MVStylizer: An Efficient Edge-Assisted Video Photorealistic Style Transfer System for Mobile Phones

Ang Li
Duke University
ang.li630@duke.edu

Chunpeng Wu
Duke University
chunpeng.wu@duke.edu

Yiran Chen
Duke University
yiran.chen@duke.edu

Bin Ni
Quantil Inc.
nibin@quantil.com

ABSTRACT

Recent research has made great progress in realizing neural style transfer of images, which denotes transforming an image to a desired style. Many users start to use their mobile phones to record their daily life, and then edit and share the captured images and videos with other users. However, directly applying existing style transfer approaches on videos, i.e., transferring the style of a video frame by frame, requires an extremely large amount of computation resources. It is still technically unaffordable to perform style transfer of videos on mobile phones. To address this challenge, we propose MVStylizer, an efficient edge-assisted photorealistic video style transfer system for mobile phones. Instead of performing stylization frame by frame, only key frames in the original video are processed by a pre-trained deep neural network (DNN) on edge servers, while the rest of stylized intermediate frames are generated by our designed optical-flow-based frame interpolation algorithm on mobile phones. A meta-smoothing module is also proposed to simultaneously upscale a stylized frame to arbitrary resolution and remove style transfer related distortions in these upscaled frames. In addition, for the sake of continuously enhancing the performance of the DNN model on the edge server, we adopt a federated learning scheme to keep retraining each DNN model on the edge server with collected data from mobile clients and syncing with a global DNN model on the cloud server. Such a scheme effectively leverages the diversity of collected data from various mobile clients and efficiently improves the system performance. Our experiments demonstrate that MVStylizer can generate stylized videos with an even better visual quality compared to the state-of-the-art method while achieving 75.5× speedup for 1920×1080 videos.

CCS CONCEPTS

• **Information systems** → **Mobile information processing systems**; **Multimedia content creation**.

KEYWORDS

edge computing, video style transfer, mobile phone

ACM Reference Format:
Ang Li, Chunpeng Wu, Yiran Chen, and Bin Ni. 2020. MVStylizer: An Efficient Edge-Assisted Video Photorealistic Style Transfer System for Mobile Phones. In *International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing (Mobihoc '20), October 11–14, 2020, Boston, MA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3397166.3409140

1 INTRODUCTION

In the past decade, deep neural networks (DNNs) have been widely applied in image transformation tasks, including style transfer [12, 23, 49], semantic segmentation [35], super resolution [9, 23], etc. DNN-based style transfer is one of the most popular techniques in image transformation, and has led to many successful industrial applications with significant commercial impacts, such as Prisma [25] and DeepArt [8]. The DNN-based style transfer aims at transforming an input image into a desired output image according to a user-specified style image. Specifically, the DNN model is trained to search for a new image that has similar neural activations as the input image's and similar feature correlations as the style image's. Figure 1 shows one example of directly applying a pre-trained DNN model to perform style transfer. Here the input image is one extracted frame from a video of road trip recorded in the daytime, while the style image is a similar scene captured at dusk. After performing stylization, the input frame is successfully transformed to the dusky scene while keeping the content unchanged as the input frame.

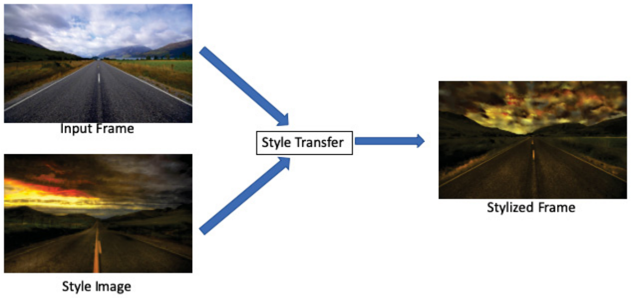


Figure 1: An example of video style transfer.

Mobihoc '20, October 11–14, 2020, Boston, MA, USA

ACM ISBN 978-1-4503-8015-7/20/10…$15.00
https://doi.org/10.1145/3397166.3409140

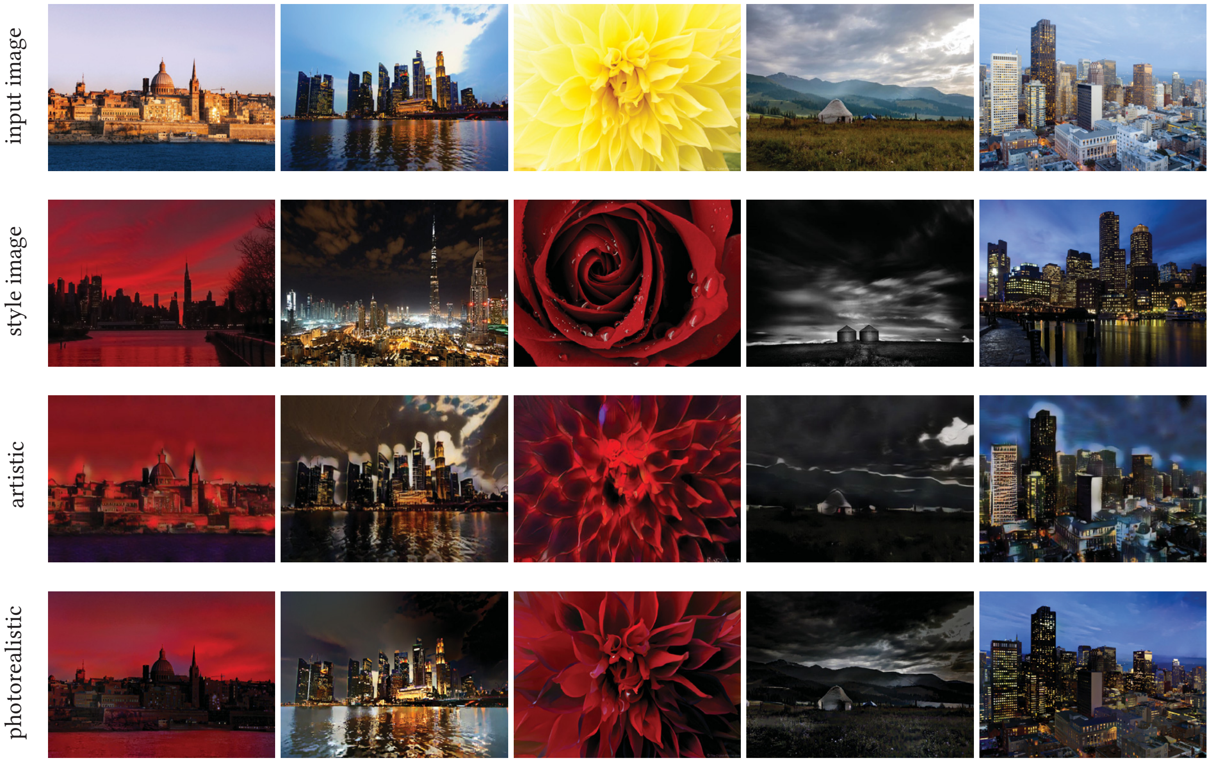


Figure 2: Examples of artistic style transfer [19] and photorealistic style transfer [36].

However, the naive extension of style transfer from images to videos is very challenging: frame-by-frame transformations are very slow [3, 18] even running on powerful GPUs. Although many users start to use mobile phones to record their daily life, and then edit and share images and videos on social networks or with friends, performing style transfer of videos on mobile phones is still unaffordable till very recently due to the limited computing sources on the phones. In addition to efficiency issues, “photorealistic” style transfer is another critical challenge. Figure 2 shows several examples of artistic style transfer and photorealistic style transfer. Even though the contents of artistic stylized images are distorted, these distortions can be tolerated and hard to be detected by human eyes due to the artistic attribute. However, compared with artistic style transfer, the target of photorealistic style transfer is to achieve photorealism, which requires loyally preserving the content structure in the stylized image. Humans are able to evaluate the visual quality of the photorealistic stylized images. It is necessary to explore the approach that can achieve efficiently photorealistic style transfer of videos on mobile phones while keeping high visual quality of the stylized videos. Some works have been done for style transfer of images and videos, most of which focus on improving either the visual quality of stylization [2, 26, 27, 43] or efficiency of style transfer [4, 10, 18, 29, 49]. However, performing photorealistic style transfer of videos on mobile phones under the constraints of computing resources has not been fully investigated.

In this paper, we propose MVStylizer – an efficient edge-assisted video style transfer system for mobile phones. Instead of performing transformation frame by frame, MVStylizer processes only extracted key frames using pre-trained DNN models on the edge server while the rest of intermediate frames are generated on-the-fly using our proposed optical-flow-based frame interpolation algorithm on mobile phones. The interpolation is done by exploiting the optical flow information between the intermediate frames and key frames. The reason why we choose edge servers but not cloud servers is that edge servers are closer to users and hence, are able to provide real-time response to mobile phones. Edge-cloud federated learning is adopted in our architecture to continuously improve the performance of the DNN models on edge server: Each DNN model on the edge server keeps re-training while performing stylization with collected data from mobile clients, and each edge server will sync the model with cloud server where a global DNN model is maintained when it is idle. Meanwhile, the global DNN is also a backup of the edge DNN, from which an edge server can be quickly restored if it suddenly crashes. We also conduct experiments to quantitatively and qualitatively evaluate our proposed system. The experimental results demonstrate that MVStylizer can successfully stylize videos with even better visual quality compared to the state-of-the-art method while achieving significant speedup with high resolutions.

The main contributions of this paper are summarized as follows:

- To the best of our knowledge, MVStylizer is the first mobile system for performing photorealistic style transfer of videos.
- An optical-flow-based frame interpolation algorithm is proposed to accelerate style transfer of videos on mobile phones.
- A meta-smoothing module is designed to efficiently tackle two problems in an end-to-end learning manner: dynamically upscaling a stylized image to multiple/arbitrary resolution and removing style transfer related distortions in these upscaled versions.
- An edge-cloud federated learning scheme is applied to continuously improve the performance of DNN-based stylizer on edge servers.
- We implement a prototype system and conduct experiments to quantitatively evaluate MVStylizer, which demonstrates that it can perform video style transfer of videos on mobile phones in an effective and efficient way.

The rest of this paper is organized as follows. Section 2 provides background information and section 3 presents the system overview and details of core modules. Section 4 shows the evaluation results. Section 5 reviews the related work. Section 6 concludes this paper.

2 BACKGROUND

2.1 Photorealistic style transfer

As Figure 1 shows, the photorealistic style transfer extracts human-perceptual "style" features from a style image and applies these style features to "decorate" the content image without changing objects' semantic structures in the original photo. Similar to artistic style transfer [22], photorealistic style transfer requires high style faithfulness for the stylized image. Unlike artistic style transfer, however, generating a stylized image with high photorealism is essential in photorealistic style transfer, which remains as a key challenge. Based on previous artistic style transfer methods [23, 31], theoretical analysis on feature correlation [32] or photorealism loss [36] are often adopted to quantitatively approximate or simulate human evaluation on photorealism.

2.2 Device to edge offloading

Offloading computational intensive tasks to edge servers is feasible and promising way to address the challenge of limited computation resources on edge devices (e.g., smartphones) [51]. Such strategy has been widely applied to many applications. Ran *et al.* [42] design a framework to dynamically determine the offloading strategy for the object detection task based on the network conditions. Yi *et al.* [52] propose a system named Lavea, which offloads the computation from the clients to nearby edge servers to provide video analytics service with low latency. Jeong *et al.* [21] propose an approach to offload DNN computations to nearby edge servers in the context of web apps. Chen *et al.* [7] conduct an empirical study to evaluate the performance of several edge computing applications in terms of latency.

In this work, we also adopt the similar strategy to offload key frames to the edge servers, where those key frames are processed by the pre-trained DNN models. We leave the lightweight optical-flow-based interpolation for intermediate frames on mobile phones.

3 OUR PROPOSED MVSTYLIZER SYSTEM

3.1 The System Overview

In this work, we propose MVStylizer for efficiently performing photorealistic style transfer for videos on mobile phones. Due to the constrained computation resources on mobiles, edge servers are leveraged to speed up the style transfer. Moreover, specific technical approaches are proposed to address two critical challenges in this system.

First, performing frame-by-frame stylization is still technically unaffordable, even with the assistance of edge servers. We propose an optical-flow-based frame interpolation algorithm and a meta-smoothing module to speed up the stylization process. Specifically, only extracted key frames will be processed by a pre-trained DNN on the edge server, while the rest of intermediate frames will be generated on-the-fly using our proposed optical-flow-based frame interpolation algorithm on mobile phones. The interpolation is done based on the stylized key frames and pre-computed optical flow information between key frames and intermediate frames. In addition, the meta-smoothing module is integrated in the edge DNN for handling upscaling and distortion issues of the stlyized frame in a single operation, accelerating the style transfer of key frames on the server.

Second, the edge DNN may be trained with limited data so that the optimal performance is not achieved. Therefore, we propose an edge-cloud federated learning scheme to continuously improve the performance of the edge DNN. While the edge server offers stylization service, the edge DNN keeps retraining based on the collected data from mobile clients. The updated edge DNN will be synced with a cloud DNN when the edge server is idle. Note that the cloud DNN has the same DNN architecture as edge DNNs, and the cloud DNN is also maintained as a backup of the edge DNN in case some edge server crashes. The cloud server is updated with averaging the aggregated parameters from each server, and the updated parameters of the cloud DNN will be synced with each edge DNN.

As illustrated in Figure 3, MVStylizer consists of three major modules: *optical-flow-based frame interpolation*, *edge DNN* and *cloud DNN*. The work flow is as follows: When a user performs the style transfer of a video on the mobile phone, the extracted key frames of this video will be sent to the edge server associated with a user-specified style image. The key frames can be identified using either standard H.264 video codec or the content-based method [40]. Afterwards, a pre-trained DNN will perform transformation on received key frames according to the style image on the edge server. While performing stylization, the edge DNN will continuously keep retraining based on the specific evaluation metric. The updated edge DNN will be synced with a cloud DNN when the edge server is idle. The post-processed stylized key frames will then be returned to the mobile client. Finally, the stylized intermediate frames will be interpolated by the proposed optical-flow-based frame interpolation algorithm.

Next, we will illustrate more details about each module.

3.2 Optical-flow-based Frame Interpolation

Transferring the style of a video on a mobile phone is always challenging due to its limited available computing resource. In this work,

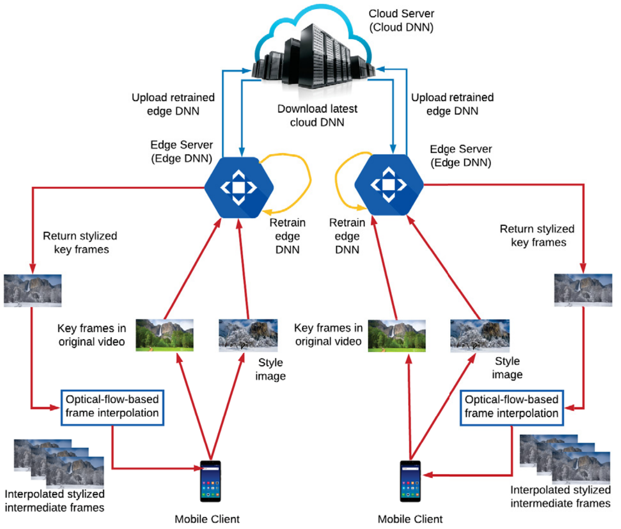


Figure 3: The system design of MVStylizer.

we attempt to process only a few key frames on the edge DNN, while a lot of more stylized intermediate frames will be interpolated based on the optical flow information. An optical-flow-based frame interpolation algorithm is designed for interpolating stylized intermediate frames based on the optical flow information between intermediate frames and key frames in the original video.

Given a video consists of n frames, among which there are j key frames $(k_0, \ldots, k_{j-1})$ and m intermediate frames $(i_0, \ldots, i_{m-1})$ where $j + m = n$. The optical flow information can be computed as $f_{sp} = F(i_s, k_p)$ for any intermediate frame i_s between the key frame k_p and k_{p+1}, where F is an optical flow estimator [38]. The f_{sp} is a two dimensional flow field, representing the displacement of each pixel from k_p to i_s. For example, given the location of a pixel $\mathbf{p}$ in k_p as $k_p(x, y)$, and assume the location of $\mathbf{p}$ in i_s is $k_p(x + \Delta x, y + \Delta y)$, that is, $i_s(x, y) = k_p(x + \Delta x, y + \Delta y)$. The optical flow of $\mathbf{p}$ from k_p to i_s is $(\Delta x, \Delta y)$. Again, only key frames will be sent to the edge server for performing style transfer using the edge DNN, and the stylized key frames $(\hat{k}_0, \ldots, \hat{k}_{j-1})$ will be returned to the mobile phone. Based on the stylized key frames $(\hat{k}_0, \ldots, \hat{k}_{j-1})$ and optical flow information $(f_0, \ldots, f_{m-1})$, we are able to generate stylized intermediate frames $(\hat{i}_0, \ldots, \hat{i}_{m-1})$ by spatial warping. We adopt the bilinear interpolation for generating the stylized intermediate frames, such as:

$$\hat{i}_p = I(f_{pq}, \hat{k}_q), \tag{1}$$

where I is a bilinear interpolation kernel. The detailed workflow of the optical-flow-based interpolation algorithm is described in **Algorithm 1**.

3.3 Edge DNN

In MVStylizer, the edge DNN is a stylizer. As depicted in Figure 4, it consists of four modules: a pretrained VGG [48] encoder for feature extraction, a colorization module for integrating style features into content features, a decoder with mirrored VGG layers for generating stylized image, and our proposed *meta-smoothing module*.

Algorithm 1 Optical-flow-based frame interpolation

```
Input:
    Stylized key frames $(\hat{k}_0, \ldots, \hat{k}_{j-1})$;
    Optical flow information between key frames and intermediate frames $(f_0, \ldots, f_{m-1})$;
    Index of each frame in original video
Output:
    Stylized intermediate frames $(\hat{i}_0, \ldots, \hat{i}_{m-1})$;
$p = 0, q = 0$
for p = 0 : m:
    if Index $(\hat{k}_q)$ < Index $(f_p)$ < Index $(\hat{k}_{q+1})$
        $\hat{i}_p = I(f_p, \hat{k}_q)$;
    else
        $q = q + 1$
    end if
end for
```

The meta-smoothing module is designed to tackle two major issues in an *end-to-end* learning manner: dynamically upscaling the decoder's output to multiple/arbitrary resolution and removing the style transfer related distortions in these upscaled versions. First, popular encoder-decoder for style transfer [20, 31] often adopt fixed architectures (VGG, ResNet [15], MobileNet [16], *etc.*). Even when replacing the encoder-decoder with a more flexible DNN transformer [46] to support manually designed architecture, the generated image still needs to be upscaled to the target resolution, and the upscaling efficiency is an issue. Second, the stylized image generated by the decoder has distortions incurred by the style transfer. Those distortions become even worsened in the upscaling process as observed in previous super resolution studies [17, 47]. In other words, adopting only super resolution methods cannot synthesize a satisfactory stylized image with high resolution.

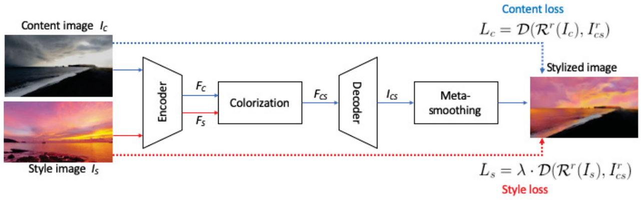


Figure 4: The design of DNN-based stylizer with meta-smoothing.

In our method, as shown in Figure 4, a content image $\boldsymbol{I_c}$ and a style image $\boldsymbol{I_s}$ are rescaled to the same size and fed into our stylizer's pretrained VGG encoder. The encoder will extract the content features $\boldsymbol{F_c}$ (blue) and style features $\boldsymbol{F_s}$ (red). After that, the colorization module will "colorize" $\boldsymbol{F_c}$ with $\boldsymbol{F_s}$. The feature maps $\boldsymbol{F_c}$ and $\boldsymbol{F_s}$ have the same dimension, since the input $\boldsymbol{I_c}$ and $\boldsymbol{I_s}$ have been rescaled to the same size. The dimensions of $\boldsymbol{F_c}$ and $\boldsymbol{F_s}$ are denoted as $H \times W \times C$ where $H \times W$ is the feature map's height and width and C is the channel number. The colorization is processed along channel dimensions of $\boldsymbol{F_c}$ and $\boldsymbol{F_s}$. Specifically, the feature maps $\boldsymbol{F_c}$ is denoted as $\{\boldsymbol{x}_c^i | i = 1, 2, ..., H \times W\}$ where the vector length $|\boldsymbol{x}^i| = C$. Similarly, the feature maps $\boldsymbol{F_s}$ is denoted as $\{\boldsymbol{y}_s^j | j = 1, 2, ..., H \times W\}$ where the vector length $|\boldsymbol{y}^j| = C$. The colorization module outputs the colorized content features $\boldsymbol{Fcs}$, which is denoted as $\{\boldsymbol{z}_{cs}^k | z = 1, 2, ..., H \times W\}$ and the vector length $|\boldsymbol{z}^k| = C$. The colorization is defined as:

$$\boldsymbol{z}_{cs}^k = \sum_{m=1}^{H \times W} (\boldsymbol{y}_s^m - \boldsymbol{x}_c^k). \tag{2}$$

Finally, $\boldsymbol{Fcs}$ is fed into the decoder, which generates a stylized image denoted as $\boldsymbol{I}_{cs}$. The meta-smoothing module upscales and smoothes $\boldsymbol{I}_{cs}$ such as

$$\boldsymbol{I}_{cs}^r = \mathcal{P}(\boldsymbol{I}_{CS}, \boldsymbol{W}_u^r) * \boldsymbol{W}_s, \tag{3}$$

where $\mathcal{P}(\cdot)$ is a deconvolution operator, r ($r > 0$) is an upscaling factor that is specified by an application user, $\boldsymbol{W}_u^r$ is our improved convolution kernel which is used in the deconvolution operator $\mathcal{P}(\cdot)$, $\boldsymbol{W}_s$ is convolution kernel $\boldsymbol{W}_s$ for smoothing distortions incurred by the style transfer, and N_s is number of feature maps output by convolution with $\boldsymbol{W}_s$. The convolution kernel $\boldsymbol{W}_u^r$ for deconvolution is defined in Equation 4:

$$\boldsymbol{W}_u^r = \begin{cases} \boldsymbol{W}_u^1 & r = 1 \\ Q(r) * \boldsymbol{W}_u^1 & r \neq 1 \end{cases} \tag{4}$$

where the function $Q(\cdot)$ is to construct a matrix filled with r. If the upscaling factor $r = 1$, the deconvolution operator $\mathcal{P}(\cdot)$ only executes convolution without upscaling. Therefore, $\boldsymbol{W}_u^1$ is denoted as a meta convolution kernel of our meta-smoothing module. If the upscaling factor $r \neq 1$, $\boldsymbol{W}_u^r$ is calculated using the meta convolution kernel $\boldsymbol{W}_u^1$ and the upscaling factor r. For $\boldsymbol{W}_u^r$ and $\boldsymbol{W}^s$, each filter's dimensions are respectively denoted as $H_u \times W_u$ and $H_s \times W_s$.

The objective function is defined as Equation 5:

$$L = \underbrace{\mathcal{D}(\mathcal{R}^r(\boldsymbol{I}_c), \boldsymbol{I}_{cs}^r)}_{\text{content loss}} + \lambda \cdot \underbrace{\mathcal{D}(\mathcal{R}^r(\boldsymbol{I}_s), \boldsymbol{I}_{cs}^r)}_{\text{style loss}}, \tag{5}$$

where the function $\mathcal{D}(\cdot)$ is to measure the perceptual distance [23], the function $\mathcal{R}^r(\cdot)$ is to rescale input data with respect to the upscaling factor r, and λ is a scale factor. $\mathcal{R}^r(\boldsymbol{I}_c)$, $\mathcal{R}^r(\boldsymbol{I}_s)$, and $\boldsymbol{I}_{cs}^r$ have the same size. In Equation (5), the first part evaluates the *content loss* defined as the perceptual distance between the input content image and the stylized image, and the second part evaluates the *style loss*, which is the perceptual distance between the input style image and the stylized image.

3.4 Cloud DNN

The cloud DNN has an identical architecture as the edge DNN. It is designed for aggregating the updated parameters of edge DNNs in the edge-cloud federated learning process. Algorithm 2 presents the details of the edge-cloud federated learning procedure. Suppose that there exist N participated edge servers, the parameters of each corresponding edge DNN are denoted as $(\theta_1, \cdots, \theta_N)$. $\bar{\theta}$ represents the parameters of the cloud DNN. When a mobile client sends a style transfer request to an edge server associated with the video data, the edge DNN will perform the style transfer on received data while retraining the model based on those data. Then, the updated parameters θ_i will be uploaded to the cloud server when the edge server is idle. The parameters of the cloud DNN can be updated by averaging the parameters of each edge DNN as $\bar{\theta}^t = \sum_{i=1}^{N} \theta_i^t$. Finally, the updated $\bar{\theta}$ will be distributed to each edge server, where the edge DNN can be updated with the latest $\bar{\theta}$. The entire process will be continuously repeated for improving the performance of the edge DNN in an efficient way. The effectiveness of the edge-cloud federated learning is evaluated in Section 4.8.

Algorithm 2 Edge-Cloud Federated Learning

```
Input:
    $N$ collected video data $\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3, \dots, \mathcal{D}_N$ on each edge server;
Output:
    federated learned neural network parameters $\bar{\theta}$;
 1: Deploy the pre-trained DNN model on the cloud server;
 2: Distribute the cloud DNN to each server as edge DNN for initialization;
 3: Initialize time stamp $t = 0$;
 4: Client executes:
 5:     for $i = 1 : N$:
 6:         Participated edge server receives latest model parameters $\bar{\theta}^t$ from the cloud server;
 7:         Update corresponding edge DNN $\theta_i^{t+1} = \bar{\theta}^t$;
 8:         $t = t + 1$;
 9:         Retrain the edge DNN based on $\mathcal{D}_i$ and send the updated parameters $\theta_i^t$;
10: Server executes:
11:     Compute average parameters $\bar{\theta}^t = \sum_{i=1}^{N} \theta_i^t$;
12:     Send aggregated parameters $\bar{\theta}^t$ to each participated edge server;
13: Continuously repeat from steps 4 to 12 for improving the performance of edge DNN;
```

Not only works for the federated learning, the cloud DNN is also maintained as a backup of the edge DNN, from which an edge server can be quickly restored after it suddenly crashes.

4 EVALUATIONS

4.1 Experiment Setup

We implement a prototype system of MVStylizer which is composed of Google Pixel 2 and servers running on Ubuntu 16.04. The mobile device is equipped with Qualcomm Snapdragon-835@2.35GHz. In this experiment, both the edge server and cloud server are equipped with an Intel Xeon E5-2630@2.6GHz, 128G RAM and a NVIDIA TITAN X Pascal GPU. But the cloud server should be more powerful than the edge server in the real life. The mobile device communicates with the server through IEEE 802.11 wireless network. The system is implemented with PyTorch and OpenCV.

4.2 Model Training

To train our proposed stylizer shown in Figure 4, the pretrained VGG encoder is frozen, while the other three modules are learned together. The stylizer is optimized using Adam [24] with parameters: $\beta_1 = 0.5$, $\beta_2 = 0.999$, and an initial learning rate of 0.0001. Batch size is set to 2. The scale factors λ in Equation 5 is set to 1.0, *i.e.*, we do not tune λ. For the meta-smoothing module, each filter of $\boldsymbol{W}_u^r$ and $\boldsymbol{W}_s$ is set to have the same dimension. Specifically, we set $H_u = H_s = W_u = W_s = 5$.

4.3 Dataset

In this work, we adopt two datasets for training and testing the proposed DNN-based stylizer, respectively. We train the stylization model on MS-COCO [34], which contains 328k images covering 91 different object types. Both content images and style images are sampled from MS-COCO for training. We also download 100 videos of different scenes from Videvo.net [1] for evaluation, which contains about 41,500 frames. One style image is assigned to each video for performing style transformation.

4.4 Stylization Speed

The efficiency issue is one major concern about photorealistic style transfer. Therefore, we evaluate the efficiency of our proposed style transfer method by comparing with the methods proposed by Luan *et al.* [37] and Li *et al.* [33]. Table 1 shows the average time to perform style transfer on one frame with different resolutions. The numbers reported in Table 1 are obtained by averaging the stylization time of 1000 frames which are randomly sampled from the testing data. Overall, our proposed method outperforms the methods proposed by Luan *et al.* and Li *et al.* with any resolution setting, and we can obtain greater speedup with increasing resolution. For example, the time of processing a 512 × 256 frame by our method is 356.7 times and 5.6 times faster than the approaches of Luan *et al.* and Li *et al.*, respectively. Besides, for performing style transfer on a 1920 × 1080 frame, the two compared methods need over 1000 seconds and 38.72 seconds separately, but our method only costs 1.51 seconds, which is greatly faster. In summary, benefiting from our proposed meta-smoothing module of the stylizer, our proposed model can perform stylization in a much more efficient way than existing work.

Table 1: Average run time (in seconds) comparison between existing photorealistic style transfer methods and ours (on an NVIDIA TITAN X Pascal).

Method	512x256	768x384	1024x512	1920x1080
Luan *et al.* [37]	186.52	380.82	650.45	>1000.00
Li *et al.* [33]	2.95	7.05	13.16	38.72
Ours	**0.52**	**0.73**	**0.99**	**1.51**

4.5 Speedup by Optical-flow-based Frame Interpolation

As described in Section 3, we design an optical-flow-based interpolation algorithm to interpolate stylized intermediate frames for improving efficiency. Therefore, we evaluate the speedup by comparing the run time of performing stylization by our pre-trained DNN model on an edge server with that of interpolating stylized intermediate frames on the mobile phone. Table 2 shows average run time of processing 1000 key frames and 1000 intermediate frames by those two methods separately. It demonstrates that the optical-flow-based interpolation method significantly outperforms performing DNN-based stylization in terms of efficiency, even though the run time of both methods will be increased with higher resolution frame. In particular, performing DNN-based stylization on a 512 × 256 frame needs 0.52 seconds, but it only takes 0.00006 seconds for interpolating a stylized intermediate frame with the same resolution, which achieves about 866.7 times speedup. More important, since mobile phones usually record a video in high quality today, we also make the evaluation for high-resolution videos. Specifically, for processing a 1920 × 1080 frame, the optical-flow-based interpolation method only costs 0.02 seconds but the DNN-based stylization requires 1.51 seconds, indicating 75.5 times speedup. It can be imagined how slow it will be to perform the DNN-based stylization frame by frame. For instance, given a 10-minute video with resolution of 1920 × 1080 and frame rate at 30 fps, it will cost 7.55 hours to perform DNN-based stylization frame by frame even on the edge server.

Table 2: Average run time (in seconds) comparison between DNN-based stylization and optical-flow-based interpolation

Resolution	**Stylization per frame (edge server)**	**Interpolation per frame (mobile)**	**Speedup**
512x256	0.52	0.0006	866.7
768x384	0.73	0.002	365
1024x512	0.99	0.006	165
1920x1080	1.51	0.02	75.5

In addition to evaluate the speedup for a single frame, we also quantitatively evaluate the speedup for performing stylization on videos. Benefiting from the optical-flow-based interpolation, we can define the speedup as Equation 6:

$$
\begin{aligned}
\text{speedup} &= \frac{\#\text{all frames} \times t_d}{\#\text{key frames} \times t_d + \#\text{intermediate frames} \times t_i} \\
&\approx \frac{\#\text{all frames} \times t_d}{\#\text{key frames} \times t_d} = \frac{\#\text{all frames}}{\#\text{key frames}} \\
&\propto \text{key frame interval}
\end{aligned}
\tag{6}
$$

where t_d represents the time of performing stylization on a frame by the DNN-based stylizer, and t_i expresses the time of stylizing a frame with optical-flow-based interpolation algorithm, which are shown in Table 2. Since $t_d \gg t_i$, speedup is approximately proportional to the *key frame interval* as shown Equation 6. The key frame interval is defined as how often a key frame appears in a particular video as Equation 2. In this experiment, we consider a video clip with 300 frames with different resolutions as an example, and the speedup with various key frame interval are displayed in Figure 5. Generally, the higher key frame interval, the greater speedup due to less key frames in the video. Regarding with different resolutions, given the same key frame interval, since the higher resolution cost more time to perform style transfer for both key frames and intermediate frames, the speedup has a slight decrease but is still directly proportional to corresponding key frame interval.

4.6 Quantitative Evaluation of Stylization Results

In addition to the efficiency evaluation, we also experimentally evaluate the visual result of style transfer, including measuring the quality of stylization by the DNN-based stylizer and comparing the

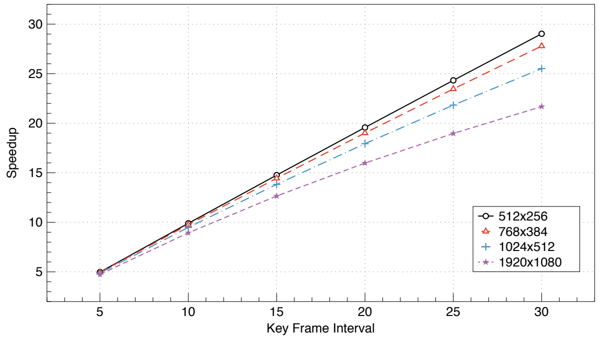


Figure 5: The speedup with different key frame intervals

similarity between frames that are stylized by DNN-based stylizer with those interpolated by the optical-flow-based interpolation algorithm.

Firstly, we randomly sample 1000 frames with the resolution 512 × 256 from the testing data, and we compare the visual quality of stylized results based on those 1000 frames which are transformed by our DNN-based stylizer with those processed by the state-of-the-art photorealistic style transfer method [33]. In this experiment, we adopt two widely applied quantitative evaluation metrics *Inception score* [45] and *Fréchet Inception Distance (FID)* [45] for our evaluation, which are designed to measure two aspects of synthesized image quality: photorealism and diversity. Note that the bigger Inception score indicates higher visual quality while the smaller FID representing better image quality. The averaged result are reported in Table 3, demonstrating our method can generate visually better stylized results than the state-of-the-art.

Table 3: Visual quality comparison of stylized frames between the state-of-the-art [33] and ours

Method	Inception score	FID
Li *et al.* [33]	129.54	168.71
Ours	**135.20**	**164.23**

Besides, we also evaluate the visual quality of interpolated stylized intermediate frames. Ideally, we expect the visual quality of interpolated frames can be as close as frames which are directly processed by the DNN-based stylizer. Figure 6 shows several examples of stylized key frames by DNN-based stylizer and interpolated intermediate frames. In this example, we choose the intermediate frame that is next to the corresponding key frame for demonstration. As Figure 6 illustrates, both the stylized key frames and interpolated intermediate frames are successfully rendered into the target style while maintaining original content structure, but it is difficult to perceptually tell the difference between them in terms of visual quality. Furthermore, we also quantitatively evaluate image similarity between the stylized key frames and interpolated intermediate frames by attempting to predict human perceptual similarity judgments. In this experiment, we adopt the widely applied metric *multi-scale structural similarity (MS-SSIM)* [39, 50] for measuring the frame similarity. MS-SSIM is a multi-scale perceptual similarity metric that attempts to pay less attention to aspects of an image that are not important for human perception. MS-SSIM values range between 0 and 1. The higher MS-SSIM values, the more perceptually similar between compared images. Specifically, we randomly choose 1000 frames from the testing data, and a pair of stylized frames for each of those 1000 frames are generated by the DNN-based stylizer and optical-flow-based interpolation algorithm, respectively. In addition, we evaluate the MS-SSIM for those 1000 pairs with different resolutions settings which are the same as used in above experiments. Table 4 shows the averaged results for those 1000 pairs, MS-SSIM is greater than 0.98 for all resolution settings, indicating that the stylized frames generated by the optical-flow-based interpolation algorithm have the perceptually comparable visual quality with frames that are directly processed by the DNN-based stylizer.

In summary, above experiments demonstrate MVStylizer can efficiently perform style transfer of videos while achieving even better visual quality compared to the state-of-the-art method.

Table 4: MS-SSIM of stylized frames processed by DNN-based stylization and optical-flow-based interpolation

Resolution	MS-SSIM
512x256	0.9845
768x384	0.9841
1024x512	0.9847
1920x1080	0.9849

4.7 Latency Comparison

We quantitatively compare the latency of sending a key frame to an edge server with that of sending a key frame to a conventional cloud server. In this experiment, we test 1000 frames that are sent by a user from Boston. The edge server is located in New York, but the other two conventional cloud servers are located in Los Angles and Hong Kong, respectively. The average results are shown in Table 5. Generally, the latency of send a key frame the edge server is about 10 times and 30 times lower than that of sending the key frame to the cloud server in Los Angles and in Hong Kong, respectively. For example, it takes 0.031 seconds to upload a 1920×1080 key frame to the edge server from the mobile user, but requires 0.318 seconds and 0.925 seconds to send the key frame to the cloud server in Los Angles and Hong Kong, respectively. The above results demonstrate that the edge server is able to provide the style transfer service to the mobile user with a significantly lower latency compared to the conventional cloud server.

4.8 Performance Improvement by Edge-Cloud Federated Learning

As described in Section 3, we apply an edge-cloud federated learning scheme to continuously improve the DNN-based stylizer on each edge server. In this experiment, we quantitatively evaluate how edge-cloud federated learning scheme can improve the performance of DNN-based stylizer on each edge server through simulations. To make simulations, we enforce each participated edge server to sync the edge DNN with the cloud DNN after training on 4000 images

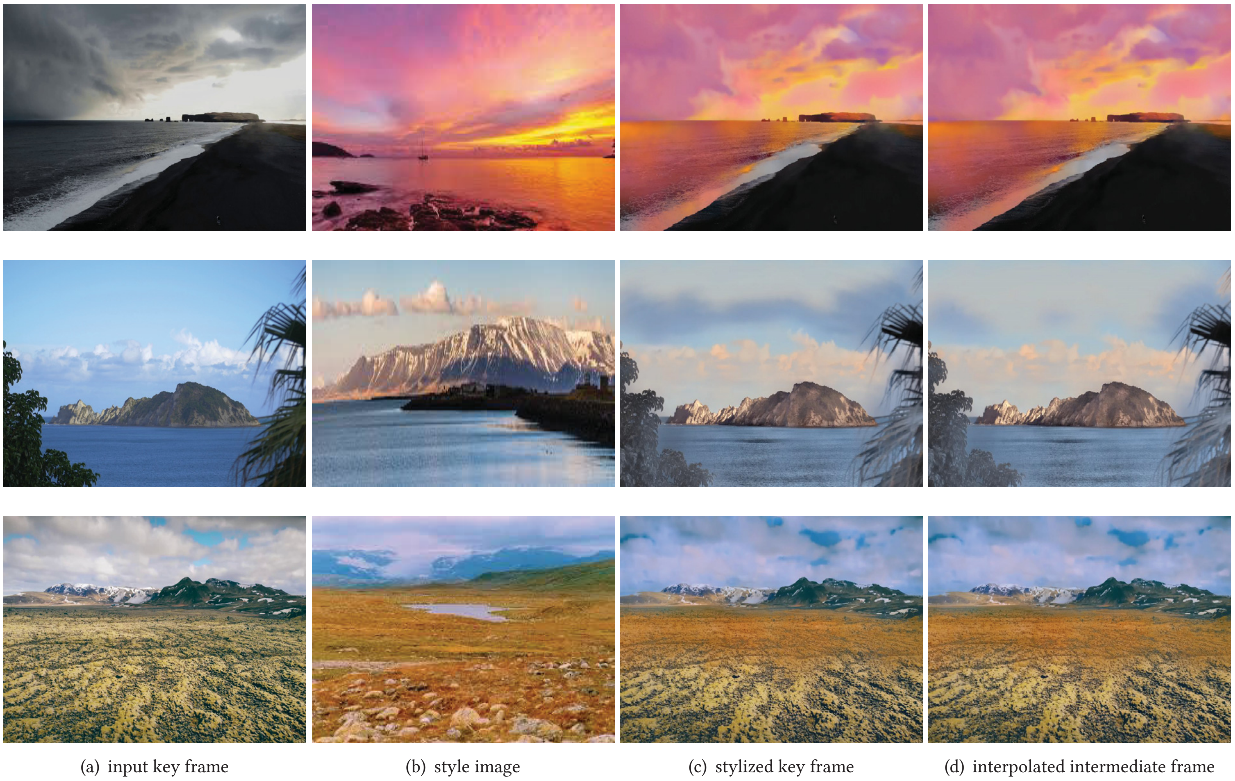


Figure 6: Examples of stylized key frames and interpolated intermediate frames

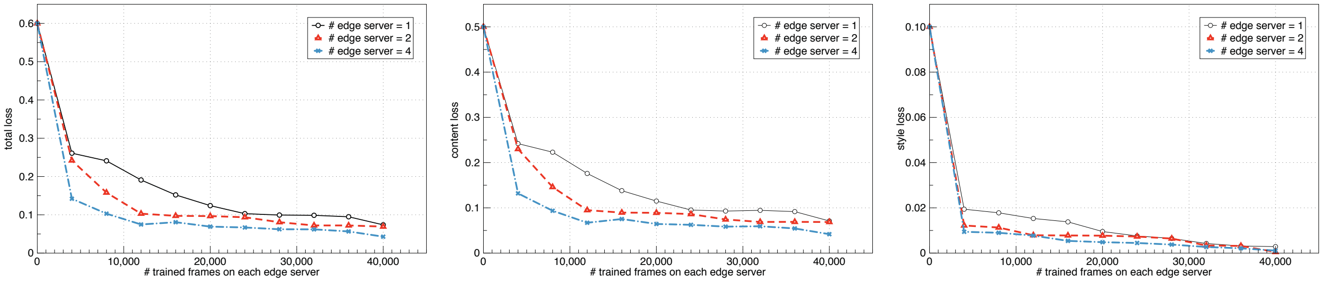
Table 5: Average latency (in seconds) comparison between sending one frame to the edge server and to the cloud server.

Resolution	Edge Server (New York)	Cloud Server (Los Angles)	Cloud Server (Hong Kong)
512x256	0.003	0.028	0.088
768x384	0.006	0.058	0.176
1024x512	0.011	0.105	0.312
1920x1080	0.031	0.318	0.925

that are randomly sampled from the training data, and the cloud DNN will make an update once receive all the synchronized parameters from all participated edge server. Then, the latest cloud DNN will be distributed to each edge server as the new edge DNN. In addition, we also change the number of participated edge servers to explore how it will affect the performance improvement. We evaluate the performance of the model based on the loss function defined as Equation 5, including the content loss and style loss. Figure 7 shows the federated learning curves during the continuously training with different number of participated edge servers. In general, the more participated edge servers, the faster and greater the performance can be improved. For example, if there are 4 participated edge servers, the total loss can be reduced to 0.0748 after 12000 images are trained on each edge server. However, it requires to train 40000 images when there is only one participated edge server for achieving the same performance, and 32000 images on each of the two participated edge servers. The results show the edge-cloud federated learning can more efficiently improve the performance with an increasing number of participated edge servers.

5 RELATED WORK

Neural Style Transfer. Current neural style transfer techniques fit one of two mainstreams [22], *image-optimization-based online neural methods* and *model-optimization-based offline neural methods*. Generally, the first category transfers the style by optimizing an image in an iterative way, while the second category aims to optimize a generative model offline which can generate the stylized image with a single forward pass. Gatys *et al.* [12, 13] proposed a seminal work demonstrating the power in style transfer by separating and recombining image content and style. It is inspired by observing that a CNN can separately extract content information from an original image and style information from a style image. Based on such observations, a CNN model can be trained to recombine the extracted content and style information to generate the target stylized image. However, this method only compares the content and stylized image in the feature space, which inevitably lose some low-level information contained in the image that can lead to distortion and abnormal artistic effects of stylized outputs. To preserve the structure coherence, Li *et al.* [28] introduced an additional Laplacian loss to constrain low-level features in pixel space. Although image-optimization-based can achieve impressive



(a) total loss

(b) content loss

(c) style loss

Figure 7: Edge-cloud federated learning curves of total loss, content loss and style loss (# participated edge server =1,2,4)

stylized results, they have a common limitation in efficiency. To address the efficiency issue, various model-optimization-based offline neural methods have been proposed. Johnson *et al.* [23] and Ulyanov *et al.* [49] proposed the first two model-optimization-based algorithms for style transfer, which generate stylized result with a single forward pass through a pre-trained style-specific CNN. Even though those algorithms can achieve real-time style transfer, they require separate generative networks to be trained for each specific style, which is quite inflexible. Therefore, multiple-style-per-model neural methods are proposed to improve the flexibility by integrating multiple styles into one single model by tuning a small number of parameters for each style [5, 11] or combining both style and content as inputs to the generative model [30, 53]. Furthermore, several works have been done for designing one single mode transfer arbitrary artistic styles by exploiting texture modeling techniques [6, 14].

Video Style Transfer. Compared with above image style transfer techniques, video style transfer algorithms need to consider the smooth transition between consecutive frames. Ruder *et al.* [43, 44] introduced a temporal consistency loss based on optical flow for video style transfer. Huang *et al.* [18] designed an augmented temporal consistency loss by computing the outputs of style transfer network for two consecutive frames. A flow subnetwork was proposed by Chen *et al.* [3] to produce feature flow, which can be used to wraps feature activations from a pre-trained stylization encoder.

Photorealistic Style Transfer. Most existing works focus on artistic style transfer which can tolerate some distortion, but photorealistic style transfer requires more strict structure preservation of the content image. Luan *et al.* [36] firstly proposed a two-stage optimization for photorealistic style transfer, which firstly renders a given photo with non-photorealistic style and then penalizes image distortions by adding a photorealistic regularization. However, this algorithm is very computational expensive. Mechrez *et al.* [41] also adopts above two-stage optimization scheme, but they refine the photorealistic rendering effect by matching the gradients in the stylized image to those in the content image. To improve the efficiency issue, Li *et al.* [33] designed a two-step photorealistic style transfer algorithm, including the *stylization step* and *smoothing step*. The stylization step aims to generate stylized output based on existing neural style transfer algorithms but replaces upsampling layers with unpooling layer for less distortion, and then the smoothing step is applied to remove structural artifacts.

Even though there exist some works on video style transfer and photorealistic style transfer, none of them is specifically designed for performing photorealistic style transfer of videos on resource-constrained devices, such as mobile phones.

6 CONCLUSION

In this paper, we designed MVStylizer to efficiently perform photorealistic style transfer of videos on mobile phones with the assistance of an edge server. Considering the stylization is very computational expensive, we proposed an optical-flow-based interpolation algorithm, so that only key frames in the video need to be uploaded to the edge server where they can be processed by the pre-trained DNN-based stylizer and the rest of stylized intermediate frames can be interpolated based on the pre-computed optical flow information from the original video and stylized key frames. A meta-smoothing module is also designed in the DNN-based stylizer for improving the efficiency of performing style transfer on the edge server. In addition, we adopt an edge-cloud federated learning scheme to continuously enhancing the performance of DNN-based stylizer. Experiments demonstrate 75.5 times speedup compared with performing style transfer frame by frame using the DNN-based stylizer even with the high resolution, while generating the stylized videos with even better visual quality compared to the state-of-the-art method. Furthermore, it also demonstrates the edge-cloud federated learning scheme can facilitate in continuously improving the performance of the DNN-based stylizer in an efficiency way.

ACKNOWLEDGMENTS

This work was supported in part by NSF-1822085, NSF-1717657, and NSF IUCRC for ASIC membership from Quantil. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSF and their contractors.

REFERENCES

[1] 2019. Videvo free footage. https://www.videvo.net.
[2] Alexander G Anderson, Cory P Berg, Daniel P Mossing, and Bruno A Olshausen. 2016. Deepmovie: Using optical flow and deep neural networks to stylize movies. *arXiv preprint arXiv:1605.08153* (2016).

[3] Dongdong Chen, Jing Liao, Lu Yuan, Nenghai Yu, and Gang Hua. 2017. Coherent online video style transfer. In *Proceedings of the IEEE International Conference on Computer Vision*. 1105–1114.

[4] Dongdong Chen, Lu Yuan, Jing Liao, Nenghai Yu, and Gang Hua. 2017. Stylebank: An explicit representation for neural image style transfer. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1897–1906.

[5] Dongdong Chen, Lu Yuan, Jing Liao, Nenghai Yu, and Gang Hua. 2017. StyleBank: An Explicit Representation for Neural Image Style Transfer. *arXiv.org* (March 2017). arXiv:cs.CV/1703.09210v2

[6] Tian Qi Chen and Mark Schmidt. 2016. Fast patch-based style transfer of arbitrary style. *arXiv preprint arXiv:1612.04337* (2016).

[7] Zhuo Chen, Wenlu Hu, Junjue Wang, Siyan Zhao, Brandon Amos, Guanhang Wu, Kiryong Ha, Khalid Elgazzar, Padmanabhan Pillai, Roberta Klatzky, et al. 2017. An empirical study of latency in an emerging class of edge computing applications for wearable cognitive assistance. In *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*. 1–14.

[8] DeepArt. [n.d.]. DeepArt: Turn your photos into art. https://deepart.io. Accessed: 2019-07-29.

[9] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. 2015. Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence* 38, 2 (2015), 295–307.

[10] Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. 2016. A learned representation for artistic style. *arXiv preprint arXiv:1610.07629* (2016).

[11] Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. 2017. A Learned Representation For Artistic Style. *ICLR* (2017).

[12] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. 2015. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576* (2015).

[13] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. 2016. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2414–2423.

[14] Golnaz Ghiasi, Honglak Lee, Manjunath Kudlur, Vincent Dumoulin, and Jonathon Shlens. 2017. Exploring the structure of a real-time, arbitrary neural artistic stylization network. *BMVC* (2017).

[15] K. He, X. Zhang, S. Ren, and J. Sun. 2016. Deep Residual Learning for Image Recognition. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016).

[16] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv:1704.04861v1* (2017).

[17] X. Hu, H. Mu, X. Zhang, Z. Wang, T. Tan, and J. Sun. 2019. Meta-SR: A Magnification-Arbitrary Network for Super-Resolution. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2019).

[18] Haozhi Huang, Hao Wang, Wenhan Luo, Lin Ma, Wenhao Jiang, Xiaolong Zhu, Zhifeng Li, and Wei Liu. 2017. Real-time neural style transfer for videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 783–791.

[19] Xun Huang and Serge Belongie. 2017. Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization. In *ICCV*.

[20] X. Huang and S. Belongie. 2017. Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization. *International Conference on Computer Vision (ICCV)* (2017).

[21] Hyuk-Jin Jeong, InChang Jeong, Hyeon-Jae Lee, and Soo-Mook Moon. 2018. Computation offloading for machine learning web apps in the edge server environment. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 1492–1499.

[22] Yongcheng Jing, Yezhou Yang, Zunlei Feng, Jingwen Ye, Yizhou Yu, and Mingli Song. 2017. Neural Style Transfer: A Review. *arXiv.org* (May 2017), arXiv:1705.04058. arXiv:cs.CV/1705.04058

[23] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. 2016. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*. Springer, 694–711.

[24] D. P. Kingma and J. L. Ba. 2015. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations (ICLR)* (2015).

[25] Prisma Labs. 2019. Prisma: Building the future of photo and video editing. https://prisma-ai.com.

[26] Hochang Lee, Sanghyun Seo, Seungtaek Ryoo, and Kyunghyun Yoon. 2010. Directional texture transfer. In *Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering*. ACM, 43–48.

[27] Chuan Li and Michael Wand. 2016. Combining markov random fields and convolutional neural networks for image synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2479–2486.

[28] Shaohua Li, Xinxing Xu, Liqiang Nie, and Tat-Seng Chua. 2017. Laplacian-steered neural style transfer. In *Proceedings of the 25th ACM international conference on Multimedia*. ACM, 1716–1724.

[29] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. 2017. Diversified texture synthesis with feed-forward networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3920–3928.

[30] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. 2017. Diversified Texture Synthesis with Feed-forward Networks. *arXiv.org* (March 2017), arXiv:1703.01664. arXiv:cs.CV/1703.01664

[31] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M.-H. Yang. 2017. Universal Style Transfer via Feature Transforms. *Conference on Neural Information Processing Systems (NeurIPS)* (2017).

[32] Yijun Li, Ming-Yu Liu, Xueting Li, Ming-Hsuan Yang, and Jan Kautz. 2018. A Closed-form Solution to Photorealistic Image Stylization. *arXiv.org* (Feb. 2018), arXiv:1802.06474. arXiv:cs.CV/1802.06474

[33] Y. Li, M.-Y. Liu, X. Li, M.-H. Yang, and J. Kautz. 2018. A Closed-Form Solution to Photorealistic Image Stylization. *European Conference on Computer Vision (ECCV)* (2018).

[34] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*. Springer, 740–755.

[35] Jonathan Long, Evan Shelhamer, and Trevor Darrell. 2015. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 3431–3440.

[36] Fujun Luan, Sylvain Paris, Eli Shechtman, and Kavita Bala. 2017. Deep photo style transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4990–4998.

[37] F. Luan, S. Paris, E. Shechtman, and K. Bala. 2017. Deep Photo Style Transfer. *International Conference on Computer Vision and Pattern Recognition (CVPR)* (2017).

[38] Bruce D Lucas, Takeo Kanade, et al. 1981. An iterative image registration technique with an application to stereo vision. (1981).

[39] Kede Ma, Qingbo Wu, Zhou Wang, Zhengfang Duanmu, Hongwei Yong, Hongliang Li, and Lei Zhang. 2016. Group mad competition-a new methodology to compare objective image quality models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1664–1673.

[40] Jiachen Mao, Qing Yang, Ang Li, Hai Li, and Yiran Chen. 2019. MobiEye: An Efficient Cloud-based Video Detection System for Real-time Mobile Applications. In *Proceedings of the 56th Annual Design Automation Conference 2019 (DAC '19)*. ACM, New York, NY, USA, Article 102, 6 pages. https://doi.org/10.1145/3316781.3317865

[41] Roey Mechrez, Eli Shechtman, and Lihi Zelnik-Manor. 2017. Photorealistic style transfer with screened poisson equation. *arXiv preprint arXiv:1709.09828* (2017).

[42] Xukan Ran, Haolianz Chen, Xiaodan Zhu, Zhenming Liu, and Jiasi Chen. 2018. Deepdecision: A mobile deep learning framework for edge video analytics. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 1421–1429.

[43] Manuel Ruder, Alexey Dosovitskiy, and Thomas Brox. 2016. Artistic style transfer for videos. In *German Conference on Pattern Recognition*. Springer, 26–36.

[44] Manuel Ruder, Alexey Dosovitskiy, and Thomas Brox. 2018. Artistic Style Transfer for Videos and Spherical Images. *International Journal of Computer Vision* (2018).

[45] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. 2016. Improved Techniques for Training GANs. *Advances in Neural Information Processing Systems (NeurIPS)* (2016).

[46] F. Shen, S. Yan, and G. Zeng. 2018. Meta Networks for Neural Style Transfer. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2018).

[47] W. Shi, J. Caballero, F. Huszar, J. Totz, and A. P. Aitken. 2016. Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016).

[48] K. Simonyan and A. Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. *International Conference on Learning Representations (ICLR)* (2015).

[49] Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, and Victor S Lempitsky. 2016. Texture Networks: Feed-forward Synthesis of Textures and Stylized Images.. In *ICML*, Vol. 1. 4.

[50] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. 2003. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, Vol. 2. Ieee, 1398–1402.

[51] Dianlei Xu, Tong Li, Yong Li, Xiang Su, Sasu Tarkoma, and Pan Hui. 2020. A Survey on Edge Intelligence. *arXiv preprint arXiv:2003.12172* (2020).

[52] Shanhe Yi, Zijiang Hao, Qingyang Zhang, Quan Zhang, Weisong Shi, and Qun Li. 2017. Lavea: Latency-aware video analytics on edge computing platform. In *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*. 1–13.

[53] Hang Zhang and Kristin J Dana. 2018. Multi-style Generative Network for Real-Time Transfer. *ECCV Workshops* (2018).