GIF2Video: Color Dequantization and Temporal Interpolation of GIF images

Yang Wang[1], Haibin Huang[2†], Chuan Wang[2], Tong He[3], Jue Wang[2], Minh Hoai[1]
[1]Stony Brook University, [2]Megvii Research USA, [3]UCLA, [†]Corresponding Author

Abstract

Graphics Interchange Format (GIF) is a highly portable graphics format that is ubiquitous on the Internet. Despite their small sizes, GIF images often contain undesirable visual artifacts such as flat color regions, false contours, color shift, and dotted patterns. In this paper, we propose GIF2Video, the first learning-based method for enhancing the visual quality of GIFs in the wild. We focus on the challenging task of GIF restoration by recovering information lost in the three steps of GIF creation: frame sampling, color quantization, and color dithering. We first propose a novel CNN architecture for color dequantization. It is built upon a compositional architecture for multi-step color correction, with a comprehensive loss function designed to handle large quantization errors. We then adapt the SuperSlomo network for temporal interpolation of GIF frames. We introduce two large datasets, namely GIF-Faces and GIF-Moments, for both training and evaluation. Experimental results show that our method can significantly improve the visual quality of GIFs, and outperforms direct baseline and state-of-the-art approaches.

1. Introduction

GIFs [1] are everywhere, being created and consumed by millions of Internet users every day on the Internet. The widespread of GIFs can be attributed to its high portability and small file sizes. However, due to heavy quantization in the creation process, GIFs often have much worse visual quality than their original source videos. Creating an animated GIF from a video involves three major steps: frame sampling, color quantization, and optional color dithering. Frame sampling introduces jerky motion, while color quantization and color dithering create flat color regions, false contours, color shift, and dotted patterns, as shown in Fig. 1.

In this paper, we propose GIF2Video, the first learning-based method for enhancing the visual quality of GIFs. Our algorithm consists of two components. First, it performs color dequantization for each frame of the animated gif sequence, removing the artifacts introduced by both color quantization and color dithering. Second, it increases the temporal resolution of the image sequence by using a modified SuperSlomo [20] network for temporal interpolation.

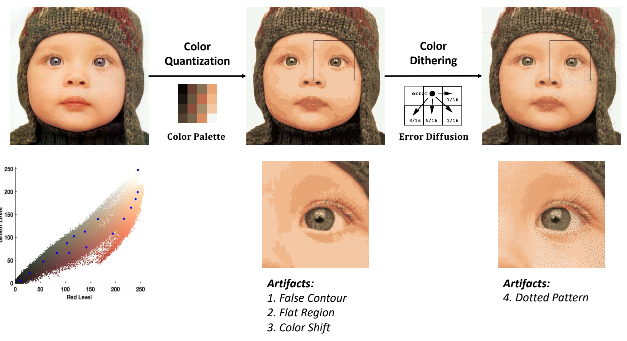

Figure 1. **Color quantization and color dithering.** Two major steps in the creation of a GIF image. These are lossy compression processes that result in undesirable visual artifacts. Our approach is able to remove these artifacts and produce a much more natural image.

The main effort of this work is to develop a method for color dequantization, i.e., removing the visual artifacts introduced by heavy color quantization. Color quantization is a lossy compression process that remaps original pixel colors to a limited set of entries in a small color palette. This process introduces quantization artifacts, similar to those observed when the bit depth of an image is reduced. For example, when the image bit depth is reduced from 48-bit to 24-bit, the size of the color palette shrinks from 2.8×10^{14} colors to 1.7×10^{7} colors, leading to a small amount of artifacts. The color quantization process for GIF, however, is far more aggressive with a typical palette of 256 distinct colors or less. Our task is to perform dequantization from a tiny color palette (e.g., 256 or 32 colors), and it is much more challenging than traditional bit depth enhancement [15, 25, 38].

Of course, recovering all original pixel colors from the quantized image is nearly impossible, thus our goal is to render a plausible version of what the original image might look like. The idea is to collect training data and train a ConvNet [16, 23, 29, 34, 44] to map a quantized image to its original version. It is however difficult to obtain a good dequantization network for a wide range of GIF images. To this end, we propose two novel techniques to improve the performance of the dequantization network. Firstly, we pose dequantization as an optimization problem, and we propose Compositional Color Dequantization Network (CCDNet), a novel network architecture for iterative color dequantization. Similar to the iterative Lucas-Kanade algorithm [28], this iterative procedure alleviates the problems associated with severe color quantization. Secondly, during training, we consider reconstruction loss and generative adversarial loss [10, 19, 32] on both pixel colors and image gradients. This turns out to be far more effective than a loss function defined on the color values only.

Another contribution of the paper is the creation of two large datasets: GIF-Faces and GIF-Moments. Both datasets contain animated GIFs and their corresponding high-quality videos. GIF-Faces is face-centric whereas GIF-Moments is more generic and diverse. Experiments on these two datasets demonstrate that our method can significantly enhance the visual quality of GIFs and reduce all types of artifacts. Comparatively, Our method outperforms its direct baselines as well as existing methods such as False Contour Detection & Removal [15] and Pix2Pix [19].

2. Related Work

False Contour Detection and Removal. Smooth areas of images and video frames should not contain color edges, but false contours are often visible in those areas after color bit-depth reduction or video codec encoding. Several false contour detection and decontouring methods [2, 5, 7, 15, 21, 24, 42] have been proposed to address this problem. Among them, False Contour Detection and Removal (FCDR) [15] is the most recent state-of-the-art approach. It first locates the precise positions of the false contours and then applies dedicated operations to suppress them. However, the color quantization artifacts in GIFs are far more severe, and GIF color dequantization requires more than removing minor false contours produced by bit-depth reduction.

Video Interpolation. Classical video interpolation methods rely on cross-frame motion estimation and occlusion reasoning [3, 4, 14, 18]. However, motion boundaries and severe occlusions are still challenging for existing optical flow estimation methods [6, 9]. Moreover, the flow computation, occlusion reasoning, and frame interpolation are separated steps that are not properly coupled. Drawing inspiration from the success of deep learning in high-level vision tasks [12, 22, 36], many deep models have been proposed for single-frame interpolation [26, 27, 30, 31] and multi-frame interpolation [20]. SuperSlomo [20] is a recently proposed state-of-the-art method for variable-length multi-frame interpolation approach. We adapt this method for GIF frame interpolation to enhance the temporal resolution of the input GIFs.

3. GIF Generation and Artifacts

The three main steps of creating a GIF from a video are: (1) frame sampling, (2) color quantization, and (3) color dithering. Frame sampling reduces the file size of the obtained GIF, but it also lowers the temporal resolution of the video content. In this section, we will provide more details about the color quantization and color dithering processes and the resulting visual artifacts as seen in Figure 1.

3.1. GIF Color Quantization

The GIF color quantization process takes an input image $I \in R^{H\times W\times 3}$ and a color palette $\mathcal{C} \in R^{N\times 3}$ of N distinct colors, and produces a color-quantized GIF image G. The quantization is computed for each pixel, thus G has the same width and height as the input image. $G_{i,j}$ at pixel (i, j) is simply set to the color in the palette $\mathcal{C}$ closest to the input color $I_{i,j}$, i.e., $G_{i,j} = \text{argmin}_{c\,\in\,\mathcal{C}} \left\|I_{i,j} - c\right\|_2^2$. The color palette $\mathcal{C}$ could be optimized with a clustering algorithm to minimize the total quantization error $\|I - G\|_2^2$. Different clustering algorithms are used in practice, but Median Cut [13] is the most popular one due to its computational efficiency.

Most of the visual artifacts in GIFs are produced by the color quantization process with a tiny color palette ($N = 256, 32, ...$). As illustrated in Figure 1, the three most noticeable types of artifacts are (1) flat regions, (2) false contours, and (3) color shift. We notice a GIF image has a lot of connected components in which the color values are the same, which will be referred to as "flat regions". Flat regions are created because neighboring pixels with similar colors are quantized into the same color bin in the palette. False contours also emerge at the boundaries between flat regions with close color values. This is because the continuity of the color space has been broken and the color change cannot be gradual. We also notice the color shift between the input image and the GIF is larger for certain small regions such as the lips of the baby in Figure 1. This is because the color palette does not spend budget on these small regions even though they have unique, distinct colors.

3.2. GIF Color Dithering

Color quantization with a small color palette yields substantial quantization error and artifacts. Color dithering is a technique that can be used to hide quantization error and alleviate large-scale visual patterns such as false contours in GIFs. The most popular color dithering approach is Floyd-Steinberg dithering [8]. It diffuses the quantization error from every pixel to its neighboring pixels in a feedback process. The dithered GIF has the same exact small color palette. However, it appears to have more colors. The idea is to use a neighborhood of mixed colors to visually approximate a color that is not in the color palette.

Color dithering produces its own visual artifacts, which are noise-like dotted patterns. These dotted patterns are apparent when one pays attention to local regions. This type of artifact is somewhat easier to remove, because the dithered GIFs preserve more color information with the help of neighboring pixels using the error-diffusion algorithm than the non-dithered GIFs. It is worth noting that, even with color dithering, GIFs still contain flat regions, false contours, and shifted colors.

4. Our Approach

Our method converts a sequence of GIF frames into a video that has a substantially higher visual quality. There are two main steps: color dequantization and frame interpolation. For color dequantization, we develop a new compositional ConvNet, inspired by the iterative optimization procedure of the Lucas-Kanade algorithm [28]. This network is trained by combining reconstruction loss and generative adversarial loss on both the color values and the image gradient vectors. After performing color dequantization, we apply an improved video frame interpolation method to increase the temporal resolution of the output video.

4.1. Color Dequantization

Let $G = f_{\mathcal{C}}(\hat{I})$ denote the color quantization function, where G and $\hat{I}$ are the GIF image and the original input image respectively, and $\mathcal{C}$ is the color palette used for quantization. The goal of color dequantization is to recover the original image given the GIF image G and the color palette $\mathcal{C}$, i.e., $I = f_{\mathcal{C}}^{-1}(G)$. However, the quantization function $f_{\mathcal{C}}$ is a many-to-one mapping, so color dequantization is an ill-posed problem. Our proposed approach embeds the quantization function $f_{\mathcal{C}}$ itself into the compositional network, which provides valuable information to guide the learning and inference of the inverse function.

4.1.1 Compositional Architecture

Given the color quantized GIF image G and the color palette $\mathcal{C}$, we seek image I that is close to the ground truth image $\hat{I}$, and at the same time satisfies the color quantization constraint $f_{\mathcal{C}}(I) = G$. This can be formulated as an optimization problem, minimizing the reconstruction error between I and $\hat{I}$ as well as between $f_{\mathcal{C}}(I)$ and $G = f_{\mathcal{C}}(\hat{I})$, i.e.,

$$\min_{I} \left\| I - \hat{I} \right\|_2^2 + \lambda \left\| f_{\mathcal{C}}(I) - G \right\|_2^2 . \tag{1}$$

The first loss term is the reconstruction error between the recovered image and the ground truth image, which can be directly computed based on the output of the neural network and the target ground truth image. However, the second

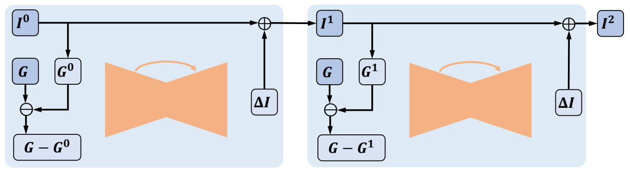

Figure 2. **Architecture of the proposed CCDNet.** Given the current image estimation I^t, we first compute its color quantified image G^t using the same color palette of input GIF G. A UNet module then takes $(I^t, G, G^t, G - G^t)$ as input and outputs ΔI, which will be added to the current image estimation. This process can be iteratively applied during training and test time.

loss term $\left\| f_{\mathcal{C}}(I) - G \right\|_2^2$ cannot be directly used as a proper loss for I because the derivative of the quantization function with respect to the input image, $\frac{\partial f_{\mathcal{C}}(I)}{\partial I}$, is 0 almost everywhere. This is because the quantization process uses a tiny color palette.

We propose to use Lucas-Kanade to iteratively optimize for the second loss term. In each iteration, we compute an update for the recovered image to further minimize the loss:

$$\min_{\Delta I} \left\| f_{\mathcal{C}}(I + \Delta I) - G \right\|_2^2 , \tag{2}$$

where ΔI is the update to the current estimation of the ground truth image I. The Lucas-Kanade algorithm assumes $f_{\mathcal{C}}(I + \Delta I)$ is a linear function of ΔI for small ΔI, and it can be well approximated by the first order Taylor series expansion, i.e., $f_{\mathcal{C}}(I + \Delta I) \approx f_{\mathcal{C}}(I) + \frac{\partial f_{\mathcal{C}}(I)}{\partial I} \Delta I$. Thus, solving Equation (2) can be approximated by solving:

$$\min_{\Delta I} \left\| f_{\mathcal{C}}(I) + \frac{\partial f_{\mathcal{C}}(I)}{\partial I} \Delta I - G \right\|_2^2 . \tag{3}$$

The above is a quadratic program with respect to ΔI, and there is a closed-form solution for the optimal update:

$$\Delta I = \left(\frac{\partial f_{\mathcal{C}}(I)}{\partial I} \right)^{+} (G - f_{\mathcal{C}}(I)), \tag{4}$$

where $^+$ denotes the pseudo-inverse operator. The Lucas-Kanade algorithm iterates between computing the above update value and updating the parameters: $I = I + \Delta I$.

Equation (4) suggests the update parameter ΔI is a simple linear function of the difference between the two GIF images. In practice, however, the true relationship between the quantization function and the input image is seldom linear. In this case, the linear approximation given by Taylor series expansion is not tight, and a simple linear model to compute the optimal update might not have enough capacity to fit the data. Instead, we propose to replace the linear function by a deep ConvNet. Specifically, we propose to use the U-Net architecture [34] to estimate the optimal update ΔI. Equation 4 becomes:

$$\Delta I = g(I, G, f_{\mathcal{C}}(I), G - f_{\mathcal{C}}(I)), \tag{5}$$

where g denotes the deep ConvNet that we need to learn. Following the iterative optimization scheme of the Lucas-Kanade algorithm, we alternate between computing the update direction and updating the de-quantized image:

$$\begin{aligned} \Delta I &= g(I^t, G, G^t, G - G^t), \\ I^{t+1} &\leftarrow I^t + \Delta I \end{aligned} \tag{6}$$

where $G^t = f_{\mathcal{C}}(I^t)$. This leads to the proposed Compositional Color Dequantization Network (CCDNet). The compositional architecture of the CCDNet is illustrated in Figure 2. Let I^t be the current estimation of the ground truth image, we first apply $f_{\mathcal{C}}$ (the same color quantization function used to generate G) to I^t to obtain G^t. Ideally G^t should be identical to G. However if there is difference between the two quantified image, the difference $G - G^t$ will provide valuable information for estimating ΔI as shown in Equation 4. Therefore, we concatenate $(I^t, G, G^t, G - G^t)$ and apply network g again to compute ΔI, which is subsequently used to update the estimated image I. This process can be iteratively applied for multiple steps.

The CCDNet can be trained by unfolding the architecture multiple times, as illustrated in Figure 2. Suppose a CCDNet is unfolded by k times, we refer to the corresponding model as CCDNet-k. Note that the same U-Net module is shared across all unfolding steps except for the first step. Reusing the same U-Net module dramatically reduces the number of model parameters compared to an alternative approach where the U-Net parameters at different stages are not shared. We allow the U-Net at the first unfolding step to have separate parameters from the rest because it expects different inputs (I^t and G^t are undefined for $t = 0$).

For a color-dithered GIF, the exact quantization function $f_{\mathcal{C}}$ is unknown, due to the missing information about the error diffusion step. Different GIF creation software programs use different error diffusion algorithms, and information about the algorithm is not stored in a GIF file. For color-dithered GIFs, we propose not to compute G^t and $G - G^t$ in CCDNet. Fortunately, color-dithered and non-dithered GIFs have different local patterns, and they can easily recognized by a simple classifier. We propose to train two separate CCDNets for the color-dithered and non-dithered GIFs, and use a trained classifier to route an input GIF to the corresponding network.

4.1.2 Color Dequantization Loss

Let G_i be the input GIF image to the CCDNet and I_i the corresponding output. We want to train a CCDNet so that

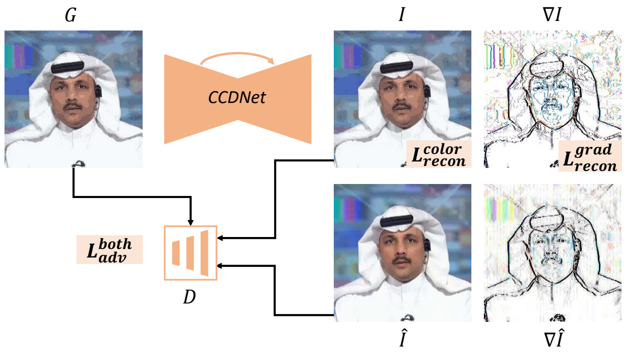

Figure 3. **Color Dequantization Loss.** The proposed loss in Equation 7 measures the differences between the estimated image I and the groundtruth image $\hat{I}$ based on both the color values and the gradient values. We can also train CCDNet using the conditional GAN framework to encourage even more realistic image outputs.

I_i is as close to the original image $\hat{I}_i$ as much as possible. We propose to use the loss function described in Equation (7) to measure the discrepancy between these two images.

$$\mathcal{L}_{recon}^{color}(I_i, \hat{I}_i) + \mathcal{L}_{recon}^{grad}(I_i, \hat{I}_i) + \lambda_{adv}\mathcal{L}_{adv}^{both}(I_i, \hat{I}_i). \tag{7}$$

This loss function measures the differences between I_i and $\hat{I}_i$ based on both the color values and the gradient values. To get sharper image estimation, we use L_1 norm to compute the reconstruction loss $\mathcal{L}_{recon}^{color}$ and $\mathcal{L}_{recon}^{grad}$:

$$\mathcal{L}_{recon}^{color} = \left\| I_i - \hat{I}_i \right\|_1, \text{ and } \mathcal{L}_{recon}^{grad} = \left\| \nabla I_i - \nabla \hat{I}_i \right\|_1. \tag{8}$$

We can also optimize the CCDNet using the conditional GAN framework, to encourage the outputs of the network to have the same distribution as the original ground truth images. This can be done by adding an adversarial loss function defined on both the color and gradient values:

$$\mathcal{L}_{adv}^{both} = \log D(G_i, \hat{I}_i, \nabla \hat{I}_i) + \log(1 - D(G_i, I_i, \nabla I_i)),$$

where D is the discriminator function for distinguishing between the set of ground truth images $\{\hat{I}_i\}$ and the set of estimated images $\{I_i\}$. λ_{adv} is set to 0.01 or 0, depending on whether the adversarial loss is enabled. Experiments show it is critical to include the losses computed on the image gradient values. Compared to the original images, GIF images have drastically different gradient signatures (due to flat regions, false contours, dotted patterns), so it is much more effective to use additional losses on the image gradients.

4.2. Temporal Interpolation

We adapt the recently proposed SuperSlomo [20] network to reverse the process of frame sampling and increase the temporal resolution of GIFs. SuperSlomo is designed

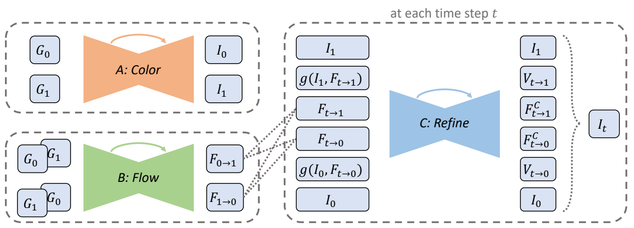

Figure 4. **Overview of the GIF2Video pipeline.** Network A performs color dequantization on two consecutive input GIF frames G_0 and G_1; Network B estimates the bidirectional flow maps between them; Network C receives the outputs from A and B, and produces the interpolated frames I_t's for $t \in (0, 1)$. We use the proposed CCDNet2 as Network A and the modified SuperSlomo as Network B and C.

for variable-length multiple-frame interpolation. Given two consecutive frames at time steps $t = 0$ and $t = 1$, SuperSlomo in one step can interpolate frames anywhere between $t = 0$ and $t = 1$. This is more efficient than methods where only the middle frame $t = 0.5$ is produced. More details about SuperSlomo can be found in [20].

We implement SuperSlomo and adapt it to our task. Figure 4 depicts the entire GIF2Video pipeline, with the adaptation is shown in (B) and (C). This algorithm has three major components. Network A performs color dequantization and outputs the estimated ground truth images I_0 and I_1. Network B estimates the bidirectional flow maps $F_{0\to1}$ and $F_{1\to0}$ between the two input frames. Network C receives the outputs of network A and B, and it produced interpolated frames I_t's for $t \in (0, 1)$. We use the proposed CCDNet as network A, while network B and C are both U-Net modules from [20]. Note that network B estimates the optical flow directly from the input GIF images, instead of using the outputs of network A. This allows networks A and B to run in parallel. Experiments show this parallel model performs similarly to the alternative sequential model.

5. Datasets

With methods presented in Section 3, we can convert any video frame to a GIF image. This allows us to train the CCDNet with a large amount of training image pairs. As a byproduct, we introduce two GIF-Video datasets: GIF-Faces and GIF-Moments. The former is designed to be face-centric, while the latter is more generic and built on real GIFs shared by Internet users. Figure 5 shows some GIF frames (non-dithered) of the two datasets. Images in the first row are from the GIF-Faces dataset, and they also cover parts of the upper-body with diverse background scenes. The second row shows images from the GIF-Moments dataset. They contain diverse content, covering a wide range of scenes including sports, movie, and animation. Details about these two datasets are provided below.

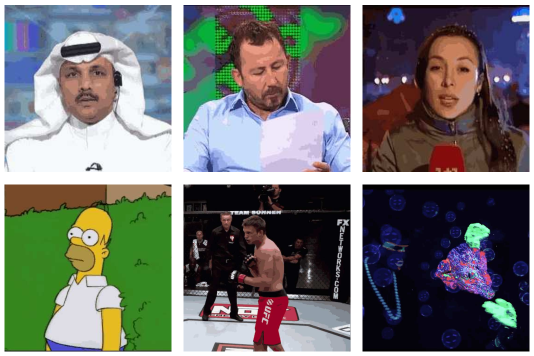

Figure 5. **Example GIF frames from GIF-Faces and GIF-Moments.** First row: GIF-Faces (face-centric); Second row: GIF-Moments (generic GIFs shared by Internet users).

5.1. GIF-Faces

A large portion of online GIFs are face-centric, and they contain noticeable artifacts on face regions such as cheeks and lips. Given the popularity of face-centric GIFs, there are strong motivations for optimizing the network on faces. The GIF-Faces dataset was designed for such a purpose.

We first extracted a large number of face-centric video clips from the FaceForensics dataset [35]. Most of the faces in FaceForensics have near-frontal poses and neutral expression changes across frames. Given a video from FaceForensics, we first detected human faces on every frame, then computed a minimal square region which covers all the detected faces. We further expanded this square region by a factor of 1.5 to increase the coverage of the scene.

A total of 987 high-quality face-centric video clips were extracted from the FaceForensics dataset. The frames of these videos were resized to have $256p\times256p$ resolution and the temporal resolution was unaltered. We used 883 videos for training and 104 for evaluation. There are around 500 frames on average in each video. The corresponding GIF frames (both dithered and non-dithered) were computed from these face-centric videos with the color palette size set to 32. We use 32 as the default color palette size, to make the color dequantization task as challenging as possible, yet not unreasonable. To deal with GIFs of different palette sizes, we can simply read their palette sizes and route them to appropriate models trained on similar levels of color quantization.

5.2. GIF-Moments

We also curated GIF-Moments, a dataset of generic GIFs shared by Internet users. Specifically, Gygli et al. [11] crawled popular GIF-sharing websites and collected 100K GIFs and their corresponding original videos. For each

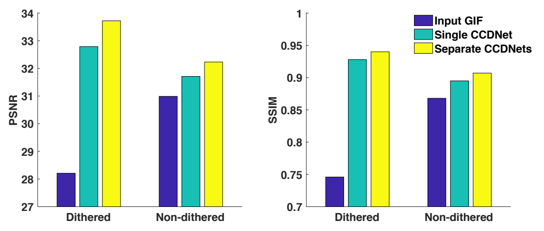

Figure 6. **Benefits of having separate dequantization networks for color-dithered and non-dithered GIFs**. Cyan: using a single CCDNet for both types of images. Yellow: using separate dedicated networks. Here we use the GIF-Faces dataset to train CCDNet1 with $\lambda_{adv} = 0.01$. The performance is measured by PSNR and SSIM (higher is better).

GIF clip, their dataset provides the corresponding YouTube video ID and the start frame and end frame. These video moments are generic and diverse, covering a wide range of video categories and topics such as sports, movie, and animation. We first downloaded all the candidate videos from YouTube in their highest resolution, and temporally cropped the videos using the annotated start and end frames. We only kept videos of high visual quality with sufficient spatial and temporal resolution: the width and height must be at least $360p$, the temporal resolution is no less than 20fps, and the total number of frames has to be more than 40.

In the end, we had a collection of 71,575 video clips, with a total of 12 million frames. We use 85%, 5%, and 10% of the videos for training, validation, and evaluation respectively. Similar to GIF-Faces, we computed the corresponding GIF frames (both dithered and non-dithered) with the color palette size set to 32.

6. Experiments

In our experiments, PSNR (Peak Signal to Noise Ratio) and SSIM (Structural Similarity Index) are used as evaluation metrics. PSNR is defined via the root mean squared error (RMSE) between the estimated images and the ground truth images. More specifically, $PSNR = 20 \log_{10} \frac{MAX}{RMSE}$. Roughly speaking, 1dB, 2dB, and 3dB improvement on the PSNR are equivalent to 10%, 20%, and 30% RMSE reduction in the image color space respectively. SSIM is a perceptual metric that quantifies image quality. We first compute PSNR and SSIM for each frame and average them within each video, and finally average them across all videos in the test set.

6.1. GIF Color Dequantization

6.1.1 Dichotomy of GIF Color Dithering Mode

The color dithering step is optional in the GIF generation process. It is up to the creation tools or the end users to decide whether or not to perform color dithering after color quantization. We observe that color-dithered GIFs are more popular than the non-dithered ones on the Internet. Color-dithering is preferred because it can reduce the noticeability of large visual artifacts such as false contours in GIFs. However, non-dithered GIFs are also widely shared. They also exhibit more artifacts and are more challenging for the task of GIF color dequantization.

Should we learn a single CCDNet or two separate CCDNets for the color-dithered and the non-dithered GIFs? The latter approach is better, as suggested by Figure 6. This figure shows the results of an experiment on the GIF-Faces dataset, where we train a CCDNet1 model with $\lambda_{adv} = 0.01$. From Figure 6, we observe that learning a single CCDNet with both dithered and non-dithered GIFs used as training data reduces the GIF color dequantization performance, measured by PSNR and SSIM. We also observe that it is easier to restore color values for the dithered GIFs. The reason is that dithered GIFs preserve more color information than non-dithered ones with the help of neighboring pixels using error-diffusion algorithms.

The benefits of having separate CCDNets for color-dithered and non-dithered GIFs is clear. However, at test time, the color-dithering mode of a GIF image is not stored in the file. Fortunately, the color-dithered and non-dithered GIFs exhibit very different local patterns especially on the gradient space. We therefore can train a classifier to infer whether an input GIF is dithered or not. We trained a simple classifier with only five Conv layers on the GIF-Faces training set. It achieves 100% and 98.6% accuracy on the GIF-Faces and GIF-Moments test sets respectively. The model is a CNN with layers: $C(9,64) \to NL \to C(64,128) \to NL \to C(128,256) \to NL \to C(512,1) \to GAP$. $C(m,n)$ represents a Conv layer with m input channels and n output channels. NL stands for Non-Linearity, which is one BatchNorm followed by one LeakyReLU (negative slope: 0.2). GAP is short for Global Average Pooling. The input is a GIF frame with its gradient maps.

6.1.2 Ablation Study on Networks & Losses

We perform extensive ablation study and quantitative evaluation on the GIF-Faces dataset. From these experiments, we draw several conclusions below.

U-Net is an effective building block for CCDNet. Function g in Equation (6) denotes a deep neural network for computing the iterative update. There are many candidate network architectures that can be used for g. We experiment with three models that have been successfully used for other tasks that are similar to color dequantization: U-Net [34], DRRN [37], and GLCIC [17]. The U-Net architecture allows multi-level information to shortcut across the network and is widely used for the task of image segmentation and image-to-image translation [19]. DRRN (Deep

#	Method	GIF-Faces Non-dithered	GIF-Faces Dithered
1	DRRN-1	30.52/0.874	-
2	GLCIC-1	30.71/0.883	-
3	UNet-1	**32.23/0.907**	**33.72/0.940**
4	UNet-1 (no grad loss)	31.20/0.884	32.68/0.927
5	UNet-1 (no adv loss)	**32.83/0.918**	**33.90/0.944**
6	UNet-2	32.65/0.911	34.31/0.943
7	UNet-3	**32.85/0.917**	**34.43/0.945**
8	UNet-2 (no adv loss)	**34.05/0.928**	**35.63/0.956**
9	UNet-3 (no adv loss)	33.75/0.927	34.59/0.950
10	Pix2Pix [19]	31.41/0.895	32.80/0.925
11	FCDR [15]	31.51/0.878	-
12	Gaussian ($\sigma = 0.5$)	31.36/0.876	31.20/0.873
13	Gaussian ($\sigma = 1.0$)	29.26/0.797	29.75/0.815
14	GIFs (palette: 32)	30.99/0.868	28.21/0.746

Table 1. **Quantitative Results of GIF Color Dequantization on GIF-Faces.** Row 1-9 are the results of CCDNet with different settings. UNet-k stands for CCDNet-k with UNet as backbone. Row 10-13 are the results of several existing methods. The performance is measured by PSNR and SSIM (higher is better).

Recursive Residual Network) is a state-of-the-art network for single image super-resolution. It can substantially reduce the amount of model parameters by applying residual learning in both global and local manners. GLCIC (Globally and Locally Consistent Image Completion) is proposed for the task of image and video inpainting [41]. The mid-layers of GLCIC are dilated Conv layers [43], allowing to compute each output pixel with a much larger input area without increasing the amount of model parameters.

The results are shown in Table 1 (Row 1-3) and Figure 7 (a). As can be observed, using U-Net as the basic module for CCDNet is significantly better than using DRRN or GLCIC. We believe that DRRN's capability of recovering colors from severely quantized GIFs is limited by its small parameter size. And GLCIC is generally bad at predicting image within regions of high-frequency textures (e.g., stripes on clothes, text on background).

It is critical to include the loss defined on the gradient values. Row 3 of Table 1 shows the color dequantization performance achieved by training CCDNet1 with all loss terms, whereas Row 4 shows the performance when the loss on the gradient values is excluded. More specifically, to disable the loss of the gradient values, we discard the gradient reconstruction loss $\mathcal{L}_{recon}^{grad}$ and stop using ∇I as input channels to the adversarial discriminator D. Comparing

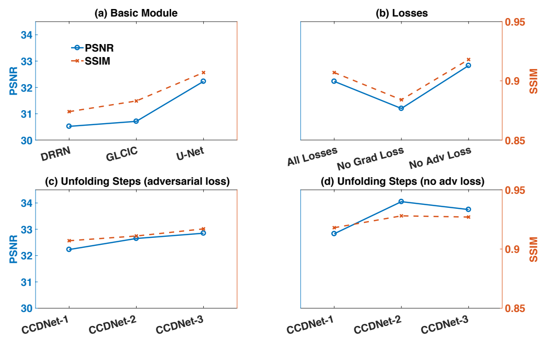

Figure 7. **Ablation study of CCDNet on GIF-Faces dataset.** (a) U-Net is a more effective building block for CCDNet than others. (b) It is critical to include the loss defined on the gradient values, and using adversarial loss yields more realistic images. (c, d) It is beneficial to unfold CCDNet by multiple steps.

Sample Rate	GIF-Faces GIF	GIF-Faces GIF2Video	GIF-Moments GIF	GIF-Moments GIF2Video
1/1	30.99/0.868	**34.05/0.928**	33.71/0.902	**36.10/0.948**
1/2	30.02/0.857	**33.27/0.921**	29.05/0.859	**31.92/0.918**
1/4	29.01/0.842	**32.08/0.908**	26.16/0.812	**28.38/0.865**
1/8	27.41/0.815	**30.20/0.884**	23.29/0.751	**24.95/0.800**

Table 2. **Results of temporal GIF frame interpolation.** The visual quality of created GIFs quickly deteriorates as the temporal downsampling factor increases from 1 to 8. The proposed GIF2Video improves the PSNR of recovered videos by 3dB on GIF-Faces dataset, that is equivalent to 30% root-mean-square-error reduction on the pixel color values.

Row 3 and 4 (also see Figure7(b)), we observe a significant quantitative performance drop after disabling the image gradient-based losses. But more importantly, we find that without the gradient-based losses, the network fails to reduce the artifacts such as flat regions and false contours. The reason can be seen from Figure 3: the difference between I and $\hat{I}$ is more apparent in the gradient space than in the original color space. Because the artifacts exhibited in GIFs have drastically different gradient signatures from the ground truth images. For example, the flat regions in GIFs have zero image gradient while the dotted patterns in GIFs exhibit noise-like gradient patterns.

Using adversarial loss yields more realistic images. Comparing Row 3 and 5 of Table 1 and also considering Figure7(b), we observe that after removing the adversarial loss $\mathcal{L}_{adv}^{both}$, the quantitative performance measured by PSNR/SSIM actually improves. This is not surprising, as it is aligned with many previous studies using adversarial learning. The adversarial loss is designed to make the output images more realistic, which is sometimes not perfectly

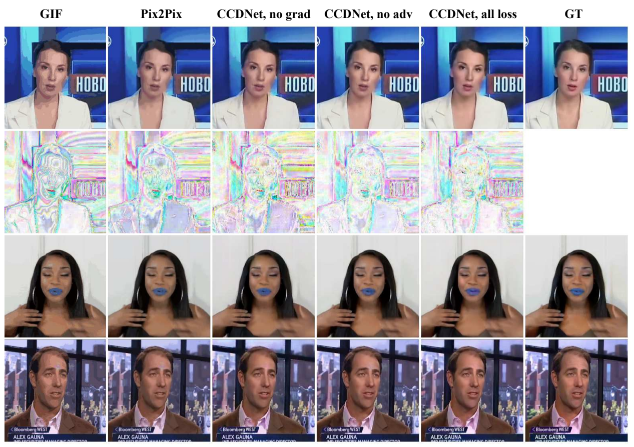

Figure 8. **Qualitative Results of GIF Color Dequantization on GIF-Faces.** Pix2Pix and CCDNet trained without image gradient-based losses cannot remove quantization artifacts such as flat regions and false contours very well. Training CCDNet with adversarial loss yields more realistic and colorful images (see the color of the skin and the lips). Best viewed on a digital device.

aligned with the goal of improving quantitative measures such as PSNR or SSIM. Looking at qualitative results, we find that the adversarial loss is indeed helpful to make the output images more realistic. We also performed a small scale user study involving five subjects. We displayed a pair of images produced by the two CCDNet2 (with and without $\mathcal{L}_{adv}$) in randomized order. The subjects chose which is more realistic to them. The three choices and the percentage being chosen are as follows. *Use* $\mathcal{L}_{adv}$: 53%; *No* $\mathcal{L}_{adv}$: 12%; *Not Sure*: 35%.

It is beneficial to unfold CCDNet by multiple steps and embed the quantization process into the CCDNet. As illustrated in Figure 2, the proposed CCDNet is a compositional architecture that can be unfolded by multiple steps. Empirically, it is beneficial to do so, as can be seen from Figure 7(c) and (d). We observe that with more unfolding steps, the CCDNet can estimate the ground truth image more accurately especially around the object boundaries. We also investigate if it is effective to embed the GIF color quantization process into the CCDNet. In Equation 4, 5 and 6, we derive that the difference image between the input GIF and the corresponding GIF of the current estimation, i.e., $G - G^t = G - f_C(I^t)$, provides valuable information and guidance on how to update the current estimation. If we remove G^t and $G - G^t$ from the input channels to the U-Net basic module, the color dequantization performance of CCDNet2 will decrease significantly. For CCDNet2 trained without $\mathcal{L}_{adv}$, PSNR/SSIM drops from 34.05/0.956 to 33.40/0.923. For CCDNet2 trained with $\mathcal{L}_{adv}$, PSNR/SSIM drops from 32.65/0.911 to 32.48/0.904.

6.1.3 Comparison to Other Methods

Table 1 Row 10-13 report the color dequantization performance of several other methods on GIF-Faces dataset. We first consider applying Gaussian Smoothing with different kernel sizes (Row 12, 13). As expected, the color dequantization performance of this naive approach is really poor. We then implement FCDR (False Contour Detection & Removal [15]), a recently proposed state-of-the-art method for image bit-depth superresolution. It can alleviate mild color quantization artifacts introduced by image bit-depth reduction. However, the color quantization used in GIF generation is far more aggressive than that in image bit-depth reduction. FCDR cannot handle severe GIF artifacts, as listed in Row 11. We also tested Pix2Pix [19], an adversarial network designed for image-to-image translation tasks. It performs similarly to our CCDNet1 trained without image gradient-based losses.

6.2. Temporal GIF Frame Interpolation

Table 2 shows the performance of the proposed GIF2Video algorithm on non-dithered GIF-Faces and GIF-Moments datasets. The performance is measured by PSNR/SSIM (higher is better). For this experiment, we use CCDNet2 trained without the adversarial loss for color dequantization. As can be observed, the visual quality of created GIFs quickly deteriorates as the temporal downsampling factor increases from 1 to 8. For a large downsampling factor, the visual quality of GIF-Moments is worse than that of GIF-Faces. This is because the GIF-Moments dataset contains more dynamic content and larger motions. With the proposed GIF2Video algorithm, we are able to improve the PSNR of recovered videos by 3dB on GIF-Faces dataset, that is equivalent to 30% root-mean-square-error reduction in the image color space.

7. Conclusions

This paper presents GIF2Video, the first learning-based method for enhancing the visual quality of GIFs in the wild. The main tasks of GIF2Video are color dequantization and frame interpolation. For the first task, we propose a novel compositional network architecture CCDNet and a comprehensive loss for training it. For the second task, we adapt SuperSlomo for variable-length multi-frame interpolation to enhance the temporal resolution of input GIFs. Experiments show our method can dramatically enhance the visual quality of input GIFs and significantly reduce quantization artifacts. We hope our method could inspire more solutions to the task of reconstructing video from GIF, such as based on the idea of viewing image sequences as a 3D volume [39, 40], or applying recurrent neural networks to enhance the inter-frame consistency [33].

References

[1] Graphics interchange format, version 89a. https://www.w3.org/Graphics/GIF/spec-gif89a.txt. 1

[2] W. Ahn and J.-S. Kim. Flat-region detection and false contour removal in the digital tv display. In *Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on*, pages 1338–1341. IEEE, 2005. 2

[3] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. J. Black, and R. Szeliski. A database and evaluation methodology for optical flow. *IJCV*, 92(1):1–31, 2011. 2

[4] J. L. Barron, D. J. Fleet, and S. S. Beauchemin. Performance of optical flow techniques. *IJCV*, 12(1):43–77, 1994. 2

[5] S. Bhagavathy, J. Llach, and J. Zhai. Multiscale probabilistic dithering for suppressing contour artifacts in digital images. volume 18, pages 1936–1945. IEEE, 2009. 2

[6] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In *Proc. ECCV*, pages 611–625. Springer, 2012. 2

[7] S. J. Daly and X. Feng. Decontouring: Prevention and removal of false contour artifacts. In *Human Vision and Electronic Imaging IX*, volume 5292, pages 130–150. International Society for Optics and Photonics, 2004. 2

[8] R. Floyd and L. Steinberg. Adaptive algorithm for spatial greyscale. In *Proceedings of the Society of Information Display*, 1976. 2

[9] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Proc. CVPR*, pages 3354–3361. IEEE, 2012. 2

[10] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *NIPS*. 2014. 2

[11] M. Gygli, Y. Song, and L. Cao. Video2gif: Automatic generation of animated gifs from video. In *Proc. CVPR*, 2016. 5

[12] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proc. CVPR*, 2016. 2

[13] P. Heckbert. *Color image quantization for frame buffer display*, volume 16. ACM, 1982. 2

[14] E. Herbst, S. Seitz, and S. Baker. Occlusion reasoning for temporal interpolation using optical flow. *Department of Computer Science and Engineering, University of Washington, Tech. Rep. UW-CSE-09-08-01*, 2009. 2

[15] Q. Huang, H. Y. Kim, W.-J. Tsai, S. Y. Jeong, J. S. Choi, and C.-C. J. Kuo. Understanding and removal of false contour in hevc compressed images. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(2):378–391, 2018. 1, 2, 7, 8

[16] Z. Huang, T. Li, W. Chen, Y. Zhao, J. Xing, C. LeGendre, L. Luo, C. Ma, and H. Li. Deep volumetric video from very sparse multi-view performance capture. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 336–354, 2018. 1

[17] S. Iizuka, E. Simo-Serra, and H. Ishikawa. Globally and locally consistent image completion. *ACM Transactions on Graphics (TOG)*, 36(4):107, 2017. 6

[18] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *Proc. CVPR*, volume 2, page 6, 2017. 2

[19] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *Proc. CVPR*, 2017. 2, 6, 7, 8

[20] H. Jiang, D. Sun, V. Jampani, M.-H. Yang, E. Learned-Miller, and J. Kautz. Super slomo: High quality estimation of multiple intermediate frames for video interpolation. 2018. 1, 2, 4, 5

[21] X. Jin, S. Goto, and K. N. Ngan. Composite model-based dc dithering for suppressing contour artifacts in decompressed video. *IEEE Transactions on Image Processing*, 20(8):2110–2121, 2011. 2

[22] A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012. 2

[23] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 1

[24] J. W. Lee, B. R. Lim, R.-H. Park, J.-S. Kim, and W. Ahn. Two-stage false contour detection using directional contrast and its application to adaptive false contour reduction. *IEEE Transactions on Consumer Electronics*, 52(1):179–188, 2006. 2

[25] J. Liu, W. Sun, and Y. Liu. Bit-depth enhancement via convolutional neural network. In *International Forum on Digital TV and Wireless Multimedia Communications*, pages 255–264. Springer, 2017. 1

[26] Z. Liu, R. A. Yeh, X. Tang, Y. Liu, and A. Agarwala. Video frame synthesis using deep voxel flow. In *Proc. ICCV*, 2017. 2

[27] G. Long, L. Kneip, J. M. Alvarez, H. Li, X. Zhang, and Q. Yu. Learning image matching by simply watching video. In *Proc. ECCV*, pages 434–450. Springer, 2016. 2

[28] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of Imaging Understanding Workshop*, 1981. 2, 3

[29] R. Natsume, S. Saito, Z. Huang, W. Chen, C. Ma, H. Li, and S. Morishima. Siclope: Silhouette-based clothed people. *arXiv preprint arXiv:1901.00049*, 2018. 1

[30] S. Niklaus, L. Mai, and F. Liu. Video frame interpolation via adaptive convolution. In *Proc. CVPR*, 2017. 2

[31] S. Niklaus, L. Mai, and F. Liu. Video frame interpolation via adaptive separable convolution. In *Proc. ICCV*, 2017. 2

[32] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv:1511.06434, 2015. 2

[33] J. S. Ren, Y. Hu, Y.-W. Tai, C. Wang, L. Xu, W. Sun, and Q. Yan. Look, listen and learn - a multimodal lstm for speaker identification. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, pages 3581–3587, 2016. 8

[34] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention*, 2015. 1, 4, 6

[35] A. Rössler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Nießner. Faceforensics: A large-scale video dataset for forgery detection in human faces. *arXiv preprint arXiv:1803.09179*, 2018. 5

[36] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556, 2014. 2

[37] Y. Tai, J. Yang, and X. Liu. Image super-resolution via deep recursive residual network. In *Proc. CVPR*, 2017. 6

[38] P. Wan, G. Cheung, D. Florencio, C. Zhang, and O. C. Au. Image bit-depth enhancement via maximum a posteriori estimation of ac signal. *IEEE Transactions on Image Processing*, 25(6):2896–2909, 2016. 1

[39] C. Wang, Y. Guo, J. Zhu, L. Wang, and W. Wang. Video object co-segmentation via subspace clustering and quadratic pseudo-boolean optimization in an mrf framework. *IEEE Transactions on Multimedia*, 16(4):903–916, 2014. 8

[40] C. Wang, J. Zhu, Y. Guo, and W. Wang. Video vectorization via tetrahedral remeshing. *IEEE Transactions on Image Processing*, 26(4):1833–1844, 2017. 8

[41] C. Wang, H. Huang, X. Han, and J. Wang. Video inpainting by jointly learning temporal structure and spatial details. *arXiv preprint arXiv:1806.08482*, 2018. 7

[42] K. Yoo, H. Song, and K. Sohn. In-loop selective processing for contour artefact reduction in video coding. *Electronics letters*, 45(20):1020–1022, 2009. 2

[43] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. In *ICLR*, 2016. 7

[44] Y. Zhao, W. Chen, J. Xing, X. Li, Z. Bessinger, F. Liu, W. Zuo, and R. Yang. Identity preserving face completion for large ocular region occlusion. *arXiv preprint arXiv:1807.08772*, 2018. 1