DeeperFour: convert BW films into cartoon style

Bo Sun
Department of Computer Science
University of Southern California
bos@usc.edu

Muxin Liang
Department of Computer Science
University of Southern California
muxinlia@usc.edu

Xi Jin
Department of Computer Science
University of Southern California
xij@usc.edu

Zhuo Han
Department of Computer Science
University of Southern California
zhuoh@usc.edu

Abstract

This paper is about the the project DeeperFour, which aims at converting black-and-white films into colorized cartoons.

1 Goal and Motivation

There are a lot of fantastic old movies for people to appreciate. However, the out-of-date filming technology and the black-and-white presentation make them less enjoyable. We want to make classic film works adorable again by endowing the old black-and-white version with vibrant colors, and even more interesting, parallel it with a cartoonized version. Our goal is to re-energize those old movies through deep learning techniques: Apply colorization on videos by convolutional networks, and animate it with different cartoon styles through GANs. And if possible, we'd like to try to research about temporal loss to stabilize the video outputs.

2 Problem formulation

Input for this problem will be a series of frames from a black and white movie, and the output will be the colorized and animated movie. And our solution will be a form of pipeline:

1. Cutting original movies into frames
2. Colorizing grayscale movie frames using a convolution neural network
3. Generating datasets with different styles
4. Animating colored frames using a GAN model
5. Stabilizing the output video

3 Models and approaches

3.1 Colorization

We chose DeOldify to colorize video frames with pre-trained weights and tuned with different render factor(quality of render colors. The model is a combined architecture composed of the following techniques:

Preprint. Work in progress.

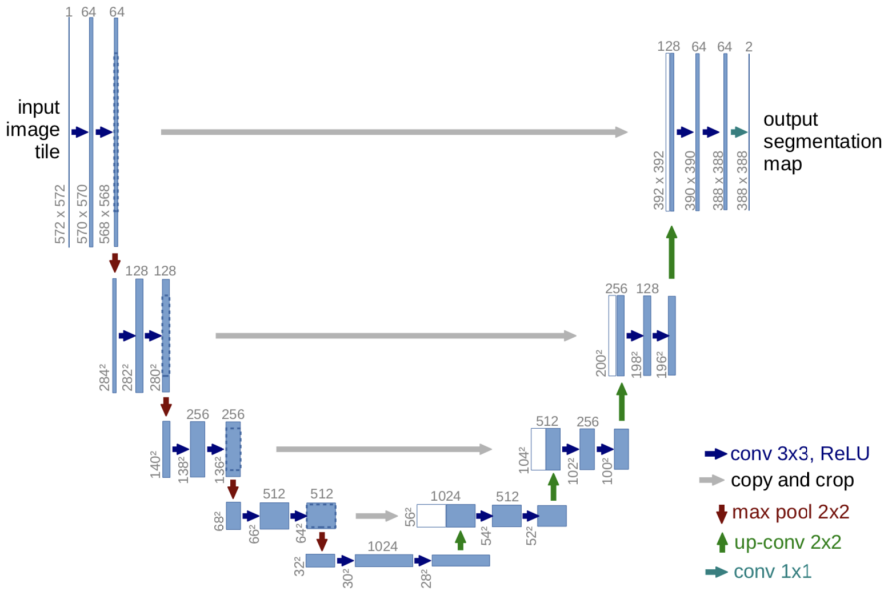

Figure 1: U-net architecture for the generator used in SAGAN

(1) A Self-Attention Generative Adversarial Network(SAGANZhang et al. [2018]), with the generator use a pre-trained U-net(Ronneberger et al. [2015]). In SAGAN, proposed attention module has been applied to both generator and discriminator, which are trained in an alternating fashion by minimizing the hinge version of the adversarial loss.

$$\mathcal{L}_D = -\mathbb{E}_{(x,y)\sim p_{data}}[\min(0, -1 + D(x,y))] - \mathbb{E}_{z\sim p_z, y\sim p_data}[min(0, -1 - D(G(z), y))] \quad (1)$$

$$\mathcal{L}_G = -\mathbb{E}_{z\sim p_z, y\sim p_data} D(G(z), y) \quad (2)$$

(2) Progressive Growing of GANs(Karras et al. [2017]), with the difference that in Deodify the number of layers stay constant, only the size is progressively changing from low resolution to high resolution.

(3) Two Time-Scale Update Rule(Heusel et al. [2017].) This is very hard, we can not understand a bit, but it is a stragety to train GAN.

3.2 Cartoonization

We use CartoonGAN, introduced by Chen et al. [2018], to convert colorized pictures into cartoon-style. In order to stablize the output, we also need to adopt method from Ruder et al. [2018] to add new loss functions.

The loss function $L(G, D)$ in Eq (3) consists of two parts: (1) the adversarial loss, which defined in Eq (6), and (2) the content loss, which defined in Eq (7).

$$\mathcal{L}(G, D) = L_{adv}(G, D) + \omega L_{con}(G, D) \quad (3)$$

In CartoonGAN, the goal of training the discriminator D is to maximize the probability of assigning the correct label to $G(p_k)$, the cartoon images without clear edges (i.e., $e_j \in S_{data}(e)$) and the real cartoon images (i.e., $c_i \in S_{data}(c)$), such that the generator G can be guided correctly by transforming the input to the correct manifold. Therefore, we define the edge-promoting adversarial loss as:

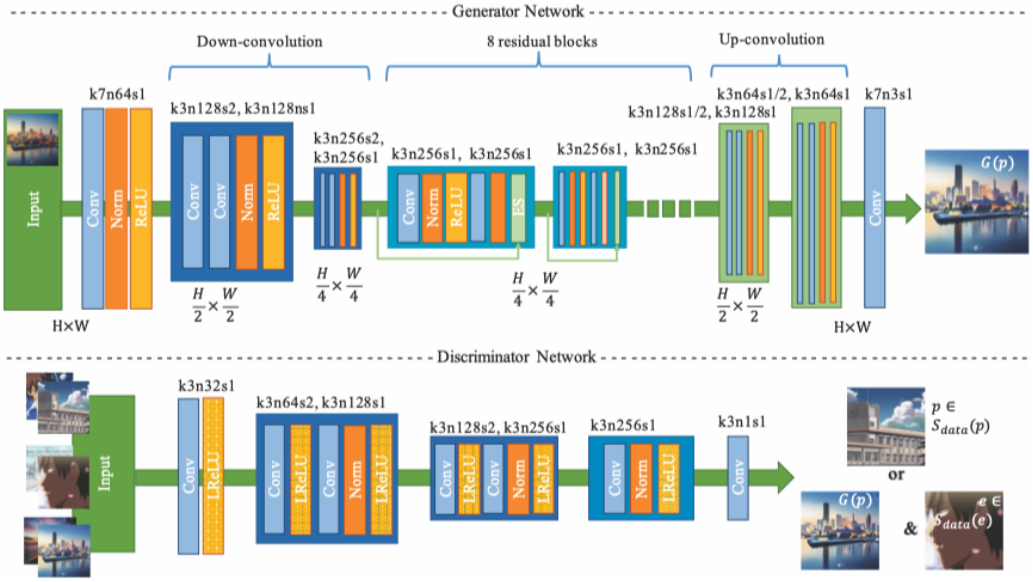

Figure 2: Architecture of the generator and discriminator networks in CartoonGAN

$$\mathcal{L}_{adv}(G, D) = \mathbb{E}_{c_i \sim S_{data}(c)}[\log D(c_i)] \tag{4}$$
$$+ \mathbb{E}_{e_j \sim S_{data}(c)}[\log(1 - D(e_j))] \tag{5}$$
$$+ \mathbb{E}_{p_k \sim S_{data}(c)}[\log(1 - D(G(p_k)))] \tag{6}$$

In CartoonGAN, it adopt the high-level feature maps in the VGG network. In Eq (7), l refers to the feature maps of a specific VGG layer.

$$\mathcal{L}_{con}(G, D) = \mathbb{E}_{p_i \sim S_{data}(p)}[\|VGG_l(G(p_i)) - VGG_l(p_i)\|_1] \tag{7}$$

4 Experimental results

We gathered a lot of Makoto Shinkai's works: *Five Centimeters per Second*, *The Garden of Words*, and *Your Name*. And preprocessed all images with edge-smoothing, e.g. Figure 3. We clipped two scenes from Hichcock's *Psycho* for testing.

(a) A cartoon image from Shinkai's *Your Name* (b) The edge-smoothed version

Figure 3: Removing clear edges in cartoon images

(a) BW origin image from *psycho* (b) Coloried result (c) Shinkai-style cartoonization

Figure 4: Sampled frame in Hitchcock's *Psycho*

The workflow is shown like Figure 4. Starting from a black/white frame like subfigure 4a.

However, sometimes two consecutive frames may have great difference, which makes the video unstable. E.g. in Figure 5, the color near the man's mouth area changed suddenly, and this color change also has significate effect on afterwards cartoonization.

(a) 5 consecutive frames after colorization

(b) 5 consecutive frames after cartoonization

Figure 5: Compare five consecutive frames in Hitchcock's *Psycho*

5 Plans

5.1 Colorize with high render factor

When we were colorizing with different render factor, we see a significant improvement in the quality of colorized frames. The value of render factor is restricted by computational resource. Thus we can work on using GPUs from Google Cloud Platform to achieve better colorizing results.

(a) Render factor = 8 (b) Render factor = 12 (c) Render factor = 20

Figure 6: Hitchcock's *Psycho*, colorized with different render factor

5.2 Stablizing

We want to divide into two groups to try two methods that we found, one is adding a long-term temporal consistency loss, which introduced in Ruder et al. [2018], and another one is Lai et al. [2018]'s recurrent convolutional network.

5.2.1 Temporal Consistency Loss

The temporal consistency loss function penalizes deviation from the warped image in regions.

$$\mathcal{L}_{temporal}(x, w, c) = \frac{1}{D} \sum_{k=1}^{D} c_k \times (x_k - w_k)^2 \tag{8}$$

Here $c \in [0, 1]^D$ is a per-pixel weighting of the loss and $D = W \times H \times C$ is the dimensionality of the image.

The short-term model only considering adjacent frames will likely have inconsistency on disoccluded areas. Taking the long-term motion into account will increase the robustness.

$$\begin{aligned} \mathcal{L}_{longterm}(f^{(i)}, a, x^{(i)}) = \alpha \mathcal{L}_{content}(f^{(i)}, x^{(i)}) + \beta \mathcal{L}_{style}(a, x^{(i)}) + \\ \gamma \sum_{j \in J : i-j \leq 1} \mathcal{L}_{temporal}(x^{(i)}, w_{i-j}^{i}(x^{(i-j)}), c_{long}^{(i-j,i)}) \end{aligned} \tag{9}$$

J denote the set of indices each frame should take into account relative to the frame number. $c(i-j, i)$ is the weights for the flow between image $i - j$ and i, as defined for the short-term model. The long-term weights are then computed as:

$$c_{long}^{(i-j),i} = max(c^{(i-j,i)} - \sum_{k \in J : i-k i-j} c^{(i-k,i)}, 0). \tag{10}$$

5.2.2 ConvLSTM

Introduced in Lai et al. [2018], The overall loss function is defined as:

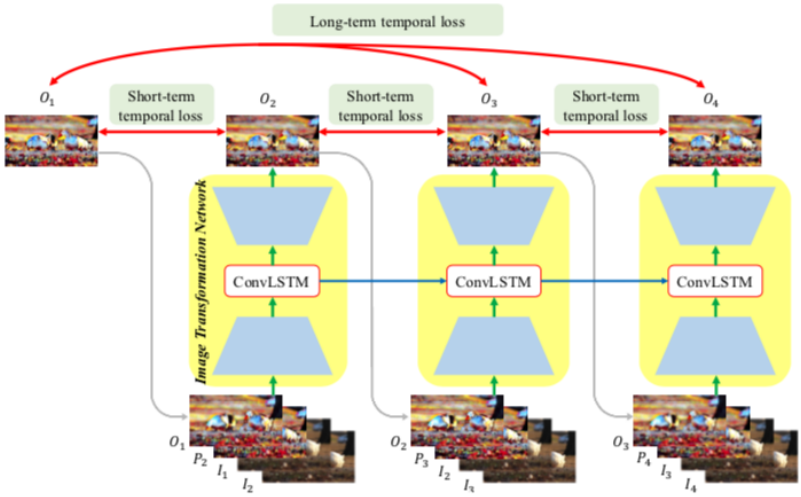

Figure 7: Temporal losses

$$\mathcal{L} = \lambda_p \mathcal{L}_p + \lambda_{st} \mathcal{L}_{st} + \lambda_{lt} \mathcal{L}_{lt} \tag{11}$$

Each loss function is define below, O_t represents a vector $\in R^3$ with RGB pixel values of the output O at time t, $\phi_l(\cdot)$ denotes the feature activation at the l-th layer of VGG-19 network ϕ.

$$\mathcal{L}_p = \sum_{t=2}^{T}\sum_{i=1}^{N}\sum_{l}\|\phi_l(O_t^{(i)}) - \phi_l(P_t^(i))\|_1 \tag{12}$$

$$\mathcal{L}_{st} = \sum_{t=2}^{T}\sum_{i=1}^{N} M_{t\Rightarrow t-1}^{(i)}\|O_t^{(i)} - \hat{O}_{t-1}^{(i)}\|_1 \tag{13}$$

$$\mathcal{L}_{lt} = \sum_{t=2}^{T}\sum_{i=1}^{N} M_{t\Rightarrow 1}^{(i)}\|O_t^{(i)} - \hat{O}_1^{(i)}\| \tag{14}$$

6 Breakdowns

Colorization Zhuo Han, Muxin Liang

Zhuo configured google cloud platform with GPU; investigated and executed several different colorization methods; generated colorized video frames via DeOldify with tuned hyperparameters.

Muxin restructed colorize source codes and tuned some hyperparameters(batch size, render factor, etc) to improve colorize performance.

Cartoonization Xi Jin, Bo Sun

Xi preprocessed gathered Shinkai's films into input dataset. Tuned edge-smoothing parameter and ω for balancing two loss functions. Wrote pipeline to generate anime from BW source.

Bo investigated and implemented PyTorch version CartoonGAN based on original Torch version, and successfully trained it on our dataset to make sure it's functional.

References

Yang Chen, Yu-Kun Lai, and Yong-Jin Liu. Cartoongan: Generative adversarial networks for photo cartoonization. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9465–9474, 2018.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, Günter Klambauer, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a nash equilibrium. *CoRR*, abs/1706.08500, 2017. URL `http://arxiv.org/abs/1706.08500`.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *CoRR*, abs/1710.10196, 2017. URL `http://arxiv.org/abs/1710.10196`.

Wei-Sheng Lai, Jia-Bin Huang, Oliver Wang, Eli Shechtman, Ersin Yumer, and Ming-Hsuan Yang. Learning blind video temporal consistency. *CoRR*, abs/1808.00449, 2018. URL `http://arxiv.org/abs/1808.00449`.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015. URL `http://arxiv.org/abs/1505.04597`.

Manuel Ruder, Alexey Dosovitskiy, and Thomas Brox. Artistic style transfer for videos and spherical images. *International Journal of Computer Vision*, 126(11):1199–1219, Nov 2018. ISSN 1573-1405. doi: 10.1007/s11263-018-1089-z. URL `https://doi.org/10.1007/s11263-018-1089-z`.

Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-Attention Generative Adversarial Networks. *arXiv e-prints*, art. arXiv:1805.08318, May 2018.