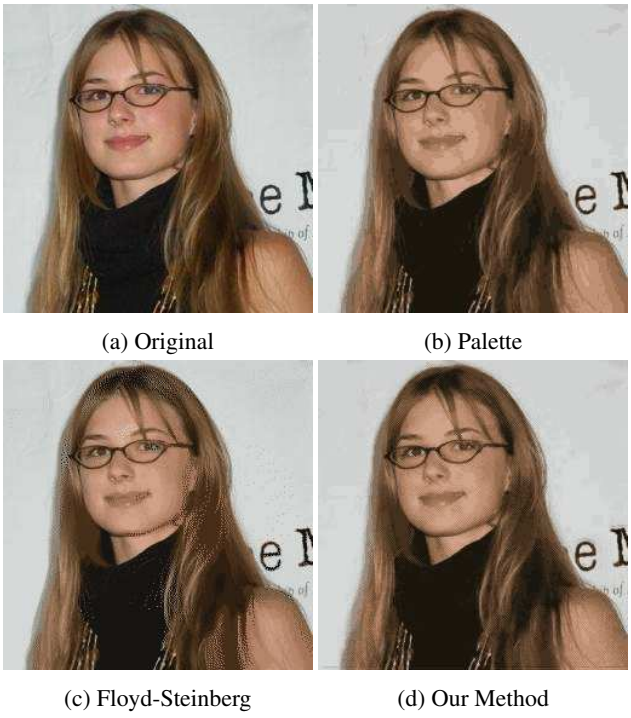
This CVPR 2020 paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

GIFnets: Differentiable GIF Encoding Framework

Innfarn Yoo, Xiyang Luo, Yilin Wang, Feng Yang, and Peyman Milanfar
Google Research - Mountain View, California
[innfarn, xyluo, yilin, fengyang, milanfar]@google.com

Abstract

Graphics Interchange Format (GIF) is a widely used image file format. Due to the limited number of palette colors, GIF encoding often introduces color banding artifacts. Traditionally, dithering is applied to reduce color banding, but introducing dotted-pattern artifacts. To reduce artifacts and provide a better and more efficient GIF encoding, we introduce a differentiable GIF encoding pipeline, which includes three novel neural networks: PaletteNet, DitherNet, and BandingNet. Each of these three networks provides an important functionality within the GIF encoding pipeline. PaletteNet predicts a near-optimal color palette given an input image. DitherNet manipulates the input image to reduce color banding artifacts and provides an alternative to traditional dithering. Finally, BandingNet is designed to detect color banding, and provides a new perceptual loss specifically for GIF images. As far as we know, this is the first fully differentiable GIF encoding pipeline based on deep neural networks and compatible with existing GIF decoders. User study shows that our algorithm is better than Floyd-Steinberg based GIF encoding.



(a) Original (b) Palette

(c) Floyd-Steinberg (d) Our Method

Figure 1: Comparison of our method against GIF encoding with no dithering and Floyd-Steinberg dithering. Compared to GIF without dithering (b) and Floyd-Steinberg (c), our method (d) shows less banding artifacts as well as less dotted noise artifacts. The examples are generated with 16 palette colors.

1. Introduction

GIF is a widely used image file format. At its core, GIF represents an image by applying color quantization. Each pixel of an image is indexed by the nearest color in some color palette. Finding an optimal color palette, which is equivalent to clustering, is an NP-hard problem. A commonly used algorithm for palette extraction is the median-cut algorithm [15], due to its low cost and relatively high quality. Better clustering algorithms such as k-means produce palettes of higher quality, but are much slower, and have $\mathcal{O}(n^2)$ time complexity [6, 21]. Nearly all classical palette extraction methods involve an iterative procedure over all the image pixels, and are therefore inefficient.

Another issue with GIF encoding is the color banding artifacts brought by color quantization as shown in Figure 1b. A popular method for suppressing banding artifacts is dithering, a technique which adds small perturbations in an input image to make the color quantized version of the input image look more visually appealing. Error diffusion is a classical method for dithering, which diffuses quantization errors to nearby pixels [8, 11, 25]. While effective, these error diffusion methods also introduce artifacts of their own, *e.g.*, dotted noise artifacts as shown in Figure 1c. Fundamentally, these methods are not aware of the higher level semantics in an image, and are therefore incapable of optimizing for higher level perceptual metrics.

To this end, we propose a differentiable GIF encoding framework consisting of three novel neural networks: Palet-

teNet, DitherNet, and BandingNet. Our architecture encompasses the entire GIF pipeline, where each component is replaced by one of the networks above. Our motivation for designing a fully differentiable architecture is two-fold. First of all, having a differentiable pipeline allows us to jointly optimize the entire model with respect to any loss functions. Instead of designing by hand heuristics for artifact removal, our network can automatically learn strategies to produce more visually appealing images by optimizing for perceptual metrics. Secondly, our design implies that both color quantization and dithering only involve a single forward pass of our network. This is inherently more efficient. Therefore, our method has $\mathcal{O}(n)$ time complexity, and is easy to be parallelized compared to the iterative procedures in the traditional GIF pipeline.

Our contributions are the following:

- To the best of our knowledge, our method is the first fully differentiable GIF encoding pipeline. Also, our method is compatible with existing GIF decoders.
- We introduce PaletteNet, a novel method that extracts color palettes using a neural network. PaletteNet shows a higher peak signal-to-noise ratio (PSNR) than the traditional median-cut algorithm.
- We introduce DitherNet, a neural network-based approach to reducing quantization artifacts. DitherNet is designed to randomize errors more on areas that suffer from banding artifacts.
- We introduce BandingNet, a neural network for detecting banding artifacts which can also be used as a perceptual loss.

The rest of the paper is structured as follows: Section 2 reviews the current literature on relevant topics. Section 3 gives a detailed explanation of our method. The experiments are given in Section 4, and conclusions in Section 5.

2. Previous Work

We categorize the related literature into four topics: dithering, palette selection, perceptual loss, and other related work.

2.1. Dithering and Error Diffusion

Dithering is a procedure to randomize quantization errors. Floyd-Steinberg error diffusion [11] is a widely used dithering algorithm. The Floyd-Steinberg algorithm sequentially distributes errors to neighboring pixels. Applying blue-noise characteristics on dithering algorithms showed improvement in perceptual quality [25, 34]. Kite *et al*. [18] provided a model for error diffusion as linear gain with additive noise, and also suggested quantification of edge sharpening and noise shaping. Adaptively changing threshold in error diffusion reduces artifact of quantization [9].

Halftoning or digital halftoning are other types of error diffusion methods, representing images as patternized gray pixels. In halftoning, Pang *et al*. [26] used structural similarity measurement (SSIM) to improve halftoned image quality. Chang *et al*. [8] reduced the computational cost of [26] by applying precomputed tables. Li and Mould [22] alleviated contrast varying problems using contrast-aware masks. Recently, Fung and Chan [12] suggested a method to optimize diffusion filters to provide blue-noise characteristics on multiscale halftoning, and Guo *et al*. [13] proposed a tone-replacement method to reduce banding and noise artifacts.

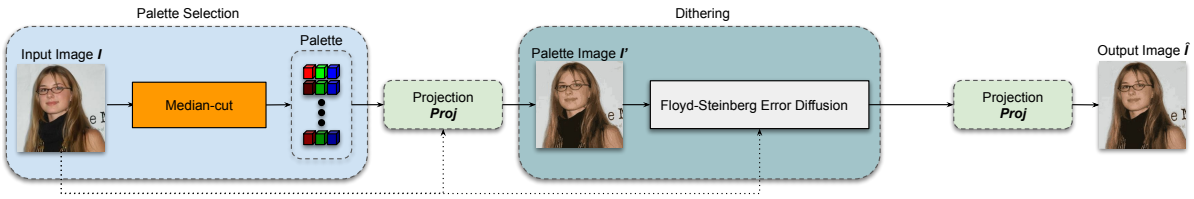
2.2. Palette Selection

Color quantization involves clustering the pixels of an image to N clusters. One of the most commonly used algorithms for GIF color quantization is the median-cut algorithm [5]. Dekker proposed using Kohonen neural networks for predicting cluster centers [10]. Other clustering techniques such as k-means [6], hierarchical clustering [7], particle swarm methods [24] have also been applied to the problem of color quantization [30].

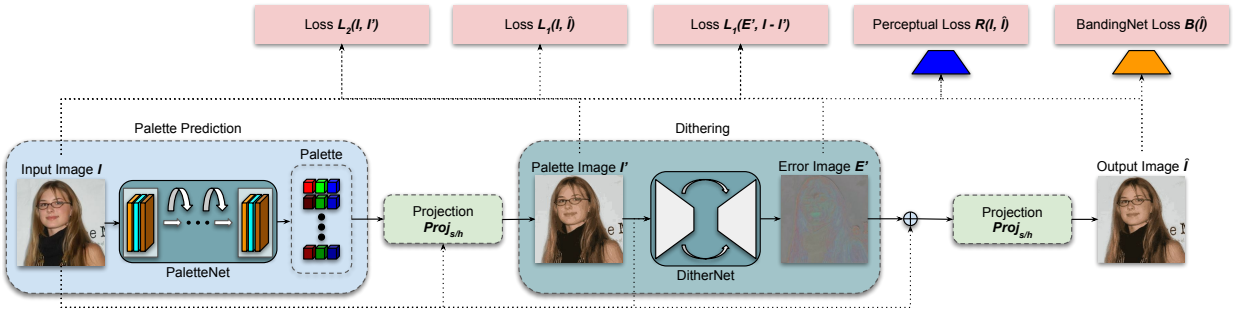
Another line of work focuses on making clustering algorithms differentiable. Jang *et al*. [16] proposed efficient gradient estimators for the Gumbel-Softmax which can be smoothly annealed to a categorical distribution. Oord *et al*. [35] proposed VQ-VAE that generates discrete encodings in the latent space. Agustsson *et al*. [1] provides another differentiable quantization method through annealing. Peng *et al*. [27] proposed a differentiable k-means network which reformulates and relaxes the k-means objective in a differentiable manner.

2.3. Perceptual Loss

Compared to quantitative metrics, *e.g*., signal-to-noise ratio (SNR), PSNR, and SSIM, perceptual quality metrics measure human perceptual quality. There have been efforts to evaluate perceptual quality using neural networks. Johnson *et al*. [17] suggested using the feature layers of convolutional neural networks as a perceptual quality metric. Talebi and Milanfar [33] proposed NIMA, a neural network based perceptual image quality metric. Blau and Michaeli [3] discussed the relationship between image distortion and perceptual quality, and provided comparisons of multiple perceptual qualities and distortions. The authors extended their discussion further in [4] and considered compression rate, distortion, and perceptual quality. Zhang *et al*. [38] compared traditional image quality metrics such as SSIM or PSNR and deep network-based perceptual quality metrics, and discussed the effectiveness of perceptual metrics.



(a) Traditional Pipeline



(b) Our Pipeline

Figure 2: Traditional GIF pipeline (a) and our GIF encoding pipeline (b). In our pipeline, PaletteNet predicts a near-optimal color palette and applies either a soft or hard projection to produce the quantized output image. DitherNet suppresses the banding artifacts by randomizing quantization errors, and BandingNet provides a perceptual loss to judge the severity of banding artifacts.

Banding is a common compression artifact caused by quantization. There are some existing works [2, 20, 37] about banding artifact detection, where the method proposed by Wang *et al*. [37] achieved strong correlations with human subjective opinions. Lee *et al*. [20] segmented the image into smooth and non-smooth regions, and computed various directional features, *e.g*., contrast and Sobel masks, and non-directional features, *e.g*., variance and entropy for each pixel in the non-smooth region to identify it as "banding boundaries" or "true edges". Baugh *et al*. [2] related the number of uniform segments to the visibility of banding. Their observation was that when the size of most of the uniform segments in an image was less than 10 pixels in area, then there was no perceivable banding in the image. Wang *et al*. [37] extracted banding edges from homogeneous segments, and defined a banding score based on length and visibility of banding edges. The proposed banding metrics have a good correlation with subjective assessment.

2.4. Other Related Work

GIF2Video [36] is a recent work that also involves the GIF format and deep learning. GIF2Video tackles the problem of artifact removal for both static or animated GIF images. This is complementary to our work since GIF2Video is a post-processing method, whereas our method is a preprocessing method.

Our method also uses several well-known neural network architectures, such as ResNet [14], U-Net [29], and Inception [31]. ResNet allowed deeper neural network connections using skip connections and showed significant performance improvement on image classification tasks. Inception used different convolution kernel sizes and merged outputs, and achieved better image classification accuracy than humans. U-Net introduced an auto-encoder with skip connections, and achieved high quality output images in auto-encoder.

3. Method

As shown in Figure 2a, GIF encoding involves several steps: (1) color palette selection; (2) pixel value quantization given the color palette; (3) dithering; (4) reapplying pixel quantization; (5) Lempel-Ziv-Welch lossless data compression. The last step is lossless compression, and it does not affect the image quality. Thus we will focus on replacing the first four steps with neural networks to improve the image quality. To make a differentiable GIF encoding pipeline, we introduce two neural networks: 1) PaletteNet, predicting the color palette from a given input image and 2) DitherNet for reducing quantization artifacts. We also introduce soft projection to make the quantization step differentiable. To suppress banding artifacts, we introduce BandingNet as an additional perceptual loss. Figure 2b shows the overall architecture of the differentiable GIF en-

coding pipeline.

3.1. PaletteNet: Palette Prediction

The goal of PaletteNet is to predict a near-optimal palette with a feed-forward neural network given an RGB image and the number of palette N_p. We emphasize here that at inference time, the output from PaletteNet will be *directly* used as the palette, and no additional iterative optimization is used. Therefore, PaletteNet in effect learns how to cluster image pixels in a feed-forward fashion.

We first state a few definitions and notations. Let $I \in \mathbb{R}^{H\times W\times 3}$ be the input image, $P \in \mathbb{R}^{N_p\times 3}$ a color palette with N_p number of palettes, $Q(I) : \mathbb{R}^{H\times W\times 3} \rightarrow \mathbb{R}^{N_p\times 3}$ be the palette prediction network. We define I' to be the quantized version of I using a given palette P, *i.e.*,

$$I'[\alpha] = P[k], \quad k = \underset{j}{\operatorname{argmin}} |I'[\alpha] - P[j]|^2, \qquad (1)$$

where α is any indexing over the pixel space.

Given the definitions above, PaletteNet is trained in a self-supervised manner with a standard L_2 loss.

$$L_{palette} = \frac{1}{H \times W} \sum_{\alpha} |I[\alpha] - P[k]|^2. \qquad (2)$$

Next, we introduce the *soft projection* operation to make the quantized image I' also differentiable with respect to the network parameters of Q. Note that since the predicted palette P is already differentiable with respect to Q, the non-differentiability of the projected image I' is due to the hard projection operation defined in Equation 1. Therefore, we define a soft projection I'_s which smooths the argmin operation in Equation 1,

$$\textit{Proj}_s(I, P, \alpha) = I'_s[\alpha] = \sum_j w_j \cdot P[j], \qquad (3)$$

where $w_j = \frac{\exp(d_j/T)}{\sum_l \exp(d_l/T)}$, $d_j = \|P[j] - I'_s[\alpha]\|^2$, and T a temperature parameter controlling the amount of smoothing. Note that the soft projection is not required for the training of PaletteNet itself, but needed if we want to chain the *quantized* image I' as a differentiable component of some additional learning system, such as in the case of our GIF encoding pipeline in Figure 2b.

3.2. DitherNet: Reducing Artifacts

Dithering is an algorithm to randomize quantization errors into neighboring pixels in images to avoid banding artifacts. After the color quantization, we define the error image as

$$E(\alpha) = I(\alpha) - \textit{Proj}_h(I(\alpha), P), \qquad (4)$$

where I is the original image, $\textit{Proj}_h$ is a hard projection function that returns the nearest color in P, and α is any indexing over the pixel space. The traditional Floyd-Steinberg algorithm diffuses errors with fixed weights, and updates neighboring pixels in a sequential fashion.

Different from Floyd-Steinberg, our DitherNet D accepts an original image I and its quantized image I' as an input, and directly predicts a randomized error image $E' = D(I, I')$. The predicted error is then added to the original image to produce the final quantized image.

$$\hat{I} = \textit{Proj}_{s/h}(I + E', P), \qquad (5)$$

where $\textit{Proj}_s$ is the soft projection and used for training to maintain differentiability. The hard projection ($\textit{Proj}_h$) is used for inference.

For the network architecture, DitherNet uses a variation of U-Net [29] as a backbone network for creating error images. Training DitherNet is inherently a multi-loss problem. The final output has to remain close to the original, while achieving good visual quality after quantization. Ideally, DitherNet should be able to reduce the errors along the banding areas in a visually pleasing way. To this end, we define the dithering loss, L_{dither}, as a combination of image fidelity measures, perceptual losses, as well as a customized banding loss term to explicitly suppress banding artifacts. We define the L_1 loss as $L_1(x, y) = \sum |x(i) - y(i)|$, where $|\cdot|$ is the absolute difference operator and i is the element index. The dither loss is defined as

$$\begin{aligned} L_{dither} = \lambda L_1(I, \hat{I}) + \gamma L_1(E', I' - I) \\ +\delta B(\hat{I}) + \theta R(I, \hat{I}), \end{aligned} \qquad (6)$$

where B is the banding loss given by BandingNet (see Section 3.3), R is a perceptual loss given by either NIMA [33] or VGG [17], and λ, γ, δ, and θ are weights of each loss. In Equation 6, $L_1(I, \hat{I})$ preserves the similarity between input and final quantized images, $L_1(E', I - I')$ is for preserving the sum of quantization errors, and $B(\hat{I})$ is the banding loss. We will discuss the effect of each term in Section 4.2.

3.3. BandingNet: Banding Score

We propose a neural network that predicts the severity of banding artifacts in an image. We train the network by distilling from the non-differentiable banding metric in [37]. The output of this network will be used as the loss to guide our DitherNet.

A straightforward way to train the model is to directly use the RGB image as input, and banding scores obtained by [37] as ground truth. We first tried to use a classical CNN model [19] to train the banding predictor, and defined the loss as the mean absolute difference (MAD) between predicted score and ground truth. However, as shown in Figure 3 (red lines), such naive approach is unstable over training, and could not achieve low MAD.

As pointed out in [37], banding artifacts tend to appear on the boundary between two smooth regions. We found

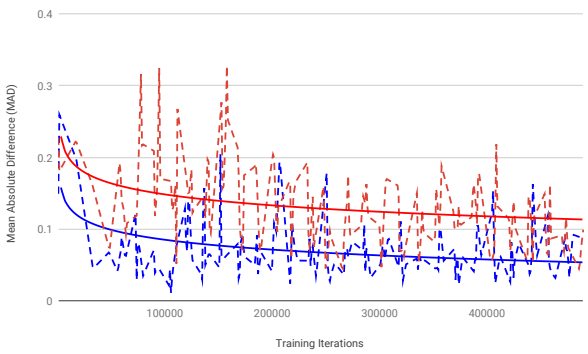


Figure 3: Mean Absolute Difference (MAD) between precomputed and predicted banding scores. Red lines represent MAD without banding edge map, and blue lines show MAD with edge map over training iteration. Solid lines represent the trend of MAD over training iterations.

that adding a map of likely banding artifact areas as input would significantly improve the model. Here we propose a way to extract potential banding edges, and form a new input map (Algorithm 1).

Algorithm 1 Input Generation for BandingNet

function BANDINGINPUTS(RGB)
  Converting RGB to YUV
  Computing gradient G_x and G_y for Y channel only
  Gradient map $G = ||{G_x}^2 + {G_y}^2||_2$
  Weight $W = ((1 - ReLU(1 - G)) * ones(7 \times 7))^2$
  Banding edge map $E = W \cdot G$
  return $M = [Y, E]$
end function

As shown in Figure 4, the extracted edge map is able to highlight potential banding edges, *e.g.*, banding on the background, and set relatively low weights for true edges like eyes and hair. By using this new input map, the training converges faster than using the RGB map, and MAD is also lowered as well, 0.057 vs. 0.088 within the banding score range [0, 1].

To use the BandingNet as a perceptual loss, we propose some additional modifications to the banding network proposed above. First, we augment the training data by adding pairs of GIF encoded images with and without Floyd-Steinberg dithering, and artificially lower the banding score loss for examples with Floyd-Steinberg dithering. This guides the loss along the Floyd-Steinberg direction, and can reduce adversarial artifacts compared to the unadjusted BandingNet. Secondly, we apply our BandingNet on a multi-scale image pyramid in order to capture the banding artifacts at multiple scales of the image. Compared to a single-scale loss, the multi-scale BandingNet promotes randomizing errors on a larger spatial range instead of only near the banding edges. To define the multi-scale loss, we construct the image pyramid in Equation 7.

(a) RGB (b) Y channel (c) Extracted edge

Figure 4: Illustration of banding edge extraction. The RGB image (a) is first converted to YUV, where the Y channel (b) is used to extract the banding edge map (c). Note that only edges in banding artifact areas are extracted.

$$G(I, \eta) = F_{up}(F_{down}(S(I), \eta), 1/\eta), \tag{7}$$

where G is a level of image pyramid, F_{up} denotes image upscaling, F_{down} denotes image downscaling, S is a smoothing function, and η is a scaling factor.

Let $Z(I) : \mathbb{R}^{H \times W \times 3} \rightarrow \mathbb{R}$ be the output of BandingNet. Our final banding loss is defined by

$$\begin{aligned} B_\eta(I) = Z(G(I, \eta)) + Z(G(I, \eta^2)) \\ + Z(G(I, \eta^3)) + Z(G(I, \eta^4)) \end{aligned} \tag{8}$$

For training DitherNet, we use $B_{1.5}(I)$. The exact training parameters for our BandingNet loss can be found in the supplementary materials.

3.4. Overall Training

The overall loss for training our networks is given in Equation 9.

$$\begin{aligned} L_{Total} = L_{palette} + L_{dither} = \\ \beta L_2(I, I') + \lambda L_1(I, \hat{I}) + \gamma L_1(E', I - I') \\ + \delta B(\hat{I}) + \theta R(I, \hat{I}) \end{aligned} \tag{9}$$

To stabilize training, we propose a three-stage method for training all of the networks. In the first stage, BandingNet and PaletteNet are first trained separately until convergence. BandingNet will be fixed and used only as a loss in the next two stages. In the second stage, we fix PaletteNet and only train DitherNet using Equation 9. In the final stage, we use a lower learning rate to fine-tune PaletteNet and DitherNet jointly. We found that the second stage is necessary to encourage DitherNet to form patterns that remove banding artifacts, whereas jointly training PaletteNet and DitherNet from the start will result in PaletteNet dominating the overall system, and a reduced number of colors in the output image.

(a) Original (b) Palette (c) FS Dither (d) Ours

Figure 5: Samples images of our method compared to Median-cut and Floyd Steinberg. (a) original images, (b) quantized GIF with only PaletteNet, (c) Median-cut + Floyd-Steinberg, and (d) PaletteNet + DitherNet (our final method).

4. Experiments

We evaluate our methods on the CelebA dataset [23], where the models are trained and evaluated on an 80/20 random split of the dataset. For both training and evaluation, we first resize the images while preserving the aspect ratio so that the minimum of the image width and height is 256. The images are then center cropped to 256×256.

4.1. PaletteNet

For PaletteNet, we use InceptionV2 [32] as the backbone network, where the features from the last layer are globally pooled before passing to a fully connected layer with $N_p \times 3$ output dimensions. The output is then mapped to $[-1, 1]$ by the tanh activation function. To train PaletteNet, we use the L_2 loss in Equation 2. The details of training parameters are discussed in the supplementary material.

We evaluate the average PSNR of the quantized image compared to the original, for both median-cut and PaletteNet. We also explored using PointNet [28], a popular architecture for processing point cloud input, as a comparison. The traditional method of palette extraction can be viewed as clustering a 3D point cloud, where each point corresponds to an RGB pixel in the image. The point cloud formed by the image pixels clearly retains rich geometric structures, and thus is highly applicable to PointNet. The details of PointNet are discussed in the supplementary material. Results for different values of $N_p \in \{16, 32, 64, 128, 256\}$ are shown in Table 1.

From Table 1, we see that PaletteNet with the InceptionV2 network outperforms median-cut across all values of N_p, where the improvement is more pronounced for lower values of N_p. We also note that the PointNet architec-

PSNR	$N_p = 16$	$N_p = 32$	$N_p = 64$	$N_p = 128$	$N_p = 256$
Median-cut	28.10	30.78	33.27	35.61	37.80
PointNet	26.05	27.93	30.41	32.67	33.09
InceptionV2	**29.24**	**31.6**	**33.75**	**35.81**	**38.08**

Table 1: Average PSNR of quantized images for different palette extraction methods. Top row: Median-cut palette, Middle row: palette from PointNet, Bottom row: palette from InceptionV2.

ture performed worse than the InceptionV2 network and Median-cut.

4.2. DitherNet

We note here that the DitherNet needs to be trained with relatively high weights on the various perceptual losses in order to produce perturbations that improve visual quality. However, raising these weights too much introduces adversarial artifacts that lowers the perceptual loss, but no longer produces examples of high visual quality. To reduce this effect, we augment the input images by changing saturation, hue, and also apply early stopping where we terminate training at epoch 3. The details of training parameters are discussed in the supplementary material.

	Without Banding Loss	With Banding Loss
PSNR	29.19	**28.65**
SSIM	0.854	**0.828**

Table 2: PSNR and SSIM with and without banding loss for color palette with 16 colors.

Table 2 shows that training without banding loss provides better PSNR and SSIM. However, DitherNet trained with banding loss provides better perceptual quality as shown in Figure 6. In our experiment, our method shows better quality with VGG [17] and NIMA [33] perceptual losses as shown in Figure 7.

(a) Original (b) Without BL (c) With BL

Figure 6: The original image before training (a), and the results of training without banding loss (b) and with banding loss (c).

Quantitative metrics, such as PSNR or SSIM, are not proper metrics to compare dithering methods, since dithering is mainly used for improving perceptual quality and

(a) Original (b) No Perc Loss

(c) VGG Loss (d) NIMA Loss

Figure 7: The results of training with and without perceptual loss ($N_p = 16$).

does not necessarily improve PSNR or SSIM over non-dithered images. Instead, to evaluate the visual quality of our algorithm compared to the standard GIF pipeline with Floyd-Steinberg error diffusion and median-cut palette, a pairwise comparison user study was conducted using Amazon's Mechanical Turk. We randomly choose 200 images from the CelebA evaluation data, and produce quantized images from the standard GIF pipeline and our model. Raters are then asked to choose from a pair which image is of higher quality. For each image pair, we collect ratings from 10 independent users.

$N_p = 16$	$N_p = 32$	$N_p = 64$
87.5	85.0	79.4
$N_p = 128$	$N_p = 256$	Average
45.1	47.1	68.8.

Table 3: Favorability of our method compared to Median-cut + Floyd-Steinberg.

Table 3 shows the favorability of our method compared to the standard median-cut with Floyd-Steinberg dithering. We see that our method outperforms the baseline by a large margin when the number of palettes is low, and has comparable performance when the number of palettes is high. There are several causes to the discrepancy in favorability

(a) 16 Palette (b) 32 Palette (c) 64 Palette (d) 128 Palette (e) 256 Palette

Figure 8: Side-by-side comparisons of our method and Floyd-Steinberg for 16, 32, 64, 128, and 256 palettes.

for different numbers of palette levels. First of all, the number of images with visible banding artifacts decreases as the number of palettes increases. On images without banding artifacts, our method is almost identical to that from the standard GIF pipeline, since raters are often not sensitive to the minute differences in PSNR. On images with banding artifacts, a key difference between low and high palette count is in the visibility of the dotted pattern artifacts. When the number of palette levels is low, the dotted patterns are much more visible in the image and often rated unfavorably compared to the patterns from DitherNet. Another reason is the performance gap between PaletteNet and median-cut shrinks as the number of palettes grows (see Table 1).

5. Conclusion

In this paper, we proposed the first fully differentiable GIF encoding pipeline by introducing DitherNet and PaletteNet. To further improve the encoding quality, we introduced BandingNet that measures banding artifact score. Our PaletteNet can predict high quality palettes from input images. DitherNet is able to distribute errors and lower banding artifacts using BandingNet as a loss. Experimental results show that our algorithm achieved better quality than Floyd-Steinberg algorithm. Our method can be extended in multiple directions as future work. For example, k-means based palette prediction and heuristic methods for dithering, *i.e.*, [21], show higher visual quality than ours. We also would like to extend our current work to image reconstruction, static to dynamic GIF, and connecting with other differentiable image file formats.

References

[1] Eirikur Agustsson, Fabian Mentzer, Michael Tschannen, Lukas Cavigelli, Radu Timofte, Luca Benini, and Luc V Gool. Soft-to-hard vector quantization for end-to-end learning compressible representations. In *NeurIPS*, pages 1141–1151. 2017.

[2] Gary Baugh, Anil Kokaram, and François Pitié. Advanced video debanding. In *CVMP*, pages 1–10, November 2014.

[3] Yochai Blau and Tomer Michaeli. The perception-distortion tradeoff. In *CVPR*, pages 6228–6237, June 2018.

[4] Yochai Blau and Tomer Michaeli. Rethinking lossy compression: The rate-distortion-perception tradeoff. In *ICML*, June 2019.

[5] Peter Burger and Duncan Gillies. *Interactive Computer Graphics; Functional, Procedural, and Device-Level Methods*. Addison-Wesley Longman Publishing Co., Inc., USA, 1st edition, 1990.

[6] M. Emre Celebi. Improving the performance of k-means for color quantization. *Image and Vision Computing*, 29(4):260–271, March 2011.

[7] M. Emre Celebi, Quan Wen, and Sae Hwang. An effective real-time color quantization method based on divisive hierarchical clustering. *Journal of Real-Time Image Processing*, 10:329–344, June 2015.

[8] Jianghao Chang, Benoundefinedt Alain, and Victor Ostromoukhov. Structure-aware error diffusion. *ACM Transactions on Graphics*, 28(5):1–8, December 2009.

[9] Niranjan Damera-Venkata and Brian L. Evans. Adaptive threshold modulation for error diffusion halftoning. *IEEE Transactions on Image Processing*, 10(1):104–116, January 2001.

[10] Anthony H. Dekker. Kohonen neural networks for optimal colour quantization. *Network: Computation in Neural Systems*, 5(3):351–367, 1994.

[11] Robert W. Floyd and Louis Steinberg. An adaptive algorithm for spatial greyscale. *Proceedings of the Society for Information Display*, 17(2):75–77, 1976.

[12] Yik-Hing Fung and Yuk-Hee Chan. Optimizing the error diffusion filter for blue noise halftoning with multiscale error diffusion. *IEEE Transactions on Image Processing*, 22(1):413–417, January 2013.

[13] Jing-Ming Guo, Jia-Yu Chang, Yun-Fu Liu, Guo-Hong Lai, and Jiann-Der Lee. Tone-replacement error diffusion for multitoning. *IEEE Transactions on Image Processing*, 24(11):4312–4321, November 2015.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, June 2016.

[15] Paul Heckbert. Color image quantization for frame buffer display. *SIGGRAPH Computer Graphics*, 16(3):297–307, July 1982.

[16] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, November 2016.

[17] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, pages 694–711, September 2016.

[18] Thomas D. Kite, Brian L. Evans, and Alan C. Bovik. Modeling and quality assessment of halftoning by error diffusion. *IEEE Transactions on Image Processing*, 9(5):909–922, May 2000.

[19] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, pages 1097–1105. 2012.

[20] Ji Won Lee, Bo Ra Lim, Rae-Hong Park, Jae-Seung Kim, and Wonseok 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, February 2006.

[21] Kornel Lesiński. Pngquant. pngquant.org, November 2019.

[22] Hua Li and David Mould. Contrast-aware halftoning. *Computer Graphics Forum*, 29(2):273–280, June 2010.

[23] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *ICCV*, pages 3730–3738, December 2015.

[24] Mahamed G. H. Omran, Andries P. Engelbrecht, and Ayed Salman. *Particle Swarm Optimization for Pattern Recognition and Image Processing*, pages 125–151. Springer, Berlin, Heidelberg, 2006.

[25] Victor Ostromoukhov. A simple and efficient error-diffusion algorithm. In *SIGGRAPH*, pages 567–572, August 2001.

[26] Wai-Man Pang, Yingge Qu, Tien-Tsin Wong, Daniel Cohen-Or, and Pheng-Ann Heng. Structure-aware halftoning. In *SIGGRAPH*, pages 1–8, August 2008.

[27] Xi Peng, Joey Tianyi Zhou, and Hongyuan Zhu. k-meansnet: When k-means meets differentiable programming. *arXiv preprint arXiv:1808.07292*, August 2018.

[28] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, pages 652–660, July 2017.

[29] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, pages 234–241, November 2015.

[30] Paul Scheunders. A comparison of clustering algorithms applied to color image quantization. *Pattern Recognition Letters*, 18(11):1379–1384, November 1997.

[31] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, pages 1–9, June 2015.

[32] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, pages 2818–2826, June 2016.

[33] Hossein Talebi and Peyman Milanfar. Nima: Neural image assessment. *IEEE Transactions on Image Processing*, 27(8):3998–4011, August 2018.

[34] Robert Ulichney. Dithering with blue noise. *Proceedings of the IEEE*, 76(1):56–79, January 1988.

[35] Aaron van den Oord, Oriol Vinyals, and koray kavukcuoglu. Neural discrete representation learning. In *NeurIPS*, pages 6306–6315. 2017.

[36] Yang Wang, Haibin Huang, Chuan Wang, Tong He, Jue Wang, and Minh Hoai. Gif2video: Color dequantization and temporal interpolation of gif images. In *CVPR*, pages 1419–1428, June 2019.

[37] Yilin Wang, Sang-Uok Kum, Chao Chen, and Anil Kokaram. A perceptual visibility metric for banding artifacts. In *ICIP*, pages 2067–2071, September 2016.

[38] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of

deep features as a perceptual metric. In *CVPR*, pages 586–595, June 2018.