Self-Contained Stylization via Steganography for Reverse and Serial Style Transfer

Hung-Yu Chen[1*†] I-Sheng Fang[1*†] Chia-Ming Cheng[2] Wei-Chen Chiu[1]
[1]National Chiao Tung University, Taiwan [2]MediaTek Inc., Taiwan
chen3381@purdue.edu nf0126@gmail.com walon@cs.nctu.edu.tw

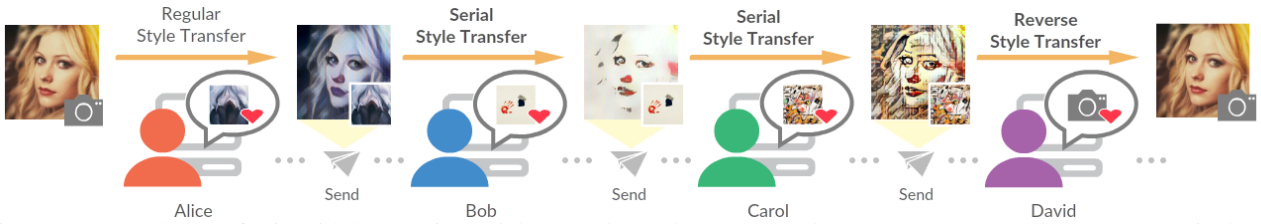


Figure 1: As style transfer is widely used in social networks such as Facebook or Instagram nowadays, two practical applications related to style transfer are studied in this paper and can be illustrated with a scenario here. **(1) Serial style transfer**: when Bob receives a stylized image from Alice, he can easily modify its style into any other arbitrary one and further share the output to others (e.g. Carol). **(2) Reverse style transfer**: any user (e.g. David) who receives a stylized image can easily reverse it back to its original photo, i.e. analogous to de-style, without requiring any additional information. Our proposed self-contained stylization approach is capable of tackling these two applications hence could be quite useful for the social networks, since the users of a social network tend to share photos and stylized images on the same platform.

Abstract

Style transfer has been widely applied to give real-world images a new artistic look. However, given a stylized image, the attempts to use typical style transfer methods for de-stylization or transferring it again into another style usually lead to artifacts or undesired results. We realize that these issues are originated from the content inconsistency between the original image and its stylized output. Therefore, in this paper we advance to keep the content information of the input image during the process of style transfer by the power of steganography, with two approaches proposed: a two-stage model and an end-to-end model. We conduct extensive experiments to successfully verify the capacity of our models, in which both of them are able to not only generate stylized images of quality comparable with the ones produced by typical style transfer methods, but also effectively eliminate the artifacts introduced in reconstructing original input from a stylized image as well as performing multiple times of style transfer in series.

1. Introduction

A style transfer approach typically aims to modify an input photo such that its content can be preserved but the associated style is revised as the one of a reference image. In comparison to the classical methods which generally rely on matching color statistics between the reference image and modified output [8, 18], the recent development of deep learning has brought a great leap by being able to capture high-level representation for the content and style of images, thus producing more photorealistic stylization. In particular, after the advent of first deep-learning style transfer work [5], many research efforts [4, 5, 11, 23] have gone with the trend to propose faster, more visually appealing, and more universal algorithms for the task of style transfer.

Without loss of generality, as these approaches basically perform transformation on the content feature of input photo according to the style feature from reference image, the appearance of photo is usually altered to have various colors or textures, which inevitably causes changes to the fine-grained details in content information. Consequently, the stylized output no longer has the same content feature as its original photo, leading to some issues for two novel applications that we proposed in this paper: *serial* and *reverse* style transfer. The former attempts to transfer an image, which is already stylized, into another arbitrary style; while the latter aims to remove the stylization effect of a stylized image and turn back to its original photo, as the example scenario illustrated in the Figure 1. Particularly, what we expect to obtain for the serial style transfer is that,

[†]Hung-Yu Chen and I-Sheng Fang are now with Purdue University and National Cheng Chi University respectively.
[*]Both authors contribute equally.

even after applying multiple times of different stylizations, the final output should be similar to the one which is produced by directly transferring the original photo into the latest style (i.e. not influenced by the previous stylization). We believe that having both serial and reverse style transfer can open the door to exciting new ways for users to interact with style transfer algorithms, not only allowing the freedom to perform numerous stylizations on a photo with having its content well preserved, but also providing the access to the original input by recovering from a stylized image.

Although these two problems intuitively seem easy to solve by performing style transfer again on the stylized image with taking the image of another artistic style or the original photo as the source of stylization respectively, the results are usually not visually satisfying and lose the content consistency. For instance, when two style transfer operations are performed in series, such characteristic brings artifacts to the final output and makes it significantly distinct from the result obtained by applying the second style transfer to the original input . Similarly, upon taking a stylized image and its corresponding original photo as sources of content and style respectively, we are not able to achieve reverse stylization of reconstructing the original input . Furthermore, there could exist a potential argument that both reverse and serial style transfer are simple once the original photo is always transmitted with the stylized image. However, this naïve solution doubles the bit-rates for transmission thus being quite inefficient for sharing stylized images on the internet or social networks.

To tackle the aforementioned issues, it calls for a framework which can not only generate visually appealing stylized images as typical style-transfer approaches do, but also maintain the important representations related to the content feature of input photo, so that the content inconsistency between the stylized image and the original photo can be compensated afterwards. In this paper we propose to achieve so by integrating the power of **steganography** [6, 1, 26] into style transfer approaches, where the content information of input photo is hidden into the style-transferred output with steganographic method. With a decoder trained to extract the hidden information from the stylized image produced by our proposed method, the issue of having severe artifacts while doing reverse or serial transfer could be resolved. As the content information is self-contained in the stylized image via the use of steganography, in the scenario of Figure 1, the serial and reverse style transfer are now naturally achievable without any additional cost of transmitting the original photo or any other forms of attaching data. It is also worth noting that, with a simple extension on our approach such as a gate to control whether the content information of original photo is provided for the steganography component or not, the users can easily control the usage right of their stylized images, i.e. allowing or forbidding the images to be further style-transferred or de-stylized.

We implement the idea with two different deep-learning architectures, where one is a two-stage model and the other one is an end-to-end model, as going to be detailed in Section 3. The two-stage model needs to hide a bigger amount of information into the image, but can be coupled with various style transfer methods, leading to a better adaptability; the end-to-end model is highly dependent on the traits of AdaIN [9], but it only needs to encrypt a small amount of information into the image, being more robust to the potential error accumulation during multiple serial style transfers. We conduct extensive experimental validation comparing to several baselines and demonstrate the efficacy of our proposed method to advance serial and reverse style transfer.

2. Related Works

Style transfer. Giving images a new artistic style or texture has long been a topic that attracts researchers' attention. Some of the early research works prior to the renaissance of deep learning tackle the style transfer by simply matching the characteristic in color, or searching for the correspondences across source and style images [3, 8]. Instead of using low-level feature cues as early works, Gatys *et al*. [4, 5] utilize representations obtained from the pre-trained convolutional neural network (CNN) to extract more semantic description on the content and style features of images. Their methods can generate visually appealing results; however, it is extremely slow due to iterative optimization for matching style features between the output and style image.

In order to speed up the process of image style transfer, several feed-forward approaches (e.g. [11, 23]) are proposed, which directly learn feed-forward networks to approximate the iterative optimization procedure with respect to the same objectives. The style transfer now can be carried out in real time, however, there usually exists a trade-off between the the processing speed and the image quality of the stylized output. For example, the result of [11] suffers from repetitive patterns in plain area. Fortunately, Ulyanov *et al*. [23] uncover that the image quality produced by the network of [11] could be greatly improved through replacing its batch normalization layers (BN) with instance normalization (IN) ones, while [24] steps further to introduce conditional instance normalization and learns to perform real-time style transfer upon multiple styles that have been seen during training. Nevertheless, all these feed-forward models are typically constrained to particular styles and hardly generalizable to arbitrary stylization. That's where adaptive instance normalization (AdaIN) [9] comes into play.

AdaIN could be roughly seen as IN with a twist. It basically follows the IN steps, except now the content feature of input photo is first normalized then affine-transformed by using the mean and standard deviation of the style features of style image. This operation matches the statistics of

content and style features in order to transfer the input photo into an arbitrary style, since the parameter applied in AdaIN is dependent on the target style. Given a content feature x and a style feature y, the procedure of AdaIN is:

$$\text{AdaIN}(x, y) = \sigma(y)\left(\frac{x - \mu(x)}{\sigma(x)}\right) + \mu(y) \qquad (1)$$

where μ and σ denote the mean and standard deviation respectively. There are other research works [13, 17] sharing the similar idea with AdaIN, where various manners are introduced for adaptively transforming the content feature of input photo in accordance to the style image. Since the simplicity of AdaIN and its ability for universal style transfer, we utilize AdaIN as the base model in our proposed method for style transfer and make extensions for handling issues of serial and reverse stylizations.

Image de-stylization / Reverse style transfer. As image style transfer typically applies artistic styles to the input images, image de-stylization or reverse style transfer attempts to remove those styles from the stylized images and recover them back to their original appearance. To the best of our knowledge, only a handful of research works tackle this topic. Shiri *et al*. [19, 20] explore the field of image de-stylization with a particular focus on human faces. Their methods learn a style removal network to recover the photo-realistic face images from the stylized portraits and retain the identity information. However, they rely on the specific properties of human faces, so it can hardly be generalized to other object categories. [22] proposes to translate artworks to photo-realistic images, which is similar to de-stylization but is limited to only few artistic styles. The naïve approach of having the original input as the style image and other methods from the image-to-image translation area (e.g. CycleGAN [27] or Pix2Pix [10]) are also incapable of achieving image de-stylization or only applicable to the seen styles (thus not universal), as already shown in [20].

Image steganography. Image steganography is a way to deliver a message secretly by hiding it into in an irrelevant cover image while minimizing the perturbations within the cover, and has been studied for a long period in the area of image processing [12, 2]. In general, traditional approaches rely on carefully and manually designed algorithms to achieve both message hiding and retrieval from the cover image. Some examples of such methods would be HUGO [16] and least significant bit steganography [6].

After the application of deep learning has grown popular, few research works [7, 1, 26] explore the possibility of having deep neural networks perform steganography on images, where the hiding and revealing processes are learned together in the manner of end-to-end training. [7] proposes to train the steganographic algorithm and steganalyzer jointly via an adversarial scheme between three-players. In comparison to handling lower bit rates of [7], [1] intends to hide an image entirely into another image of the same size, but has a potential drawback of being detectable. For the method proposed in [26], it hides relatively smaller amount of message into an irrelevant cover image but specifically tackles the problem of making the hidden message robust to noises.

3. Proposed Methods

3.1. Two-Stage Model

Our two-stage model is a pipeline built upon a straightforward integration of style transfer and steganography networks, as shown in Figure 2(a). In the first stage, we stylize the content image I_c according to the style image I_s based on a style transfer model. Afterward in the second stage, the steganography network learns an encoder to hide the content information of I_c into the stylized image I_t from the previous stage, as well as a paired decoder which is able to retrieve the hidden information from the encoded image.

3.1.1 Style Transfer Stage

We adopt AdaIN [9] as our primary reference method while our two-stage model is capable of incorporating with other style transfer algorithms (e.g. WCT [13] or [23], please refer to the supplementary materials for more details). The architecture is composed of a pre-trained VGG19 [21] encoder E_{VGG} and a decoder D_{AdaIN}. The concept of the training procedure can be briefly summarized as follows:

1) the encoder extracts the content feature $v_c = E_{VGG}(I_c)$ and style feature $v_s = E_{VGG}(I_s)$ from content image I_c and style image I_s respectively; 2) based on Eq. 1, the content feature v_c is adaptively redistributed according to the statistics of style feature v_s to obtain the target feature v_t; 3) the stylized output I_t is finally produced by $D_{AdaIN}(v_t)$.

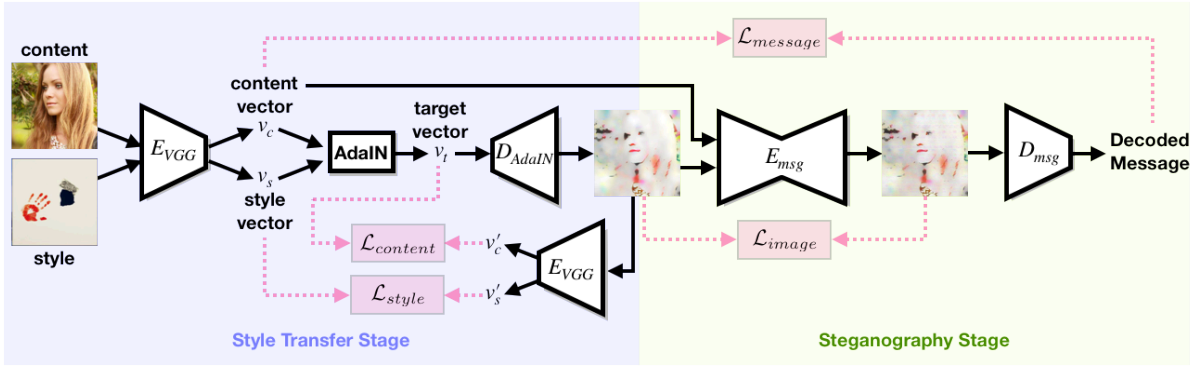
While encoder E_{VGG} is pre-trained and fixed (based on first few layers of VGG19 up to `relu4_1`), the learning of AdaIN style transfer focuses on training D_{AdaIN} with respect to the objective (same as in AdaIN [9]):

$$\mathcal{L}_{style-transfer} = \mathcal{L}_{content} + \lambda_{style}\mathcal{L}_{style} \qquad (2)$$

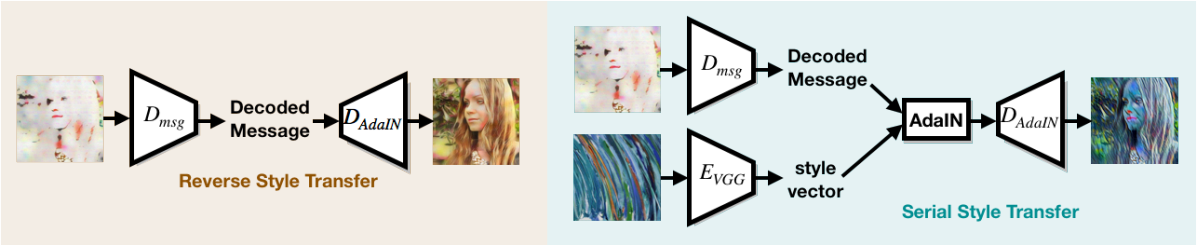
in which the content loss $\mathcal{L}_{content}$ and style loss $\mathcal{L}_{style}$ are defined as follows with their balance controlled by λ_{style} (which is set to 10 in all our experiments, as used in [9]):

$$\mathcal{L}_{content} = \|E_{VGG}(I_t) - v_t\|_2 \qquad (3)$$

$$\mathcal{L}_{style} = \sum_{i}^{L} \|\mu(l_i(I_t)) - \mu(l_i(I_s))\|_2 + \sum_{i}^{L} \|\sigma(l_i(I_t)) - \sigma(l_i(I_s))\|_2 \qquad (4)$$



(a) Visualization of the architecture and training objectives for our proposed two-stage model, which is composed of a style transfer stage (shaded in purple) and a steganography stage (shaded in green).



(b) Illustrations of how to apply our two-stage model in the tasks of reverse and serial style transfer respectively.

Figure 2: Overview of the training and testing procedure of our two-stage model. Please refer to Section 3.1 for details.

where each l_i denotes the feature map obtained from a layer in VGG19, and $L = \{$ `relu1_1`, `relu2_1`, `relu3_1`, `relu4_1` $\}$ in our experiments.

In addition to the objective function above which encourages $D_{AdaIN}(v_t)$ to output the stylized image I_t with its content feature $E_{VGG}(I_t)$ close to target v_t and similar style as I_t, we also train D_{AdaIN} with *identity mapping*, i.e. reconstructing content image $\tilde{I}_c$ solely from its content feature v_c, for the purpose of better dealing with reverse style transfer later on. To achieve identity mapping, we occasionally place the same photo for both content and style images during the training of D_{AdaIN}, so that the content feature v_c and target feature v_t are identical. Thus, the output I_t of D_{AdaIN} is similar to I_c by the same objectives as Eq. 3.

3.1.2 Steganography Stage

The steganography stage in our model contains a message encoder E_{msg} and a corresponding message decoder D_{msg}. The message encoder E_{msg} aims to hide content feature v_c into stylized image I_t and produce the encoded image $I_e = E_{msg}(I_t, v_c)$, which is exactly the output of our two-stage model, while the message decoder D_{msg} tries to decode v_c out from I_e. As the typical scheme of steganography, the difference between the encoded image I_e and stylized image I_t should be visually undetectable, therefore the E_{msg} is trained to minimize the objective defined as:

$$\mathcal{L}_{image} = \|I_e - I_t\|_2 \tag{5}$$

On the other hand, the message decoder D_{msg} is optimized to well retrieve the message v_c hidden in I_e, with respect to the objective:

$$\mathcal{L}_{message} = \|D_{msg}(I_e) - v_c\|_2 \tag{6}$$

where the architecture designs of both E_{msg} and D_{msg} follow the ones used in the recent steganography paper [26]. The objective for the steganography stage is summarized as:

$$\mathcal{L}_{steganography} = \lambda_{img}\mathcal{L}_{image} + \lambda_{msg}\mathcal{L}_{message} \tag{7}$$

where λ_{img} and λ_{msg} are used to balance $\mathcal{L}_{image}$ and $\mathcal{L}_{message}$ respectively.

3.1.3 Reverse & Serial Stylization by Two-Stage Model

Reverse style transfer. As shown in the left portion of Figure 2(b), by using the decoder D_{AdaIN}, which is capable of performing identity mapping, to decode the content feature v_c from a given encoded image I_e, the original content image I_c can now be recovered by $D_{AdaIN}(D_{msg}(I_e))$.

Serial style transfer. To transfer the encoded image I_e (which is already stylized) into another style given by I'_s, as shown in the right portion of Figure 2(b), the content feature $v_c = D_{msg}(I_e)$ decoded from I_e and the style feature $v'_s = E_{VGG}(I'_s)$ extracted from I'_s are taken as inputs for AdaIN transformation, then the serial style transfer is achieved by computing $I'_t = D_{AdaIN}(\text{AdaIN}(D_{msg}(I_e), v'_s))$. In addition, performing multiple times of style transfer in series

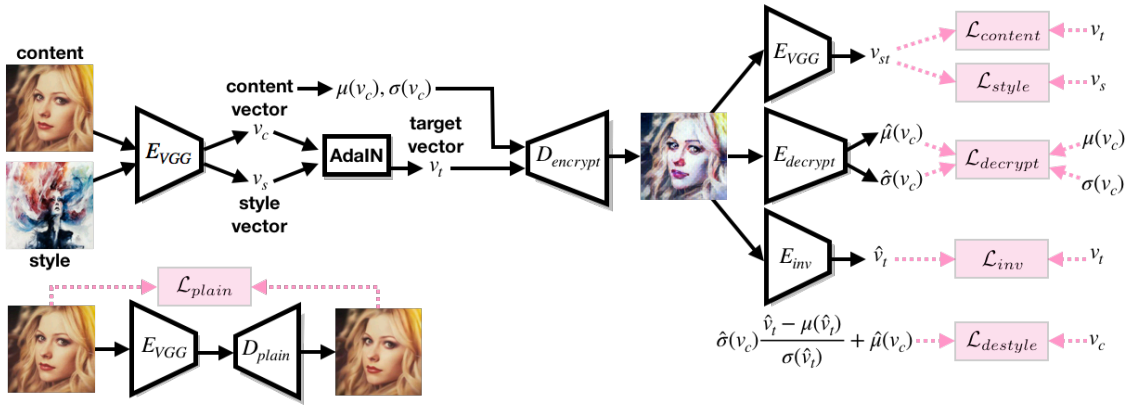


Figure 3: Overview of our end-to-end model and its training objectives, where the image stylization and content information encryption are now particularly performed jointly in a single network $D_{encrypt}$. Please refer to Section 3.2 for more details.

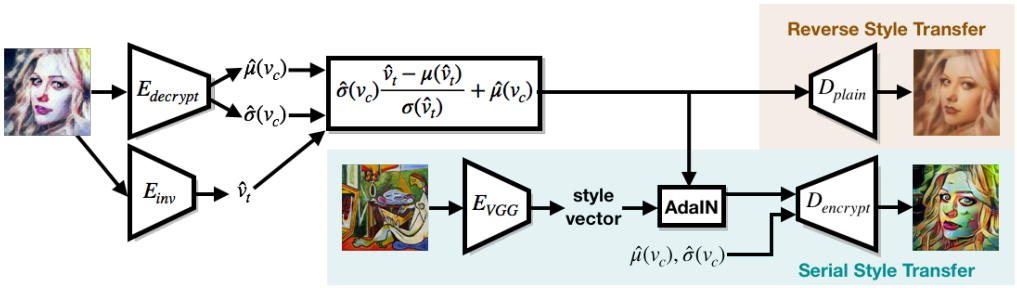


Figure 4: Illustration of applying proposed end-to-end model, especially the reconstructed content feature (cf. Eq. 10), for both tasks of reverse and serial style transfer network, denoted by different colors respectively.

is naturally doable when the steganography is applied for encoding v_c into I'_t again.

3.2. End-to-End Model

Aside from the two-stage model which can take several style transfer methods as its base (e.g. WCT [13] or [23], please refer to our supplementary material), our end-to-end model digs deeply into the characteristic of AdaIN for enabling image stylization and content information encryption simultaneously in a single network. As we know, the procedure of AdaIN (cf. Eq. 1) produce a target feature v_t by transforming the content feature v_c to match the statistics of the style feature v_s, i.e. mean $\mu(v_s)$ and standard deviation $\sigma(v_s)$. Assume there exists an inverse function which can estimate the corresponding target feature v_t of a stylized image I_{st}, we hypothesize that the content feature v_c is derivable from v_t by $\sigma(v_c)\frac{v_t-\mu(v_t)}{\sigma(v_t)}+\mu(v_c)$ once its original statistics $\{\mu(v_c), \sigma(v_c)\}$ is available.

Based on this hypothesis, our end-to-end model is designed to have several key components as shown in Figure 3: 1) a encrypted image decoder $D_{encrypt}$ which takes $v_t, \mu(v_c), \sigma(v_c)$ as input and produce a stylized image I_{st} with $\{\mu(v_c), \sigma(v_c)\}$ being encrypted into it; 2) a decrypter $E_{decrypt}$ which is able to decrypt $\{\mu(v_c), \sigma(v_c)\}$ out from I_{st}; and 3) an inverse target encoder E_{inv} which is capable of estimating v_t from the given stylized image I_{st}. In the following and Figure 3 we detail the overall computation of our model and the objectives for training.

First, the output image I_{st} of the encrypted image decoder $D_{encrypt}$, which is simultaneously encrypted and stylized, should still have the similar content/style feature as the one in the content/style image respectively (i.e. $\{v_c, v_s\}$). The same objective functions, $\mathcal{L}_{content}$ and $\mathcal{L}_{style}$, defined in Eq. 3, can then be adopted to optimize $D_{encrypt}$ by simply replacing I_t with I_{st} here.

Second, we see that the $\{\mu(v_c), \sigma(v_c)\}$ encrypted into I_{st} with $D_{encrypt}$ should be retrievable by using the corresponding decrypter $E_{decrypt}$. Therefore, the output of $E_{decrypt}$, $\{\hat{\mu}(v_c), \hat{\sigma}(v_c)\}$, is compared to the original $\{\mu(v_c), \sigma(v_c)\}$, leading to the decryption loss $L_{decrypt}$ for jointly optimizing $D_{encrypt}$ and $E_{decrypt}$:

$$\mathcal{L}_{decrypt} = \|\hat{\mu}(v_c) - \mu(v_c)\|_2 + \|\hat{\sigma}(v_c) - \sigma(v_c)\|_2 \quad (8)$$

Third, as motivated in our hypothesis, there should be an inverse target encoder E_{inv} which is able to recover the

Figure 5: Example results of regular style transfer produced by different methods. First two columns show the pairs of content/style images; third to last columns present the results from Gatys [5], AdaIN [9], our two-stage model, and our end-to-end model respectively. We observe that our models are able to generate results with quality comparable to the baselines.

target vector v_t used for generating I_{st}. It is worth noting that, here we design E_{inv} to have the same architecture as E_{VGG}, but it is trained to ignore the influence caused by the encrypted information in I_{st} and focus on retrieving the target vector v_t. With denoting the feature vector estimated by E_{inv} as $\hat{v}_t = E_{inv}(I_{st})$, the objective function for training E_{inv} is then defined as

$$\mathcal{L}_{inv} = \|\hat{v}_t - v_t\|_2 \tag{9}$$

Fourth, with having $\{\hat{\mu}(v_c), \hat{\sigma}(v_c)\}$ and $\hat{v}_t$ obtained from $E_{decrypt}$ and $E_{inv}(I_{st})$ respectively, we can reconstruct the content feature $\hat{v}_c$ according to:

$$\hat{v}_c = \hat{\sigma}(v_c)\frac{\hat{v}_t - \mu(\hat{v}_t)}{\sigma(\hat{v}_t)} + \hat{\mu}(v_c) \tag{10}$$

Then an objective is defined based on the difference between $\hat{v}_c$ and the original v_c:

$$\mathcal{L}_{destyle} = \|\hat{v}_c - v_c\|_2 \tag{11}$$

where it can update $D_{encrypt}$, $E_{decrypt}$, and E_{inv} jointly.

Fifth, as a similar idea of having identity mapping in our two-stage model, here we learn a plain image decoder D_{plain} which can map a content feature back to the corresponding content image I_c. Its training is simply done by:

$$\mathcal{L}_{plain} = \|D_{plain}(E_{VGG}(I_c)) - I_c\|_2 \tag{12}$$

The overall objective function $\mathcal{L}_{end2end}$ for our end-to-end model training is then summarized as below, where λ parameters are used to balance weights of different losses:

$$\begin{aligned}\mathcal{L}_{end2end} = \lambda_c\mathcal{L}_{content} + \lambda_s\mathcal{L}_{style} + \lambda_{dec}\mathcal{L}_{decrypt} \\ + \lambda_{inv}\mathcal{L}_{inv} + \lambda_{des}\mathcal{L}_{destyle} + \lambda_p\mathcal{L}_{plain}\end{aligned} \tag{13}$$

3.2.1 Reverse & Serial Stylization by End-to-End Model

After our end-to-end model are properly trained, since the content vector of the original content image can be reconstructed by using Eq. 10, the reverse and serial style transfer are now straightforwardly achievable, as shown in Figure 4.

Reverse style transfer. The reverse style transfer, which recovers the original image based on a stylized image I_{st}, is done by having the decrypted content feature $\hat{v}_c$ go through the plain image decoder D_{plain}.

Serial style transfer. Given a stylized image I_{st}, by decrypting content feature $\hat{v}_t$ from I_{st} and extracting style feature v'_s from a new style image, the serial style transfer is then produced based on $D_{plain}(\text{AdaIN}(\hat{v}_c, v'_s))$. Please note here we can encrypt the statistics of content feature into the output again, as shown in the lower part of Figure 4, for enabling multiple times of style transfer in series.

4. Experiment

Dataset We follow the similar setting in [9] to build up the training set for our models. We randomly sample 10K content and 20K style images respectively from the training set of MS-COCO [14] and the training set of WikiArt [15]. These training images are first resized to have the smallest dimension be 512 while the aspect ratio is kept, then randomly cropped to the size of 256×256.

4.1. Qualitative Evaluation

We compare our proposed models to the baselines from Gatys *et al*. [5] and AdaIN [9] (more baselines, e.g. [13, 23], in the supplement), based on the qualitative results for the tasks of regular, reverse and serial style transfer. In particular, we apply two times of stylization sequentially in the serial style transfer experiments, for both qualitative and quantitative evaluations (subsection 4.1 and 4.2 respectively). Please note that all the style images used in qualitative evaluation have never been seen during our training.

4.1.1 Regular Style Transfer

As the goal of our proposed models is not aiming to improve the quality of regular style transfer, we simply examine whether the stylization produced by our models is

Figure 6: Two sets of example results for reverse style transfer. The rows show (1) stylized images and (2) the de-stylized results. The corresponding content image is provided in the left of each set. Our models better reconstruct the overall structure of original content images.

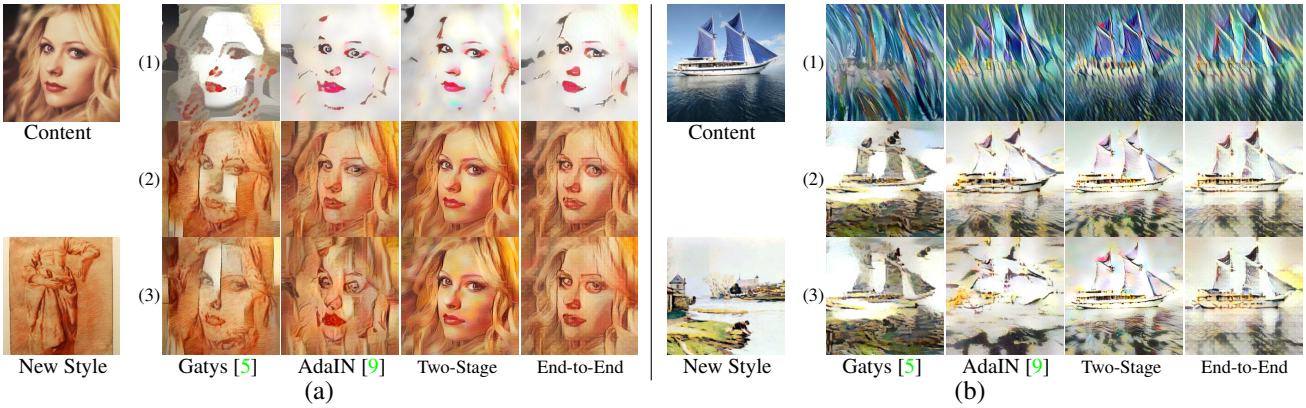


Figure 7: Two sets of example results of serial style transfer. The rows from top to bottom sequentially show (1) stylized images based on the first style, (2) the expectation of new stylization on the content image, and (3) results of serial stylization produced by different methods. The corresponding content image and the new style going to be applied on it are provided in the left of each set. Our models have results better aligned w.r.t. the corresponding expectations.

reasonable in comparison to the baselines. Figure. 5 provides example results of regular style transfer generated by using different methods. We can see that although both our two-stage and end-to-end models have different stylized results w.r.t their base AdaIN approach, they retain comparable quality where the global structure of content image is maintained and the stylization is effective.

4.1.2 Reverse Style Transfer

The goal of reverse style transfer is performing de-stylization on a stylized image, such that the content image can be reconstructed as close to its original appearance as possible. As the baselines, Gatys *et al*. [5] and AdaIN [9], have no corresponding procedures for reverse style transfer, we thus utilize a naïve solution for them, where the stylization is applied to a given stylized image with having the original content image as source of target style. Please note here that this naïve solution of reverse style transfer for baselines needs the access to original content image, while our proposed models can perform de-stylization solely with the given stylized image. Two sets of example results for the task of reverse style transfer are shown in Figure 6. From set (a), both baselines, especially AdaIN, fail to preserve the contour of the face. Although the results of our two-stage and end-to-end models have some mild color patches and slight color shift respectively, they both well reconstruct the overall structure of the content image. Similar observation also exists in set (b). The results of both our models are unaffected by the fuzzy patterns in stylized images, and have clear boundaries between objects, while the baselines could not discriminate the actual contours from the edges caused by stylization, which leads to the results with severe artifacts. These experimental results verify the capability of our models toward resolving the issue of reverse style transfer.

4.1.3 Serial Style Transfer

Serial style transfer attempts to transfer a stylized image into another different style, while keeping the result minimally affected by the previous stylization. Ideally, the result of serial style transfer is expected to be close to the one obtained by stylizing the original content image with the new style image. Two sets of example results of serial style transfer are shown in Figure 7. It is obvious that the results produced by our proposed method are more similar to their respective expectations than the ones from baselines which are deeply influenced by the previous stylization. Therefore our proposed models are successfully verified for their competence on dealing with serial style transfer.

		Gatys [5]	AdaIN [9]	Two-Stage	End-to-End
Reverse Style Transfer	L2	4.4331	0.0368	**0.0187**	0.0193
	SSIM	0.2033	0.3818	0.4796	**0.5945**
	LPIPS	0.3684	0.4614	**0.3323**	0.3802
Serial Style Trasfer	L2	7.5239	0.0213	0.0148	**0.0104**
	SSIM	0.0472	0.5470	0.7143	**0.8523**
	LPIPS	0.4317	0.3637	0.2437	**0.1487**

Table 1: The averaged L2 distance, SSIM, and LPIPS [25] between the results and their corresponding expectations.

4.2. Quantitative Evaluation

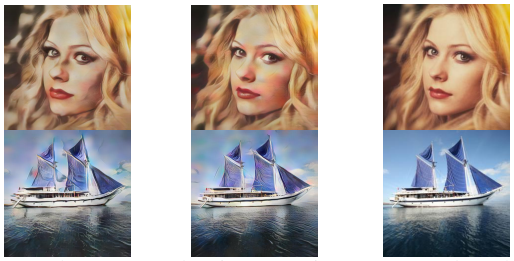
We conduct experiments to quantitatively evaluate the performance of our proposed models in both reverse and serial style transfer. A test set is built upon 1000 content images randomly sampled from the testing set of MS-COCO, with each of them transferred into 5 random styles that have never been used in the training phase. We perform reverse and serial style transfer with different models and compare the outputs with respect to their corresponding expectations. The averaged L2 distance, structural similarity (SSIM), and learned perceptual image patch similarity (LPIPS [25]) are used to measure the difference and the results are shown in Table 1. Both our models perform better than the baselines. Particularly, our two-stage model performs the best for reverse style transfer while the end-to-end model does so for serial style transfer. We believe that our two stage model benefits from its larger amount of encrypted information and the design of identity mapping, leading to the better result in reverse style transfer, and the end-to-end model shows its advantage in having less information to hide, making it more robust to the propagated error caused by serial style transfer.

4.3. Ablation Study

Here we perform ablation studies to verify the benefits of some design choices in our proposed models. Due to page limit, please refer to our supplement for more studies.

Identity mapping of two-stage model As described in Sec. 3.1.1, for the decoder D_{AdaIN}, we have an additional objective based on identity mapping. From the example results provided in Figure 8, we can see the ones produced by our D_{AdaIN} have less artifacts, which clearly demonstrate the benefits to the task of reverse style transfer brought by using identity mapping in our proposed model, in comparison to the decoder used in the typical style transfer method.

Using E_{inv} to recover v_t from I_{st} in end-to-end model There is a potential argument that we could replace E_{inv} with E_{VGG} due to the similarity between $\mathcal{L}_{inv}$ and $\mathcal{L}_{content}$ in our end-to-end model (please note that E_{VGG} is pre-trained and kept fixed). Hence we perform experiments accordingly in the task of reverse style transfer, and observe



AdaIN | Identity mapping | Ground truth

Figure 8: Comparison between results of using the AdaIN decoder trained w/o and w/ identity mapping in the task of reverse style transfer. It shows the AdaIN decoder trained w/ identity mapping generates results with less artifacts.

Content I_c | Stylized I_{st} | E_{inv} | E_{VGG}

Figure 9: Comparison between using E_{inv} and E_{VGG} to extract the target feature v_t in reverse style transfer. The ones of using E_{inv} show less influence from the stylization.

that the results of using E_{inv} preserve the overall content structure better, while the ones of using E_{VGG} tend to have severe interference from stylization as shown in Figure 9. The benefit of having E_{inv} in our model is thus verified. Please note that more results and videos are available in the supplementary material. All the source code and datasets (or trained models) will be made available to the public.

5. Conclusion

In this paper, we introduce the issues and artifacts that are inevitably introduced by typical style transfer methods in the scenarios of serial and reverse style transfer. We successfully address these problems by proposing a two-stage and an end-to-end approach while retaining the image quality of stylized output comparable to the state-of-the-art style transfer method simultaneously. Our methods are novel on uniquely integrating the steganography technique into style transfer for preserving the important characteristic of content features extracted from input photo, and the extensive experiments clearly verify the capability of our networks.

Acknowledgements This project is supported by MediaTek Inc., MOST-108-2636-E-009-001, MOST-108-2634-F-009 -007, and MOST-108-2634-F-009-013. We are grateful to the National Center for Highperformance Computing for computer time and facilities.

References

[1] S. Baluja. Hiding images in plain sight: Deep steganography. In *Advances in Neural Information Processing Systems (NIPS)*, 2017. 2, 3

[2] A. Cheddad, J. Condell, K. Curran, and P. Mc Kevitt. Digital image steganography: Survey and analysis of current methods. *Signal Processing*, 2010. 3

[3] A. A. Efros and W. T. Freeman. Image quilting for texture synthesis and transfer. *ACM Transactions on Graphics (TOG)*, 2001. 2

[4] L. A. Gatys, A. S. Ecker, and M. Bethge. Texture synthesis using convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2015. 1, 2

[5] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1, 2, 6, 7, 8

[6] S. Gupta, A. Goyal, and B. Bhushan. Information hiding using least significant bit steganography and cryptography. *International Journal of Modern Education and Computer Science (IJMECS)*, 2012. 2, 3

[7] J. Hayes and G. Danezis. Generating steganographic images via adversarial training. In *Advances in Neural Information Processing Systems (NIPS)*, 2017. 3

[8] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin. Image analogies. *ACM Transactions on Graphics (TOG)*, 2001. 1, 2

[9] X. Huang and S. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *IEEE International Conference on Computer Vision (ICCV)*, 2017. 2, 3, 6, 7, 8

[10] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 3

[11] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision (ECCV)*, 2016. 1, 2

[12] G. C. Kessler and C. Hosmer. An overview of steganography. *Advances in Computers*, 2011. 3

[13] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M.-H. Yang. Universal style transfer via feature transforms. In *Advances in Neural Information Processing Systems (NIPS)*, 2017. 3, 5, 6

[14] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision (ECCV)*, 2014. 6

[15] K. Nichol. Painter by numbers, wikiart. https://www.kaggle.com/c/painter-by-numbers, 2016. 6

[16] T. Pevný, T. Filler, and P. Bas. Using high-dimensional image models to perform highly undetectable steganography. In *Proceedings of the International Conference on Information Hiding (IH)*, 2010. 3

[17] L. Sheng, Z. Lin, J. Shao, and X. Wang. Avatar-net: Multi-scale zero-shot style transfer by feature decoration. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 3

[18] Y. Shih, S. Paris, F. Durand, and W. T. Freeman. Data-driven hallucination of different times of day from a single outdoor photo. *ACM Transactions on Graphics (TOG)*, 2013. 1

[19] F. Shiri, X. Yu, P. Koniuszand, and F. Porikli. Face destylization. In *Proceedings of the International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, 2017. 3

[20] F. Shiri, X. Yu, F. Porikli, R. Hartley, and P. Koniusz. Identity-preserving face recovery from portraits. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2018. 3

[21] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015. 3

[22] M. Tomei, M. Cornia, L. Baraldi, and R. Cucchiara. Art2Real: Unfolding the Reality of Artworks via Semantically-Aware Image-to-Image Translation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 3

[23] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *International Conference on Machine Learning (ICML)*, 2016. 1, 2, 3, 5, 6

[24] M. K. Vincent Dumoulin, Jonathon Shlens. A learned representation for artistic style. In *International Conference on Learning Representations (ICLR)*, 2017. 2

[25] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 8

[26] J. Zhu, R. Kaplan, J. Johnson, and L. Fei-Fei. Hidden: Hiding data with deep networks. In *European Conference on Computer Vision (ECCV)*, 2018. 2, 3, 4

[27] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2017. 3