* $\boldsymbol{M} \in \mathbb{R}^{HW\times HW}$ *and* $\boldsymbol{\mathcal{M}} = \text{fold}(\boldsymbol{M})$, *if for* $\forall i,j,k,l$, $\exists \boldsymbol{A}^i \in \mathbb{R}^{W\times W}$ *and* $\boldsymbol{B}^l \in \mathbb{R}^{H\times H}$, *s.t.,* $\boldsymbol{\mathcal{M}}_{i,j,k,l} = [\boldsymbol{A}^i]_{j,l} \times [\boldsymbol{B}^l]_{i,k}$, *then* $\boldsymbol{M}$ *can be decomposed as:*

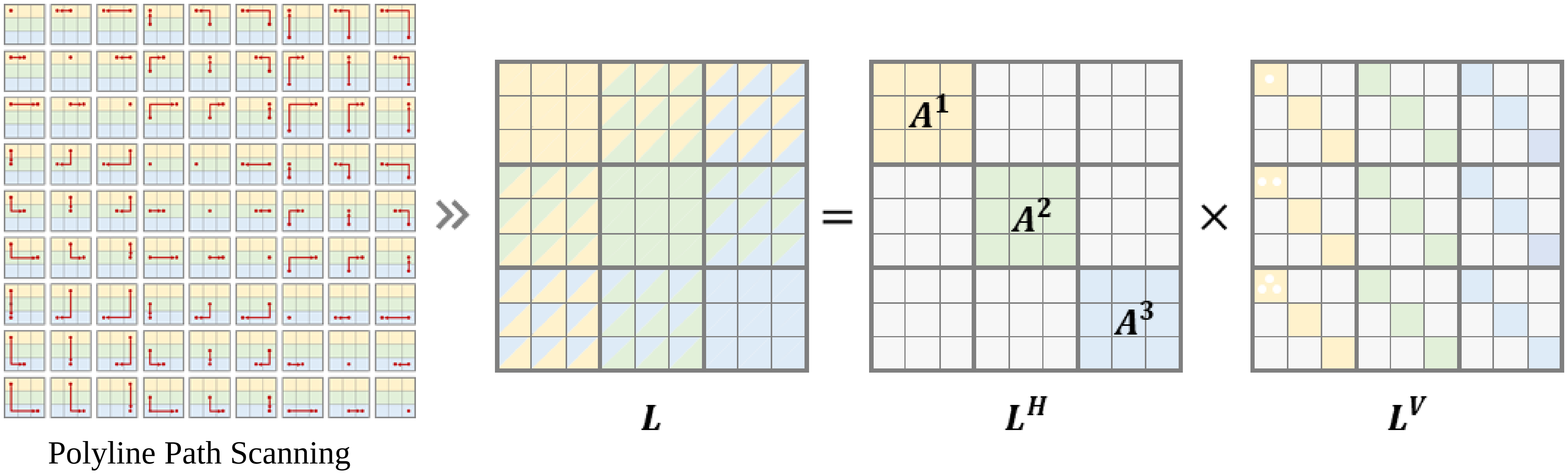
$$\boldsymbol{M} = \boldsymbol{M}^A \times \boldsymbol{M}^B = \hat{\boldsymbol{M}}^A \odot \hat{\boldsymbol{M}}^B, \tag{6}$$

where $\boldsymbol{M}^A, \boldsymbol{M}^B, \hat{\boldsymbol{M}}^A, \hat{\boldsymbol{M}}^B \in \mathbb{R}^{HW\times HW}$, *which satisfy*

$$\boldsymbol{M}^A = \text{unfold}(\boldsymbol{\mathcal{M}}^A), \boldsymbol{M}^B = \text{unfold}(\boldsymbol{\mathcal{M}}^B), \text{ s.t., } \boldsymbol{\mathcal{M}}^A_{i,:,k,:} = \begin{cases} \boldsymbol{A}^i & k=i \\ 0 & k\neq i \end{cases}, \boldsymbol{\mathcal{M}}^B_{:,j,:,l} = \begin{cases} \boldsymbol{B}^l & j=l \\ 0 & j\neq l \end{cases}, \tag{7}$$

$$\hat{\boldsymbol{M}}^A = \text{unfold}(\hat{\boldsymbol{\mathcal{M}}}^A), \ \hat{\boldsymbol{M}}^B = \text{unfold}(\hat{\boldsymbol{\mathcal{M}}}^B), \ \text{ s.t., } \ \hat{\boldsymbol{\mathcal{M}}}^A_{i,:,k,:} = \boldsymbol{A}^i, \ \hat{\boldsymbol{\mathcal{M}}}^B_{:,j,:,l} = \boldsymbol{B}^l. \tag{8}$$

The matrix $\boldsymbol{L}$ satisfies the conditions in Theorem 1 with $[\boldsymbol{A}^i]_{j,l} = \alpha_{i,j:l}$ and $[\boldsymbol{B}^l]_{i,k} = \beta_{i:k,l}$.



(a) An illustration of the Polyline Path Mask Decomposition

$$
L=\begin{bmatrix}
\alpha_{1,1:1}\beta_{1:1,1} & \alpha_{1,1:2}\beta_{1:1,2} & \alpha_{1,1:3}\beta_{1:1,3} & \alpha_{1,1:1}\beta_{1:2,1} & \alpha_{1,1:2}\beta_{1:2,2} & \alpha_{1,1:3}\beta_{1:2,3} & \alpha_{1,1:1}\beta_{1:3,1} & \alpha_{1,1:2}\beta_{1:3,2} & \alpha_{1,1:3}\beta_{1:3,3} \\
\alpha_{1,2:1}\beta_{1:1,1} & \alpha_{1,2:2}\beta_{1:1,2} & \alpha_{1,2:3}\beta_{1:1,3} & \alpha_{1,2:1}\beta_{1:2,1} & \alpha_{1,2:2}\beta_{1:2,2} & \alpha_{1,2:3}\beta_{1:2,3} & \alpha_{1,2:1}\beta_{1:3,1} & \alpha_{1,2:2}\beta_{1:3,2} & \alpha_{1,2:3}\beta_{1:3,3} \\
\alpha_{1,3:1}\beta_{1:1,1} & \alpha_{1,3:2}\beta_{1:1,2} & \alpha_{1,3:3}\beta_{1:1,3} & \alpha_{1,3:1}\beta_{1:2,1} & \alpha_{1,3:2}\beta_{1:2,2} & \alpha_{1,3:3}\beta_{1:2,3} & \alpha_{1,3:1}\beta_{1:3,1} & \alpha_{1,3:2}\beta_{1:3,2} & \alpha_{1,3:3}\beta_{1:3,3} \\
\alpha_{2,1:1}\beta_{2:1,1} & \alpha_{2,1:2}\beta_{2:1,2} & \alpha_{2,1:3}\beta_{2:1,3} & \alpha_{2,1:1}\beta_{2:2,1} & \alpha_{2,1:2}\beta_{2:2,2} & \alpha_{2,1:3}\beta_{2:2,3} & \alpha_{2,1:1}\beta_{2:3,1} & \alpha_{2,1:2}\beta_{2:3,2} & \alpha_{2,1:3}\beta_{2:3,3} \\
\alpha_{2,2:1}\beta_{2:1,1} & \alpha_{2,2:2}\beta_{2:1,2} & \alpha_{2,2:3}\beta_{2:1,3} & \alpha_{2,2:1}\beta_{2:2,1} & \alpha_{2,2:2}\beta_{2:2,2} & \alpha_{2,2:3}\beta_{2:2,3} & \alpha_{2,2:1}\beta_{2:3,1} & \alpha_{2,2:2}\beta_{2:3,2} & \alpha_{2,2:3}\beta_{2:3,3} \\
\alpha_{2,3:1}\beta_{2:1,1} & \alpha_{2,3:2}\beta_{2:1,2} & \alpha_{2,3:3}\beta_{2:1,3} & \alpha_{2,3:1}\beta_{2:2,1} & \alpha_{2,3:2}\beta_{2:2,2} & \alpha_{2,3:3}\beta_{2:2,3} & \alpha_{2,3:1}\beta_{2:3,1} & \alpha_{2,3:2}\beta_{2:3,2} & \alpha_{2,3:3}\beta_{2:3,3} \\
\alpha_{3,1:1}\beta_{3:1,1} & \alpha_{3,1:2}\beta_{3:1,2} & \alpha_{3,1:3}\beta_{3:1,3} & \alpha_{3,1:1}\beta_{3:2,1} & \alpha_{3,1:2}\beta_{3:2,2} & \alpha_{3,1:3}\beta_{3:2,3} & \alpha_{3,1:1}\beta_{3:3,1} & \alpha_{3,1:2}\beta_{3:3,2} & \alpha_{3,1:3}\beta_{3:3,3} \\
\alpha_{3,2:1}\beta_{3:1,1} & \alpha_{3,2:2}\beta_{3:1,2} & \alpha_{3,2:3}\beta_{3:1,3} & \alpha_{3,2:1}\beta_{3:2,1} & \alpha_{3,2:2}\beta_{3:2,2} & \alpha_{3,2:3}\beta_{3:2,3} & \alpha_{3,2:1}\beta_{3:3,1} & \alpha_{3,2:2}\beta_{3:3,2} & \alpha_{3,2:3}\beta_{3:3,3} \\
\alpha_{3,3:1}\beta_{3:1,1} & \alpha_{3,3:2}\beta_{3:1,2} & \alpha_{3,3:3}\beta_{3:1,3} & \alpha_{3,3:1}\beta_{3:2,1} & \alpha_{3,3:2}\beta_{3:2,2} & \alpha_{3,3:3}\beta_{3:2,3} & \alpha_{3,3:1}\beta_{3:3,1} & \alpha_{3,3:2}\beta_{3:3,2} & \alpha_{3,3:3}\beta_{3:3,3}
\end{bmatrix}
$$

Complexity: $\mathcal{O}(N^{\frac{5}{2}})$

$$
=\begin{bmatrix}
\alpha_{1,1:1} & \alpha_{1,1:2} & \alpha_{1,1:3} & 0 & 0 & 0 & 0 & 0 & 0 \\
\alpha_{1,2:1} & \alpha_{1,2:2} & \alpha_{1,2:3} & 0 & 0 & 0 & 0 & 0 & 0 \\
\alpha_{1,3:1} & \alpha_{1,3:2} & \alpha_{1,3:3} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \alpha_{2,1:1} & \alpha_{2,1:2} & \alpha_{2,1:3} & 0 & 0 & 0 \\
0 & 0 & 0 & \alpha_{2,2:1} & \alpha_{2,2:2} & \alpha_{2,2:3} & 0 & 0 & 0 \\
0 & 0 & 0 & \alpha_{2,3:1} & \alpha_{2,3:2} & \alpha_{2,3:3} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \alpha_{3,1:1} & \alpha_{3,1:2} & \alpha_{3,1:3} \\
0 & 0 & 0 & & & & \alpha_{3,2:1} & \alpha_{3,2:2} & \alpha_{3,2:3} \\
0 & 0 & 0 & & & & \alpha_{3,3:1} & \alpha_{3,3:2} & \alpha_{3,3:3}
\end{bmatrix}
\times
\begin{bmatrix}
\beta_{1:1,1} & 0 & 0 & \beta_{1:2,1} & 0 & 0 & \beta_{1:3,1} & 0 & 0 \\
0 & \beta_{1:1,2} & 0 & 0 & \beta_{1:2,2} & 0 & 0 & \beta_{1:3,2} & 0 \\
0 & 0 & \beta_{1:1,3} & 0 & 0 & \beta_{1:2,3} & 0 & 0 & \beta_{1:3,3} \\
\beta_{2:1,1} & 0 & 0 & \beta_{2:2,1} & 0 & 0 & \beta_{2:3,1} & 0 & 0 \\
0 & \beta_{2:1,2} & 0 & 0 & \beta_{2:2,2} & 0 & 0 & \beta_{2:3,2} & 0 \\
0 & 0 & \beta_{2:1,3} & 0 & 0 & \beta_{2:2,3} & 0 & 0 & \beta_{2:3,3} \\
\beta_{3:1,1} & 0 & 0 & \beta_{3:2,1} & 0 & 0 & \beta_{3:3,1} & 0 & 0 \\
0 & \beta_{3:1,2} & 0 & & & & 0 & \beta_{3:3,2} & 0 \\
0 & 0 & \beta_{3:1,3} & & & & 0 & 0 & \beta_{3:3,3}
\end{bmatrix}
$$

$\boldsymbol{L^H}$: $\mathcal{O}(N^{\frac{3}{2}})$ $\boldsymbol{L^V}$: $\mathcal{O}(N^{\frac{3}{2}})$

$$
=\begin{bmatrix}
\alpha_{1,1:1} & \alpha_{1,1:2} & \alpha_{1,1:3} & \alpha_{1,1:1} & \alpha_{1,1:2} & \alpha_{1,1:3} & \alpha_{1,1:1} & \alpha_{1,1:2} & \alpha_{1,1:3} \\
\alpha_{1,2:1} & \alpha_{1,2:2} & \alpha_{1,2:3} & \alpha_{1,2:1} & \alpha_{1,2:2} & \alpha_{1,2:3} & \alpha_{1,2:1} & \alpha_{1,2:2} & \alpha_{1,2:3} \\
\alpha_{1,3:1} & \alpha_{1,3:2} & \alpha_{1,3:3} & \alpha_{1,3:1} & \alpha_{1,3:2} & \alpha_{1,3:3} & \alpha_{1,3:1} & \alpha_{1,3:2} & \alpha_{1,3:3} \\
\alpha_{2,1:1} & \alpha_{2,1:2} & \alpha_{2,1:3} & \alpha_{2,1:1} & \alpha_{2,1:2} & \alpha_{2,1:3} & \alpha_{2,1:1} & \alpha_{2,1:2} & \alpha_{2,1:3} \\
\alpha_{2,2:1} & \alpha_{2,2:2} & \alpha_{2,2:3} & \alpha_{2,2:1} & \alpha_{2,2:2} & \alpha_{2,2:3} & \alpha_{2,2:1} & \alpha_{2,2:2} & \alpha_{2,2:3} \\
\alpha_{2,3:1} & \alpha_{2,3:2} & \alpha_{2,3:3} & \alpha_{2,3:1} & \alpha_{2,3:2} & \alpha_{2,3:3} & \alpha_{2,3:1} & \alpha_{2,3:2} & \\
\alpha_{3,1:1} & \alpha_{3,1:2} & \alpha_{3,1:3} & \alpha_{3,1:1} & \alpha_{3,1:2} & \alpha_{3,1:3} & \alpha_{3,1:1} & \alpha_{3,1:2} & \alpha_{3,1:3} \\
\alpha_{3,2:1} & \alpha_{3,2:2} & \alpha_{3,2:3} & \alpha_{3,2:1} & \alpha_{3,2:2} & \alpha_{3,2:3} & \alpha_{3,2:1} & \alpha_{3,2:2} & \alpha_{3,2:3} \\
\alpha_{3,3:1} & \alpha_{3,3:2} & \alpha_{3,3:3} & \alpha_{3,3:1} & \alpha_{3,3:2} & \alpha_{3,3:3} & \alpha_{3,3:1} & \alpha_{3,3:2} & \alpha_{3,3:3}
\end{bmatrix}
\odot
\begin{bmatrix}
\beta_{1:1,1} & \beta_{1:1,2} & \beta_{1:1,3} & \beta_{1:2,1} & \beta_{1:2,2} & \beta_{1:2,3} & \beta_{1:3,1} & \beta_{1:3,2} & \beta_{1:3,3} \\
\beta_{1:1,1} & \beta_{1:1,2} & \beta_{1:1,3} & \beta_{1:2,1} & \beta_{1:2,2} & \beta_{1:2,3} & \beta_{1:3,1} & \beta_{1:3,2} & \beta_{1:3,3} \\
\beta_{1:1,1} & \beta_{1:1,2} & \beta_{1:1,3} & \beta_{1:2,1} & \beta_{1:2,2} & \beta_{1:2,3} & \beta_{1:3,1} & \beta_{1:3,2} & \beta_{1:3,3} \\
\beta_{2:1,1} & \beta_{2:1,2} & \beta_{2:1,3} & \beta_{2:2,1} & \beta_{2:2,2} & \beta_{2:2,3} & \beta_{2:3,1} & \beta_{2:3,2} & \beta_{2:3,3} \\
\beta_{2:1,1} & \beta_{2:1,2} & \beta_{2:1,3} & \beta_{2:2,1} & \beta_{2:2,2} & \beta_{2:2,3} & \beta_{2:3,1} & \beta_{2:3,2} & \beta_{2:3,3} \\
 & \beta_{2:1,2} & \beta_{2:1,3} & \beta_{2:2,1} & \beta_{2:2,2} & \beta_{2:2,3} & \beta_{2:3,1} & \beta_{2:3,2} & \beta_{2:3,3} \\
\beta_{3:1,1} & \beta_{3:1,2} & \beta_{3:1,3} & \beta_{3:2,1} & \beta_{3:2,2} & \beta_{3:2,3} & \beta_{3:3,1} & \beta_{3:3,2} & \beta_{3:3,3} \\
\beta_{3:1,1} & \beta_{3:1,2} & \beta_{3:1,3} & \beta_{3:2,1} & \beta_{3:2,2} & \beta_{3:2,3} & \beta_{3:3,1} & \beta_{3:3,2} & \beta_{3:3,3} \\
\beta_{3:1,1} & \beta_{3:1,2} & \beta_{3:1,3} & \beta_{3:2,1} & \beta_{3:2,2} & \beta_{3:2,3} & \beta_{3:3,1} & \beta_{3:3,2} & \beta_{3:3,3}
\end{bmatrix}
$$

$\mathcal{O}(N^2)$

Complexity: $\mathcal{O}(N^2)$

5.2 PPMA: Method (Efficient Computation Theory)

- Efficient Computation of Polyline Path Mask Matrix Multiplication
  - **Naive Computation**: for a rank-N matrix $\boldsymbol{L} \in \mathbb{R}^{N\times N}$ and a vector $\boldsymbol{x} \in \mathbb{R}^{N}$, $\boldsymbol{Lx}$ requires a complexity of $\mathcal{O}(N^2)$
  - **Efficient Computation**: for decomposed polyline path mask, $\boldsymbol{Lx} = \boldsymbol{L}^H \times \boldsymbol{L}^V \times \boldsymbol{x}$ requires a complexity of $\mathcal{O}(N)$

Theorem 2 (Efficient Matrix Multiplication). *For matrices $\boldsymbol{M}^A, \boldsymbol{M}^B$ defined in Theorem 1, $\forall \boldsymbol{x} \in \mathbb{R}^{HW}$, the following equation holds:*

$$\boldsymbol{y} = \boldsymbol{M}^A \times \boldsymbol{M}^B \times \boldsymbol{x} \quad \Leftrightarrow \quad \boldsymbol{Z}_{:,l} = \boldsymbol{B}^l \times \boldsymbol{X}_{:,l},\ \boldsymbol{Y}_{i,:} = \boldsymbol{A}^i \times \boldsymbol{Z}_{i,:}, \tag{9}$$

where $\boldsymbol{y} \in \mathbb{R}^{HW}$, $\boldsymbol{X} = \mathrm{unvec}(\boldsymbol{x}) \in \mathbb{R}^{H\times W}$, $\boldsymbol{Y} = \mathrm{unvec}(\boldsymbol{y}) \in \mathbb{R}^{H\times W}$, $\boldsymbol{Z} \in \mathbb{R}^{H\times W}$, and the operator $\mathrm{vec}(\cdot)$ *vectorizes a matrix by stacking its columns and* $\mathrm{unvec}(\cdot)$ *is its inverse operator.*

Algorithm 1: Efficient Masked Attention Computation.

Input: decay factors α, β of $\boldsymbol{L}$, vector $\boldsymbol{x} \in \mathbb{R}^{HW}$;

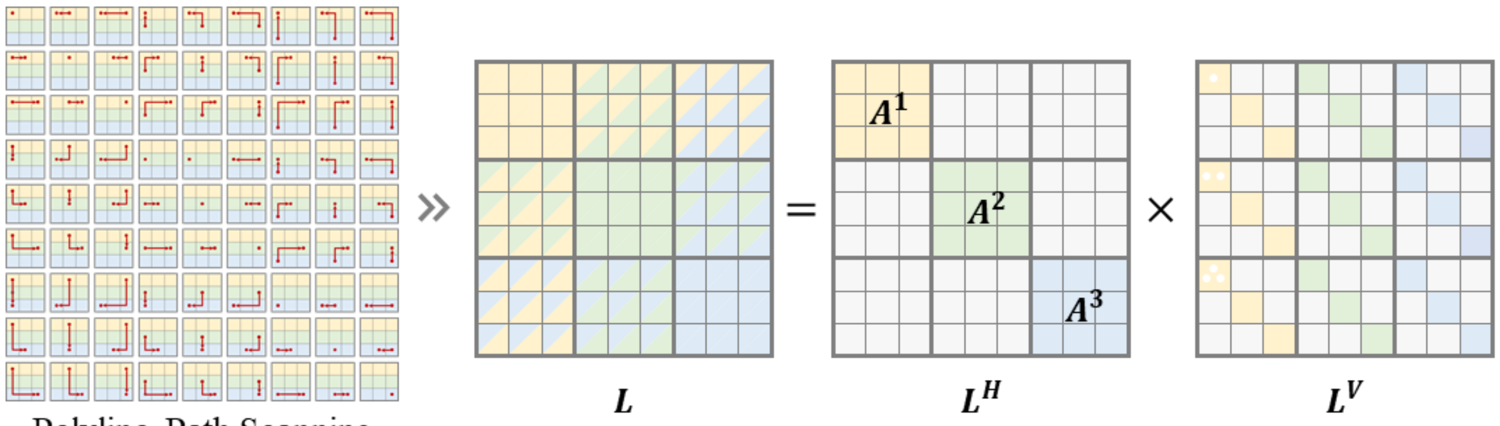
1: Compute $\boldsymbol{X} = \mathrm{unvec}(\boldsymbol{x}) \in \mathbb{R}^{H\times W}$;
2: Compute $\boldsymbol{B}^l \in \mathbb{R}^{H\times H}$, where for $l = 1:W$, $[\boldsymbol{B}^l]_{i,k} = \beta_{i:k,l}$;
3: Compute $\boldsymbol{Z} \in \mathbb{R}^{H\times W}$, where $\boldsymbol{Z}_{:,l} = \boldsymbol{B}^l \times \boldsymbol{X}_{:,l}$;
4: Compute $\boldsymbol{A}^i \in \mathbb{R}^{W\times W}$, where for $i = 1:H$, $[\boldsymbol{A}^i]_{j,l} = \alpha_{i,j:l}$;
5: Compute $\boldsymbol{Y} \in \mathbb{R}^{H\times W}$, where $\boldsymbol{Y}_{i,:} = \boldsymbol{A}^i \times \boldsymbol{Z}_{i,:}$;

Output: $\boldsymbol{y} = \mathrm{vec}(\boldsymbol{Y})$;

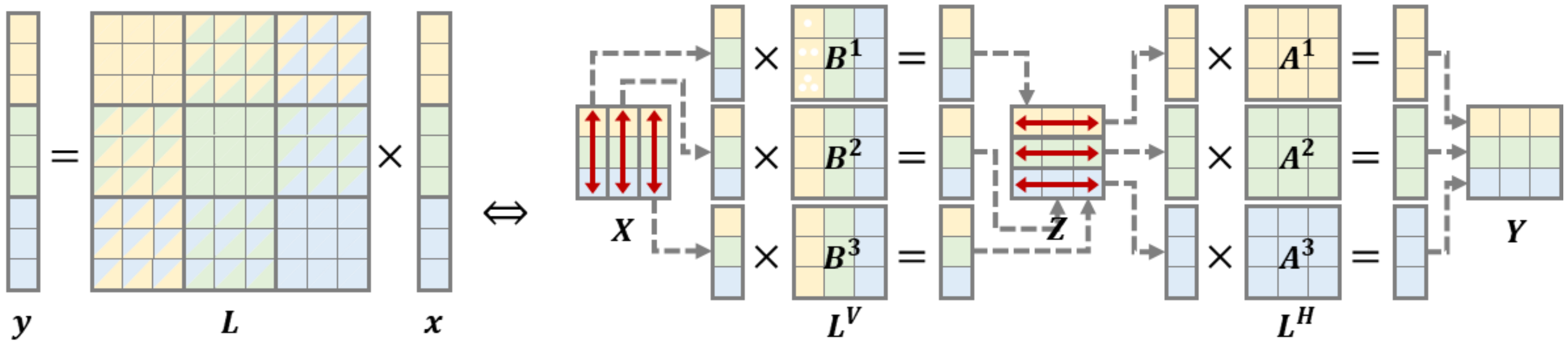
Theorem 2 (Efficient Matrix Multiplication). *For matrices $\boldsymbol{M}^A, \boldsymbol{M}^B$ defined in Theorem 1, $\forall \boldsymbol{x} \in \mathbb{R}^{HW}$, the following equation holds:*

$$\boldsymbol{y} = \boldsymbol{M}^A \times \boldsymbol{M}^B \times \boldsymbol{x} \quad \Leftrightarrow \quad \boldsymbol{Z}_{:,l} = \boldsymbol{B}^l \times \boldsymbol{X}_{:,l},\ \boldsymbol{Y}_{i,:} = \boldsymbol{A}^i \times \boldsymbol{Z}_{i,:}, \tag{9}$$

where $\boldsymbol{y} \in \mathbb{R}^{HW}$, $\boldsymbol{X} = \mathrm{unvec}(\boldsymbol{x}) \in \mathbb{R}^{H\times W}$, $\boldsymbol{Y} = \mathrm{unvec}(\boldsymbol{y}) \in \mathbb{R}^{H\times W}$, $\boldsymbol{Z} \in \mathbb{R}^{H\times W}$, and the operator $\mathrm{vec}(\cdot)$ *vectorizes a matrix by stacking its columns and* $\mathrm{unvec}(\cdot)$ *is its inverse operator.*



(a) Polyline Path Mask Decomposition



(b) Polyline Path Mask Multiplication

Theorem 2 (Efficient Matrix Multiplication). *For matrices $\boldsymbol{M}^A, \boldsymbol{M}^B$ defined in Theorem 1, $\forall \boldsymbol{x} \in \mathbb{R}^{HW}$, the following equation holds:*

$$\boldsymbol{y} = \boldsymbol{M}^A \times \boldsymbol{M}^B \times \boldsymbol{x} \quad \Leftrightarrow \quad \boldsymbol{Z}_{:,l} = \boldsymbol{B}^l \times \boldsymbol{X}_{:,l}, \; \boldsymbol{Y}_{i,:} = \boldsymbol{A}^i \times \boldsymbol{Z}_{i,:}, \tag{9}$$

where $\boldsymbol{y} \in \mathbb{R}^{HW}$, $\boldsymbol{X} = \mathrm{unvec}(\boldsymbol{x}) \in \mathbb{R}^{H\times W}$, $\boldsymbol{Y} = \mathrm{unvec}(\boldsymbol{y}) \in \mathbb{R}^{H\times W}$, $\boldsymbol{Z} \in \mathbb{R}^{H\times W}$, and the operator $\mathrm{vec}(\cdot)$ vectorizes a matrix by stacking its columns and $\mathrm{unvec}(\cdot)$ is its inverse operator.

Algorithm 1: Efficient Masked Attention Computation.

Input: decay factors α, β of $\boldsymbol{L}$, vector $\boldsymbol{x} \in \mathbb{R}^{HW}$;

1: Compute $\boldsymbol{X} = \mathrm{unvec}(\boldsymbol{x}) \in \mathbb{R}^{H\times W}$;

2: Compute $\boldsymbol{B}^l \in \mathbb{R}^{H\times H}$, where for $l = 1:W$, $[\boldsymbol{B}^l]_{i,k} = \beta_{i:k,l}$;

3: Compute $\boldsymbol{Z} \in \mathbb{R}^{H\times W}$, where $\boldsymbol{Z}_{:,l} = \boldsymbol{B}^l \times \boldsymbol{X}_{:,l}$; — Complexity: $\mathcal{O}(H^2W) \longrightarrow \mathcal{O}(HW)$

4: Compute $\boldsymbol{A}^i \in \mathbb{R}^{W\times W}$, where for $i = 1:H$, $[\boldsymbol{A}^i]_{j,l} = \alpha_{i,j:l}$;

5: Compute $\boldsymbol{Y} \in \mathbb{R}^{H\times W}$, where $\boldsymbol{Y}_{i,:} = \boldsymbol{A}^i \times \boldsymbol{Z}_{i,:}$; — Complexity: $\mathcal{O}(HW^2) \longrightarrow \mathcal{O}(HW)$

Output: $\boldsymbol{y} = \mathrm{vec}(\boldsymbol{Y})$;

$$L = \begin{bmatrix}
\alpha_{1,1:1}\beta_{1:1,1} & \alpha_{1,1:2}\beta_{1:1,2} & \alpha_{1,1:3}\beta_{1:1,3} & \alpha_{1,1:1}\beta_{1:2,1} & \alpha_{1,1:2}\beta_{1:2,2} & \alpha_{1,1:3}\beta_{1:2,3} & \alpha_{1,1:1}\beta_{1:3,1} & \alpha_{1,1:2}\beta_{1:3,2} & \alpha_{1,1:3}\beta_{1:3,3} \\
\alpha_{1,2:1}\beta_{1:1,1} & \alpha_{1,2:2}\beta_{1:1,2} & \alpha_{1,2:3}\beta_{1:1,3} & \alpha_{1,2:1}\beta_{1:2,1} & \alpha_{1,2:2}\beta_{1:2,2} & \alpha_{1,2:3}\beta_{1:2,3} & \alpha_{1,2:1}\beta_{1:3,1} & \alpha_{1,2:2}\beta_{1:3,2} & \alpha_{1,2:3}\beta_{1:3,3} \\
\alpha_{1,3:1}\beta_{1:1,1} & \alpha_{1,3:2}\beta_{1:1,2} & \alpha_{1,3:3}\beta_{1:1,3} & \alpha_{1,3:1}\beta_{1:2,1} & \alpha_{1,3:2}\beta_{1:2,2} & \alpha_{1,3:3}\beta_{1:2,3} & \alpha_{1,3:1}\beta_{1:3,1} & \alpha_{1,3:2}\beta_{1:3,2} & \alpha_{1,3:3}\beta_{1:3,3} \\
\alpha_{2,1:1}\beta_{2:1,1} & \alpha_{2,1:2}\beta_{2:1,2} & \alpha_{2,1:3}\beta_{2:1,3} & \alpha_{2,1:1}\beta_{2:2,1} & \alpha_{2,1:2}\beta_{2:2,2} & \alpha_{2,1:3}\beta_{2:2,3} & \alpha_{2,1:1}\beta_{2:3,1} & \alpha_{2,1:2}\beta_{2:3,2} & \alpha_{2,1:3}\beta_{2:3,3} \\
\alpha_{2,2:1}\beta_{2:1,1} & \alpha_{2,2:2}\beta_{2:1,2} & \alpha_{2,2:3}\beta_{2:1,3} & \alpha_{2,2:1}\beta_{2:2,1} & \alpha_{2,2:2}\beta_{2:2,2} & \alpha_{2,2:3}\beta_{2:2,3} & \alpha_{2,2:1}\beta_{2:3,1} & \alpha_{2,2:2}\beta_{2:3,2} & \alpha_{2,2:3}\beta_{2:3,3} \\
\alpha_{2,3:1}\beta_{2:1,1} & \alpha_{2,3:2}\beta_{2:1,2} & \alpha_{2,3:3}\beta_{2:1,3} & \alpha_{2,3:1}\beta_{2:2,1} & \alpha_{2,3:2}\beta_{2:2,2} & \alpha_{2,3:3}\beta_{2:2,3} & \alpha_{2,3:1}\beta_{2:3,1} & \alpha_{2,3:2}\beta_{2:3,2} & \alpha_{2,3:3}\beta_{2:3,3} \\
\alpha_{3,1:1}\beta_{3:1,1} & \alpha_{3,1:2}\beta_{3:1,2} & \alpha_{3,1:3}\beta_{3:1,3} & \alpha_{3,1:1}\beta_{3:2,1} & \alpha_{3,1:2}\beta_{3:2,2} & \alpha_{3,1:3}\beta_{3:2,3} & \alpha_{3,1:1}\beta_{3:3,1} & \alpha_{3,1:2}\beta_{3:3,2} & \alpha_{3,1:3}\beta_{3:3,3} \\
\alpha_{3,2:1}\beta_{3:1,1} & \alpha_{3,2:2}\beta_{3:1,2} & \alpha_{3,2:3}\beta_{3:1,3} & \alpha_{3,2:1}\beta_{3:2,1} & \alpha_{3,2:2}\beta_{3:2,2} & \alpha_{3,2:3}\beta_{3:2,3} & \alpha_{3,2:1}\beta_{3:3,1} & \alpha_{3,2:2}\beta_{3:3,2} & \alpha_{3,2:3}\beta_{3:3,3} \\
\alpha_{3,3:1}\beta_{3:1,1} & \alpha_{3,3:2}\beta_{3:1,2} & \alpha_{3,3:3}\beta_{3:1,3} & \alpha_{3,3:1}\beta_{3:2,1} & \alpha_{3,3:2}\beta_{3:2,2} & \alpha_{3,3:3}\beta_{3:2,3} & \alpha_{3,3:1}\beta_{3:3,1} & \alpha_{3,3:2}\beta_{3:3,2} & \alpha_{3,3:3}\beta_{3:3,3}
\end{bmatrix}$$

Theorem 2 (Efficient Matrix Multiplication). *For matrices $\boldsymbol{M}^A, \boldsymbol{M}^B$ defined in Theorem 1,* $\forall \boldsymbol{x} \in \mathbb{R}^{HW}$*, the following equation holds:*

$$\boldsymbol{y} = \boldsymbol{M}^A \times \boldsymbol{M}^B \times \boldsymbol{x} \quad \Leftrightarrow \quad \boldsymbol{Z}_{:,l} = \boldsymbol{B}^l \times \boldsymbol{X}_{:,l}, \; \boldsymbol{Y}_{i,:} = \boldsymbol{A}^i \times \boldsymbol{Z}_{i,:}, \tag{9}$$

where $\boldsymbol{y} \in \mathbb{R}^{HW}$, $\boldsymbol{X} = \mathrm{unvec}(\boldsymbol{x}) \in \mathbb{R}^{H\times W}$, $\boldsymbol{Y} = \mathrm{unvec}(\boldsymbol{y}) \in \mathbb{R}^{H\times W}$, $\boldsymbol{Z} \in \mathbb{R}^{H\times W}$, *and the operator* $\mathrm{vec}(\cdot)$ *vectorizes a matrix by stacking its columns and* $\mathrm{unvec}(\cdot)$ *is its inverse operator.*

Algorithm 1: Efficient Masked Attention Computation.

Input: decay factors α, β of $\boldsymbol{L}$, vector $\boldsymbol{x} \in \mathbb{R}^{HW}$;

1: Compute $\boldsymbol{X} = \mathrm{unvec}(\boldsymbol{x}) \in \mathbb{R}^{H\times W}$;

2: Compute $\boldsymbol{B}^l \in \mathbb{R}^{H\times H}$, where for $l = 1:W$, $[\boldsymbol{B}^l]_{i,k} = \beta_{i:k,l}$;

3: Compute $\boldsymbol{Z} \in \mathbb{R}^{H\times W}$, where $\boldsymbol{Z}_{:,l} = \boldsymbol{B}^l \times \boldsymbol{X}_{:,l}$; — Complexity: $\mathcal{O}(H^2W) \longrightarrow \mathcal{O}(HW)$

4: Compute $\boldsymbol{A}^i \in \mathbb{R}^{W\times W}$, where for $i = 1:H$, $[\boldsymbol{A}^i]_{j,l} = \alpha_{i,j:l}$;

5: Compute $\boldsymbol{Y} \in \mathbb{R}^{H\times W}$, where $\boldsymbol{Y}_{i,:} = \boldsymbol{A}^i \times \boldsymbol{Z}_{i,:}$; — Complexity: $\mathcal{O}(HW^2) \longrightarrow \mathcal{O}(HW)$

Output: $\boldsymbol{y} = \mathrm{vec}(\boldsymbol{Y})$;

$$L = \begin{bmatrix}
\alpha_{1,1:1} & \alpha_{1,1:2} & \alpha_{1,1:3} & 0 & 0 & 0 & 0 & 0 & 0 \\
\alpha_{1,2:1} & \alpha_{1,2:2} & \alpha_{1,2:3} & 0 & 0 & 0 & 0 & 0 & 0 \\
\alpha_{1,3:1} & \alpha_{1,3:2} & \alpha_{1,3:3} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \alpha_{2,1:1} & \alpha_{2,1:2} & \alpha_{2,1:3} & 0 & 0 & 0 \\
0 & 0 & 0 & \alpha_{2,2:1} & \alpha_{2,2:2} & \alpha_{2,2:3} & 0 & 0 & 0 \\
0 & 0 & 0 & \alpha_{2,3:1} & \alpha_{2,3:2} & \alpha_{2,3:3} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \alpha_{3,1:1} & \alpha_{3,1:2} & \alpha_{3,1:3} \\
0 & 0 & 0 & 0 & 0 & 0 & \alpha_{3,2:1} & \alpha_{3,2:2} & \alpha_{3,2:3} \\
0 & 0 & 0 & 0 & 0 & 0 & \alpha_{3,3:1} & \alpha_{3,3:2} & \alpha_{3,3:3}
\end{bmatrix} \times \begin{bmatrix}
\beta_{1:1,1} & 0 & 0 & \beta_{1:2,1} & 0 & 0 & \beta_{1:3,1} & 0 & 0 \\
0 & \beta_{1:1,2} & 0 & 0 & \beta_{1:2,2} & 0 & 0 & \beta_{1:3,2} & 0 \\
0 & 0 & \beta_{1:1,3} & 0 & 0 & \beta_{1:2,3} & 0 & 0 & \beta_{1:3,3} \\
\beta_{2:1,1} & 0 & 0 & \beta_{2:2,1} & 0 & 0 & \beta_{2:3,1} & 0 & 0 \\
0 & \beta_{2:1,2} & 0 & 0 & \beta_{2:2,2} & 0 & 0 & \beta_{2:3,2} & 0 \\
0 & 0 & \beta_{2:1,3} & 0 & 0 & \beta_{2:2,3} & 0 & 0 & \beta_{2:3,3} \\
\beta_{3:1,1} & 0 & 0 & \beta_{3:2,1} & 0 & 0 & \beta_{3:3,1} & 0 & 0 \\
0 & \beta_{3:1,2} & 0 & 0 & \beta_{3:2,2} & 0 & 0 & \beta_{3:3,2} & 0 \\
0 & 0 & \beta_{3:1,3} & 0 & 0 & \beta_{3:2,3} & 0 & 0 & \beta_{3:3,3}
\end{bmatrix}$$

Theorem 2 (Efficient Matrix Multiplication). *For matrices $\boldsymbol{M}^A, \boldsymbol{M}^B$ defined in Theorem 1, $\forall \boldsymbol{x} \in \mathbb{R}^{HW}$, the following equation holds:*

$$\boldsymbol{y} = \boldsymbol{M}^A \times \boldsymbol{M}^B \times \boldsymbol{x} \quad \Leftrightarrow \quad \boldsymbol{Z}_{:,l} = \boldsymbol{B}^l \times \boldsymbol{X}_{:,l}, \ \boldsymbol{Y}_{i,:} = \boldsymbol{A}^i \times \boldsymbol{Z}_{i,:}, \tag{9}$$

where $\boldsymbol{y} \in \mathbb{R}^{HW}$, $\boldsymbol{X} = \mathrm{unvec}(\boldsymbol{x}) \in \mathbb{R}^{H\times W}$, $\boldsymbol{Y} = \mathrm{unvec}(\boldsymbol{y}) \in \mathbb{R}^{H\times W}$, $\boldsymbol{Z} \in \mathbb{R}^{H\times W}$, and the operator $\mathrm{vec}(\cdot)$ vectorizes a matrix by stacking its columns and $\mathrm{unvec}(\cdot)$ is its inverse operator.

Algorithm 1: Efficient Masked Attention Computation.

Input: decay factors α, β of $\boldsymbol{L}$, vector $\boldsymbol{x} \in \mathbb{R}^{HW}$;

1: Compute $\boldsymbol{X} = \mathrm{unvec}(\boldsymbol{x}) \in \mathbb{R}^{H\times W}$;

2: Compute $\boldsymbol{B}^l \in \mathbb{R}^{H\times H}$, where for $l = 1:W$, $[\boldsymbol{B}^l]_{i,k} = \beta_{i:k,l}$;

3: Compute $\boldsymbol{Z} \in \mathbb{R}^{H\times W}$, where $\boldsymbol{Z}_{:,l} = \boldsymbol{B}^l \times \boldsymbol{X}_{:,l}$;

4: Compute $\boldsymbol{A}^i \in \mathbb{R}^{W\times W}$, where for $i = 1:H$, $[\boldsymbol{A}^i]_{j,l} = \alpha_{i,j:l}$;

5: Compute $\boldsymbol{Y} \in \mathbb{R}^{H\times W}$, where $\boldsymbol{Y}_{i,:} = \boldsymbol{A}^i \times \boldsymbol{Z}_{i,:}$;

Output: $\boldsymbol{y} = \mathrm{vec}(\boldsymbol{Y})$;

Chunkwise algorithm

Complexity: $\mathcal{O}(H^2W) \rightarrow \mathcal{O}(HW)$

Complexity: $\mathcal{O}(HW^2) \rightarrow \mathcal{O}(HW)$

$$L = \begin{bmatrix}
\alpha_{1,1:1} & \alpha_{1,1:2} & \alpha_{1,1:3} & 0 & 0 & 0 & 0 & 0 & 0 \\
\alpha_{1,2:1} & \alpha_{1,2:2} & \alpha_{1,2:3} & 0 & 0 & 0 & 0 & 0 & 0 \\
\alpha_{1,3:1} & \alpha_{1,3:2} & \alpha_{1,3:3} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \alpha_{2,1:1} & \alpha_{2,1:2} & \alpha_{2,1:3} & 0 & 0 & 0 \\
0 & 0 & 0 & \alpha_{2,2:1} & \alpha_{2,2:2} & \alpha_{2,2:3} & 0 & 0 & 0 \\
0 & 0 & 0 & \alpha_{2,3:1} & \alpha_{2,3:2} & \alpha_{2,3:3} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \alpha_{3,1:1} & \alpha_{3,1:2} & \alpha_{3,1:3} \\
0 & 0 & 0 & 0 & 0 & 0 & \alpha_{3,2:1} & \alpha_{3,2:2} & \alpha_{3,2:3} \\
0 & 0 & 0 & 0 & 0 & 0 & \alpha_{3,3:1} & \alpha_{3,3:2} & \alpha_{3,3:3}
\end{bmatrix}
\times
\begin{bmatrix}
\beta_{1:1,1} & 0 & 0 & \beta_{1:2,1} & 0 & 0 & \beta_{1:3,1} & 0 & \boldsymbol{B}^1 \\
0 & \beta_{1:1,2} & 0 & 0 & \beta_{1:2,2} & 0 & 0 & \beta_{1:3,2} & 0 \\
0 & 0 & \beta_{1:1,3} & 0 & 0 & \beta_{1:2,3} & 0 & 0 & \beta_{1:3,3} \\
\beta_{2:1,1} & 0 & 0 & \beta_{2:2,1} & 0 & 0 & \beta_{2:3,1} & 0 & 0 \\
0 & \beta_{2:1,2} & 0 & 0 & \beta_{2:2,2} & 0 & 0 & \beta_{2:3,2} & 0 \\
0 & 0 & \beta_{2:1,3} & 0 & 0 & \beta_{2:2,3} & 0 & 0 & \beta_{2:3,3} \\
\beta_{3:1,1} & 0 & 0 & \beta_{3:2,1} & 0 & 0 & \beta_{3:3,1} & 0 & 0 \\
0 & \beta_{3:1,2} & 0 & 0 & \beta_{3:2,2} & 0 & 0 & \beta_{3:3,2} & 0 \\
0 & 0 & \beta_{3:1,3} & 0 & 0 & \beta_{3:2,3} & 0 & 0 & \beta_{3:3,3}
\end{bmatrix}$$

$\boldsymbol{A}^1$, $\boldsymbol{A}^2$, $\boldsymbol{A}^3$

Theorem 2 (Efficient Matrix Multiplication). *For matrices $\boldsymbol{M}^A, \boldsymbol{M}^B$ defined in Theorem 1,* $\forall \boldsymbol{x} \in \mathbb{R}^{HW}$*, the following equation holds:*

$$\boldsymbol{y} = \boldsymbol{M}^A \times \boldsymbol{M}^B \times \boldsymbol{x} \quad \Leftrightarrow \quad \boldsymbol{Z}_{:,l} = \boldsymbol{B}^l \times \boldsymbol{X}_{:,l}, \ \boldsymbol{Y}_{i,:} = \boldsymbol{A}^i \times \boldsymbol{Z}_{i,:}, \tag{9}$$

where $\boldsymbol{y} \in \mathbb{R}^{HW}$, $\boldsymbol{X} = \mathrm{unvec}(\boldsymbol{x}) \in \mathbb{R}^{H\times W}$, $\boldsymbol{Y} = \mathrm{unvec}(\boldsymbol{y}) \in \mathbb{R}^{H\times W}$, $\boldsymbol{Z} \in \mathbb{R}^{H\times W}$*, and the operator* $\mathrm{vec}(\cdot)$ *vectorizes a matrix by stacking its columns and* $\mathrm{unvec}(\cdot)$ *is its inverse operator.*

Algorithm 1: Efficient Masked Attention Computation.

Input: decay factors α, β of $\boldsymbol{L}$, vector $\boldsymbol{x} \in \mathbb{R}^{HW}$;

1: Compute $\boldsymbol{X} = \mathrm{unvec}(\boldsymbol{x}) \in \mathbb{R}^{H\times W}$;

2: Compute $\boldsymbol{B}^l \in \mathbb{R}^{H\times H}$, where for $l = 1:W$, $[\boldsymbol{B}^l]_{i,k} = \beta_{i:k,l}$;

3: Compute $\boldsymbol{Z} \in \mathbb{R}^{H\times W}$, where $\boldsymbol{Z}_{:,l} = \boldsymbol{B}^l \times \boldsymbol{X}_{:,l}$;

4: Compute $\boldsymbol{A}^i \in \mathbb{R}^{W\times W}$, where for $i = 1:H$, $[\boldsymbol{A}^i]_{j,l} = \alpha_{i,j:l}$;

5: Compute $\boldsymbol{Y} \in \mathbb{R}^{H\times W}$, where $\boldsymbol{Y}_{i,:} = \boldsymbol{A}^i \times \boldsymbol{Z}_{i,:}$;

Output: $\boldsymbol{y} = \mathrm{vec}(\boldsymbol{Y})$;

Chunkwise algorithm

Complexity: $\mathcal{O}(H^2W) \rightarrow \mathcal{O}(HW)$

Complexity: $\mathcal{O}(HW^2) \rightarrow \mathcal{O}(HW)$

$$L = \begin{bmatrix} \alpha_{1,1:1} & \alpha_{1,1:2} & \alpha_{1,1:3} & 0 & 0 & 0 & 0 & 0 & 0 \\ \alpha_{1,2:1} & \alpha_{1,2:2} & \alpha_{1,2:3} & 0 & 0 & 0 & 0 & 0 & 0 \\ \alpha_{1,3:1} & \alpha_{1,3:2} & \alpha_{1,3:3} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \alpha_{2,1:1} & \alpha_{2,1:2} & \alpha_{2,1:3} & 0 & 0 & 0 \\ 0 & 0 & 0 & \alpha_{2,2:1} & \alpha_{2,2:2} & \alpha_{2,2:3} & 0 & 0 & 0 \\ 0 & 0 & 0 & \alpha_{2,3:1} & \alpha_{2,3:2} & \alpha_{2,3:3} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \alpha_{3,1:1} & \alpha_{3,1:2} & \alpha_{3,1:3} \\ 0 & 0 & 0 & 0 & 0 & 0 & \alpha_{3,2:1} & \alpha_{3,2:2} & \alpha_{3,2:3} \\ 0 & 0 & 0 & 0 & 0 & 0 & \alpha_{3,3:1} & \alpha_{3,3:2} & \alpha_{3,3:3} \end{bmatrix} \times \begin{bmatrix} \beta_{1:1,1} & 0 & 0 & \beta_{1:2,1} & 0 & 0 & \beta_{1:3,1} & 0 & \\ 0 & \beta_{1:1,2} & 0 & 0 & \beta_{1:2,2} & 0 & 0 & \beta_{1:3,2} & 0 \\ 0 & 0 & \beta_{1:1,3} & 0 & 0 & \beta_{1:2,3} & 0 & 0 & \beta_{1:3,3} \\ \beta_{2:1,1} & 0 & 0 & \beta_{2:2,1} & 0 & 0 & \beta_{2:3,1} & 0 & 0 \\ 0 & \beta_{2:1,2} & 0 & 0 & \beta_{2:2,2} & 0 & 0 & \beta_{2:3,2} & 0 \\ 0 & 0 & \beta_{2:1,3} & 0 & 0 & \beta_{2:2,3} & 0 & 0 & \beta_{2:3,3} \\ \beta_{3:1,1} & 0 & 0 & \beta_{3:2,1} & 0 & 0 & \beta_{3:3,1} & 0 & 0 \\ 0 & \beta_{3:1,2} & 0 & 0 & \beta_{3:2,2} & 0 & 0 & \beta_{3:3,2} & 0 \\ 0 & 0 & \beta_{3:1,3} & 0 & 0 & \beta_{3:2,3} & 0 & 0 & \beta_{3:3,3} \end{bmatrix}$$

$\boldsymbol{A}^1$, $\boldsymbol{A}^2$, $\boldsymbol{A}^3$, $\boldsymbol{B}^2$

Theorem 2 (Efficient Matrix Multiplication). *For matrices $\boldsymbol{M}^A, \boldsymbol{M}^B$ defined in Theorem 1, $\forall \boldsymbol{x} \in \mathbb{R}^{HW}$, the following equation holds:*

$$\boldsymbol{y} = \boldsymbol{M}^A \times \boldsymbol{M}^B \times \boldsymbol{x} \quad \Leftrightarrow \quad \boldsymbol{Z}_{:,l} = \boldsymbol{B}^l \times \boldsymbol{X}_{:,l}, \ \boldsymbol{Y}_{i,:} = \boldsymbol{A}^i \times \boldsymbol{Z}_{i,:}, \tag{9}$$

where $\boldsymbol{y} \in \mathbb{R}^{HW}$, $\boldsymbol{X} = \mathrm{unvec}(\boldsymbol{x}) \in \mathbb{R}^{H\times W}$, $\boldsymbol{Y} = \mathrm{unvec}(\boldsymbol{y}) \in \mathbb{R}^{H\times W}$, $\boldsymbol{Z} \in \mathbb{R}^{H\times W}$, and the operator $\mathrm{vec}(\cdot)$ vectorizes a matrix by stacking its columns and $\mathrm{unvec}(\cdot)$ is its inverse operator.

Algorithm 1: Efficient Masked Attention Computation.

Input: decay factors α, β of $\boldsymbol{L}$, vector $\boldsymbol{x} \in \mathbb{R}^{HW}$;

1: Compute $\boldsymbol{X} = \mathrm{unvec}(\boldsymbol{x}) \in \mathbb{R}^{H\times W}$;

2: Compute $\boldsymbol{B}^l \in \mathbb{R}^{H\times H}$, where for $l = 1:W$, $[\boldsymbol{B}^l]_{i,k} = \beta_{i:k,l}$;

3: Compute $\boldsymbol{Z} \in \mathbb{R}^{H\times W}$, where $\boldsymbol{Z}_{:,l} = \boldsymbol{B}^l \times \boldsymbol{X}_{:,l}$; — Complexity: $\mathcal{O}(H^2W) \longrightarrow \mathcal{O}(HW)$

4: Compute $\boldsymbol{A}^i \in \mathbb{R}^{W\times W}$, where for $i = 1:H$, $[\boldsymbol{A}^i]_{j,l} = \alpha_{i,j:l}$;

5: Compute $\boldsymbol{Y} \in \mathbb{R}^{H\times W}$, where $\boldsymbol{Y}_{i,:} = \boldsymbol{A}^i \times \boldsymbol{Z}_{i,:}$; — Complexity: $\mathcal{O}(HW^2) \longrightarrow \mathcal{O}(HW)$

Output: $\boldsymbol{y} = \mathrm{vec}(\boldsymbol{Y})$;

Chunkwise algorithm

$$L = \begin{bmatrix} \alpha_{1,1:1} & \alpha_{1,1:2} & \alpha_{1,1:3} & 0 & 0 & 0 & 0 & 0 & 0 \\ \alpha_{1,2:1} & \alpha_{1,2:2} & \alpha_{1,2:3} & 0 & 0 & 0 & 0 & 0 & 0 \\ \alpha_{1,3:1} & \alpha_{1,3:2} & \alpha_{1,3:3} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \alpha_{2,1:1} & \alpha_{2,1:2} & \alpha_{2,1:3} & 0 & 0 & 0 \\ 0 & 0 & 0 & \alpha_{2,2:1} & \alpha_{2,2:2} & \alpha_{2,2:3} & 0 & 0 & 0 \\ 0 & 0 & 0 & \alpha_{2,3:1} & \alpha_{2,3:2} & \alpha_{2,3:3} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \alpha_{3,1:1} & \alpha_{3,1:2} & \alpha_{3,1:3} \\ 0 & 0 & 0 & 0 & 0 & 0 & \alpha_{3,2:1} & \alpha_{3,2:2} & \alpha_{3,2:3} \\ 0 & 0 & 0 & 0 & 0 & 0 & \alpha_{3,3:1} & \alpha_{3,3:2} & \alpha_{3,3:3} \end{bmatrix} \times \begin{bmatrix} \beta_{1:1,1} & 0 & 0 & \beta_{1:2,1} & 0 & 0 & \beta_{1:3,1} & 0 & \\ 0 & \beta_{1:1,2} & 0 & 0 & \beta_{1:2,2} & 0 & 0 & \beta_{1:3,2} & 0 \\ 0 & 0 & \beta_{1:1,3} & 0 & 0 & \beta_{1:2,3} & 0 & 0 & \beta_{1:3,3} \\ \beta_{2:1,1} & 0 & 0 & \beta_{2:2,1} & 0 & 0 & \beta_{2:3,1} & 0 & 0 \\ 0 & \beta_{2:1,2} & 0 & 0 & \beta_{2:2,2} & 0 & 0 & \beta_{2:3,2} & 0 \\ 0 & 0 & \beta_{2:1,3} & 0 & 0 & \beta_{2:2,3} & 0 & 0 & \beta_{2:3,3} \\ \beta_{3:1,1} & 0 & 0 & \beta_{3:2,1} & 0 & 0 & \beta_{3:3,1} & 0 & 0 \\ 0 & \beta_{3:1,2} & 0 & 0 & \beta_{3:2,2} & 0 & 0 & \beta_{3:3,2} & 0 \\ 0 & 0 & \beta_{3:1,3} & 0 & 0 & \beta_{3:2,3} & 0 & 0 & \beta_{3:3,3} \end{bmatrix}$$

$\boldsymbol{A}^1$, $\boldsymbol{A}^2$, $\boldsymbol{A}^3$ (diagonal blocks of the first matrix); $\boldsymbol{B}^3$ (entries $\beta_{1:1,3}, \beta_{1:2,3}, \beta_{1:3,3}, \beta_{2:1,3}, \beta_{2:2,3}, \beta_{2:3,3}, \beta_{3:1,3}, \beta_{3:2,3}, \beta_{3:3,3}$ of the second matrix)

5.2 PPMA: Method (Efficient Computation Theory)

- Efficient Computation of Polyline Path Mask Matrix Multiplication
  - **Naive Computation**: for a rank-N matrix $\boldsymbol{L} \in \mathbb{R}^{N \times N}$ and a vector $\boldsymbol{x} \in \mathbb{R}^{N}$, $\boldsymbol{Lx}$ requires a complexity of $\mathcal{O}(N^2)$
  - **Efficient Computation**: for decomposed polyline path mask, $\boldsymbol{Lx} = \boldsymbol{L}^H \times \boldsymbol{L}^V \times \boldsymbol{x}$ requires a complexity of $\mathcal{O}(N)$

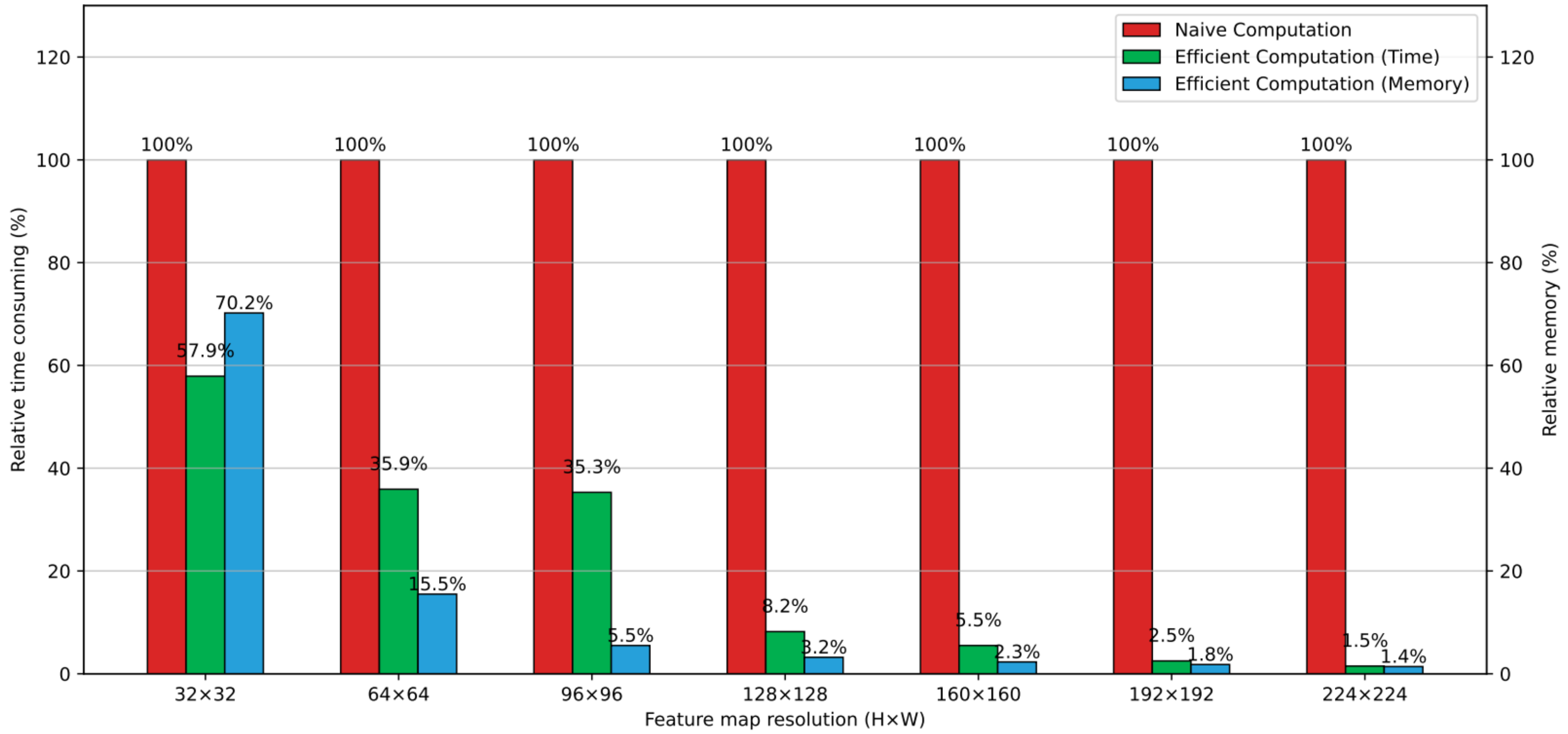


Figure 10: The comparison of the relative time consuming and memory usage between the naive computation and efficient computation (Algorithm 1) of $\boldsymbol{Lx}$.

Remarks. Intuitively, as illustrated in Fig. 4, Algorithm 1 shows that the 2D polyline path scanning on 2D tokens (i.e., $\boldsymbol{L}\boldsymbol{x}$) can be decomposed as the 1D vertical scanning along each column of $\boldsymbol{X}$ (i.e., $\boldsymbol{Z}_{:,l} = \boldsymbol{B}^l \times \boldsymbol{X}_{:,l}$) followed by the 1D horizontal scanning along each row of $\boldsymbol{Z}$ (i.e., $\boldsymbol{Y}_{i,:} = \boldsymbol{A}^i \times \boldsymbol{Z}_{i,:}$). This equivalence offers an intuitive understanding of the physical meaning of the decomposed polyline path mask $\boldsymbol{L} = \boldsymbol{L}^H \boldsymbol{L}^V$ and enables its natural extension to 3D or higher-dimensional tokens, as detailed in Appendix C.2.

Algorithm 1: Efficient Masked Attention Computation.

Input: decay factors α, β of $\boldsymbol{L}$, vector $\boldsymbol{x} \in \mathbb{R}^{HW}$;

1: Compute $\boldsymbol{X} = \text{unvec}(\boldsymbol{x}) \in \mathbb{R}^{H \times W}$;

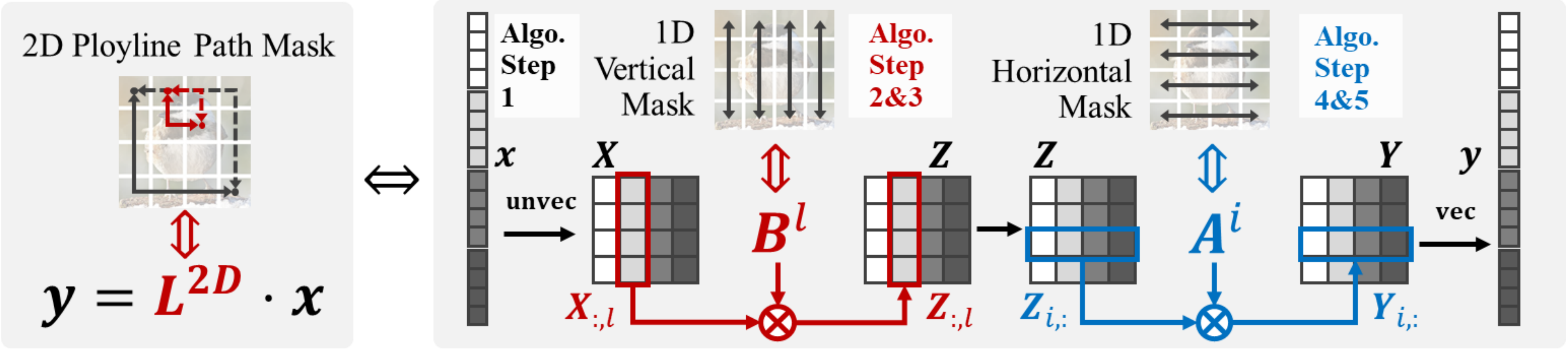
2: Compute $\boldsymbol{B}^l \in \mathbb{R}^{H \times H}$, where for $l = 1:W$, $[\boldsymbol{B}^l]_{i,k} = \beta_{i:k,l}$;

3: Compute $\boldsymbol{Z} \in \mathbb{R}^{H \times W}$, where $\boldsymbol{Z}_{:,l} = \boldsymbol{B}^l \times \boldsymbol{X}_{:,l}$; — Complexity: $\mathcal{O}(H^2W) \rightarrow \mathcal{O}(HW)$

4: Compute $\boldsymbol{A}^i \in \mathbb{R}^{W \times W}$, where for $i = 1:H$, $[\boldsymbol{A}^i]_{j,l} = \alpha_{i,j:l}$;

5: Compute $\boldsymbol{Y} \in \mathbb{R}^{H \times W}$, where $\boldsymbol{Y}_{i,:} = \boldsymbol{A}^i \times \boldsymbol{Z}_{i,:}$; — Complexity: $\mathcal{O}(HW^2) \rightarrow \mathcal{O}(HW)$

Output: $\boldsymbol{y} = \text{vec}(\boldsymbol{Y})$;



Remarks. Intuitively, as illustrated in Fig. 4, Algorithm 1 shows that the 2D polyline path scanning on 2D tokens (i.e., $\boldsymbol{L}\boldsymbol{x}$) can be decomposed as the 1D vertical scanning along each column of $\boldsymbol{X}$ (i.e., $\boldsymbol{Z}_{:,l} = \boldsymbol{B}^l \times \boldsymbol{X}_{:,l}$) followed by the 1D horizontal scanning along each row of $\boldsymbol{Z}$ (i.e., $\boldsymbol{Y}_{i,:} = \boldsymbol{A}^i \times \boldsymbol{Z}_{i,:}$). This equivalence offers an intuitive understanding of the physical meaning of the decomposed polyline path mask $\boldsymbol{L} = \boldsymbol{L}^H \boldsymbol{L}^V$ and enables its natural extension to 3D or higher-dimensional tokens, as detailed in Appendix C.2.

C.2 3D Extension of Polyline Path Mask

Based on the decomposability, we naturally extend the 2D polyline path mask to 3D applications. As illustrated in Fig. 14, the 3D polyline path mask $\boldsymbol{L}^{3D}$ can be decomposed as the multiplication of three 1D structured masks, $\boldsymbol{L}^H \times \boldsymbol{L}^V \times \boldsymbol{L}^D$, representing the horizontal, vertical, and depth scanning masks, respectively. Specifically, for each token pair $(\boldsymbol{x}_{i,j,k}, \boldsymbol{x}_{l,m,n})$ in the 3D grid, the 3D polyline path mask is defined as:

$$\mathcal{L}^{3D}_{(i,j,k),(l,m,n)} = \alpha_{i,j,k:n}\beta_{i,j:m,n}\gamma_{i:l,m,n}, \tag{40}$$

where $\mathcal{L}^{3D}$ is the tensor form of matrix $\boldsymbol{L}^{3D}$, α, β, and γ are the decay factors along the horizontal, vertical, and depth axes, respectively. Compared to the cross-scanning strategy [30], the 3D polyline path scanning strategy better preserves the adjacency relationships of 3D tokens.

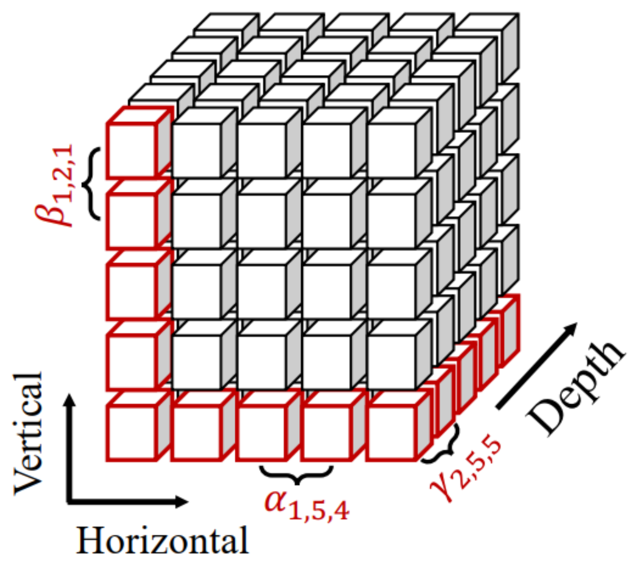


Figure 14: Illustration of the 3D extension of polyline path mask.

5.2 PPMA: Method (Polyline Path Masked Attention)

- Polyline path mask can be integrated into various attention in a plug-and-play manner

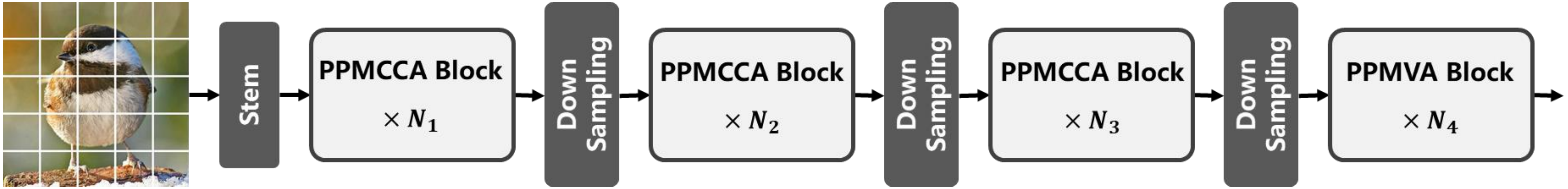
Basic Paradigm: $\mathrm{PPMA}(\boldsymbol{X}) = (\mathrm{Attn}(\boldsymbol{Q},\boldsymbol{K}) \odot \boldsymbol{L}^{2D})\boldsymbol{V} = (\mathrm{Attn}(\boldsymbol{Q},\boldsymbol{K}) \odot \boldsymbol{L})\boldsymbol{V} + \left(\mathrm{Attn}(\boldsymbol{Q},\boldsymbol{K}) \odot \tilde{\boldsymbol{L}}\right)\boldsymbol{V}$

1) Masked **Vanilla self-attention**: $\mathrm{PPMVA}(\boldsymbol{X}) = (\mathrm{softmax}(\boldsymbol{Q}\boldsymbol{K}^\top) \odot \boldsymbol{L}^{2D})\boldsymbol{V}$ — Complexity: $\mathcal{O}(N^2)$

2) Masked **linear attention**: $\mathrm{PPMLA}(\boldsymbol{X}) = (\boldsymbol{Q}\boldsymbol{K}^\top \odot \boldsymbol{L}^{2D})\boldsymbol{V} = \boldsymbol{Q} \star \left(\boldsymbol{L}^{2D} \times (\boldsymbol{K} \star \boldsymbol{V})\right)$ — Complexity: $\mathcal{O}(N)$

3) Masked **criss-cross attention**: $\mathrm{PPMCCA}(\boldsymbol{X}) = \left(\left(\boldsymbol{S}^H \times \boldsymbol{S}^V\right) \odot \boldsymbol{L}^{2D}\right)\boldsymbol{V}$ — Complexity: $\mathcal{O}(N^{\frac{3}{2}})$

4) Masked **decomposed attention** : $\mathrm{PPMDA}(\boldsymbol{X}) = ((\boldsymbol{S}_1 \times \boldsymbol{S}_2) \odot \boldsymbol{L}^{2D})\boldsymbol{V} = \boldsymbol{S}_1 \star \left(\boldsymbol{L}^{2D} \times (\boldsymbol{S}_2 \star \boldsymbol{V})\right)$



5.2 PPMA: Method (Polyline Path Masked Attention)

- Decomposed Criss-Cross Attention[1]

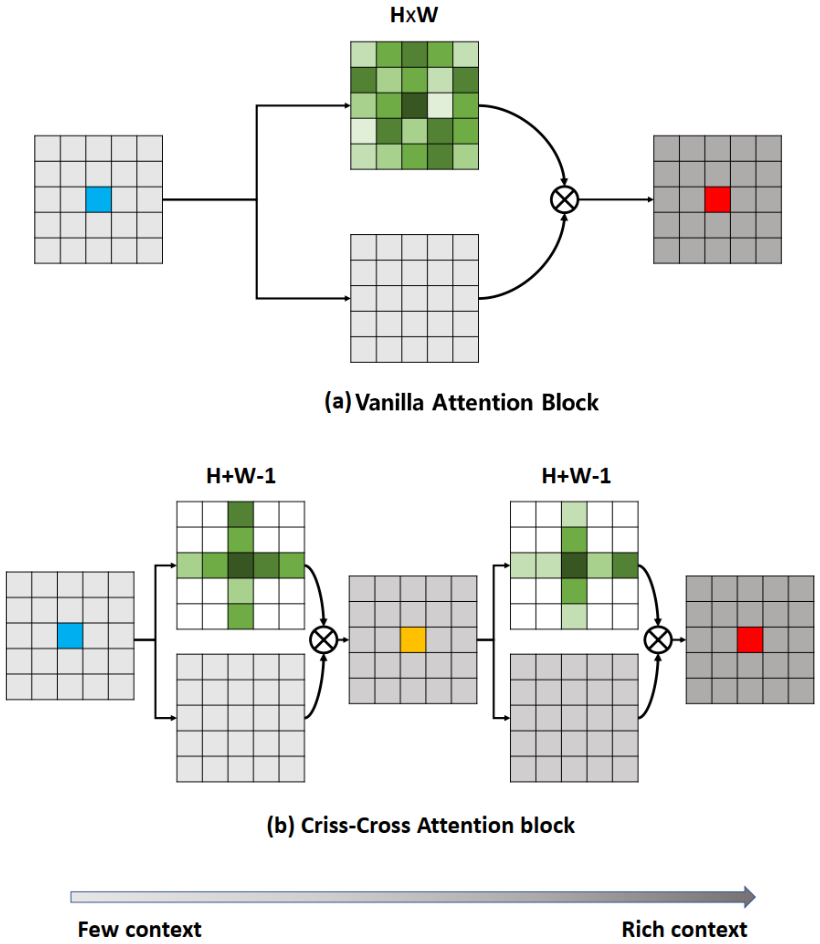
$$S^H = \mathrm{softmax}(Q_H K_H^T) \odot L_H$$

$$S^V = \mathrm{softmax}(Q_V K_V^T) \odot L_V$$

$$Y = 0.5 \times \left(S^V (S^H V)^\top\right)^\top + 0.5 \times \left(S^H (S^V V^\top)^\top\right)$$

3) Masked **criss-cross attention**: $\mathrm{PPMCCA}(X) = \left((S^H \times S^V) \odot L^{2D}\right) V$

$$\begin{aligned}
\left((S^H \times S^V) \odot L\right) V &= \left((S^H \times S^V) \odot (L^H \times L^V)\right) V \\
&= \left((\hat{S}^H \odot \hat{S}^V) \odot (\hat{L}^H \odot \hat{L}^V)\right) V \\
&= \left((\hat{S}^H \odot \hat{L}^H) \odot (\hat{S}^V \odot \hat{L}^V)\right) V \\
&= \left((S^H \odot L^H) \times (S^V \odot L^V)\right) V \\
&= (S^H \odot L^H) \times \left((S^V \odot L^V) \times V\right)
\end{aligned}$$

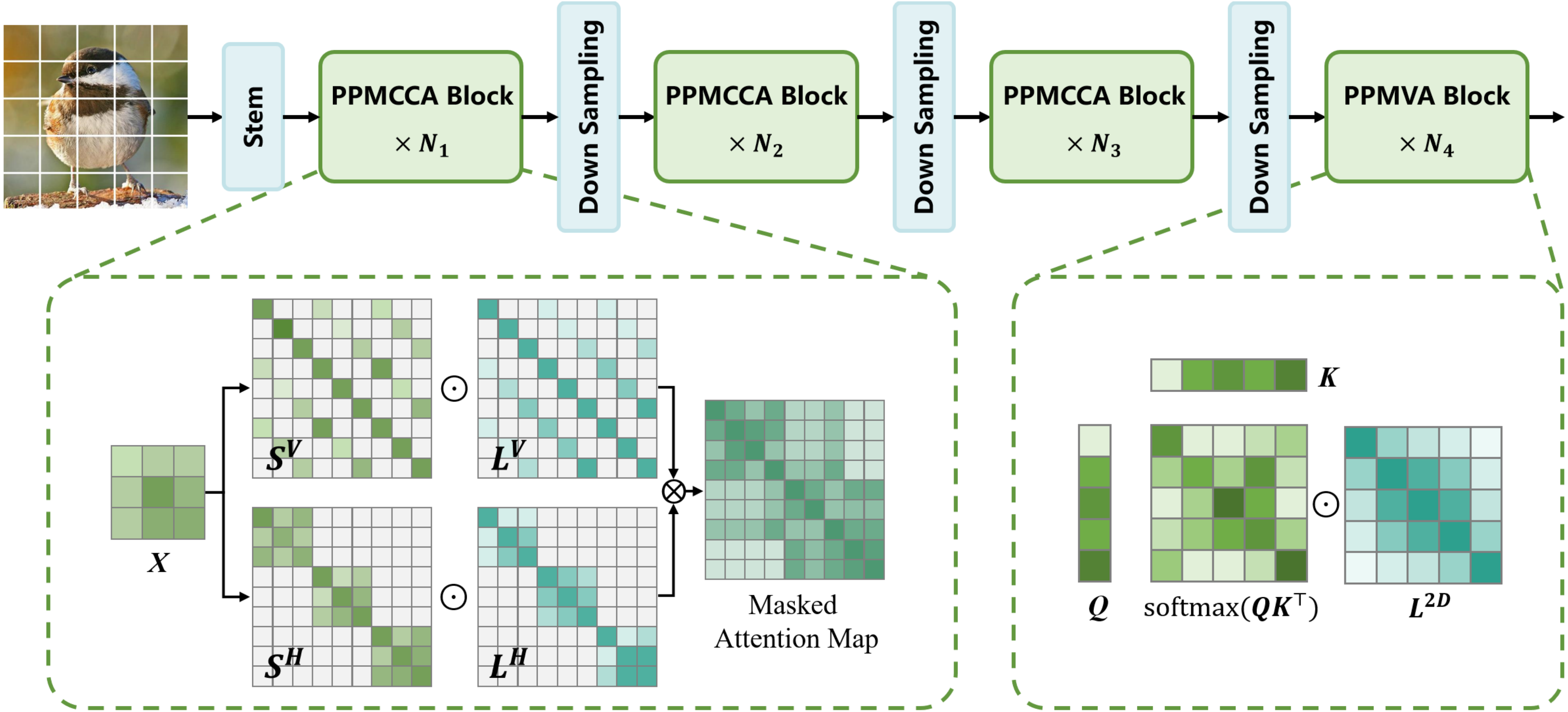


(a) Vanilla Attention Block

(b) Criss-Cross Attention block

[1] Huang Z, Wang X, Huang L, et al. Ccnet: Criss-cross attention for semantic segmentation[C], 2019ICCV & 2020TPAMI. (cited:3646)

5.2 PPMA: Method (Polyline Path Masked Attention)



(a) Polyline Path Masked Criss-Cross Attention

(b) Polyline Path Masked Vanilla Attention

5.3 PPMA: Experiments (Image Classification)

Table 1: Image classification performance on the ImageNet-1K validation set.

Model	Arch.	#Param. (M)	FLOPs (G)	Top-1 (%)
RegNetY-1.6G [34]	CNN	11	1.6	78.0
EffNet-B3 [41]	CNN	12	1.8	81.6
Vim-T [57]	SSM	7	1.5	76.1
MSVMamba-M [36]	SSM	12	1.5	79.8
BiFormer-T [56]	Trans.	13	2.2	81.4
NAT-M [16]	Trans.	20	2.7	81.8
SMT-T [29]	Trans.	12	2.4	82.2
RMT-T [10]	Trans.	14	2.5	82.4
PPMA-T		14	2.7	**82.6**
RegNetY-4G [34]	CNN	21	4.0	80.0
ConvNeXt-T [32]	CNN	29	4.5	82.1
EffNet-B4 [41]	CNN	19	4.2	82.9
VMamba-T [30]	SSM	30	4.9	82.6
2DMamba-T [52]	SSM	31	4.9	82.8
GrootVL-T [48]	SSM	30	4.8	83.4
Spatial-Mamba-T [46]	SSM	27	4.5	83.5
MLLA-T [15]	SSM	25	4.2	83.5
Swin-T [31]	Trans.	29	4.5	82.1
CSWin-T [8]	Trans.	23	4.3	82.7
MambaVision-T2 [17]	Trans.	35	5.1	82.7

Model	Arch.	#Param. (M)	FLOPs (G)	Top-1 (%)
NAT-T [16]	Trans.	28	4.3	83.2
BiFormer-S [56]	Trans.	26	4.5	83.8
RMT-S [10]	Trans.	27	4.5	84.0
PPMA-S		27	4.9	**84.2**
RegNetY-8G [34]	CNN	39	8.0	81.7
ConvNeXt-S [32]	CNN	50	8.7	83.1
EffNet-B5 [41]	CNN	30	9.9	83.6
VMamba-S [30]	SSM	50	8.7	83.6
2DMamba-S [52]	SSM	50	8.8	83.8
GrootVL-S [48]	SSM	51	8.5	84.2
MLLA-S [15]	SSM	43	7.3	84.4
Spatial-Mamba-S [46]	SSM	43	7.1	84.6
Swin-S [31]	Trans.	50	8.7	83.0
NAT-S [16]	Trans.	51	7.8	83.7
CSWin-B [8]	Trans.	78	15.0	84.2
MambaVision-B [17]	Trans.	98	15.0	84.2
BiFormer-B [56]	Trans.	57	9.8	84.3
iFormer-B [37]	Trans.	48	9.4	84.6
RMT-B [10]	Trans.	54	9.7	84.9
PPMA-B		54	10.6	**85.0**

5.3 PPMA: Experiments (Object Detection)

Table 2: Object detection and instance segmentation performance with Mask R-CNN [18] detector on COCO val2017. FLOPs are calculated with input resolution of 1280×800.

Backbone	#Param. (M)	FLOPs (G)	AP^b	AP^b_{50}	AP^b_{75}	AP^m	AP^m_{50}	AP^m_{75}
Vim-T [57]	–	–	45.7	63.9	49.6	39.2	60.9	41.7
MSVMamba-M [36]	32	201	43.8	65.8	47.7	39.9	62.9	42.9
MPViT-XS [25]	30	231	44.2	66.7	48.4	40.4	63.4	43.4
RMT-T [10]	33	218	46.7	68.6	51.6	42.1	65.3	45.2
PPMA-T	33	218	**47.1**	**68.7**	**51.7**	**42.4**	**65.9**	**45.7**
ResNet-50 [19]	44	260	38.2	58.8	41.4	34.7	55.7	37.2
ConvNeXt-T [32]	48	262	44.2	66.6	48.3	40.1	63.3	42.8
MLLA-T [15]	44	255	46.8	69.5	51.5	42.1	66.4	45.0
GrootVL-T [48]	49	265	47.0	69.4	51.5	42.7	66.4	46.0
VMamba-T [30]	50	271	47.3	69.3	52.0	42.7	66.4	45.9
Spatial-Mamba-T [46]	46	216	47.6	69.6	52.3	42.9	66.5	46.2
Swin-T [31]	48	267	43.7	66.6	47.7	39.8	63.3	42.7
CSWin-T [8]	42	279	46.7	68.6	51.3	42.2	65.6	45.4
BiFormer-S [56]	–	–	47.8	69.8	52.3	43.2	66.8	46.5
RMT-S [10]	46	262	48.8	**70.8**	53.4	43.6	**67.4**	**47.3**
PPMA-S	46	263	**49.2**	70.7	**54.0**	**43.8**	**67.4**	47.1
ResNet-101 [19]	63	336	40.4	61.1	44.2	36.4	57.7	38.8
ConvNeXt-S [32]	70	348	45.4	67.9	50.0	41.8	65.2	45.1
GrootVL-S [48]	70	341	48.6	70.3	53.5	43.6	67.5	47.1
VMamba-S [30]	70	349	48.7	70.0	53.4	43.7	67.3	47.0
Spatial-Mamba-S [46]	63	315	49.2	70.8	54.2	44.0	67.9	47.5
MLLA-S [15]	63	319	49.2	71.5	53.9	44.2	68.5	47.2
Swin-S [31]	69	359	45.7	67.9	50.4	41.1	64.9	44.2
CSWin-S [8]	54	342	47.9	70.1	52.6	43.2	67.1	46.2
BiFormer-B [56]	–	–	48.6	70.5	53.8	43.7	67.6	47.1
RMT-B [10]	73	373	50.7	72.0	55.7	45.1	69.2	49.0
PPMA-B	73	374	**51.1**	**72.5**	**55.9**	**45.5**	**69.7**	**49.1**

5.3 PPMA: Experiments (Semantic Segmentation)

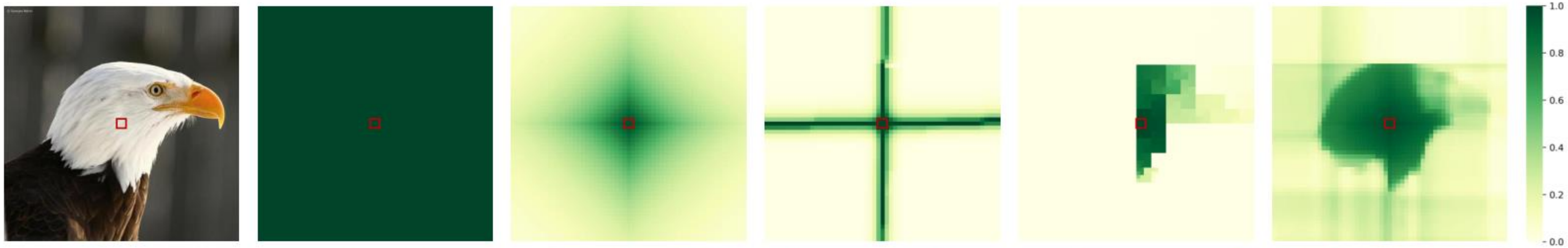
Table 3: Semantic segmentation performance with UPerNet [47] segmentor on ADE20K val set. 'SS' and 'MS' represent single-scale and multi-scale testing, respectively.

Backbone	#Param. (M)	FLOPs (G)	mIoU(%) SS	mIoU(%) MS
LocalVim-T [21]	36	181	43.4	44.4
MSVMamba-M [36]	42	875	45.1	45.4
NAT-M [16]	50	900	45.1	46.4
RMT-T [10]	43	977	48.0	48.8
PPMA-T	43	983	**48.7**	**49.1**
ResNet-50 [19]	67	953	42.1	42.8
ConvNeXt-T [32]	60	939	46.0	46.7
VMamba-T [30]	62	949	48.0	48.8
2DMamba-T [52]	62	950	48.6	49.3
GrootVL-T [48]	60	941	48.5	49.4
Spatial-Mamba-S [46]	57	936	48.6	49.4
Swin-T [31]	60	945	44.4	45.8
MambaVision-T [17]	55	945	46.6	–
NAT-T [16]	58	934	47.1	48.4
CSWin-S [8]	60	959	49.3	50.7

Backbone	#Param. (M)	FLOPs (G)	mIoU(%) SS	mIoU(%) MS
BiFormer-S [56]	–	–	49.8	50.8
RMT-S [10]	56	937	49.8	49.7
PPMA-S	56	984	**51.1**	**52.0**
ResNet-101 [19]	85	1030	42.9	44.0
ConvNeXt-S [32]	82	1027	48.7	49.6
VMamba-S [30]	82	1028	50.6	51.2
Spatial-Mamba-S [46]	73	992	50.6	51.4
GrootVL-S [48]	82	1019	50.7	51.7
Swin-S [31]	81	1039	47.6	49.5
NAT-S [16]	82	1010	48.0	49.5
MambaVision-S [17]	84	1135	48.2	–
CSWin-S [8]	65	1027	50.4	51.5
BiFormer-B [56]	–	–	51.0	51.7
RMT-B [10]	83	1051	52.0	52.1
PPMA-B	83	1137	**52.3**	**53.0**

5.3 PPMA: Experiments (Ablation Study)

- Ablation study on the polyline path mask design



(a) Input Image (b) W/o Mask (c) RMT Decay Mask (d) Cross Scan Mask (e) Hilbert Scan Mask (f) Polyline Path Mask

Figure 6: Illustration of various structured masks.

Table 9: Ablation study of structured mask designs in PPMA-T on ImageNet-1K and ADE20K.

Structured Mask	#Param. (M)	FLOPs (G)	Top-1 (%)	mIoU SS (%)
Baseline (w/o mask)	14.33	2.65	82.28	47.78
+ RMT Decay Mask	14.33	2.65	82.35	48.01
+ Cross Scan Mask	14.34	2.71	82.44	48.14
+ V2H Polyline Path Mask	14.34	2.71	82.44	48.57
+ 2D Polyline Path Mask	14.34	2.71	**82.60**	**48.73**
Shared Decay factors ($\alpha_{i,j} = \beta_{i,j}$)	14.33	2.71	82.37	48.27
Different Decay factors ($\alpha_{i,j} \neq \beta_{i,j}$)	14.34	2.71	**82.60**	**48.73**

5.3 PPMA: Experiments (Visualization)

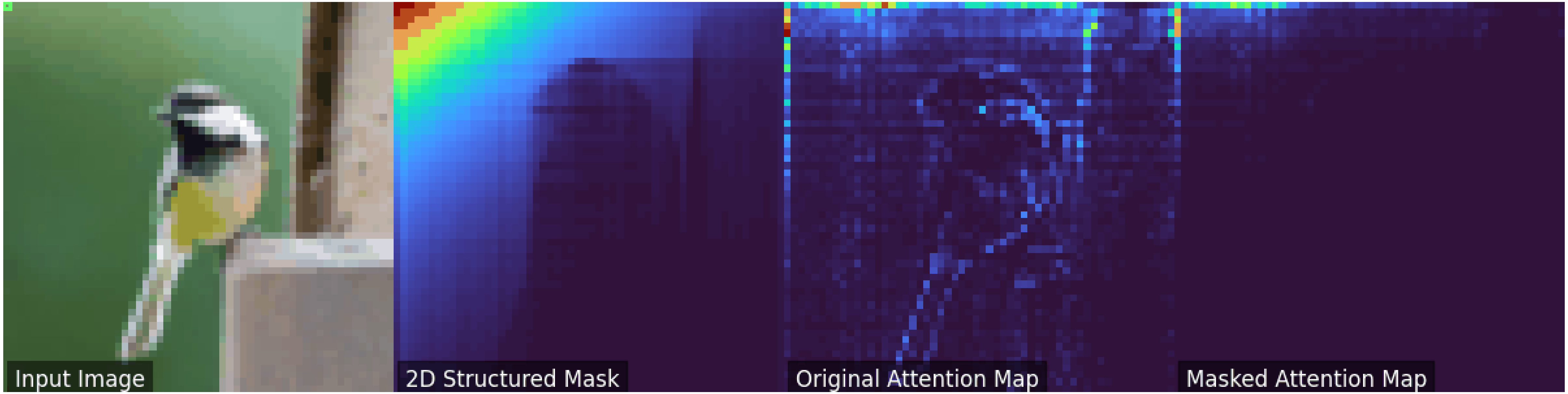
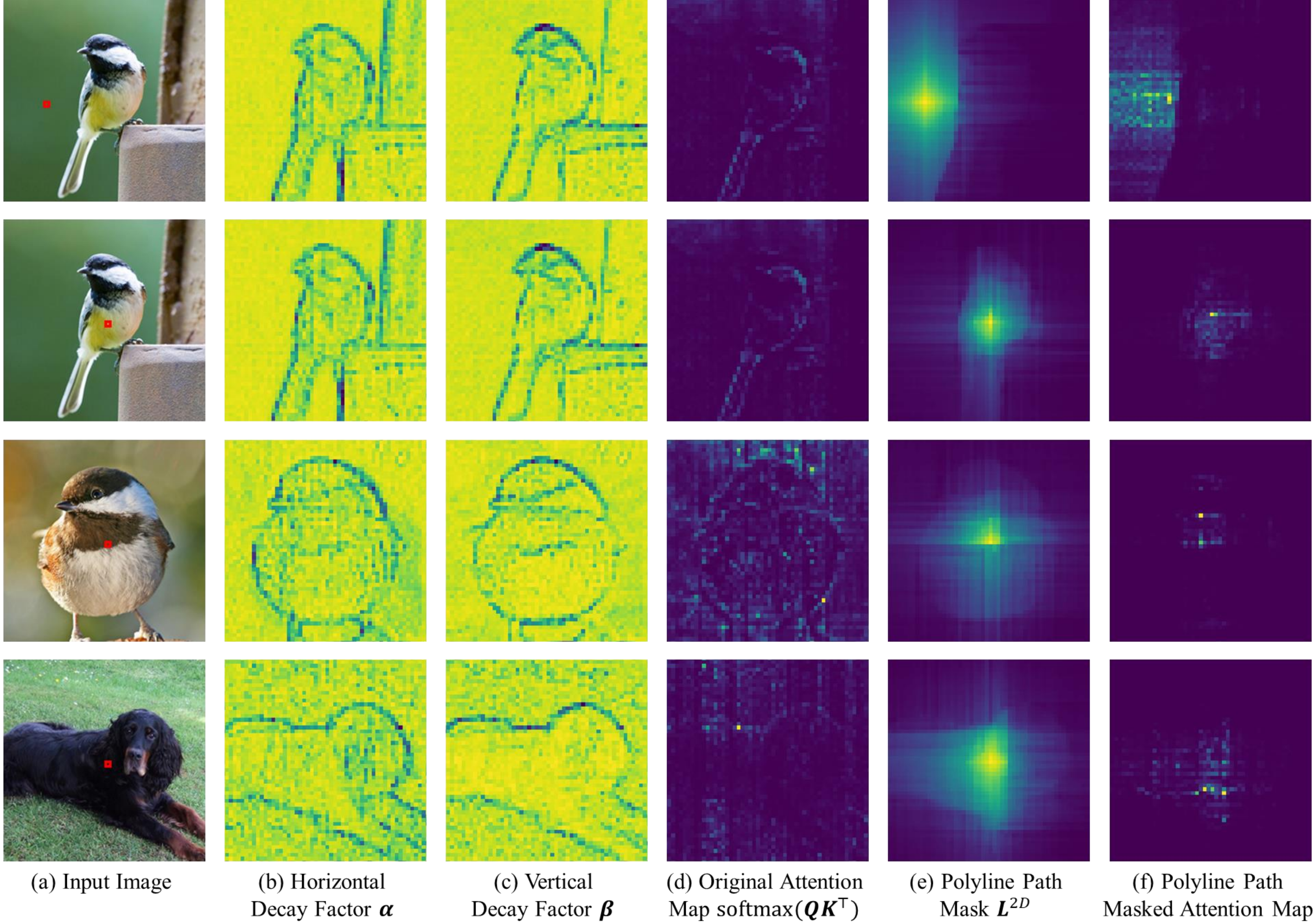
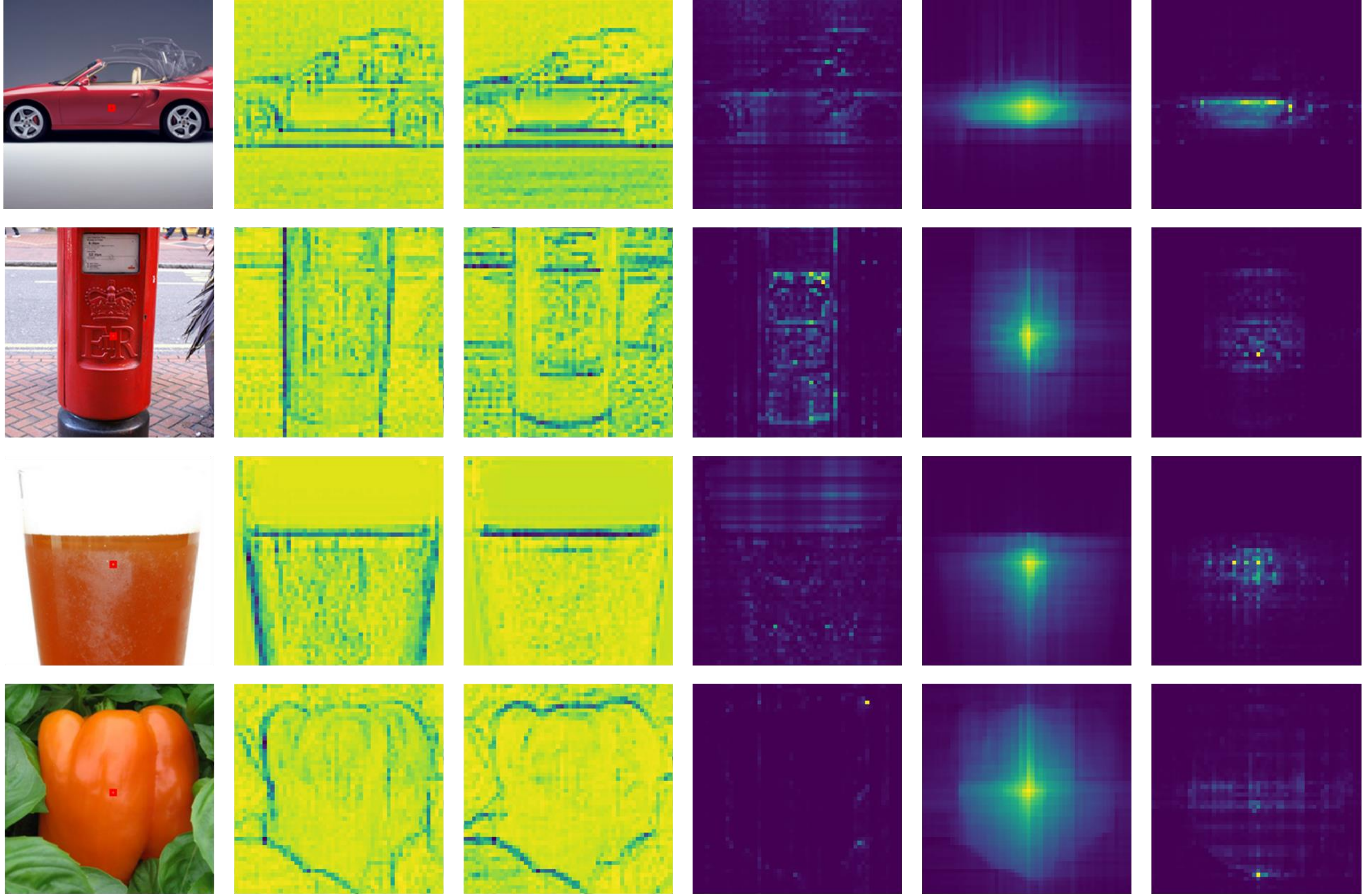


Figure 8: Visualizations of the Polyline Path Masked Attention


(a) Input Image
(b) Horizontal Decay Factor $\boldsymbol{\alpha}$
(c) Vertical Decay Factor $\boldsymbol{\beta}$
(d) Original Attention Map softmax$(\boldsymbol{QK}^\top)$
(e) Polyline Path Mask $\boldsymbol{L}^{2D}$
(f) Polyline Path Masked Attention Map



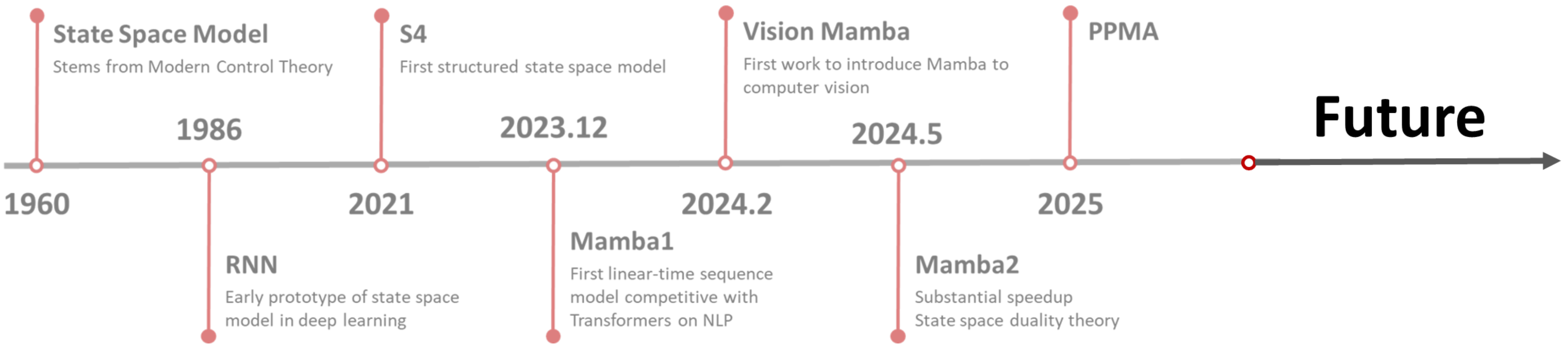
(a) Input Image | (b) Horizontal Decay Factor $\boldsymbol{\alpha}$ | (c) Vertical Decay Factor $\boldsymbol{\beta}$ | (d) Original Attention Map softmax($\boldsymbol{Q}\boldsymbol{K}^\top$) | (e) Polyline Path Mask $\boldsymbol{L}^{2D}$ | (f) Polyline Path Masked Attention Map

What is next for Mamba?

- Stable and scalable linear-time foundational model remains a worthwhile subject
- Hybrid architecture maybe the future

9.2.3 Hybrid Models: Combining SSD Layer with MLP and Attention

Recent and concurrent work (Dao, D. Y. Fu, et al. 2023; De et al. 2024; Glorioso et al. 2024; Lieber et al. 2024) suggests that a hybrid architecture with both SSM layers and attention layers could improve the model quality over that of a Transformer, or a pure SSM (e.g., Mamba) model, especially for in-context learning. We explore the different ways that SSD layers can be combined with attention and MLP to understand the benefits of each. Empirically we find that having around 10% of the total number of layers being attention performs best. Combining SSD layers, attention layers, and MLP also works better than either pure Transformer++ or Mamba-2.



西安交通大學
XI'AN JIAOTONG UNIVERSITY

Thanks！