

# All-to-key Attention for Arbitrary Style Transfer

Mingrui Zhu<sup>\*1</sup> Xiao He<sup>\*1</sup> Nannan Wang<sup>†1</sup> Xiaoyu Wang<sup>2</sup> Xinbo Gao<sup>3</sup>

<sup>\*</sup>equal technical contribution <sup>†</sup>corresponding author

<sup>1</sup>State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an, China

<sup>2</sup>School of Computer Science and Technology, University of Science and Technology of China, Hefei, China

<sup>3</sup>Chongqing Key Laboratory of Image Cognition, Chongqing University of Posts and Telecommunications, Chongqing, China

## Abstract

Attention-based arbitrary style transfer studies have shown promising performance in synthesizing vivid local style details. They typically use the all-to-all attention mechanism—each position of content features is fully matched to all positions of style features. However, all-to-all attention tends to generate distorted style patterns and has quadratic complexity, limiting the effectiveness and efficiency of arbitrary style transfer. In this paper, we propose a novel all-to-key attention mechanism—each position of content features is matched to stable key positions of style features—that is more in line with the characteristics of style transfer. Specifically, it integrates two newly proposed attention forms: distributed and progressive attention. Distributed attention assigns attention to key style representations that depict the style distribution of local regions; Progressive attention pays attention from coarse-grained regions to fine-grained key positions. The resultant module, dubbed StyA2K, shows extraordinary performance in preserving the semantic structure and rendering consistent style patterns. Qualitative and quantitative comparisons with state-of-the-art methods demonstrate the superior performance of our approach.

## 1. Introduction

Arbitrary style transfer (AST) is an important computer vision task. It aims to render a natural image (i.e., content image) with the artistic style of an arbitrary painting (i.e., style image), enabling the generated image to imitate any artistic style. There have been notable improvements in feature transformation modules [13, 32, 12, 29, 46, 7, 27, 17, 47], novel architectures [24, 31, 1, 40, 6], and practical objectives [19, 5, 48]. The core of AST is the matching of content features and style features in the feed-forward procedure. Holistic feature distribution matching [13, 24, 17, 47] and locality-aware feature matching [32, 12, 29, 46, 27] are two categories of existing approaches.



Figure 1. Image/Video style transfer results of AdaAttN [27] (second column) and our StyA2K (third column). Adobe Acrobat is recommended to view the videos in the first row.

The attention-based method is the research focus of the locality-aware feature matching category for its capability to capture long-range dependencies. Typically, it establishes a dense correspondence between point-wise tokens of the content and style features via an all-to-all attention mechanism [49]. The transferred feature of each local position is computed as the weighted sum of the local style features of all positions, where the weights are computed by applying a *softmax* function to the normalized dot products' results. Despite its encouraging results, the attention-based AST method suffers from two main predicaments. The first one is the introduction of distorted style patterns and unstable matching effects. As shown in Fig. 1, the image stylization result of AdaAttN is seriously affected by eye patterns, which significantly affects visual perception. Besides, the video stylization result of AdaAttN has an apparent flickering phenomenon, which also affects the visual quality. The second one is the high computational complexity. Handling image features with high spatial resolution and multiple layers with all-to-all attention requires significant computational consumption.

We argue that the inherent limitations of the all-to-all at-

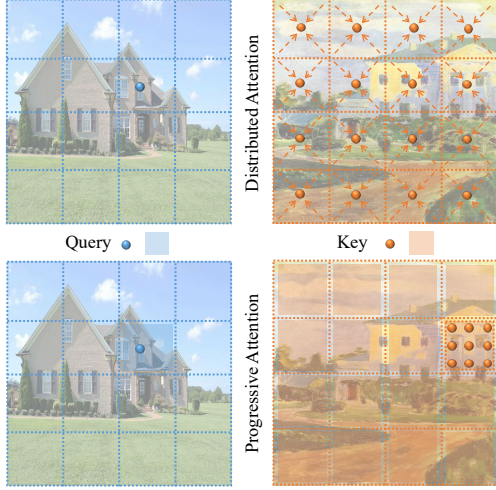


Figure 2. Illustration of A2K attention.

tention mechanism cause the above problems. (1) All-to-all attention has no error tolerance and is sensitive to position variation. It tends to concentrate on the most similar value since *softmax* shows strong exclusiveness in attention score due to exponential computation. A distorted style pattern appears when the most similar key is semantically distinct from the query. For example, the reason for the appearance of irrational eye patterns in AdaAttN’s results is that the all-to-all attention almost exclusively concentrates on the eye patterns in the style image for the positions with edge-like patterns in the content image. Its sensitivity is embodied in that slight changes in query at different positions will lead to complete semantic changes in matched keys. For instance, the object motion and the light change in the video cause severe flickering phenomena between consecutive stylized frames that are frame-by-frame independently generated by AdaAttN. (2) All-to-all attention has quadratic computational complexity since it establishes a fully connected correspondence between queries and keys. Its computational complexity scales quadratically to image size.

How to maintain the advantage of the attention mechanism (enhancing local semantics) and mitigate the disadvantages of all-to-all attention (no error tolerance, unstable, time-consuming)? Our solution is a novel all-to-key attention (A2K) mechanism that matches each query with stable “key” keys. A2k comprises two inventions. (1) It learns distributed keys that depict the style distribution of all local regions of the style features; thus, each query of the content features is matched with these stable and representative keys. (2) It gradually concentrates its attention from coarse-grained regions to fine-grained keys; thus, queries within a local region are matched to the same stable keys within a local region. The former, dubbed as distributed attention (DA), improves matching error tolerance since the most similar key matched is a regional style representation rather than an isolated position. The latter, dubbed as pro-

gressive attention (PA), mitigates the problem of “cannot see the woods for the trees” existing in all-to-all attention and ensures semantic correctness from a more macro perspective. In addition, the keys of DA and PA are stable, so A2K can reduce the sensitivity of position variation and render consistent style patterns. Furthermore, we implement DA and PA in a blocked, sparse fashion, saving considerable computational costs. Finally, an effective and efficient arbitrary style transfer model based on all-to-key attention (StyA2K) arrives. We summarize the main contributions of this paper as the following points:

- We point out the inherent limitations of all-to-all attention and present a novel all-to-key attention mechanism for effective and efficient arbitrary style transfer.
- We propose distributed attention to improve matching error tolerance and progressive attention to ensure semantic correctness, both of which enjoy the stable matching property and save significant computational consumption.
- We conduct extensive experiments to demonstrate the superiority of our approach over state-of-the-art methods in preserving semantic structures and rendering consistent style patterns.

## 2. Related Work

### 2.1. Arbitrary Style Transfer

Neural style transfer has become a hot-spot topic and has attracted wide attention since the pioneering work of Gatys *et al.* [10]. Follow-up studies [16, 20, 21, 34, 3, 25, 23, 9, 11, 35, 39] are devoted to improving the capabilities of neural style transfer algorithms in terms of visual quality, computational efficiency, style diversity, structural consistency, and factor controllability. Arbitrary style transfer [4, 13, 24, 32, 12, 22, 29, 42, 46, 19, 15, 7, 17, 27, 1, 40, 5, 6, 47, 48] has received increasing attention recently, depending on its advantage of using a single feed-forward neural model to transfer the style of an arbitrary image. Existing AST methods can be divided into two main categories: holistic feature distribution matching method and locality-aware feature matching method.

The holistic feature distribution matching method adjusts the holistic content feature distribution to match the style feature distribution. Based on the Gaussian prior assumption, AdaIN [13] and WCT [24] match feature distributions with first- or second-order statistics. The method introduced in [22] makes the transformation matrix learnable. In order to break through the theoretical and practical limitations of first-order and second-order statistics, high-order statistics are introduced in [17] and [47] to perform more exact distribution matching.

By comparison, the locality-aware feature matching method emphasizes the consistency of local semantics when

matching content and style features. StyleSwap [4] replaces content features patch-by-patch with the style features. The deep feature reshuffle module introduced in [12] reshuffles the style features according to the content features via a constrained normalized cross-correlation. Avatar-Net [32] decorates the content features with aligned style features obtained through a relaxed normalized cross-correlation. MST [46] employs clustering to divide style features into multi-modal style representations, which are matched with local content features via graph-based style matching. With the rise of self-attention [37], many studies [29, 42, 7, 27] apply it to AST. SANet [29] directly adopts attention-based feature matching for AST. AdaAttN [27] adopts attention scores to adaptively perform attentive normalization on the content features by calculating the per-point attention-weighted mean and variance of style features. However, these methods neglect the limitations of all-to-all attention and inevitably produce compromised results.

## 2.2. Attention Mechanism

Since being proposed, the attention mechanism has been widely used in the field of natural language processing (NLP) [37, 41] and computer vision [38, 8]. With the development of the Transformer [28, 36, 49, 43, 49], extensive variants of attention mechanisms are introduced to improve its capability and computational efficiency. Discovering the scalability and availability of attention mechanism in computing dense matching, many studies [44, 14, 29, 27] adapt self-attention to match features with distinct distributions. Due to the distinctions in content and style, directly applying all-to-all attention to their matching causes many problems. This work fully considers content-style matching characteristics and explores a novel attention mechanism more suitable for AST tasks.

## 3. Method

### 3.1. Overall Architecture

An overview of our framework is presented in Fig. 3. Given a content image  $I_c$  and a style image  $I_s$ , a pre-trained VGG-19 [33] network with fixed parameters is utilized as an encoder  $E_{nc}$  to extract their multi-scale features. Extracted features at each layer  $l$  can be denoted as:

$$F_c^l = E_{nc}(I_c), F_s^l = E_{nc}(I_s), \quad (1)$$

where  $l \in ReLU\{3.1, 4.1, 5.1\}$ ,  $F_*^l \in \mathbb{R}^{C_l \times H_l \times W_l}$  and  $*$  can be  $c$  or  $s$  representing content and style respectively.

The key ingredient of this framework is the A2K module. It integrates two effective and efficient attention forms (distributed attention and progressive attention as illustrated in Fig. 4) to establish meaningful sparse correspondence between the content feature  $F_c^l$  and the style feature  $F_s^l$  and

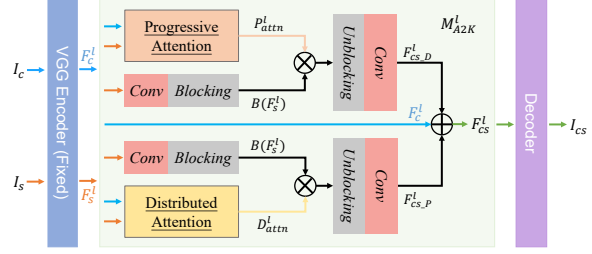


Figure 3. Framework of StyA2K.

thus faithfully synthesize the transferred features  $F_{cs}^l$ :

$$F_{cs}^l = M_{A2K}^l(F_c^l, F_s^l), \quad (2)$$

where  $M_{A2K}^l$  denotes the A2K module and  $l$  indicates that  $M_{A2K}^l$  operates on each layer of the multi-scale features.

Finally we can invert the multi-scale transferred features  $\{F_{cs}^l\}$  to the stylized image  $I_{cs}$  through a decoder  $D_{ec}$ :

$$I_{cs} = D_{ec}(\{F_{cs}^l\}). \quad (3)$$

The decoder implemented in this work follows the setting of [27], which mirrors the encoder and takes the multi-scale transferred features as input.

### 3.2. All-to-key Attention

**Revisit All-to-all Attention in AST.** Attention mechanism in AST is derived from self-attention [37] and is commonly used as a locality-aware feature matching method. Formally, given the content feature  $F_c^l$  and the style feature  $F_s^l$  with the size of  $(H^l, W^l, C^l)$  at layer  $l$ ,  $Q$  (query),  $K$  (key) and  $V$  (value) are formulated as:

$$Q = f(N(F_c^l)), K = g(N(F_s^l)), V = h(F_c^l), \quad (4)$$

where  $f$ ,  $g$ , and  $h$  are learnable convolution layers and  $N(*)$  denotes normalization operation. The attention score  $AttN$  can be calculated as:

$$AttN = softmax(Q \cdot K^T), \quad (5)$$

where  $\cdot$  denotes the dot product. The attention score  $AttN$ , with the size of  $(H \times W, H \times W)$ , is the dense similarity correspondence matrix of all the tokens in  $F_c^l$  and  $F_s^l$ . Since this attention mechanism regards feature vectors of all spatial positions as tokens and establishes full correspondence (as shown in Fig. 5 (b)), it is called all-to-all attention. Although all-to-all attention has been a key factor in many methods [29, 42, 27], its defects have not yet been found and studied. We argue that the all-to-all attention form is inadequate for AST due to its inherent limitations, as analyzed in Introduction 1.

**Distributed Attention.** To alleviate the problem caused by all-to-all attention, distributed attention first learns *distributed* keys that depict the style distribution of all local regions of the style features. Then, each query of the content

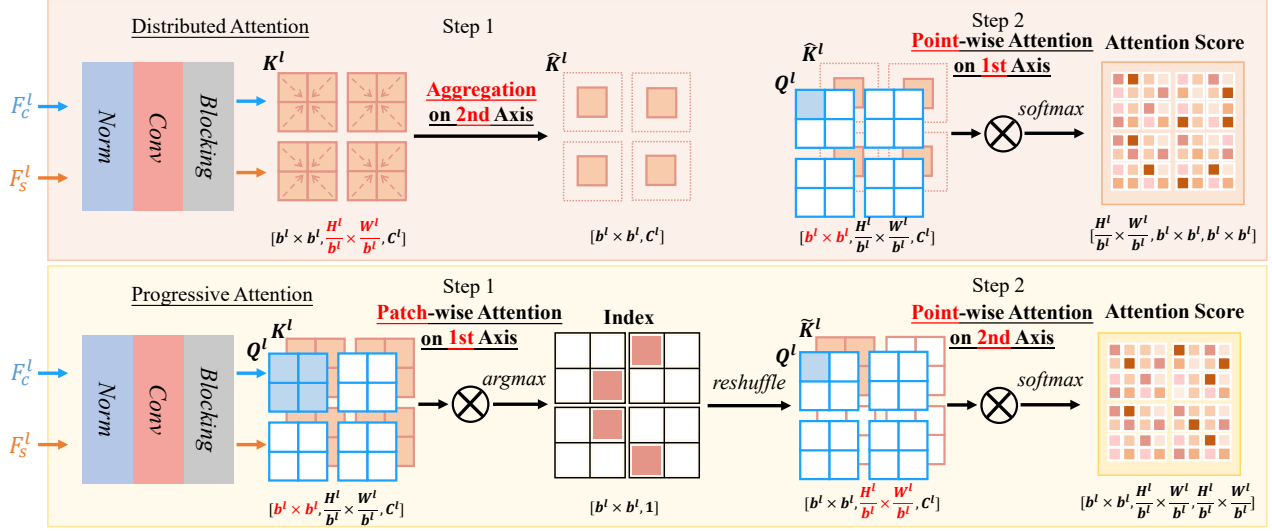


Figure 4. Distributed attention (DA) and progressive attention (PA) are implemented in a blocked, sparse fashion. DA aggregates regional style information to obtain representative keys  $\hat{K}^l$  in the first step and then calculates point-wise attention score along the first (global) axis of  $Q^l$  and  $\hat{K}^l$  in the second step. PA calculates patch-wise similarity index along the first axis of  $Q^l$  and  $K^l$  in the first step to reshuffle  $K^l$  to  $\hat{K}^l$  and then calculates the point-wise attention score on the second (regional) axis of  $Q^l$  and  $\hat{K}^l$  in the second step.

features is matched with these representative keys (as shown in Fig. 5 (e)). Distributed attention can improve matching error tolerance since the matched keys are regional style representations rather than isolated positions. Even if the exclusivity of *softmax* causes the attention to only focus on the most similar key, the most similar key can also reflect the style information of a region so that it will not directly match the isolated style pattern (as shown in Fig. 5 (f)). In addition, since these keys are fixed to depict several local regions, the matching effect of DA has high stability for queries in different positions. Technically, we implement distributed attention in a blocked, sparse fashion, as shown in Fig. 4. The content feature  $F_c^l$  and the style feature  $F_s^l$  with the size of  $(H^l, W^l, C^l)$  at each layer  $l$  are spatially blocked into tensors that stand for  $Q^l$  and  $K^l$  respectively:

$$Q^l = B^l(N^l(F_c^l)), K^l = B^l(N^l(F_s^l)), \quad (6)$$

where  $N^l(*)$  denotes instance normalization and  $B^l(*)$  denotes  $1 \times 1$  Conv-Blocking operation. The shape of  $Q^l$  and  $K^l$  is  $(b^l \times b^l, \frac{H^l}{b^l} \times \frac{W^l}{b^l}, C^l)$ , representing  $b^l \times b^l$  non-overlapping blocks each with the size of  $(\frac{H^l}{b^l}, \frac{W^l}{b^l})$ . Therefore, each block contains  $n = \frac{H^l}{b^l} \times \frac{W^l}{b^l}$  points. We denote the points in each block as  $k_i$ . The first step of distributed attention is regional style aggregation. We calculate the mean of all points in each block as the initial style representation:

$$k_m = \frac{1}{n} \sum_{i=1}^n k_i. \quad (7)$$

Then, we dynamically aggregate all points in a block based on the similarities to the mean point. Assuming the similar-

ity between the  $n$  points and the mean point is  $s \in \mathfrak{R}^n$ , the aggregated regional style representation is given by:

$$k = \frac{1}{n} (k_m + \sum_{i=1}^n \text{sig}(\alpha s_i + \beta) k_i), \quad (8)$$

where  $\alpha$  and  $\beta$  are learnable scalars to scale and shift the similarity and *sig* is a sigmoid function to re-scale the similarity to (0, 1). After aggregation, we obtain  $b^l \times b^l$  new keys, denoted as  $\hat{K}^l$ . Note that we also adopt the same aggregation method for  $B(F_s^l)$  to get new values, expressed as  $\hat{B}(F_s^l)$ . In the second step, distributed attention  $D^l$  calculates point-wise attention on the first axis of  $Q^l$  and  $\hat{K}^l$ :

$$D_{attn}^l = D^l(Q^l, \hat{K}^l). \quad (9)$$

The attention score matrix  $D_{attn}^l$ , with the size of  $(\frac{H^l}{b^l} \times \frac{W^l}{b^l}, b^l \times b^l, b^l \times b^l)$ , stores the similarity correspondence of point-wise tokens across  $b^l \times b^l$  blocks and indexed in range  $\frac{H^l}{b^l} \times \frac{W^l}{b^l}$ . Note that attention along a specific axis of  $Q^l$  and  $K^l$  can be realized straightforwardly by einsum operation, which most deep learning frameworks have implemented. In addition, inspired by the multi-head attention in transformer [37], we use multiple heads to split the tensors along channel dimensions and project the divided tensors into different spaces for attention calculation.

**Progressive Attention.** All-to-all attention focuses directly on a specific position. PA *progressively* concentrates its attention from the coarse-grained region to the fine-grained position (as shown in Fig. 5 (g) (h) (i)). Paying attention to the coarse-grained (as shown in Fig. 5 (g)) in the



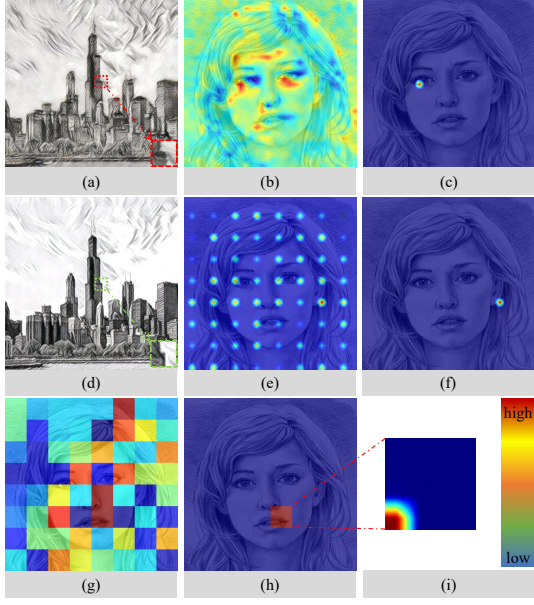


Figure 5. Visualization of attention distribution. (a) Stylized result of AdaAttN. (b) and (c) All-to-all attention distribution before and after *softmax*. (d) Stylized result of StyA2K. (e), (f) Distributed attention distribution before and after *softmax*. (g), (h) Progressive attention distribution on the coarse-grained region before and after *argmax*. (i) Progressive attention distribution on the fine-grained position after *softmax*.

first step contributes to matching style patterns on a larger scale; thus, the attention after *argmax* (as shown in Fig. 5 (h)) can concentrate on a coarse style pattern with more similar semantics. Fine-grained positions can be further located via point-wise attention within this coarse-grained region, as shown in Fig. 5. In addition, since queries within a local region are matched to the same keys within a local region, their transferred features also have regional stability. The technical implementation of PA is shown in Fig. 4.  $Q^l$  and  $K^l$  are obtained similarly as distributed attention but using a different  $1 \times 1$  Conv. The first step of progressive attention is implemented as patch-wise attention along the first axis, which takes a block region as a token instead of a specific position. Specifically, we use *argmax* to match only the most similar coarse-grained region, and therefore, the output of this step is the index matrix that stores sparse indices of patch-wise tokens across  $b^l \times b^l$  blocks:

$$P_{idx}^l = \text{argmax}(P_1^l(Q^l, K^l)), \quad (10)$$

where  $P_1^l$  denotes the first step of PA. With  $P_{idx}^l$ , we can reshuffle the tokens of  $K^l$  and  $B(F_s^l)$  to semantically matching the spatial arrangement of the tokens of  $Q^l$ :

$$\begin{aligned} \tilde{K}^l &= \text{reshuffle}(K^l, P_{idx}^l), \\ \tilde{B}(F_s^l) &= \text{reshuffle}(B(F_s^l), P_{idx}^l), \end{aligned} \quad (11)$$

where  $\text{reshuffle}(*, *)$  denotes the reshuffle operation.

The second step of progressive attention  $P_2^l$  is implemented as regional attention, where tokens attend to their neighbors within non-overlapped blocks. Attention score in this step is calculated along the second axis of  $Q^l$  and  $\tilde{K}^l$ :

$$P_{attn}^l = P_2^l(Q^l, \tilde{K}^l). \quad (12)$$

The attention score matrix  $P_{attn}^l$ , with the size of  $(b^l \times b^l, \frac{H^l}{b^l} \times \frac{W^l}{b^l}, \frac{H^l}{b^l} \times \frac{W^l}{b^l})$ , stores the sparse similarity correspondence of point-wise tokens within blocks with the size of  $\frac{H^l}{b^l} \times \frac{W^l}{b^l}$  indexed in the range  $b^l \times b^l$ . Both steps of progressive attention can be implemented with einsum.

**Feature Transformation.** With the output attention score, the feature transformation can be performed by:

$$\begin{aligned} F_{cs,D}^l &= U(D_{attn}^l \cdot \hat{B}(F_s^l)), \\ F_{cs,P}^l &= U(P_{attn}^l \cdot \tilde{B}(F_s^l)), \end{aligned} \quad (13)$$

where  $U(*)$  denotes the Unblocking-Conv operation,  $\cdot$  represents the dot product between a specific axis of two tensors which can be implemented with einsum. The transferred feature at each layer  $l$  can be eventually obtained by:

$$F_{cs}^l = F_{cs,D}^l + F_{cs,P}^l + F_c^l, \quad (14)$$

where  $F_{cs,D}^l$ ,  $F_{cs,P}^l$ , and  $F_c^l$  are the output of the distributed attention path, the output of the progressive attention path, and the content feature, respectively.

**Complexity Analysis.** The computational complexity of A2K is:

$$\begin{aligned} \Omega &= \Omega_{D_1} + \Omega_{D_2} + \Omega_{P_1} + \Omega_{P_2} \\ &= [5 + (b)^2 + (b)^2 + \frac{H}{b} \frac{W}{b}]HWC, \end{aligned} \quad (15)$$

which saves  $\mathcal{O}(\sqrt{HW})$  computational consumption with respect to image size  $HW$  when  $b^2 \approx \frac{H}{b} \frac{W}{b}$ .

### 3.3. Loss Function

The loss function for training the model consists of two terms. One of them is the style loss  $\mathcal{L}_{gs}$  followed by [13], which penalizes the Euclidean distances of mean  $\mu$  and standard deviation  $\sigma$  between stylized image and style image in VGG feature space to ensure global stylization effect:

$$\begin{aligned} \mathcal{L}_{gs} &= \sum_{l=1}^4 \|\mu(E_{nc}^l(I_{cs})) - \mu(F_s^l)\|_2 \\ &\quad + \sum_{l=1}^4 \|\sigma(E_{nc}^l(I_{cs})) - \sigma(F_s^l)\|_2, \end{aligned} \quad (16)$$

where  $E_{nc}^l(*)$  denotes feature extracted from the  $l$ th layer of the pre-trained VGG encoder. The second term is an

attention-based feature matching loss that penalizes the Euclidean distance between the transferred features of the A2K module and the features of the stylized image:

$$\mathcal{L}_{A2K^*} = \sum_{l=2}^5 \|E_{nc}^l(I_{cs}) - M_{A2K^*}^l(F_c^l, F_s^l)\|_2, \quad (17)$$

where  $M_{A2K^*}^l$  denotes a non-parametric version that removes the learnable  $1 \times 1$  Conv because the supervision signal should be deterministic. The full loss is:

$$\mathcal{L} = \lambda_1 \mathcal{L}_{gs} + \lambda_2 \mathcal{L}_{A2K^*}. \quad (18)$$

We empirically set  $\lambda_1$  and  $\lambda_2$  as 10 and 1.25. See supplementary material for a detailed analysis.

## 4. Experiments

### 4.1. Implementing Details

We use images from MS-COCO [26] as content and images from WikiArt [30] as style to train our model. The head number is set to 8. The batch size is set to 8, and the training lasts for five epochs (400K iterations) on a single NVIDIA GeForce RTX 3090 GPU. Adam [18] with momentum parameters  $\beta_1 = 0.9$  and  $\beta_2 = 0.999$  is used for optimization. The learning rate is set to  $2e-4$  for the first two epochs and linearly decayed to 0 for the rest three epochs. Images are randomly cropped to  $256 \times 256$  during training and loaded with  $512 \times 512$  during inference. To ensure  $b^l \times b^l \approx \frac{H^l}{b^l} \times \frac{W^l}{b^l}$ , the block size  $b$  is set to 16, 8, 8, and 4 for *ReLU2\_1*, *3\_1*, *4\_1* and *5\_1* layers, respectively.

### 4.2. Comparison with Prior Arts

To validate the effectiveness of the proposed method, we compare it with six previous state-of-the-art AST methods: StyTr<sup>2</sup> [6], AdaAttN [27], SANet [29], MST [46], Avatar-Net [32], and AdaIN [13]. Among them, SANet and AdaAttN are typically all-to-all attention-based methods. We obtain the results of the comparison methods by running their official code with their default configurations.

**Qualitative Comparisons.** We show the visual comparisons between our model and other AST methods in Fig. 5. AdaIN adjusts the holistic feature distribution and has no local content awareness; therefore, its results can only roughly transfer the holistic style. Avatar-Net utilizes a relaxed normalized cross-correlation to fulfill locality-aware feature matching so that its results partially maintain semantic consistency. The results are significantly distorted, though, because Avatar-Net only matches the most similar style pattern. MST clusters the complex style distribution into sub-style components and manipulates the features by graph-based patch matching. Despite its promising results, distorted structures still exist. Additionally, it heavily relies on the effect of clustering. SANet adopts an attention

mechanism to match style features for local content features attentively. AdaAttN combines attention mechanism and adaptive instance normalization to integrate the advantage of locality-aware and holistic feature matching. Their results show fine local style details. However, due to the limitation of all-to-all attention, their results contain many unreasonable, inconsistent style patterns (2nd, 3rd, and 5th row). StyTr<sup>2</sup> adapts transformer architecture to AST and achieves cutting-edge performance. Their results still have unreasonable textures (2nd, 4th, and 6th row). Our proposed StyA2K outperforms other methods in maintaining semantic structures and generating consistent style patterns.

**Quantitative Comparisons.** Following [6], we compute the average content loss between the generated results and input content images and the average style loss between the generated results and input style images to measure how well the input content and style are preserved. The smaller the value is, the better the input content/style is preserved. In addition, we adopt the Learned Perceptual Image Patch Similarity (LPIPS) [45] metric to calculate the quality difference between the generated and input content images. A lower score indirectly indicates better quality of generated images. We randomly selected 15 content and 20 style images to generate 300 stylized images. Table 1 shows the corresponding quantitative results. StyA2K shows superior performance on content loss and LPIPS score, which indicates its superiority in preserving semantic structures and rendering consistent style patterns. The style loss of StyA2K’s results is slightly higher than SANet but lower than other methods. Therefore, our results can significantly improve the structure and texture consistency while achieving comparable style performance.

**User Study.** We conduct a user study to evaluate human preference on the results generated by different methods. We reuse the images in the quantitative comparisons and randomly sample 20 groups. Each group consists of a content image, a style image, and seven stylized images generated by different methods. The order of stylized images in each group is shuffled. We ask participants to select their favorite stylized image in each group from three perspectives: content preservation, stylization effect, and overall preference. We collect 1600 votes for each view from 80 participants. Table 1 shows the statistics of the votes, which indicate that the results generated by our method are more appealing than other methods.

**Running time Comparison.** We report the average inference time of StyA2K and other AST methods under  $512 \times 512$  resolution in Table 1. StyA2K achieves 70 FPS at  $512 \times 512$  resolution, which can run in real time. StyA2K is faster than AdaAttN. The reason why StyA2K is slower than SANet is that SANet only performs feature transformation on two layers of VGG features.

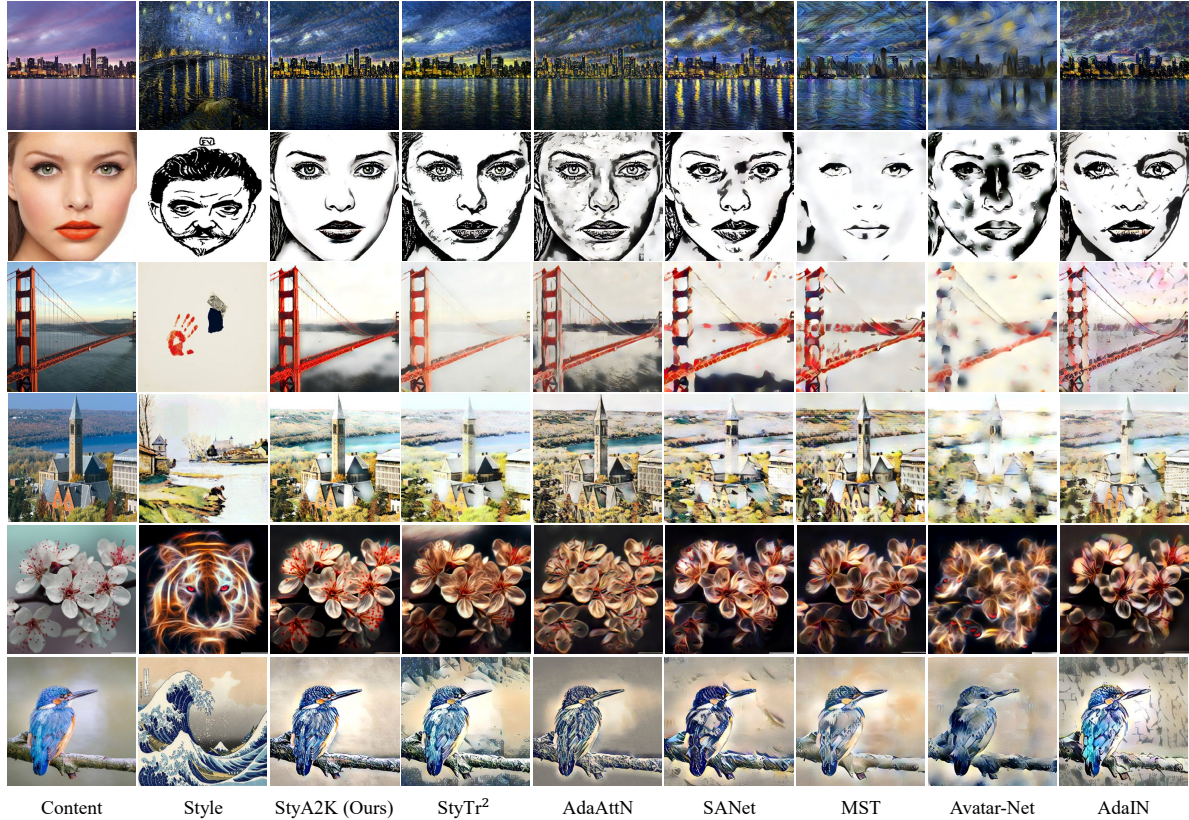


Figure 6. Visual comparisons among different AST methods. Zoom in for a better view.

Table 1. Statics of quantitative comparison, user study, and inference time. The best results are in **bold** and the second best results are marked with an underline.

Method		StyA2K (Ours)	StyTr <sup>2</sup>	AdaAttN	SANet	MST	Avatar-Net	AdaIN
Evaluation Metrics	Content Loss ↓	<b>0.55</b>	0.83	1.00	0.96	<u>0.77</u>	1.26	0.91
	Style Loss ↓	<u>1.04</u>	1.20	1.24	<b>0.99</b>	2.65	3.96	1.16
	LPIPS ↓	<b>0.52</b>	0.57	0.59	0.63	<u>0.57</u>	0.64	0.62
User Study	Content Per. ↑	<b>52.69</b>	9.81	<u>15.56</u>	7.12	<u>8.19</u>	2.25	4.38
	Style Per. ↑	<b>29.06</b>	14.81	12.38	<u>16.13</u>	11.31	9.13	7.19
	Overall Per. ↑	<b>36.06</b>	<u>15.81</u>	14.50	11.44	11.63	3.56	7.00
Inference Time (Sec./Image) ↓		0.0143	0.1004	0.0269	<u>0.0041</u>	1.3906	0.3348	<b>0.0038</b>

### 4.3. Ablation Study

We conduct an ablation study to verify the effectiveness of the key ingredients in our method. Visual and quantitative results are shown in Fig. 7 and Table 2, respectively.

**Effect of All-to-key Attention.** To demonstrate the superiority of our proposed all-to-key attention over all-to-all attention, we replace the all-to-key attention in our full model with all-to-all attention and observe the changes in visual quality and evaluation scores. As shown in Fig. 7, the model using all-to-all attention produces distorted and inconsistent style patterns in its result. The evaluation scores of all-to-all attention’s results are all worse than all-to-key attention, as shown in Table 2. The results suggest that all-to-key attention alleviates the defects caused by all-to-all

attention and thus achieves superior performance.

**Effect of DA and PA.** To validate the respective effectiveness of DA and PA, we set up three variants: 1) full model without DA, 2) full model without PA, and 3) full model without the first step of PA. The results of these variants are shown in 7. The model without DA can match correct style patterns with high semantic similarity but lack in rendering spatial consistent style patterns. The model without PA has high style consistency in local regions but produces patterns with incorrect semantics. When removing the first step, PA directly performs regional attention within the same location. In other words, it directly introduces the content semantics from the reference style. Therefore, its result is mixed with the spatially copied style patterns, as shown in 7, and its results obtain the lowest style loss, as



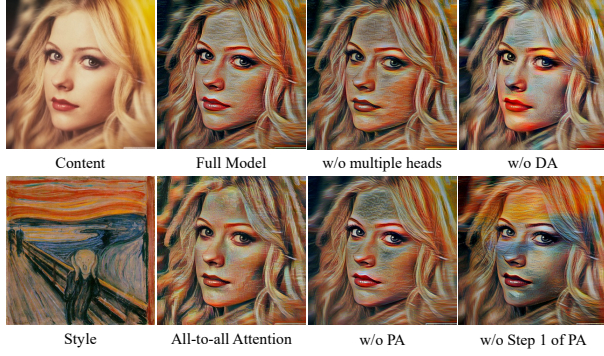


Figure 7. Ablation study on the key ingredients of StyA2K.

Table 2. Ablation study on the key ingredients of the proposed method. The best results are in **bold** and the second best results are marked with an underline.

Model	Content Loss ↓	Style Loss ↓	LPIS ↓
All-to-all Attention	0.73	1.16	0.56
w/o DA	0.59	1.48	<b>0.51</b>
w/o PA	<u>0.58</u>	<u>1.00</u>	0.59
w/o Step 1 of PA	0.65	<b>0.73</b>	0.56
w/o multiple heads	0.65	1.05	0.54
Full Model	<b>0.55</b>	1.04	<u>0.52</u>

shown in Table 2. However, introducing content semantics from the reference style image deviates from the setting of the style transfer task. In addition, almost all the average content loss and the average LPIPS score increase when removing one of the three components. Therefore, we can conclude that all three ingredients are critical to the final effect of the model.

**Effect of Multiple Heads.** The removal of multiple heads (i.e., using only one head) leads to slight visual quality degradation and loss increase, as shown in Fig. 7 and Table 2. We can conclude that multiple heads help to improve the final performance of the proposed method.

#### 4.4. Multi-style Transfer

Following previous studies [7, 29, 27], StyA2K can achieve style interpolation between different styles and accomplish multi-style transfer that integrates multiple styles in one output image, as shown in Fig. 8. These results demonstrate the flexibility of StyA2K.

#### 4.5. Video Style Transfer

Since A2K has excellent stability to position changes (such as object motion and light changes), we can directly apply it to video style transfer. We do not add any mechanism based on the video prior, do not conduct any training on video data, and independently process each frame. Fig. 1 shows a stylized video by our method. See supplementary material for more results. We also directly apply AdaAttN (*softmax* version) to video style transfer. Through comparison, we can find that our method has almost no flicker

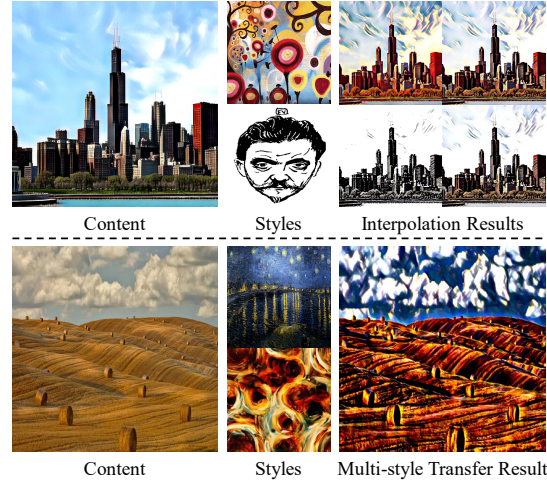


Figure 8. Results of style interpolation and multi-style transfer.

Table 3. Optical flow error of AdaAttN and StyA2K. Lower values indicate better temporal consistency.

Model	Style1	Style2	Style3	Style4	Style5	Mean
AdaAttN	3.03	3.46	6.00	3.78	8.45	4.94
StyA2K	2.58	2.56	4.99	3.21	7.23	4.24

phenomenon and shows high temporal consistency. We calculate the optical flow error [2] on five stylized video clips to quantitatively verify the stability of our method. As shown in Tab. 3, StyA2K has a lower optical flow error than AdaAttN in all five stylized video clips. Note that we have no intention to propose a SOTA video style transfer method because it requires introducing other technologies, such as temporal consistency constraint. We directly apply StyA2K to video style transfer to verify its stability.

## 5. Conclusion and Limitation

This paper proposes a novel all-to-key attention mechanism to achieve effective and efficient arbitrary style transfer. It integrates two inventions: distributed attention and progressive attention. Distributed attention assigns attention to key style representations. Progressive attention pays attention from coarse to fine. They jointly promote maintaining semantic structures and synthesizing spatially consistent style texture. Extensive experiments verify the superiority of our method.

**Limitation.** In this study, we are committed to solving the problems caused by all-to-all attention and improving the semantic structure preservation, style pattern consistency, and stability of style transfer. However, we have not explored the aspect of improving style expressiveness. Recently, contrastive learning has shown strong performance in learning style representation [48]. Therefore, we will explore the feasibility of combining all-to-key attention with contrastive learning in our future work to improve the style expression of the method.



## References

- [1] Jie An, Siyu Huang, Yibing Song, Dejing Dou, Wei Liu, and Jiebo Luo. Artflow: Unbiased image style transfer via reversible neural flows. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 862–871, 2021. 1, 2
- [2] Dongdong Chen, Jing Liao, Lu Yuan, Nenghai Yu, and Gang Hua. Coherent online video style transfer. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1105–1114, 2017. 8
- [3] Dongdong Chen, Lu Yuan, Jing Liao, Nenghai Yu, and Gang Hua. Stylebank: An explicit representation for neural image style transfer. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1897–1906, 2017. 2
- [4] Tian Qi Chen and Mark Schmidt. Fast patch-based style transfer of arbitrary style. *arXiv preprint arXiv:1612.04337*, 2016. 2, 3
- [5] Jiaxin Cheng, Ayush Jaiswal, Yue Wu, Pradeep Natarajan, and Prem Natarajan. Style-aware normalized loss for improving arbitrary style transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 134–143, 2021. 1, 2
- [6] 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. 1, 2, 6
- [7] Yingying Deng, Fan Tang, Weiming Dong, Wen Sun, Feiyue Huang, and Changsheng Xu. Arbitrary style transfer via multi-adaptation network. In *Proceedings of the 28th ACM international conference on multimedia*, pages 2719–2727, 2020. 1, 2, 3, 8
- [8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 3
- [9] Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. A learned representation for artistic style. *arXiv preprint arXiv:1610.07629*, 2016. 2
- [10] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2414–2423, 2016. 2
- [11] Leon A Gatys, Alexander S Ecker, Matthias Bethge, Aaron Hertzmann, and Eli Shechtman. Controlling perceptual factors in neural style transfer. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3985–3993, 2017. 2
- [12] Shuyang Gu, Congliang Chen, Jing Liao, and Lu Yuan. Arbitrary style transfer with deep feature reshuffle. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8222–8231, 2018. 1, 2, 3
- [13] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE international conference on computer vision*, pages 1501–1510, 2017. 1, 2, 5, 6
- [14] Wentao Jiang, Si Liu, Chen Gao, Jie Cao, Ran He, Jiashi Feng, and Shuicheng Yan. Psgan: Pose and expression robust spatial-aware gan for customizable makeup transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5194–5202, 2020. 3
- [15] Yongcheng Jing, Xiao Liu, Yukang Ding, Xinchao Wang, Errui Ding, Mingli Song, and Shilei Wen. Dynamic instance normalization for arbitrary style transfer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 4369–4376, 2020. 2
- [16] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer, 2016. 2
- [17] Nikolai Kalischek, Jan D Wegner, and Konrad Schindler. In the light of feature distributions: moment matching for neural style transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9382–9391, 2021. 1, 2
- [18] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6
- [19] Dmytro Kotovenko, Artsiom Sanakoyeu, Sabine Lang, and Bjorn Ommer. Content and style disentanglement for artistic style transfer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4422–4431, 2019. 1, 2
- [20] Chuan Li and Michael Wand. Combining markov random fields and convolutional neural networks for image synthesis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2479–2486, 2016. 2
- [21] Chuan Li and Michael Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. In *European conference on computer vision*, pages 702–716. Springer, 2016. 2
- [22] Xueting Li, Sifei Liu, Jan Kautz, and Ming-Hsuan Yang. Learning linear transformations for fast image and video style transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3809–3817, 2019. 2
- [23] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. Diversified texture synthesis with feed-forward networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3920–3928, 2017. 2
- [24] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. Universal style transfer via feature transforms. *Advances in neural information processing systems*, 30, 2017. 1, 2
- [25] Jing Liao, Yuan Yao, Lu Yuan, Gang Hua, and Sing Bing Kang. Visual attribute transfer through deep image analogy. *arXiv preprint arXiv:1705.01088*, 2017. 2
- [26] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In

- Proceedings of the European Conference on Computer Vision (ECCV)*, pages 740–755. Springer, 2014. 6
- [27] Songhua Liu, Tianwei Lin, Dongliang He, Fu Li, Meiling Wang, Xin Li, Zhengxing Sun, Qian Li, and Errui Ding. Adaattn: Revisit attention mechanism in arbitrary neural style transfer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6649–6658, 2021. 1, 2, 3, 6, 8, 11
- [28] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021. 3
- [29] Dae Young Park and Kwang Hee Lee. Arbitrary style transfer with style-attentional networks. In *proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5880–5888, 2019. 1, 2, 3, 6, 8
- [30] Fred Phillips and Brandy Mackintosh. Wiki art gallery, inc.: A case for critical thinking. *Issues in Accounting Education*, 26(3):593–608, 2011. 6
- [31] Falong Shen, Shuicheng Yan, and Gang Zeng. Neural style transfer via meta networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8061–8069, 2018. 1
- [32] Lu Sheng, Ziyi Lin, Jing Shao, and Xiaogang Wang. Avatar-net: Multi-scale zero-shot style transfer by feature decoration. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8242–8250, 2018. 1, 2, 3, 6
- [33] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 3
- [34] Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, and Victor S Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *International Conference on Machine Learning*, page 4, 2016. 2
- [35] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6924–6932, 2017. 2
- [36] Ashish Vaswani, Prajit Ramachandran, Aravind Srinivas, Niki Parmar, Blake Hechtman, and Jonathon Shlens. Scaling local self-attention for parameter efficient visual backbones. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12894–12904, 2021. 3
- [37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 3, 4
- [38] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018. 3
- [39] Xin Wang, Geoffrey Oxholm, Da Zhang, and Yuan-Fang Wang. Multimodal transfer: A hierarchical deep convolutional neural network for fast artistic style transfer. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5239–5247, 2017. 2
- [40] Xiaolei Wu, Zhihao Hu, Lu Sheng, and Dong Xu. Style-former: Real-time arbitrary style transfer via parametric style composition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14618–14627, 2021. 1, 2
- [41] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32, 2019. 3
- [42] Yuan Yao, Jianqiang Ren, Xuansong Xie, Weidong Liu, Yong-Jin Liu, and Jun Wang. Attention-aware multi-stroke style transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1467–1475, 2019. 2, 3
- [43] Tan Yu, Gangming Zhao, Ping Li, and Yizhou Yu. Boat: Bilateral local attention vision transformer. *arXiv preprint arXiv:2201.13027*, 2022. 3
- [44] Pan Zhang, Bo Zhang, Dong Chen, Lu Yuan, and Fang Wen. Cross-domain correspondence learning for exemplar-based image translation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5143–5153, 2020. 3
- [45] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. 6
- [46] Yulun Zhang, Chen Fang, Yilin Wang, Zhaowen Wang, Zhe Lin, Yun Fu, and Jimei Yang. Multimodal style transfer via graph cuts. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5943–5951, 2019. 1, 2, 3, 6
- [47] Yabin Zhang, Minghan Li, Ruihuang Li, Kui Jia, and Lei Zhang. Exact feature distribution matching for arbitrary style transfer and domain generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8035–8045, 2022. 1, 2
- [48] Yuxin Zhang, Fan Tang, Weiming Dong, Haibin Huang, Chongyang Ma, Tong-Yee Lee, and Changsheng Xu. Domain enhanced arbitrary image style transfer via contrastive learning. In *ACM SIGGRAPH 2022 Conference Proceedings*, pages 1–8, 2022. 1, 2, 8
- [49] Long Zhao, Zizhao Zhang, Ting Chen, Dimitris Metaxas, and Han Zhang. Improved transformer for high-resolution gans. *Advances in Neural Information Processing Systems*, 34:18367–18380, 2021. 1, 3

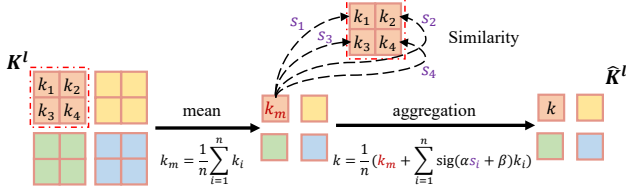


Figure 9. Detailed illustration of the step 1 in DA.

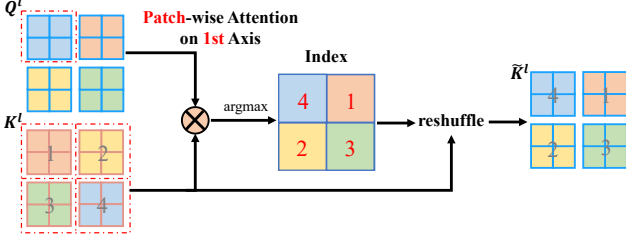


Figure 10. Detailed illustration of the step 1 in PA.

## A. Implementation Details

### A.1. Code

The code will be made public soon.

### A.2. Step 1 of DA

Fig. 9 provides a more detailed illustration of the first step in distributed attention (DA). The first step of DA is regional style aggregation. In this step, we aggregate the information of all points in a block to a point, representing the block’s style information. For example, we aggregate  $K_1$ ,  $K_2$ ,  $K_3$ , and  $K_4$  points in the first block of  $K^l$  to a point  $k$ . Note that all blocks perform regional style aggregation operations in parallel, so we will finally get the regional style representations corresponding to all blocks, which form new keys  $\hat{K}^l$ .

### A.3. Step 1 of PA

Fig. 10 gives a more detailed illustration of the first step in progressive attention (PA). The first step of PA is implemented as patch-wise attention along the first axis, which takes a block region as a token instead of a specific position. In this step, we apply argmax to the attention score to obtain the indices of the most similar coarse-grained region. For example, the 4th block region in  $K^l$  is matched as the coarse-grained region most similar to the blue block region in  $Q^l$ , so the index of this region is set to 4. With the indices, we can reshuffle the tokens of  $K^l$  to semantically match the spatial arrangement of the tokens of  $Q^l$ . The reshuffled  $\tilde{K}^l$  is further utilized in the second step of PA.

### A.4. Decoder Architecture

The decoder implemented in this work follows the setting of [27], which mirrors the encoder and takes the multi-

Table 4. Full configuration of the decoder.

Stage	Output	Architecture
$F^5$	$512 \times \frac{H}{8} \times \frac{W}{8}$	Input $F_{cs}^5$ Upsample scale 2 Add $F_{cs}^4$ $3 \times 3$ Conv, 512, ReLU
$F^4$	$256 \times \frac{H}{4} \times \frac{W}{4}$	$3 \times 3$ Conv, 256, ReLU Upsample scale 2
$F^3$	$128 \times \frac{H}{2} \times \frac{W}{2}$	Concatenate $F_{cs}^3$ $(3 \times 3$ Conv, 256, ReLU) $\times 3$ $3 \times 3$ Conv, 128, ReLU Upsample scale 2
$F^2$	$64 \times H \times W$	$3 \times 3$ Conv, 128, ReLU $3 \times 3$ Conv, 64, ReLU Upsample scale 2
$F^1$	$3 \times H \times W$	$3 \times 3$ Conv, 64, ReLU $3 \times 3$ Conv, 3

scale transferred features as input. Full decoder configuration is shown in Table 4. The decoder takes the multi-scale transferred features  $F_{cs}^l$  as input and gradually synthesizes the final image  $I_{cs}$ .

### A.5. All-to-key Attention Algorithm

The PyTorch code for our proposed all-to-key attention mechanism is shown in Algorithm 1. The implementation is elegant with the usage of einsum notation.

### A.6. Blocking and Unblocking Algorithm

The PyTorch code for feature blocking and unblocking operation in the all-to-key attention mechanism is shown in Algorithm 2.

### A.7. Loss Function

**Setting of  $\lambda_1$  and  $\lambda_2$**  We empirically set the weight of each loss term to 10 and 1.25. Since we only have two loss terms, we can fix the weight of one loss and choose the appropriate weight according to the impact of adjusting the weight of another loss. We fix  $\lambda_1$  to 10 and statistically analyze the evaluation scores of the results generated under different settings of  $\lambda_2$ , as shown in Table 5. As the proportion of  $\lambda_2$  increases, content loss, LPIPS score decreases, and style loss increases. Therefore, there is a trade-off in style and content. We finally set  $\lambda_2$  to 1.25 to render consistent style texture while maintaining semantic structure.

## B. More Results

### B.1. All-to-key Attention VS. All-to-all Attention

To further illustrate the superiority of our proposed all-to-key attention to all-to-all attention in producing high-quality stylized images, we provide more visual comparisons in Figure 11. By replacing all-to-key attention in

Table 5. Evaluation scores of the results generated under different ratios. The best results are in **bold**.

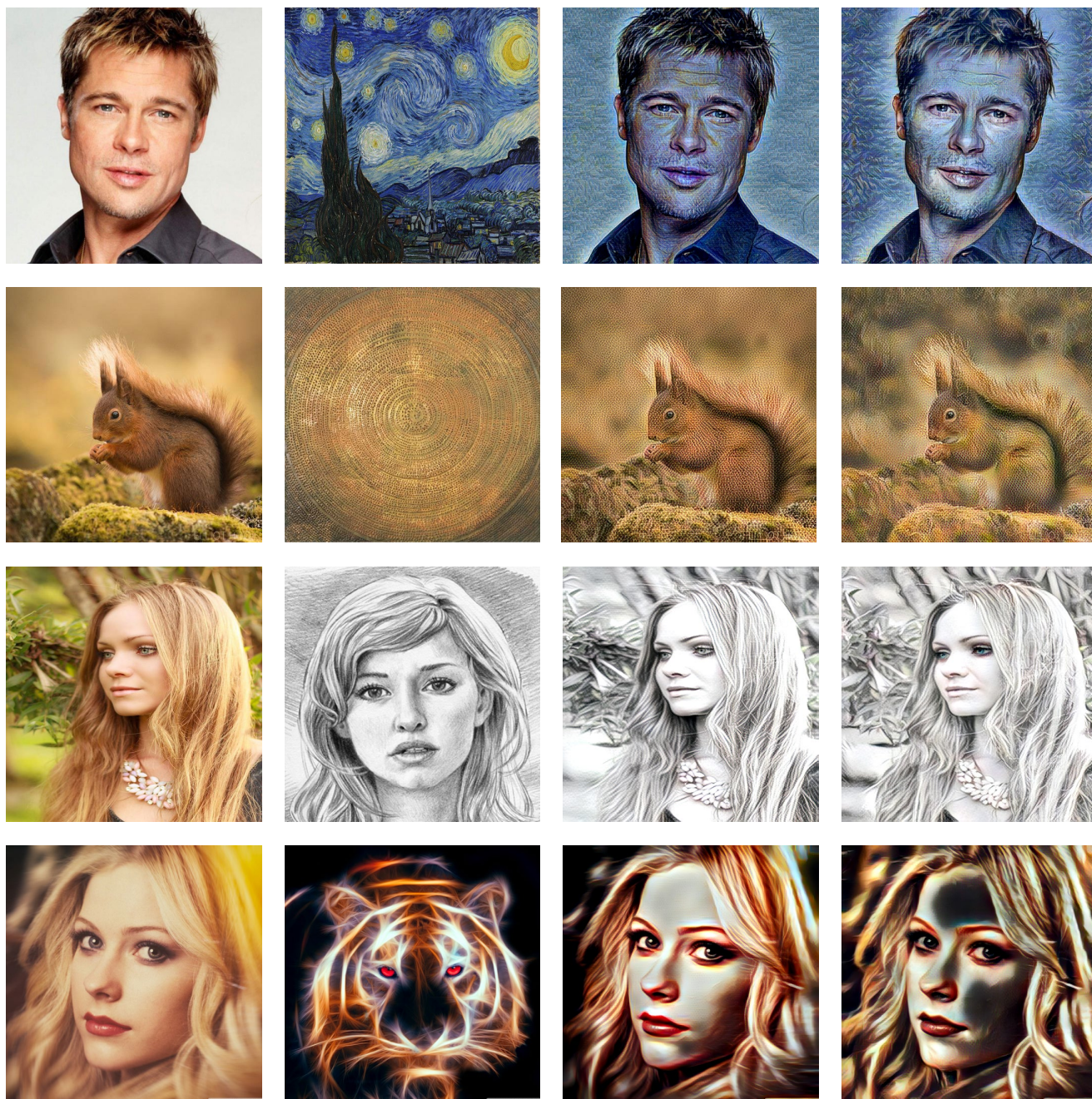
$\lambda_2$	Content Loss ↓	Style Loss ↓	LPIPS ↓
0.5	0.72	<b>0.75</b>	0.57
0.75	0.63	0.82	0.55
1	0.58	0.98	0.54
1.25	0.55	1.04	0.53
1.5	0.52	1.20	0.52
2	<b>0.49</b>	1.38	<b>0.49</b>

our full model with all-to-all attention, the visual quality of the stylized images decreases significantly, affected by distorted style patterns.

## B.2. Arbitrary Style Transfer

To further demonstrate the effectiveness and robustness of our proposed StyA2K on arbitrary style transfer, we provide more stylization results of pair-wise combinations between 10 content images and 8 style images (total 80 stylized images) in Figure 12 and Figure 13. Our method can faithfully generate visually appealing results with consistent style textures.





Content

Style

All-to-key Attention

All-to-all Attention

Figure 11. More visual comparisons between our proposed all-to-key attention and all-to-all attention.

---

**Algorithm 1** Pytorch code implementing all-to-key attention.

---

```
1 class A2K(nn.Module):
2     '''All-to-key Attention'''
3     def __init__(self, in_dim):
4         super().__init__()
5         self.Dq = nn.Conv2d(in_dim, in_dim, (1, 1))
6         self.Dk = nn.Conv2d(in_dim, in_dim, (1, 1))
7         self.Dv = nn.Conv2d(in_dim, in_dim, (1, 1))
8         self.Pq = nn.Conv2d(in_dim, in_dim, (1, 1))
9         self.Pk = nn.Conv2d(in_dim, in_dim, (1, 1))
10        self.Pv = nn.Conv2d(in_dim, in_dim, (1, 1))
11        self.sim_alpha = nn.Parameter(torch.ones(1), require_grad=True)
12        self.sim_beta = nn.Parameter(torch.zeros(1), require_grad=True)
13        self.fusion_D = nn.Conv2d(in_dim, in_dim, (1, 1))
14        self.fusion_P = nn.Conv2d(in_dim, in_dim, (1, 1))
15
16    def forward(self, content, style):
17        # Get Q K V
18        DA_q = block(self.Dq(mean_variance_norm(content)), p_size, stride)
19        DA_k = block(self.Dk(mean_variance_norm(style)), p_size, stride)
20        DA_v = block(self.Dv((style)), p_size, stride)
21        PA_q = block(self.Pq(mean_variance_norm(content)), p_size, stride)
22        PA_k = block(self.Pk(mean_variance_norm(style)), p_size, stride)
23        PA_v = block(self.Pv(style), p_size, stride)
24        # Distributed Attention Step 1
25        DA_k_m = torch.mean(DA_k, dim=-1)
26        DA_v_m = torch.mean(DA_v, dim=-1)
27        dis = torch.einsum("bhcx,bhcx->bhxy", DA_k_m, DA_k)
28        sim = torch.sigmoid(self.sim_beta+self.sim_alpha*dis)
29        DA_k_a = (torch.einsum("bhxy,bhcx->bhcx", sim, DA_k) + DA_k_m) / p_size
30        DA_v_a = (torch.einsum("bhxy,bhcx->bhcx", sim, DA_v) + DA_v_m) / p_size
31        # Distributed Attention Step 2
32        logits = torch.einsum("bhcx,bhcy->bhyxz", DA_q, DA_k_a)
33        scores = softmax(logits)
34        DA = torch.einsum("bhyxz,bhvzy->bhcx", scores, DA_v_a)
35        DA_unblock = unblock(DA)
36        # Progressive Attention Step 1
37        PA1_logits = torch.einsum("bhcx,bhcy->bhxz", PA_q, PA_k)
38        index = torch.argmax(PA1_logits, dim = -1).expand_as(PA_k)
39        PA_k_reshuffle = torch.gather(PA_k, -2, index)
40        PA_V_reshuffle = torch.gather(PA_v, -2, index)
41        # Progressive Attention Step 2
42        logits2 = torch.einsum("bhcx,bhcx->bhxyz", PA_q, PA_k_reshuffle)
43        scores2 = softmax(logits2)
44        PA = torch.einsum("bhxyz,bhcx->bhcx", scores2, PA_V_reshuffle)
45        PA_unblock = unblock(PA)
46        # Feature Transformation
47        O_DA = self.fusion_D(DA_unblock)
48        O_PA = self.fusion_P(PA_unblock)
49        O = O_DA + O_PA
50        out = O + Content
51        return out
```

---

---

**Algorithm 2** Pytorch code implementing feature blocking and unblocking.

---

```
1  def block(X, patch_size , stride):
2      '''feature blocking.
3      Args:
4          X: a tensor with shape [b, c, h, w], where b is batch size , c is the
5              channel dimension, h is feature height , and w is feature width.
6              patch_size: an integer for the patch (block) size
7              stride: the parameter of torch.nn.functional.unfold
8      Returns:
9          Y: a tensor with shape [b, c, n, r], where n is patch sequence length
10             and r is the patch size.
11      '''
12      b, c, h, w = X.shape
13      r = int(patch_size**2)
14      Y = torch.nn.functional.unfold(X, kernel_size=patch_size , stride=stride)
15      Y = Y.view(b, c, r, -1).permute(0, 1, 3, 2)
16      return Y
17
18  def unblock(X, patch_size , stride , h):
19      '''feature unblocking.
20      Args:
21          X: a tensor with shape [b, c, n, r], where b is batch size , c is
22              channel dimension, n is patch sequence length , r is the patch size.
23              patch_size: an integer for the patch (block) size
24              stride: the parameter of torch.nn.functional.unfold
25              h: the output height of the feature
26      Returns:
27          Y: a tensor with shape [b, c, h, w], where h is is feature height
28             and w is feature width.
29      '''
30      b, c, n, r = X.shape
31      X = X.permute(0, 2, 1, 3)
32      X = X.contiguous().view(b, n, -1).permute(0, 2, 1)
33      Y = torch.nn.functional.unfold(X, h, kernel_size=patch_size , stride=stride)
34      return Y
```

---





Figure 12. More image stylization results.





Figure 13. More image stylization results.