Supplementary Material for "Mamba-ST: State Space Model for Efficient Style Transfer"

Filippo Botti Alex Ergasti Leonardo Rossi Tomaso Fontanini
Claudio Ferrari Massimo Bertozzi Andrea Prati
University of Parma, Department of Engineering and Architecture
Parma, Italy
{filippo.botti, alex.ergasti, leonardo.rossi, claudio.ferrari2,
massimo.bertozzi, andrea.prati}@unipr.it

1. Style Transfer computation

In this section we formalize the relation between style transfer computed via Transformers cross-attention and our Mamba-ST. In particular, we take inspiration from [1] and [2], where a formalization of the relationship between Mamba and self-attention was first proposed. We start by noting that self-attention can be written as matrix transformation:

$$Y = MX \tag{1}$$

where $M = \text{softmax}(QK^T)$ and $X = V$. Considering Mamba discretized equations:

$$\begin{aligned} h_k &= \bar{A}h_{k-1} + \bar{B}x_k \\ y_k &= Ch_k + Dx_k \end{aligned} \tag{2}$$

following [1], we can represent Eq. (2) as:

$$y_t = C_t \sum_{j=1}^{t} \Big(\prod_{k=j+1}^{t} \bar{A}_k \Big) \bar{B}_j x_j \tag{3}$$

To simplify the calculus, we remove Dx_k since it can be seen as a skip connection multiplied by a scale factor D. We can also define the matrices derivation as:

$$\begin{aligned} C_i &= \text{Lin}_\text{C}(x_i) \\ \Delta_k &= \text{softplus}\big(\text{Lin}_\Delta(x_k)\big) \\ \bar{A}_k &= \exp\Big(\Delta_k A\Big) \\ \bar{B}_j &= \Delta_k \text{Lin}_\text{B}(x_j) \end{aligned} \tag{4}$$

Then, we define $y_i = \sum_{j=1}^{i} y_{i,j}$, which maps the contribution of each input x_j on the output y_i. In particular, $y_{i,j}$ is as follow:

$$y_{i,j} = \underbrace{\text{Lin}_\text{C}(x_i)}_{C} \Bigg(\prod_{k=j+1}^{i} \underbrace{\exp\big(\overbrace{\text{softplus}(\text{Lin}_\Delta(x_k))}^{\Delta_k} A\big)}_{\overline{A}_k} \Bigg) \cdot \underbrace{\overbrace{\text{softplus}(\text{Lin}_\Delta(x_j))}^{\Delta_k} \text{Lin}_\text{B}(x_j)}_{\overline{B}_j} x_j \tag{5}$$

By simply applying exponential property we can rewrite it as:

$$y_{i,j} = \underbrace{\text{Lin}_\text{C}(x_i)}_{C} \Bigg(\underbrace{\exp\Bigg(\sum_{k=j+1}^{i} \overbrace{\text{softplus}(\text{Lin}_\Delta(x_k))}^{\Delta_k} A \Bigg)}_{\overline{A}_k} \Bigg) \cdot \underbrace{\overbrace{\text{softplus}(\text{Lin}_\Delta(x_j))}^{\Delta_k} \text{Lin}_\text{B}(x_j)}_{\overline{B}_j} x_j \tag{6}$$

Furthermore, the two Softplus functions can be approximated summing only over positive values and using a ReLU:

$$y_{i,j} \approx \underbrace{\text{Lin}_\text{C}(x_i)}_{C} \Bigg(\underbrace{\exp \sum_{\substack{k=j+1 \\ \text{Lin}_\Delta(x_k)>0}}^{i} \overbrace{\text{Lin}_\Delta(x_k)}^{\Delta_k} A}_{\overline{A}_k} \Bigg) \underbrace{\overbrace{\text{ReLU}(\text{Lin}_\Delta(x_j))}^{\Delta_k} \text{Lin}_\text{B}(x_j)}_{\overline{B}_j} x_j \tag{7}$$

Algorithm 1 Base VSSM

Require: Patch sequences p
Ensure: Encoded patch p_e

$x \leftarrow \text{Lin}(p)$
$z \leftarrow \text{Lin}(p)$
$x \leftarrow \text{DWConv}(x)$
$x \leftarrow \text{SiLU}(x)$
for d in {scan-directions} **do**
  $B \leftarrow \text{Lin}_\text{B}(x)$
  $C \leftarrow \text{Lin}_\text{C}(x)$
  $\Delta \leftarrow \text{Lin}_\Delta(x)$
  $\bar{A} \leftarrow \Delta \otimes \text{parameter}^A$
  $\bar{B} \leftarrow \Delta \otimes B$
  $y_d \leftarrow \text{SelectiveScan}(\bar{A}, \bar{B}, C)(x)$
end for
for d in {scan-directions} **do** $y \leftarrow y + y_d$
end for
$y \leftarrow \text{LayerNorm}(y)$
$y \leftarrow y * \text{SiLU}(z)$
$y \leftarrow \text{Lin}(y)$

Algorithm 2 ST-VSSM

Require: Content c, Shuffled Style s
Ensure: Stylized patches y

$x, s \leftarrow \text{Lin}(c), \text{Lin}(s)$
$z \leftarrow \text{Lin}(c)$
$x, s \leftarrow \text{DWConv}(x), \text{DWConv}(s)$
$x, s \leftarrow \text{SiLU}(x), \text{SiLU}(s)$
for d in {scan-directions} **do**
  $B \leftarrow \text{Lin}_\text{B}(s)$
  $C \leftarrow \text{Lin}_\text{C}(x)$
  $\Delta \leftarrow \text{Lin}_\Delta(s)$
  $\bar{A} \leftarrow \Delta \otimes \text{parameter}^A$
  $\bar{B} \leftarrow \Delta \otimes B$
  $y_d \leftarrow \text{SelectiveScan}(\bar{A}, \bar{B}, C)(s)$
end for
for d in {scan-directions} **do** $y \leftarrow y + y_d$
end for
$y \leftarrow \text{LayerNorm}(y)$
$y \leftarrow y * \text{SiLU}(z)$
$y \leftarrow \text{Lin}(y)$

Comparison between the base VSSM block (on the left) and our ST-VSSM (on the right) that enables style injection.

Finally, considering the following values:

$$\begin{aligned} Q_i &= \text{Lin}_\text{C}(x_i)) \\ K_j &= \text{ReLU}(\text{Lin}_\Delta(x_j)\text{Lin}_\text{B}(x_j)) \\ H_{i,j} &= \exp \sum_{\substack{k=j+1 \\ \text{Lin}_\Delta(x_k)>0}}^{i} \Big(\text{Lin}_\Delta(x_k)\Big) A\Big) \\ X &= x_j \end{aligned} \tag{8}$$

We obtain that:

$$Y = Q_i H_{i,j} K_j \cdot X \tag{9}$$

which is a matrix transformation like 1. That demonstrates that we can simulate attention inside SSM, by assuming that B, C and Δ play the role of Key, Query and Value from transformers attention, respectively. Additionally, the matrix A is the matrix that modulates the values of the previous patches, because is the one that is multiplied with the states $i - j$ inside the state space equations 2.

Finally, in order to compute style transfer in a similar way of StyTr2 [3], but exploiting Mamba equations, we make the matrix B, Δ, and consequentially A, dependent from the style s, and the matrix C dependent from the content image x, employing a linear projection:

$$B = \text{Lin}_\text{B}(s), \Delta = \text{Lin}_\Delta(s), C = \text{Lin}_\text{C}(x) \tag{10}$$

Additionally, we make the inner state, hence, the matrix $H_{i,j}$ style-dependent. By doing so, the output can be seen as a modulation of the inner state by the matrix C, which is content dependent.

2. VSSM code comparison

In this section we present the two versions of our ST-VSSM algorithm. On the left (alg. 1) is shown the Base VSSM algorithm used inside the two encoders in order to extract features from the images. As it can be seen, all the matrices depend only on the single input x and are obtained with linear projections, as stated in [4]. On the right (alg. 2) the proposed ST-VSSM is shown, which is used to fuse the style and the content information in a single output. The first difference is that ST-VSSM takes as input both style and content information. It is worth noting that the depth-wise convolution shares the same weights for both the inputs. The core of ST-VSSM algorithm relies on how the matrices A, B, C and Δ are computed: as it can be seen, the matrices B and Δ (and consequentially the matrix A) are derived from the style source, while the matrix C is derived from the content source. Moreover, the input of the selective scan is the style source and not the content. As stated, this is justified by the fact that we want to make the state dependent only from the style source, while modulating it with the content information in order to compute the output.

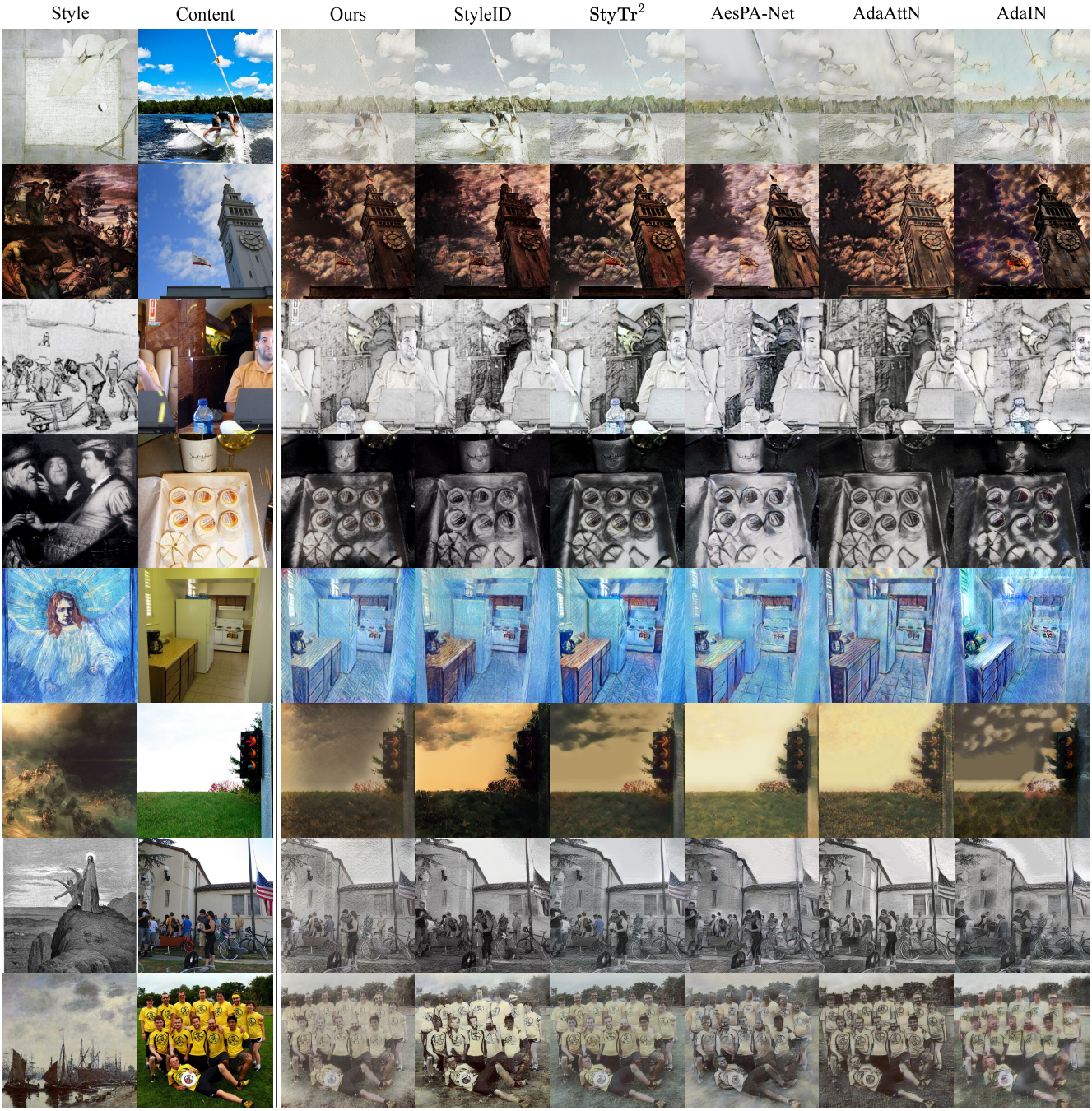

Figure 1. Additional comparisons with the current state-of-the-art models.

3. Results

In Fig. 1 we present additional results of our model compared with several state of the art architectures.

References

[1] Ameen Ali, Itamar Zimerman, and Lior Wolf. The hidden attention of mamba models. *arXiv preprint arXiv:2403.01590*, 2024.

[2] Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. *arXiv preprint arXiv:2405.21060*, 2024.

[3] Yingying Deng, Fan Tang, Weiming Dong, Chongyang Ma, Xingjia Pan, Lei Wang, and Changsheng Xu. Stytr2: Image style transfer with transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11326–11336, 2022.

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