Survey of Approaches to Non-Realistic Neural Style Transfer

Arina Varlamova, Victor Kitov
Lomonosov Moscow State University
Moscow, Russia
avarlamova1996@gmail.com, v.v.kitov@yandex.ru

***Abstract*—Despite researchers interest toward style transfer problem, there is still no foremost method available. Difficulties in problem formalization make a comparison of methods especially complicated. This paper covers twelve widely used algorithms in order to provide their comprehensive description and advantages one over another.**

I. Introduction

Style transfer is an actively studied problem at the intersection of art and technology. In general, style transfer is defined as a generation of a new image for which the content of one image will be preserved but displayed in the style of the second image (Fig. 1).

This kind of unrealistic image rendering provokes user interest. For example, the Prisma app was the leading application on Google Play in the CIS countries for the first ten days after release and continues to be popular. Of course, Prisma is not a single app with such functionality. Other examples are Artiso or Vinci mobile apps and online image stylings services such as alterdraw.com and deepart.io.

So, what makes style transfer a complex problem? The main challenge is a formalization of style. A style is characterized by multiple parameters such as the color composition, the shape of the strokes, the ratio of light and shadow, and many others. How to describe them mathematically? This question reveals the main difficulties a researcher in this field would face.

As in many other tasks related to image processing, methods based on convolutional neural networks have shown particular success in the field. This paper aims to provide a comprehensive comparison of the most applicable approaches to neural style transfer. Usually, in review articles that consider this problem in detail, a computational comparison is made or a description of algorithms is given without comparison. A distinctive feature of this work is the qualitative comparison in addition to the computational one.

Fig. 1. Example of style transfer. From left to right: style image, content image, generated stylized image produced by Gatys method

As part of the study, twelve methods were selected and compared according to user preferences and computational performance, such as memory and time consumption. Methods were selected based on the conceptual novelty of network architecture or loss function, mostly followed by a high number of citations. Among the considered methods are approaches based on optimization, patch replacement, fast generalized and non-generalized style transfer methods.

II. Neural style transfer approaches

A. Optimization-based offline neural methods

Convolutional neural networks were first applied to the style transfer problem in [1] (Gatys). The main idea here was to use the feature space of the VGG network [2]. Let I_c be the content image, VGG_c encoded input image at some layer l with N_l feature maps size of M_l. In these definitions, feature maps at layer l are represented as matrix $VGG_c^l \in \mathbb{R}^{N_l \times M_l}$, where VGG_{cij}^l is an activation of i-th convolution at j-th position of layer l.

To visualize information encoded in such a way the gradient descent on white noise image can be used. It would allow finding an image that would correspond to the same activation as the original one. Let I_g be this generated image and VGG_g^l its corresponding feature maps. The loss function then can be defined as follows:

$$\mathcal{L}_{content}(I_c, I_g, l) = \frac{1}{2} \sum_{i,j} (VGG_{cij}^l - VGG_{g_{ij}}^l)^2.$$

A randomly generated image I_g can be modified this way to meet the same activation as a content image I_c. When a convolution network is trained, it learns to transform an image into a numerical representation on every layer. With each layer, these representations get more and more sensitive to complex objects (content) but are not restricted to pixel values.

For style representation authors proposed to use correlations between activation of different convolutions at some layer l in form of Gram matrices $\mathcal{G}_{ij}^l \in \mathbb{R}^{N_l \times N_l}$, where $\mathcal{G}_{ij}^l$ is an inner product between i-th and j-th feature maps:

$$\mathcal{G}_{ij}^l = \sum_k VGG_{ik}^l VGG_{jk}^l.$$

Let I_s be a style image and VGG_s^l its corresponding feature maps. Then a contribution of l layer to loss function is:

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} ({\mathcal{G}_g}_{ij}^{\ l} - {\mathcal{G}_s}_{ij}^{\ l})^2,$$

and full style loss is defined as follows:

$$\mathcal{L}_{style}(I_s, I_g) = \sum_{l=0}^{L} w_l E_l,$$

where coefficients w_l aim to give weights to different layers.

So, in introduced terms, the style transfer can be achieved through a generation of a new image that would meet style and content loss simultaneously:

$$\mathcal{L}_{total}(I_g, I_c, I_s) = \alpha \mathcal{L}_{content}(I_g, I_c) + \beta \mathcal{L}_{style}(I_g, I_s), \quad (1)$$

with α and β as weight parameters.

The Gatys method usually provides visually pleasing results and still is a baseline method to compare with. A stylized image produced by this method can be seen on Fig. 1.

Another example of optimization-based approach proposed in [3] (STROSS). The loss function is also minimized through image I_g:

$$\mathcal{L}(I_g, I_c, I_s) = \frac{\alpha \ell_c + \ell_m + \ell_r + \frac{1}{\alpha}\ell_p}{2 + \alpha + \frac{1}{\alpha}},$$

but content and style loss are different.

For a set of layers $l_1, ..., l_L$ feature maps $VGG^{l_1}(I), ..., VGG^{l_L}(I)$ are interpolated to keep their resolutions the same as in I_c and are further concatenated. This approach allows getting an image representation that would include features from all network layers.

The main idea of the style loss function comes from earth mover distance (EMD). Relaxed earth mover distance (REMD) was proposed to speed up computations:

$$\ell_r = REMD(VGG_g, VGG_s) = max\left(\frac{1}{n}\sum_i \underset{j}{min} C_{ij}, \frac{1}{m}\sum_j \underset{i}{min} C_{ij}\right),$$

where C is a matrix that defines how far an element from VGG_g lies from an element from VGG_s:

$$C_{ij} = cos(VGG_{g_i}, VGG_{s_j}) = 1 - \frac{VGG_{g_i} \cdot VGG_{s_j}}{||VGG_{g_i}||||VGG_{s_j}||}.$$

Cosine distance ignores vector magnitude, so to avoid artifacts additional component ℓ_m is introduced:

$$\ell_m = \frac{1}{d}||\mu_g - \mu_s||_1 + \frac{1}{d^2}||\Sigma_g - \Sigma_s||_1,$$

where μ and Σ are mean and covariance.

The last component of style loss is ℓ_p which aims to match color palettes between I_g and I_c. It is achieved through REMD between pixels color in generated image $I_g^{(t)}$ at iteration t and style image I_s but with euclidean distance instead of cosine.

Content loss is based on matrices of paired cosine distance D^{I_c} and D^{I_g} between feature vectors from I_c and $I_g^{(t)}$ correspondingly:

$$\mathcal{L}_{content}(I_g, I_c) = \frac{1}{n^2}\sum_{i,j}\left|\frac{D_{ij}^{I_g}}{\sum_i D_{ij}^{I_g}} - \frac{D_{ij}^{I_c}}{\sum_i D_{ij}^{I_c}}\right|.$$

The method is applied iteratively to images with increasing resolution. Although computations take a significant amount of time, the method is highly competitive since it allows to keep large features from style images as shown in Fig. 2.

Fig. 2. Stylized image produced by STROSS method. Sunflowers are kept through style transfer

B. Fast non-generalized methods

A natural development of the Gatys method [1] was proposed in [4]. To achieve real-time performance authors used an additional transformer network $T(\cdot)$ which was trained with loss function introduced in Eq. 1:

$$\mathcal{L} = \mathcal{L}(N(I_c), I_c, I_s).$$

A simple encoder-decoder architecture was used in [4], but this approach got further improved in [5] with the usage of a U-Net architecture (UNet) [6]. Produced images are visually pleasing but mostly keep only color palette rather than the shape of stroke or other style characteristics. It was also noticed during evaluation that for some styles trained transformers tend to produce artifacts on a generated image as illustrated in Fig. 3.

C. Fast generalized methods

Another step was to give a transformation network ability to generalize on several styles. For instance, method MSG-NET [7] uses an additional layer to achieve it. Image is also generated through loss function as in Eq. 1, which is approximated as:

$$I_g = Dim^{-1}\left[Dim(VGG_c^l)^T W \mathcal{G}(VGG_s^l)\right]^T,$$

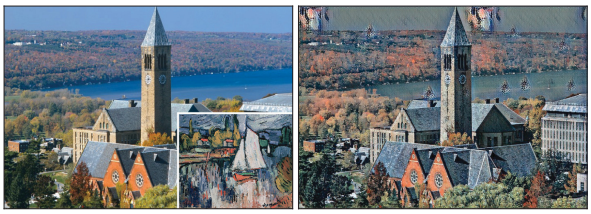

Fig. 3. Stylized image with artefacts generated by UNet method

where $W \in \mathbb{R}^{C_i \times C_i}$ is a weight matrix and $Dim(\cdot)$ transforms feature maps to $\mathbb{R}^{C_l \times (H_l W_l)}$ dimension.

Siamese network $N(\cdot)$ takes both content and style images and their feature maps are blended with the usage of layer W.

The method is resistant to artifacts, but sometimes color palettes do not align perfectly between produced and style images, as shown in Fig. 4.

Another approach is to use adaptive instance normalization (AdaIn) which was introduced in [8]. Instead of learnable parameters, authors proposed to use a special normalization layer:

$$AdaIN(I_c, I_s) = \sigma(I_s)\left(\frac{I_c - \mu(I_c)}{\sigma(I_c)}\right) + \mu(I_s).$$

Then an encoder-decoder network with fixed weights in the encoder part $Enc(\cdot)$ is trained. This network takes both images of content and style and applies a normalization afterward in feature space:

$$t = AdaIN(Enc(I_c), Enc(I_s)).$$

A decoder $Dec(\cdot)$ is then trained to transform t back to image space with a loss function defined as follows:

$$\mathcal{L}_c = ||Enc(Dec(t)) - t||_2,$$

and

$$\begin{aligned}\mathcal{L}_s = \sum_{l=1}^{L} ||\mu(VGG^l(Dec(t))) - \mu(VGG^l(I_s))||_2 + \\ \sum_{l=1}^{L} ||\sigma(VGG^l(Dec(t))) - \sigma(VGG^l(s))||_2.\end{aligned} \quad (2)$$

During the evaluation, the method showed a great performance on a variety of content and style images. It was observed, however, that sometimes it produces artifacts on a homogeneous background as shown in Fig. 5.

An approach with a fixed (but potentially unlimited) amount of styles was proposed in [9] (StyleBank). It is achieved through a network composed of three parts: an encoder, a decoder, and a StyleBank layer $\mathcal{K}$ with two learnable branches: an autoencoder ($Enc \rightarrow Dec$) and a styling branch ($Enc \rightarrow \mathcal{K} \rightarrow Dec$). A stylized image can be obtained by convolving content image features with the StyleBank layer

Fig. 4. Image stylized with MSG-Net method

Fig. 5. Artifacts on background produced by AdaIn method

$\{K_i\}, i = 1, 2, ..., n$ for n styles and decoding them with the decoder part $Dec(\cdot)$.

Two loss functions were used in the process of training. The first one is applied in the autoencoder branch:

$$\mathcal{L}_{\mathcal{I}}(I_c, I_g) = ||I_g - I_c||^2,$$

and in styling branch classical Gatys loss function from Eq. 1 were used.

Approach similar to AdaIN was introduced in [10] (WCT). Instead of using adaptive instance normalization authors proposed to apply whitening and coloring transformation in feature space.

Whitening transformation is performed as follows. First of all, a vector μ_c of mean is subtracted from feature maps VGG_c. They are then linearly transformed to $V\hat{G}G_c$ in such a way to meet $V\hat{G}G_c V\hat{G}G_c^T = I$:

$$V\hat{G}G_c = E_c D_c^{-\frac{1}{2}} E_c^T F_l,$$

where D_c a diagonal matrix consisting of eigenvalues of matrix $VGG_c VGG_c^T$ and E_c is an orthogonal matrix of eigenvectors such $VGG_c VGG_c^T = E_c D_c E_c^T$.

The next step is a coloring transform. As in whitening transform firstly a vector of μ_s of means is subtracted from VGG_s. A goal here is to obtain $VGG_{cs} : V\hat{G}G_{cs} V\hat{G}G_{cs}^T = VGG_s VGG_s^T$:

$$V\hat{G}G_{cs} = E_s D_s^{\frac{1}{2}} E_s^T \hat{F}_l.$$

To conclude, a vector μ_s is added to $V\hat{G}G_{cs}$.

A transforming network is then trained with the following loss function:

$$\mathcal{L} = ||I_o - I_i||_2^2 + \lambda ||VGG(I_o) - VGG(I_i)||_2^2.$$

An interesting peculiarity of the WCT method is how it affects human faces. As can be seen in Fig. 6, the form of eyes, lips, and nose had changed significantly during the style transfer process.

An approach with an additional transformer network is proposed in [11] (StylePredictor). A network $P(\dot{})$ takes a style image as input and predicts its vector representation $\vec{S}$. It is further concated with feature maps of styling transformer network which is trained with loss function introduced in Eq. 1.

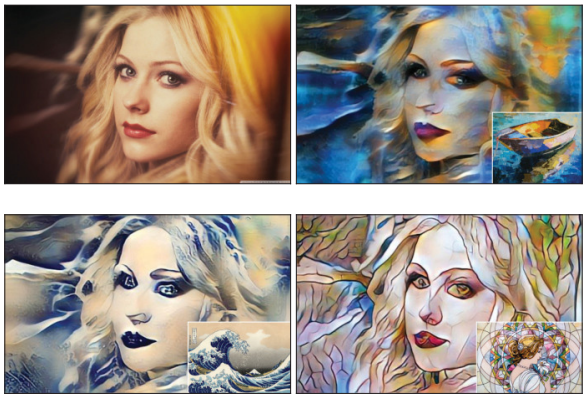

Fig. 6. Images stylized with WCT method

Similar to AdaIn, a background problem was observed for this method as well. To elaborate, several stylized images with artifacts on the sky area are demonstrated in Fig. 7.

An approach utilizing an attention mechanism (SANET) was proposed in [12]. It is based on an additional module that uses feature maps of content and style images to make a stylized internal representation:

$$Enc^i_{cs} = \frac{1}{C}\sum_{\forall j} exp(f(\overline{Enc^i_c})g(\overline{Enc^j_s}))h(Enc^j_s),$$

where $f(\overline{Enc_c}) = W_c\overline{Enc_c}$, $g(\overline{Enc_s}) = W_g\overline{F_s}$ and $h(Enc_s)) = W_hEnc_s$, and W_f, W_g, W_h are convolutions 1×1. Normalizing factor is represented by $C = \sum_{\forall j} exp(f(\overline{Enc^i_c})^T g(\overline{Enc^j_s}))$, where i – index of output position, and j is an index from set of all possible positions.

Convolution 1×1 is applied to obtained feature maps and then is added to feature maps for content image:

$$Enc_{csc} = Enc_c + Enc_{cs}.$$

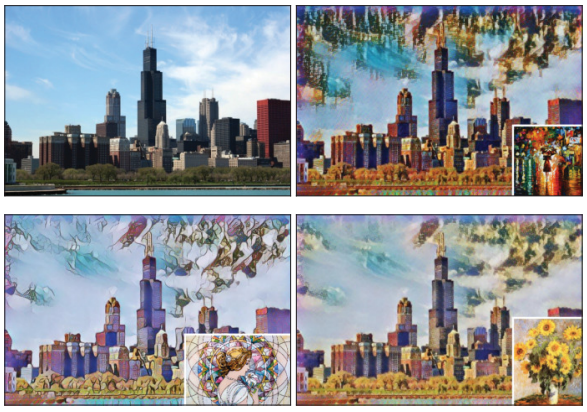

Fig. 7. Images stylized with StylePredictor method

Such feature maps are obtained for layers $Relu\_4\_1$ and $Relu\_5\_1$ are then combined into one:

$$Enc^m_{csc} = conv_{3\times3}(Enc^{r\_4\_1}_{csc} + upsampling(Enc^{r\_5\_1}_{csc})),$$

which is further decoded into stylized image.

A transformer network is trained with a following loss function:

$$\mathcal{L} = \lambda_c\mathcal{L}_c + \lambda_s\mathcal{L}_s + \mathcal{L}_{identity},$$

where $\mathcal{L}_s$ corresponds to a loss defined in Eq. 2 and a content loss is defined as:

$$\mathcal{L}_c = ||\overline{Enc(I_g)^{r\_4\_1}} - \overline{Enc(I_c)^{r\_4\_1}}||_2 + ||\overline{Enc(I_g)^{r\_5\_1}} - \overline{Enc(I_c)^{r\_5\_1}}||_2,$$

and $\mathcal{L}_{identity}$:

$$\mathcal{L}_{identity} = \lambda_{i1}(||(I_{cc} - I_c)||_2 + ||(I_{ss} - I_s)||_2) + \lambda_{i2}\sum_{i=1}^{L}(||VGG_i(I_{cc}) - VGG_i(I_c)||_2 + ||VGG_i(I_{ss}) - VGG_i(I_s)||_2),$$

where I_{cc} (or I_{ss}) is a stylized image from two images of content (style).

In the paper [13] authors paid attention to the "content loss" problem. They stated that in methods [8], [10], [14] content can be lost if style transfer is applied repeatedly. To address this issue authors proposed to use a projection-transfer-inversion approach instead of standard encoding-transfer-decoding (ArtFlow).

The architecture of the proposed network is composed of PFN (Projection Flow Network) blocks which are fully reversible. Each of them consists of three parts: additive coupling, reversible 1×1 convolution, and Actnorm.

1) Additive coupling: Let x and y be input and output tensors for additive coupling (AC). Then y is obtained as

$$x_a, x_b = split(x)$$
$$y_b = NN(x_a) + x_b$$
$$y = concat(x_a, y_b).$$

Here function $split(\cdot)$ splits a tensor into two parts along channels dimension and $NN(\cdot)$ is a some neural network where sizes of an input and an output are the same.

2) Reversible convolution: Since AC layer is applied only to half of the feature maps, it is necessary to change their number in order to make each dimension influence others. Such transformation may be defined with the usage of 1×1 convolution:

$$y_{i,j} = Wx_{i,j},$$

and the reverse transformation is $x_{i,j} = W^{-1}y_{i,j}$.

3) Actnorm: Actnorm is an alternative to batch normalization. It is defined as

$$y_{i,j} = w \otimes x_{i,j} + b,$$

where i, j is a spatial position in tensor, w and b are parameters of scale and translation for affine transformation which can be learned. Reverse transformation can be achieved as $x_{i,j} = \frac{(y_{i,j}-b)}{w}$.

As an algorithm for neural style transfer itself AdaIn or WCT are proposed.

D. Patch-based methods

One of the most early methods was proposed in [15]. It's based on procedure called style swap:

1) First step is to split feature maps VGG_s and VGG_c into intersecting patches $\{vgg_s^i\}_{i\in n_s}$ and $\{vgg_c^i\}_{i\in n_c}$.
2) For each patch of content the nearest style patch is selected, defined by normalized cross-correlation measure:

$$\phi_i^{ss}(I_c, I_s) = \underset{s_j(I_s), j=1,\ldots,n_s}{\operatorname{argmax}} \frac{\langle vgg_c^i, vgg_s^i \rangle}{||vgg_c^i|| \cdot ||vgg_s^i||}. \quad (3)$$

3) Each patch of content is replaced by the closest patch of style.
4) Patches are averaged into new feature map $\Phi^{ss}(I_c, I_s)$.

Stylized image can then be found as a solution for the optimization task

$$I_g(I_c, I_s) = \underset{I_g \in \mathbb{R}^{h\times w\times d}}{argmin} ||VGG_g - \Phi^{ss}(I_c, I_s)||_{\mathcal{F}}^2 + \lambda \ell_{TV}(I_g), \quad (4)$$

where

$$\ell_{TV}(I) = \sum_{i=1}^{h-1}\sum_{j=1}^{w}\sum_{k=1}^{d}(I_{i+i,j,k} - I_{i,j,k})^2 + \sum_{i=1}^{h}\sum_{j=1}^{w-1}\sum_{k=1}^{d}(I_{i,j+1,k} - I_{i,j,k})^2.$$

To speed up computations authors proposed to train another CNN to get an approximate solution for Eq. 4.

The method is generalized by multiple styles and shows good performance, however resulting images do not keep all style features. The color scheme is also preserved poorly as can be seen in Fig. 8.

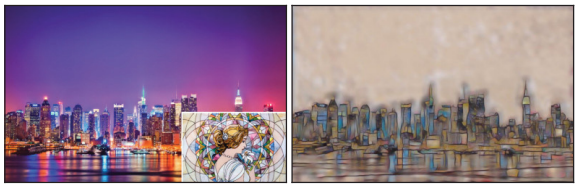

Fig. 8. Image stylized with StyleSwap method

Fig. 9. Image stylized with Avatar-Net method

An approach presented in [14] (AvatarNet) unifies methods introduced in [15], [10], [8]. Firstly a projection of feature maps Enc_c and Enc_s into a similar space is performed:

$$\overline{Enc_c} = W_c \otimes (Enc_c - \mu(Enc_c)),$$
$$\overline{Enc_s} = W_s \otimes (Enc_s - \mu(Enc_s)),$$

where W_c and W_s are some form of decorrelating matrices.

The second step is to replace patches from $\overline{Enc_c}$ with the closest patches of $\overline{Enc_s}$ as in Eq. 3. A coloring transformation is then applied to an obtained representation Enc_{cs}.

A transformer network is trained as a simple autoencoder with the following loss function:

$$\mathcal{L} = ||I_c - I_g||_2^2 + \lambda_1 \frac{1}{|\mathcal{I}|}\sum_{l\in\mathcal{I}} ||VGG_c^l - VGG_g^l||_2^2 + \lambda_2 \mathcal{L}_{TV}(I_g),$$

where $\mathcal{I}$ contains layers conv1_1, conv2_1, conv3_1 and conv4_1.

As in StyleSwap, this method still shows poor performance on keeping main style features, but the color scheme is preserved more precisely as shown in Fig. 9.

III. Evaluation

11 images of content and 16 images of style were selected to compare methods. For evaluation, official Pytorch versions or unofficial popular Pytorch versions were used. For the StylePredictor method a Pytorch version wasn't found, so a TensorFlow-based one was used instead.

NVIDIA Tesla P100 GPU was used for calculation purposes. For qualitative evaluation, 4 styles and 6 contents (24 combinations in total) were selected. Respondents were presented with a selection of stylized images to choose the most visually pleasing version in a manner of a blind study. Image examples are shown in Fig. 10.

The comparison was made based on user preferences (n=74). Fig. 11 (a and b) represents the comparison based on the overall amount of votes per method and an amount of content-style pairs where the method got the most votes. Fig. 11 (c) shows the entropy calculated on percentages of votes to different content-style pairs (the lower value indicates the pair where the most popular method got more votes compared to others). It can be easily seen that respondents prefer the STROSS method, since it is leading by overall votes, and got the highest score in pairs amount. It also leads with more certainty when chosen (has lower values of entropy).

The STROSS method, however, is optimization-based. It cannot be used in real-time and shows the worst performance

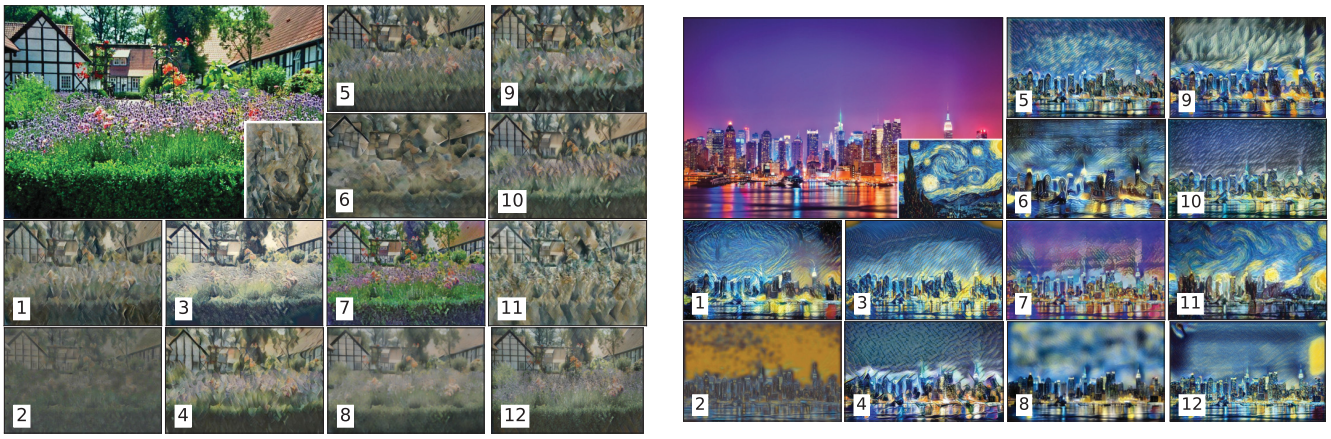

Fig. 10. An example of stylized images for respondents. Left image represents high level of disagreement between annotators (high entropy). Right image represents high level of concordance between annotators (low entropy). Upper left corner for both images contain style and content images. Other images are stylized version of images for: 1) Gatys, 2) StyleSwap, 3) MSG-Net, 4) AdaIN, 5) StyleBank, 6) WCT, 7) StylePredictor, 8) Avatar-Net, 9) SANET, 10) ArtFlow, 11) STROSS, 12) Unet methods

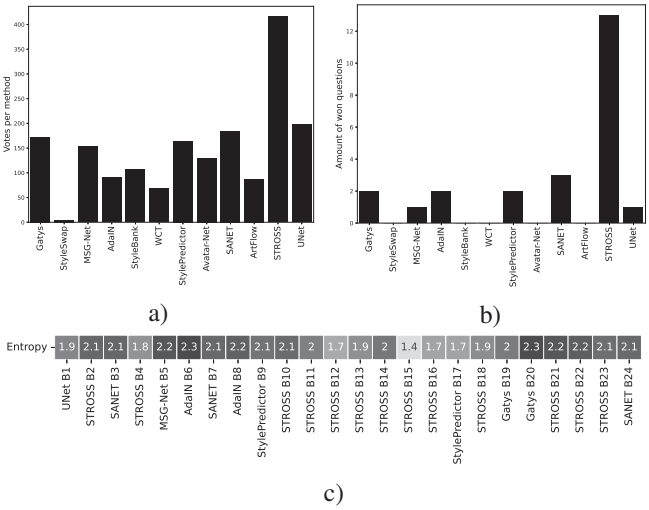

Fig. 11. Comparison of methods by user preferences

(Table I). Considering computational resources the optimal methods are AdaIn, StylePredictor, SANET, and UNet with a mean computational time of fewer than 0.1 seconds and mean memory consumption of less than 3000 MB. UNet and SANET are also second and third best methods based on user preferences.

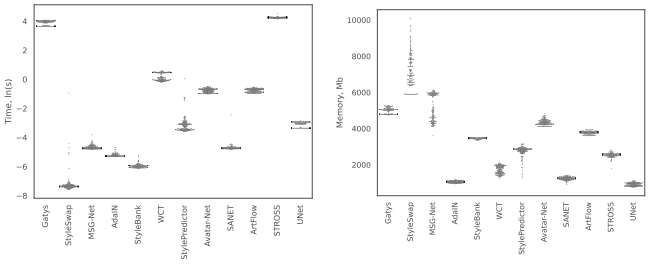

Fig. 12. Comparison of time and memory consumption

TABLE I. RESOURCES BASED COMPARISON

Method	Time (s)	Memory (MB)
Gatys [1]	47.17 ± 7.998	4955.686 ± 163.938
StyleSwap [15]	**0.003 ± 0.030**	7307.233 ± 958.834
MSG-Net [7]	0.009 ± 0.002	5168.98 ± 726.420
AdaIN [8]	0.005 ± 0.001	1053.921 ± 52.421
StyleBank [9]	**0.003 ± 0.0004**	3457.338 ± 35.056
WCT [10]	1.265 ± 0.325	1715.886 ± 221.031
StylePredictor [11]	0.055 ± 0.089	2737.717 ± 317.063
Avatar-Net [14]	0.465 ±0.066	4342.512 ± 150.471
SANET [12]	0.009 ± 0.006	1247.193 ± 82.209
ArtFlow [13]	0.466 ± 0.057	3777.144 ± 80.016
STROSS [3]	70.356 ± 3.012	2545.589± 107.3
UNet [5]	0.044 ± 0.008	**914.560724 ± 91.432**

Fig. 12 shows how resources consumption changes depending on different image sizes. In AdaIn, SANET, and UNet methods increased size has a lower impact on performance compared to StylePredictor.

In addition, in Fig. 14 shown stylized images with higher amounts of votes. For all four presented style images, there is at least one content image where the method STROSS was chosen as the best one. The same, however, cannot be said about content images. Fig. 13 demonstrates an example of such image and corresponding stylizations.

This may potentially indicate, that for the STROSS method the greater impact on result is made by content image rather than style.

Fig. 13. Stylized images produced by STROSS method for flowers content image

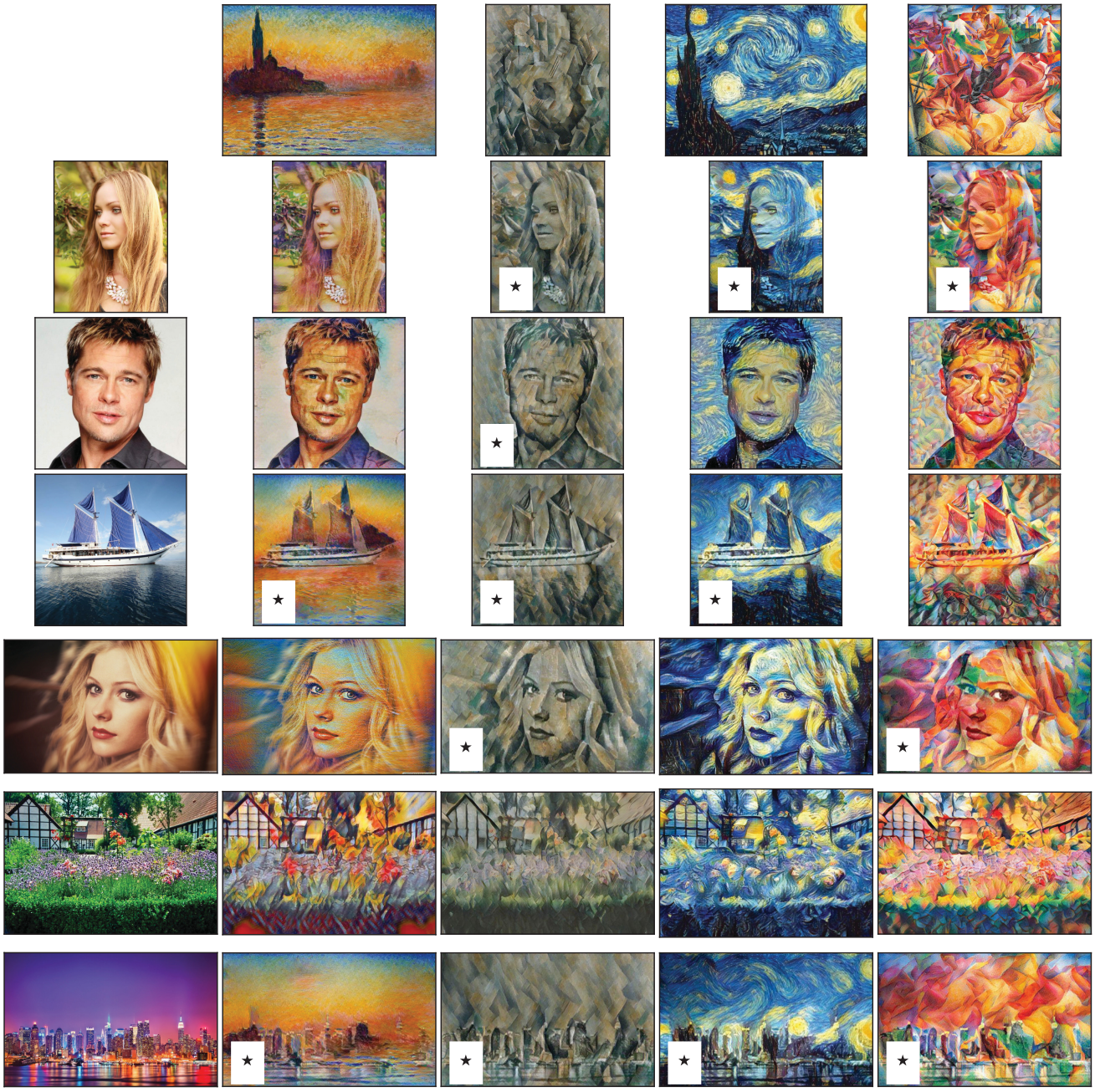

Fig. 14. Stylized images with the highest amount of votes. ⋆ indicates images produced by STROSS

IV. Conclusion

Usage of convolutional neural networks allowed to achieve greater performance in the image processing field, which is also true for the problem of style transfer. The absence of common comparison techniques, however, makes it difficult to understand the advantage of one method over another.

This survey offers a detailed description of widely used methods of neural style transfer and provides their qualitative and quantitative comparison. STROSS showed the best results based on user preferences. This method, however, cannot be used in a variety of applications requiring real-time performance. In the case of limited resource availability, the best results were achieved by UNet, which can be applied only to one style, and SANET which does not have this disadvantage.

References

[1] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2414–2423.

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
[3] N. Kolkin, J. Salavon, and G. Shakhnarovich, "Style transfer by relaxed optimal transport and self-similarity," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 051–10 060.
[4] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *European conference on computer vision*. Springer, 2016, pp. 694–711.
[5] M.-M. Ho, J. Zhou, and Y. Fan, "Respecting low-level components of content with skip connections and semantic information in image style transfer," in *European Conference on Visual Media Production*, 2019, pp. 1–9.
[6] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.
[7] H. Zhang and K. Dana, "Multi-style generative network for real-time transfer," in *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, 2018, pp. 0–0.
[8] X. Huang and S. Belongie, "Arbitrary style transfer in real-time with adaptive instance normalization," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1501–1510.
[9] D. Chen, L. Yuan, J. Liao, N. Yu, and G. Hua, "Stylebank: An explicit representation for neural image style transfer," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1897–1906.
[10] 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*, 2017.
[11] G. Ghiasi, H. Lee, M. Kudlur, V. Dumoulin, and J. Shlens, "Exploring the structure of a real-time, arbitrary neural artistic stylization network," 2017.
[12] D. Y. Park and K. H. Lee, "Arbitrary style transfer with style-attentional networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5880–5888.
[13] J. An, S. Huang, Y. Song, D. Dou, W. Liu, and J. Luo, "Artflow: Unbiased image style transfer via reversible neural flows," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 862–871.
[14] L. Sheng, Z. Lin, J. Shao, and X. Wang, "Avatar-net: Multi-scale zero-shot style transfer by feature decoration," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8242–8250.
[15] T. Q. Chen and M. Schmidt, "Fast patch-based style transfer of arbitrary style," 2016.