Deep Unsupervised Pixelization

CHU HAN, Guangdong Provincial Key Laboratory of Computer Vision and Virtual Reality Technology, SIAT, The Chinese University of Hong Kong, and South China University of Technology
QIANG WEN*, South China University of Technology
SHENGFENG HE†, South China University of Technology
QIANSHU ZHU, South China University of Technology
YINJIE TAN, South China University of Technology
GUOQIANG HAN, South China University of Technology
TIEN-TSIN WONG, The Chinese University of Hong Kong and Guangdong Provincial Key Laboratory of Computer Vision and Virtual Reality Technology, SIAT

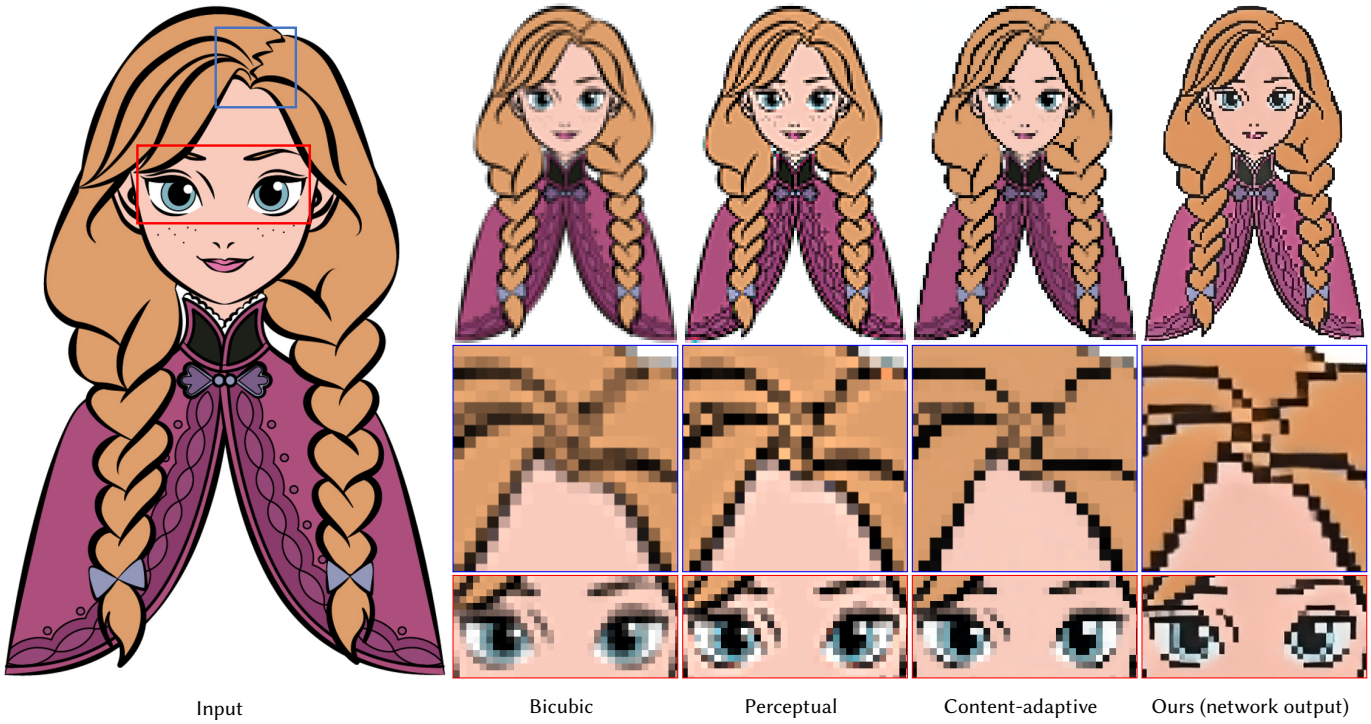


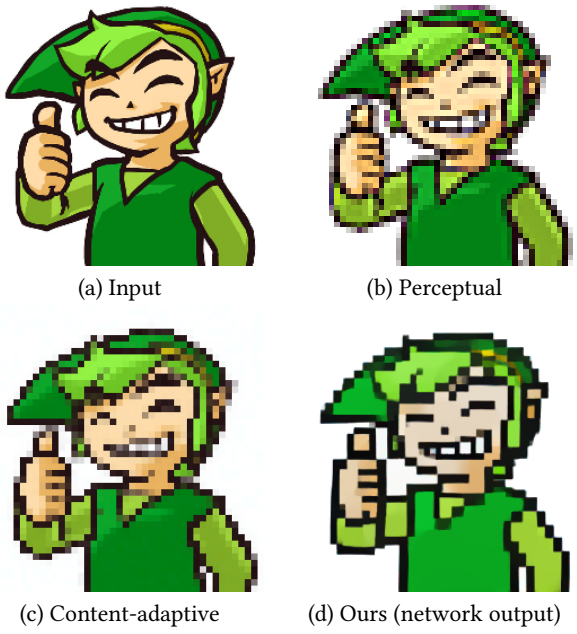
Fig. 1. Existing image downscaling methods are not able to generate pixel arts with sharp enough edges. Our method synthesizes pixel arts with crisp edges and fine local details. The scaling factor of all results is 1/6.

In this paper, we present a novel unsupervised learning method for pixelization. Due to the difficulty in creating pixel art, preparing the paired training data for supervised learning is impractical. Instead, we propose an unsupervised learning framework to circumvent such difficulty. We leverage the dual nature of the pixelization and depixelization, and model these two tasks in the same network in a bi-directional manner with the input itself as

*Chu Han and Qiang Wen are the joint first authors.
†Shengfeng He is the corresponding author (shengfenghe7@gmail.com).

Authors' addresses: Chu Han, Guangdong Provincial Key Laboratory of Computer Vision and Virtual Reality Technology, SIAT, The Chinese University of Hong Kong, South China University of Technology; Qiang Wen, South China University of Technology; Shengfeng He, South China University of Technology; Qianshu Zhu, South China University of Technology; Yinjie Tan, South China University of Technology; Guoqiang Han, South China University of Technology; Tien-Tsin Wong, ttwong@cse.cuhk.edu.hk, The Chinese University of Hong Kong, Guangdong Provincial Key Laboratory of Computer Vision and Virtual Reality Technology, SIAT.

0730-0301/2018/11-ART243 $15.00
https://doi.org/10.1145/3272127.3275082



(a) Input (b) Perceptual

(c) Content-adaptive (d) Ours (network output)

Fig. 2. Sharpness in thin edges. This cartoon character has very thin black edge surrounding his face, hair, and fingers. (b) "Perceptual" is blurry and has artifacts. (c) "Content-adaptive" can give a sharper and clearer result but is still blur thin edges. Our result (d) which is directly generated by our network shows a clean pixel art with sharpest edges even the edges are thin. (Input image ©Nintendo Co., Ltd.)

training supervision. These two tasks are modeled as a cascaded network which consists of three stages for different purposes. *GridNet* transfers the input image into multi-scale grid-structured images with different aliasing effects. *PixelNet* associated with *GridNet* to synthesize pixel arts with sharp edges and perceptually optimal local structures. *DepixelNet* connects the previous network and aims to recover the pixelized result to the original image. For the sake of unsupervised learning, the mirror loss is proposed to hold the reversibility of feature representations in the process. In addition, adversarial, L1, and gradient losses are involved in the network to obtain pixel arts by retaining color correctness and smoothness. We show that our technique can synthesize crisper and perceptually more appropriate pixel arts than state-of-the-art image downscaling methods. We evaluate the proposed method with extensive experiments on many images. The proposed method outperforms state-of-the-art methods in terms of visual quality and user preference.

CCS Concepts: • **Computing methodologies** → **Neural networks**; **Image processing**;

Additional Key Words and Phrases: Pixelization, Generative adversarial network, Image-to-image translation

ACM Reference Format:
Chu Han, Qiang Wen, Shengfeng He, Qianshu Zhu, Yinjie Tan, Guoqiang Han, and Tien-Tsin Wong. 2018. Deep Unsupervised Pixelization. *ACM Trans. Graph.* 37, 6, Article 243 (November 2018), 11 pages. https://doi.org/10.1145/3272127.3275082

1 INTRODUCTION

Pixel arts stem from the limited resolution and limited colors on display of early gaming devices (e.g. Game & Watch) and computer systems [Goldberg and Flegal 1982]. Nowadays, pixel arts has already become an art form. Artists had to carefully and manually design the graphics in a pixel-by-pixel manner. Such process is tedious and time consuming. A pleasant pixel art typically exhibits clear global structure and sufficient local details. This becomes very challenging when the input picture is very busy.

A natural computational method to generate pixel arts is to downsample the input image. Öztireli and Gross [2015] proposed an optimization solution using perceptual image quality metric. It is able to retain the perceptually important features, but it tends to blur the edges (Fig. 2(b)). Moreover, the optimization process is independently performed on individual color channels, color shifting artifacts may be resulted (hairs in Fig. 2(b)). Kopf et al. [2013] proposed a content adaptive method by aligning the kernels with the local image features in a bilateral manner. Although their results is crisper than that from image downscaling, edges may still be blurry (Fig. 2(c)) due to its kernel-based nature. Note that such antialiased edges is not desirable in pixel arts. Besides, handdrawn pixel arts also *summarize* the original visual content by selectively retaining important ones. All these are very challenging in generating pleasant pixel arts.

In this paper, we propose a deep unsupervised learning approach for pixelization. The rationale we do not use supervised learning is because preparing pixelized training data is very costly, as high-quality pixel arts are mostly by hand. Instead, we propose an unsupervised learning utilizing a neural network architecture sharing the similar spirit as cycleGAN [Yi et al. 2017; Zhu et al. 2017]. We train our network in a bi-directional manner. In the forward direction, we pixelize an input image. In the reverse direction, the pixel art is converted back to the original image. These two tasks are modeled as a cascaded network. While training in the forward direction, our network takes a cartoon art C as input. GridNet produces multi-scale grid-structured images, and followed by PixelNet to generate pixel art P. DepixelNet recovers P back to cartoon art C', i.e., $C \rightarrow P \rightarrow C'$. In the backward direction, given a pixel art P, we have $P \rightarrow C \rightarrow P'$, just similar to forward. In order to hold the reversibility of our network, we want $C \approx C'$ and $P \approx P'$. We also propose a mirror loss to minimize the difference between feature maps in pixelization and depixelization (Fig. 3). It ensures the identity of the input image after a training cycle. The direct output of the forward network is an image of the same resolution as the input clip art, but with an appearance of "low-resolution" pixel art. Through a post-processing nearest-neighboring step, the true low-resolution pixel art is obtained. With this network design, the entire network can be trained using the input image itself without the high-quality corresponding pixel art as in supervised learning.

We have evaluated the proposed method on various types of input including cartoon, vector drawing, and even natural photograph, convincing pixel arts are obtained in most cases. In addition, due to our bi-directional training design, a side effect of our method is depixelization. To validate the impact of each subnetwork and loss, we examine different combinations of architectures and losses via

experiments. Our method shows clear preference in pixelization over existing methods.

2 RELATED WORKS

2.1 Image Downscaling

The most straightforward way to generate pixel art is image downscaling, based on the sampling theory [Shannon 1949]. Filters (e.g., bilinear, bicubic, and Lanczos) have been widely used in image downscaling. However, while a filter removes one artifact (e.g., aliasing, ringing or blurring), it may introduce another. All these artifacts lower the visual quality of pixel art. Recently, [Öztireli and Gross 2015] proposed an optimization method to preserve the fine details and the local structures, using a perceptual image quality metric. [Weber et al. 2016] preserve high-frequency details based on convolutional filters. [Kopf et al. 2013] proposed a content adaptive method by aligning the kernels with the local image feature in a bilateral manner. However, these methods tend to eliminate the aliasing appearance in which pixel artists intend to preserve. In addition, due to their kernel-based nature, they can hardly synthesize pixel art with sharp edges.

2.2 Optimization Approach

Rendering the content in a low-resolution image can be regarded as an optimization problem. Dippé et al. [1985] applied antialiasing to reduce artifact. Inglis et al. [Inglis and Kaplan 2012; Inglis et al. 2013] proposed method to rasterize vector line arts and curves with visually pleasing results. The latter method supports "manual antialiasing." To synthesize pixel art animation, [Kuo et al. 2016] first generates an initial animation sequence by applying image warping to a set of selected key frames. They are then jointly optimized to preserve the prominent feature lines of each frame. [Gerstner et al. 2012] proposed an iterative process to generate abstraction of image in low resolution and with limited colors. Their approach updates SLIC superpixels [Achanta et al. 2012] and refine the color alternatively. It cannot synthesize perceptually correct results where many colors gather within a small region. However, all above optimization-based approaches pay more attention to the accuracy than the aesthetic consideration as in our pixel art application, and hence cannot be directly applied.

2.3 Image-to-image Translation

Neural network becomes an effective and efficient approach to image-to-image translation problem. Isola et al.[2017] proposed to use the generative adversarial network [Goodfellow et al. 2014], for translating one image to another domain, e.g., labels to street scene or edge to photo. Several variants of CNN-based models [Chen and Koltun 2017; Isola et al. 2017; Long et al. 2015; Mirza and Osindero 2014; Odena et al. 2016; Xie and Tu 2015] have been proposed recently. Their performances are highly dependent on paired training data. However, paired training data is hard to gather or prepare in our pixel art application, due to the difficulty in creating pixel art.

Two concurrent works [Zhu et al. 2017] and [Yi et al. 2017] proposed to use cycle consistency loss to relieve the requirement of paired data. However, directly applying these models to pixel art may result in uneven pixel grid sizes and may not retain perceptually important structure.

3 APPROACH

The proposed cascaded network architecture is shown in Fig. 3. The bi-directional training process allows us to generate a pixel art from a cartoon art and vice versa. This network has two goals that correspond to the two training directions. We denote them as *Forward* and *Backward* respectively.

Forward: Given an input image c_i, we want to generate its pixel art image p_{c_i}, that can be recovered to an image c_i' such that $c_i \approx c_i'$. The forward training flow is: $GridNet \rightarrow PixelNet \rightarrow DepixelNet$ (Fig. 3).

Backward: This is the reverse of the forward one. Given a pixel art p_j, we want to depixelize it to c_{p_j}, that can be further recovered to a pixel art p_j' such that $p_j \approx p_j'$. The backward training flow is: $DepixelNet \rightarrow GridNet \rightarrow PixelNet$ (Fig. 3).

3.1 Training Data

We denote our input training dataset as $\mathbf{S} = \{\mathbf{P}, \mathbf{C}\}$, where $\mathbf{P} = \{p_n, n = 1, ..., N\}$ indicates the pixel art images and $\mathbf{C} = \{c_m, m = 1, ..., M\}$ indicates the cartoon arts. We collect all our training data from the internet. Our training data are all unpaired since manually creating pixel art from its corresponding cartoon is tremendously tedious and time consuming. In our training dataset, 900 cartoon art images and 900 pixel art images are collected. Fig 4(a) & (b) show some of our training pixel arts and training cartoon arts, respectively. During each training iteration, we randomly select one cartoon art image $c_i \in \mathbf{C}$ and one pixel art image $p_j \in \mathbf{P}$. c_i and p_j are regarded as the reference image of each other.

3.2 Network Architecture

We design a cascaded network which consists of three subnetworks, GridNet (*GN*), PixelNet (*PN*) and DepixelNet (*DN*). Each serves for different purposes. GridNet associated with PixelNet can be regarded as the generator of pixel art. GridNet itself is designed to extract features and generates initial pixel art images with multi-scale grid structures. PixelNet aims at producing crispy edges while keeping perceptually important regions. DepixelNet depixelizes the pixel art to generate cartoon art.

3.2.1 GridNet and PixelNet. GridNet is modeled by a sequence of blocks consisting of three types of layers, including convolution layer, instance normalization layer and activation layer. In the middle of this subnetwork, we insert several residual blocks [He et al. 2016] to reduce artifacts and accelerate the learning process.

To generate different aliasing effects, we concatenate a 7×7 Conv-InstanceNorm-ReLU layer to obtain the initial pixel art results for the last three conv-blocks. Reflection padding is utilized to eliminate the artifacts at the border of image. Three output images corresponding to $\frac{1}{4}$, $\frac{3}{16}$ and $\frac{1}{8}$ of the input resolution are obtained. Training the network with multiple scales not only generates results with different aliasing effects, but also improves the generalization and representability of our network. Because the features from different layers represent details and structure information under different scales, training the network in multi-scale can be regarded as a

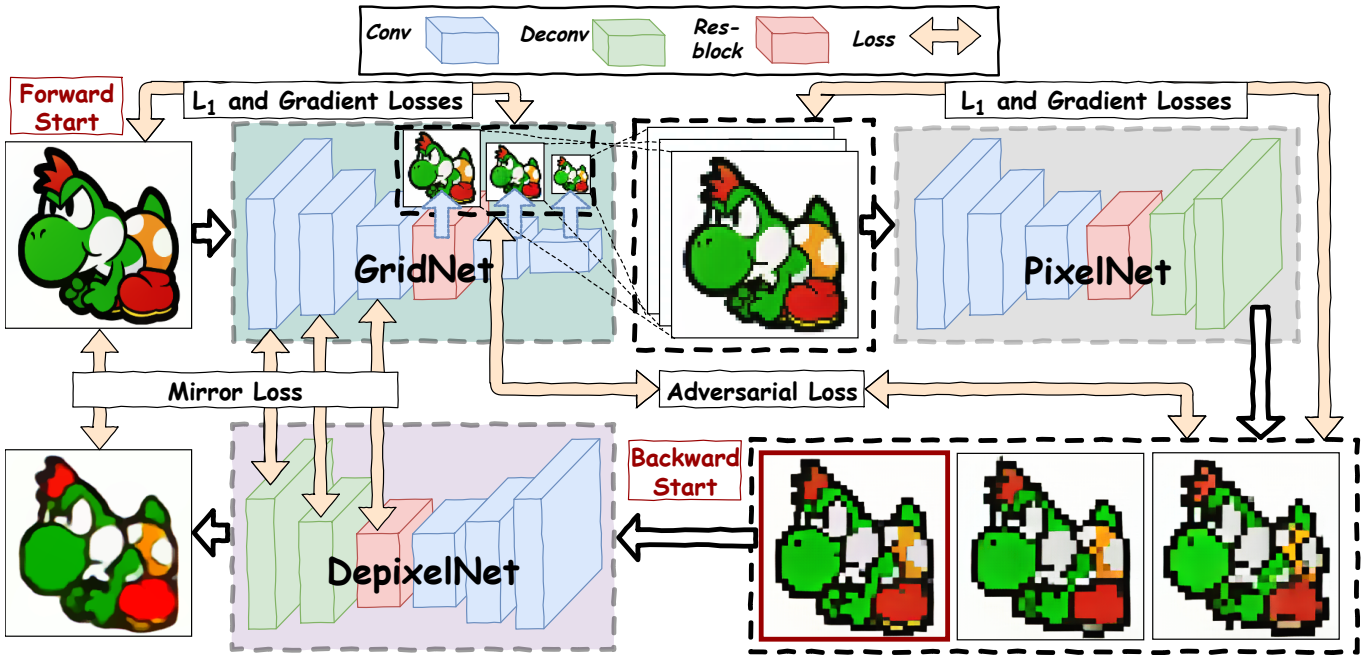
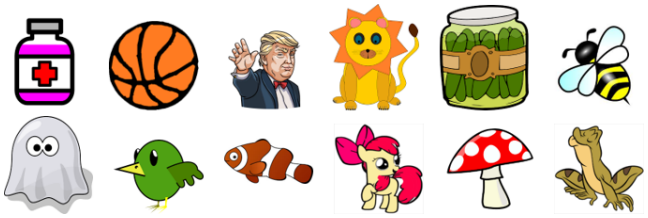


Fig. 3. Network architecture for unsupervised pixelization. It consists of three subnetworks, GridNet, PixelNet and DepixelNet. *GridNet* is designed for initializing the cartoon art to grid-structured image with different aliasing effects. *PixelNet* further processes these images with refined and abstracted image content. *DepixelNet* is used for recovering cartoon art from pixel art. *Forward* path starts from GridNet and ends at DepixelNet. It generates pixel art from cartoon art then recovers back to pixel art. On the contrary, the goal of *backward* path is the reverse of forward path. The mirror loss in both directions is designed to keep the regularity of image. (©Nintendo Co., Ltd.)

(a) Pixel arts



(b) Cartoon arts

Fig. 4. Samples of our training data.

coarse-to-fine training process. We evaluate the performance with and without multi-scale training in Section 4.3.

GridNet acts as a downscaling component and its results contain weak and blurry edges. As a consequence, we design PixelNet to produce crispy edges and retain perceptually important content. It is a fully convolutional network [Long et al. 2015] which consists of three types of blocks, including conv-block, res-block and deconv-block. Similarly, each conv-block contains three different types of layers, including convolution layer, instance normalization layer and activation layer. After three consecutive conv-blocks, we concatenate 9 res-blocks and 2 deconv-blocks to PixelNet. Deconv-block consists of three types of layers, deconvolution layer, instance normalization layer and activation layer.

Fig. 5 illustrates the process and effectiveness of our two-combo pixel art generator. Given a cartoon art image, GridNet generates three scales output images. Each corresponds to a different resolution (grid size), and hence different aliasing effect. But the small resolution restricts the generation of pixel art. Hence, we upsample their outputs to the resolution of the input image using nearest neighbor. As shown in Fig. 5, the generated results by GridNet contain different scales of details, but the edges are weak and blurry. PixelNet takes the upsampled output from GridNet as input, and generates refined and abstracted pixel art.

3.2.2 DepixelNet. The network structure of DepixelNet is exactly the same as PixelNet. It can be regarded as the generator of cartoon art. Given three scales of pixel art images, DepixelNet randomly selects one and generates its cartoon output.

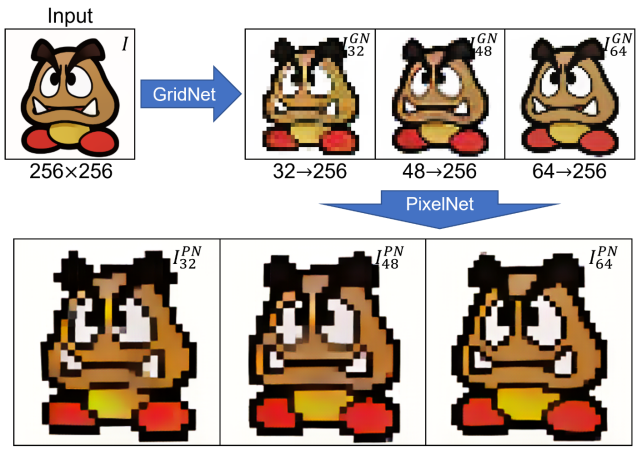


Fig. 5. Our two-combo pixel art generator is able to generate crisp and abstracted pixel art. (Input image ©Nintendo Co., Ltd.)

3.3 Objective Function

During the training process, we introduce several losses to constrain the network. Since each subnetwork has its own goal, each subnetwork has a different objective function.

GridNet takes a cartoon art as input then outputs an initial pixel art. Its objective function is:

$$\begin{aligned}\mathcal{L}_{GN} =& \mathcal{L}_{\text{GAN}}(GN, \mathcal{D}_{GN}, F) + \mathcal{L}_{L1\&\text{grad}}(GN, F) + \\ & \mathcal{L}_{L1\&\text{grad}}(GN, B),\end{aligned} \tag{1}$$

where F and B denote the training directions of forward and backward, respectively. As in the forward direction, ground truth pixel art is not available, we apply adversarial loss [Goodfellow et al. 2014] to discriminate the distribution ($C \rightarrow P$). $\mathcal{D}_{GN}$ is the discriminator in GridNet. For the same reason, we use the L_1 loss to maintain the correctness of color, and the gradient loss is introduced to guarantee the smoothness and sharp edge of image. In the backward direction, the L_1 and gradient losses are compared to the input pixel art.

As we mentioned in Section 3.2.1, the upsampled output of GridNet is fed to PixelNet, so as to get a refined pixel art. Hence, the complete objective of PixelNet is defined as follow:

$$\begin{aligned}\mathcal{L}_{PN} =& \mathcal{L}_{\text{GAN}}(PN, \mathcal{D}_{PN}, F) + \mathcal{L}_{L1\&\text{grad}}(PN, F) + \\ & \mathcal{L}_{\text{mirr}}(DN \rightarrow PN, B),\end{aligned} \tag{2}$$

where $\mathcal{D}_{PN}$ is the discriminator in PixelNet. L_1 and gradient losses are designed for the color and edge correctness. Mirror loss in the backward direction is introduced to ensure the reversibility of the whole network. The mirror loss is detailed in the following section.

DepixelNet takes a pixel art as input, then output a cartoon art. Thus, the objective contains an adversarial, L_1, and gradient losses in the backward direction. Mirror loss is used in the forward direction to regularize the network:

$$\begin{aligned}\mathcal{L}_{DN} =& \mathcal{L}_{\text{GAN}}(DN, \mathcal{D}_{DN}, B) + \mathcal{L}_{L1\&\text{grad}}(DN, B) + \\ & \mathcal{L}_{\text{mirr}}(GN \rightarrow DN, F).\end{aligned} \tag{3}$$

3.3.1 Mirror Loss. Due to the unsupervised training nature, we propose mirror losses in both directions to ensure the reversibility of our bi-directional training process. In the forward direction, we take the cartoon art c_i as input and generate its pixel art p_{c_i} then convert it back to the cartoon art c_i'. Mirror loss here is to guarantee $c_i \approx c_i'$ and the feature maps in corresponding layers to be as similar as possible. Following is the definition of mirror loss in the forward direction:

$$\mathcal{L}_{\text{mirr}}(GN \rightarrow DN, F) = \sum^{l} \omega_l E^{l}_{\text{mirr}}(F) + \omega_{\text{out}} E^{\text{out}}_{\text{mirr}}(F), \tag{4}$$

where F denotes the forward direction. $E^{\cdot}_{\text{mirr}}$ is the mirror loss between feature maps (and output image). E^{l}_{mirr} is defined as:

$$E^{l}_{\text{mirr}}(F) = \sum \left\| f^{l}_{GN} - f^{N-l}_{DN} \right\|_1, \tag{5}$$

where N indicates the total number of blocks in each subnetwork (all of them have the same number of blocks). f denotes the feature maps in the final layer of a block. We calculate the L_1 loss of feature maps between block l in GridNet and block $N-l$ in DepixelNet. We want the feature maps in corresponding blocks to be as similar as possible, just like a mirror. In practice, we let $l \in \{1, 2, 3\}$.

Mirror loss $E^{\text{out}}_{\text{mirr}}$ between input and output is defined as:

$$E^{\text{out}}_{\text{mirr}}(F) = \sum \left\| c_i - c_i' \right\|_1, \tag{6}$$

where c_i is the input cartoon and c_i' is the output cartoon generated by DepixelNet.

Mirror loss in the backward direction is similar to that in the forward direction. The only difference is that we calculate the mirror loss in backward between DepixelNet and PixelNet (the first and the last subnetwork respectively). It is defined as:

$$\mathcal{L}_{\text{mirr}}(DN \rightarrow PN, B) = \sum^{l} \omega_l E^{l}_{\text{mirr}}(B) + \omega_{\text{out}} E^{\text{out}}_{\text{mirr}}(B). \tag{7}$$

We define the backward mirror loss for block l as:

$$E^{l}_{\text{mirr}}(B) = \sum \left\| f^{l}_{DN} - f^{N-l}_{PN} \right\|_1. \tag{8}$$

Backward mirror loss between the input and output is defined as:

$$E^{\text{out}}_{\text{mirr}}(B) = \sum \left\| p_j - p_j' \right\|_1. \tag{9}$$

In Section 4, we conduct an experiment to evaluate the performance of our network with and without the mirror loss.

3.3.2 Adversarial Loss. We introduce an adversarial loss [Goodfellow et al. 2014] for each subnetwork. The loss in GridNet is defined as:

$$\begin{aligned}\mathcal{L}_{\text{GAN}}(GN, \mathcal{D}_{GN}, F) =& \mathbb{E}_{p \sim p_{\text{data}(p)}}[\log(\mathcal{D}_{GN}(p))] + \\ & \mathbb{E}_{c \sim p_{\text{data}(c)}}[\log(1 - \mathcal{D}_{GN}(GN(c)))],\end{aligned} \tag{10}$$

where $\mathcal{D}_{GN}$ is the discriminator. The Discriminator's objective is to maximize Eq. (10) while GN is to minimize it. Then we have $\min_{GN} \max_{\mathcal{D}_{GN}} \mathcal{L}_{\text{GAN}}(GN, \mathcal{D}_{GN}, F)$. Same as GridNet, PixelNet and DepixelNet also have the adversarial losses $\mathcal{L}_{\text{GAN}}(PN, \mathcal{D}_{PN}, F)$ and $\mathcal{L}_{\text{GAN}}(DN, \mathcal{D}_{DN}, B)$.

3.3.3 L1 and Gradient Losses. Adversarial loss has been demonstrated to be effective in many image-to-image translation problems. However, our training data are unpaired. We cannot guarantee the color correctness along the whole process by only applying adversarial loss. For each subnetwork, we apply L_1 loss to ensure the color consistency and gradient loss to ensure image smoothness and sharpness of edges. The L_1 and gradient loss for GridNet, PixelNet and DepixelNet are defined in Eq. (11), Eq. (12) and Eq. (13) respectively,

$$\mathcal{L}_{L1\&\text{grad}}(GN, F) = \sum^{r} (E^r_{L1}(GN, F) + E^r_{\text{grad}}(GN, F)), \tag{11}$$

$$\mathcal{L}_{L1\&\text{grad}}(PN, F) = \sum^{r} (E^r_{L1}(PN, F) + E^r_{\text{grad}}(PN, F)), \tag{12}$$

$$\mathcal{L}_{L1\&\text{grad}}(DN, B) = \sum^{r} (E^r_{L1}(DN, B) + E^r_{\text{grad}}(DN, B)), \tag{13}$$

where r indicates a specific aliasing scale, as our method is able to synthesize pixel arts with different aliasing effects. We here sum up L_1 and gradient losses under all aliasing scales.

We define L_1 loss in the forward direction at a specific aliasing scale r by:

$$E^r_{L1}(GN, F) = \sum \left\| I^r_{\text{in}} - I^r_{\text{out}} \right\|_1. \tag{14}$$

We resize $I_{\text{in}} \rightarrow I^r_{\text{in}}$ such that the resolutions of I^r_{in} and I^r_{out} are identical. We define gradient loss in the forward direction at resolution r by:

$$\begin{aligned} &E^r_{\text{grad}}(GN, F) = \\ &\sum \Big[\Big\| |I^r_{\text{in}}(x, y) - I^r_{\text{in}}(x-1, y)| - |I^r_{\text{out}}(x, y) - I^r_{\text{out}}(x-1, y)| \Big\|_1 + \\ &\Big\| |I^r_{\text{in}}(x, y) - I^r_{\text{in}}(x, y-1)| - |I^r_{\text{out}}(x, y) - I^r_{\text{out}}(x, y-1)| \Big\|_1 \Big]. \end{aligned} \tag{15}$$

L_1 and gradient losses in other subnetworks are similar to Eq. (14) and Eq. (15).

3.4 Training Details

Both training pixel art and cartoon data contain 900 images. We resize all images to 256×256 during the training process. Note that our model can handle input images with any resolution. When we train this network in the forward direction, we choose one of the three refined pixel art results randomly as its input, in order to increase the generalization of DepixelNet. Following the training strategy of [Shrivastava et al. 2017], discriminators are updated using the history of generated images instead of the latest one. Thus, we create a buffer to store the last 30 generated images. For all the experiments, we use the Adam solver [Kingma and Ba 2014] with the batch size of 1. All networks are trained with a learning rate of 0.0002 initially. After 100 epochs, we linearly decay the rate to zero in 150 epochs.

Mirror loss in the backward direction is different from that in the forward one. Since PixelNet outputs three results, there are three mirror losses between input and output in Eq. (7). We choose the minimal one as our objective in order to pass the most correct one to our network, as the aliasing scale is unknown for the input pixel art.

4 RESULTS AND DISCUSSION

We have tested our method over images collected from the internet, and test images from [Kopf et al. 2013]. They are excluded in the training set. All our results are generated using the $(N-2)$-th conv-block in the GridNet. This effectively means our network output always have a pixelized appearance in the scale of approximately 1/6 of the input image resolution.

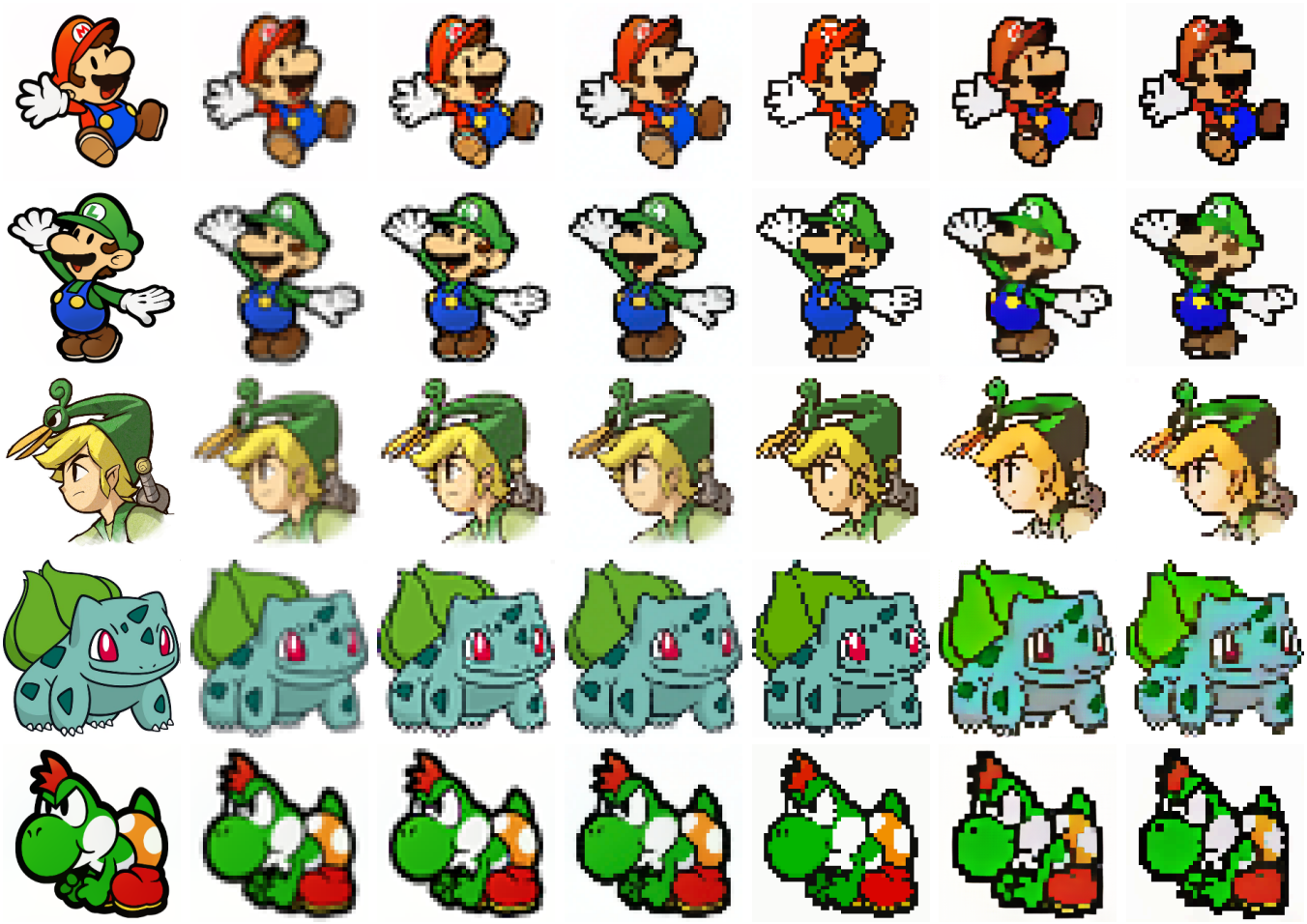
4.1 Comparison to Existing Methods

We compare our pixel art generation results to four image downscaling techniques, including bicubic, "content-adaptive" [Kopf et al. 2013], "perceptual" [Öztireli and Gross 2015] and an image abstraction method [Gerstner et al. 2012]. Their results are generated using the implementation provided by their original authors.

Note that our network output has the same resolution as the input, while our competitors are in low resolution. To have a fair comparison, we downsample our network output to the same low-resolution image by voting the major color within the high-resolution pixels that corresponding to the low-resolution pixel. In the rest of the paper, we denote our results directly from the network as "network output," while the downsampled one as "voting." Due to the page limit, we cannot show both "network output" and "voting" in all examples. But they are all shown in the supplementary material.

Fig. 1, 2 & 6 visually compare the results. Edges in results of "perceptual" and "content-adaptive" tend to be more blurry due to their kernel-based nature. The minority colors, such as edges, tend to be dominated by the surrounding majority colors. For example, the delicate eye in the blue box of Fig. 2(a). In contrast, our result (Fig. 2(d)) preserves the crispy edges. In Fig. 6, the bicubic results are the most blurry ones. The "perceptual" results in Fig. 6(c) exhibit the color shifting artifacts at regions crowded with multiple colors. "Content-adaptive" results (Fig. 6(d)) produce sharper edge than that of bicubic (Fig. 6(b)) and "perceptual" (Fig. 6(c)). However, weak edges are still observable, and some visually important details are lost. Our competitors generate fewer jaggy edges and aliasing appearance that are desirable in pixel art. Fig. 6(e) shows results generated by [Gerstner et al. 2012]. Their edges (Fig. 6(e)) are crisper than other three competitors. However, there can be discontinuous edges. In the last two columns in Fig. 6, we show our results "network output" and "voting." The "network output" is always the output from the PixelNet fed with the output from the $(N-2)$-th conv-block of GridNet. Our results show the sharpest edges while preserving the local details and avoiding discontinuous edges. Note that even the edge occupies a small portion of the final pixel, our model can somehow "amplify" the edges in final pixel art. In the red box of Fig. 1, our method successfully preserves the details on the eyes of the girl while other methods cannot achieve that. Our method also produces clearer edges than others in the blue box of Fig. 1.

Fig. 7 shows results on real photographs. The resolution of the input image ranges from 800×800 to 256×256. Note that real photographs usually have smooth color transition. To create more pixel-art appearance, we color-quantize our results as well as our competitors' results to 16 colors using a k-means color reduction. Even with such color quantization, results from our competitors remain



(a) Input (b) Bicubic (c) Perceptual (d) Content-adaptive (e) Image abstraction (f) Ours (network output) (g) Ours (voting)

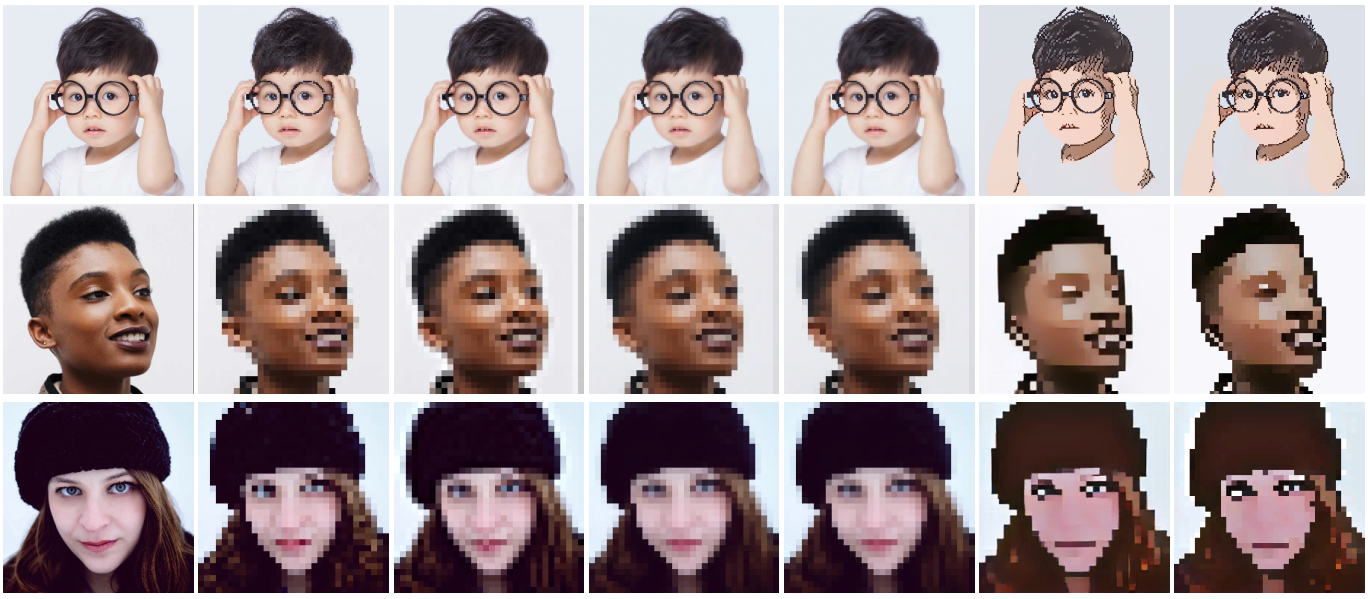
Fig. 6. Comparison of pixel art results. The resolution of downscaled images in (b), (c), (d), (e) & (g) are 32×32. (Input image ©Nintendo Co., Ltd.)

quite smooth and looks too "photorealistic." This is due to the fact that image downscaling methods intend to suppress the aliasing. For our competitors' results on the upper two rows, even with the scaling factor of 1/8, they do not exhibit pixelized appearance due to the high resolution of the input. On the contrary, our method (Fig 7(f)) preserves the color-quantized appearance desired in pixel art application. Thanks to the multi-scale training strategy and fully convolutional architecture, our method can preserve more details in high-resolution images while low resolution results are more abstract. Our "voting" results are visually similar to our "network output" counterparts.

The factor of downscaling strongly influences the quality of the pixel arts for downscaling approaches. However, choosing an appropriate downscaling factor is image-dependent. Fig. 8 compares our method to "content-adaptive" [Kopf et al. 2013] and image abstraction [Gerstner et al. 2012] in two scales, 1/6 and 1/8. For the upper example, a larger downscaling rate of 1/8 (Fig. 8(c) & (e)) produce visually better pixel arts than the smaller 1/6 (Fig. 8(b) & (d)) for both of our competitors. On the other hand, for the lower example, the smaller scaling factor of 1/6 performs better. This is because different images contain different levels of details. Selecting an appropriate downscaling factor is hence an art. In contrast, our network does not need to specify the scaling factor and produces visually pleasant pixel art (Fig. 8(f)). To have a fair comparison, the last two column shows our "voting" results rendered with scaling factor of 1/6 and 1/8, which are similar to our "network output."

4.2 Comparison to Alternative CNN Models

We also compare our method to CycleGAN [Zhu et al. 2017], and two variants of our CNN models including, "GridNet alone," and "PixelNet alone." CycleGAN is also an unsupervised learning method with a cycle consistency loss. It can be directly applied to our pixelization application. The "GridNet alone" model is actually GridNet+DepixelNet with the PixelNet being dropped. The last model "PixelNet Alone" is PixelNet + DepixelNet with the GridNet being dropped. These two variants are used to evaluate the importance



(a) Input (b) Subsampling (c) Perceptual (d) Content-adaptive (e) Image abstraction (f) Ours (network output) (g) Ours (voting)

Fig. 7. Pixel art of real photographs. Results in (f) are our "network output", and results in (g) are our "voting" based on (f). The results of other methods from the same row are all downscaled by the same scaling factor. The scaling factors from top to bottom are 1/8, 1/8, 1/6, and 1/6. Our results are always 1/6. Photos from public domain.

Table 1. Different loss combinations.

Loss1:	$\mathcal{L}_{L1} + \mathcal{L}_{\text{mirr}} + \mathcal{L}_{\text{GAN}}$
Loss2:	$\mathcal{L}_{L1} + \mathcal{L}_{\text{grad}} + \mathcal{L}_{\text{GAN}}$
Loss3:	$\mathcal{L}_{L1} + \mathcal{L}_{\text{grad}} + \mathcal{L}_{\text{mirr}}$
Loss4:	$\mathcal{L}_{L1} + \mathcal{L}_{\text{grad}} + \mathcal{L}_{\text{mirr}} + \mathcal{L}_{\text{GAN}}$ (all w/o multi-scale)
Loss5:	$\mathcal{L}_{L1} + \mathcal{L}_{\text{grad}} + \mathcal{L}_{\text{mirr}} + \mathcal{L}_{\text{GAN}}$ (all w/ multi-scale)

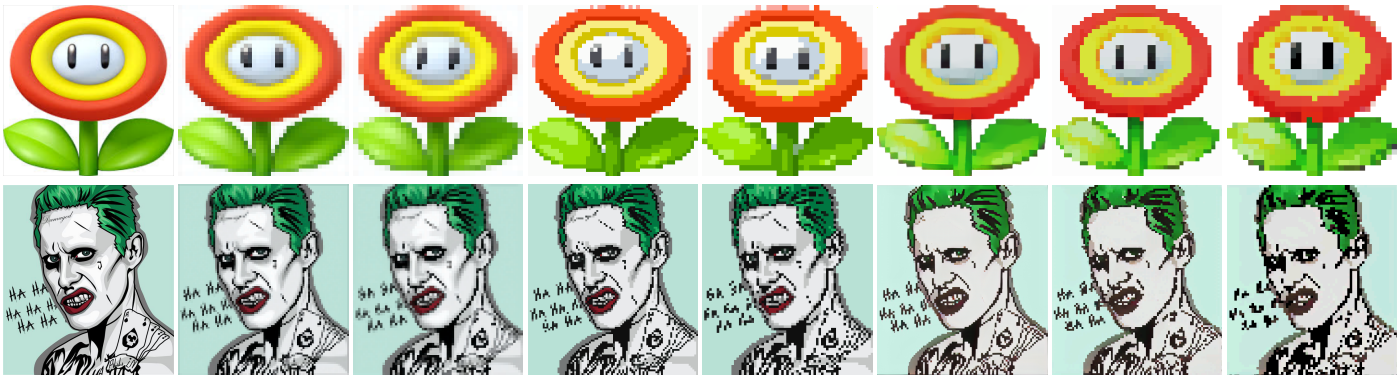
of the components in our model. All three alternatives are trained with the same training data.

Fig. 9 compares the proposed one to the three alternative models. Although CycleGAN tries to transfer the input image to pixel art style, its results contain a lot of artifacts which come from the gridlines in pixel art training data. For "GridNet alone" model, the results are too realistic images and do not look like pixel arts. This is because GridNet serves as a downscaling component while the PixelNet serves to simulate the pixelized appearance. Dropping PixelNet reduces the pixelized appearance. For the results of "PixelNet alone" model, PixelNet does generate pixelized appearance, but fails to preserve the fine edges and present color-quantized appearance. This is because the network is not constrained to generate grid-structured colors and edges, which is the main objective of GridNet. Last column (Fig. 9(e)) shows results generated from our complete network. Our results are visually more pleasant as it presents the least artifacts and the clearest structure among all methods

4.3 Impact of Losses

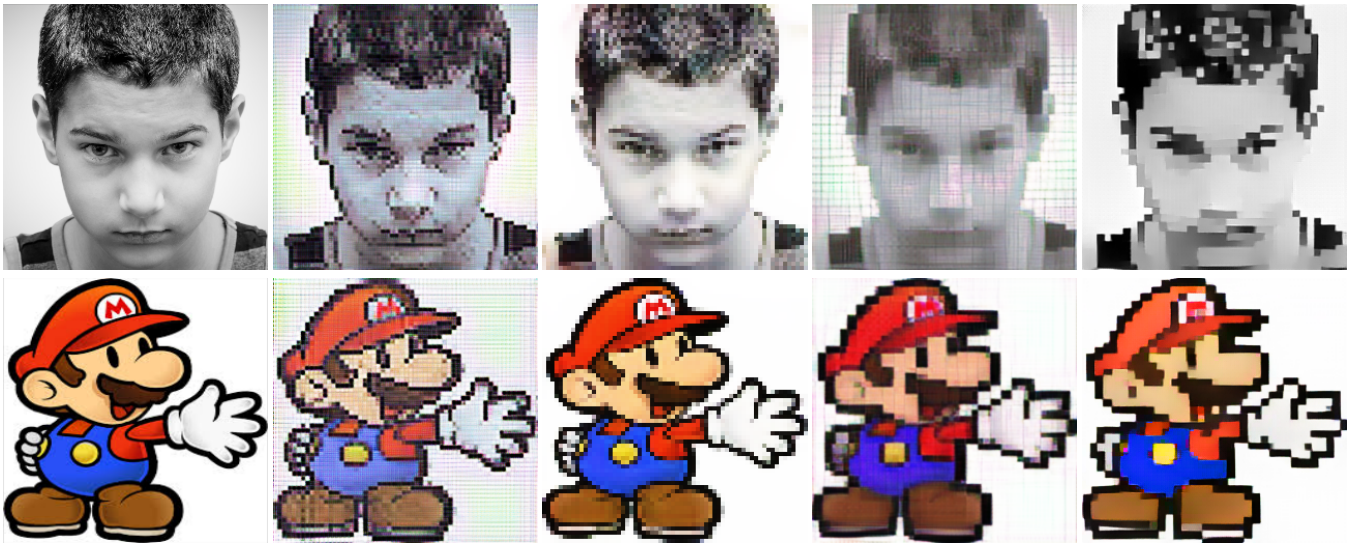
We conduct an experiment to evaluate the impact of each loss, especially adversarial and mirror losses. Different loss combinations are summarized in Table 1. Same training configuration is applied for each loss combination.

Fig. 10 shows the results of our network training with different loss configurations. Without the gradient loss (Fig 10(b)), the colors and edges of our results deviate from the input image. Moreover, the results are not smooth especially at the edges which are shown in box A. Fig 10(c) shows the result of Loss2 (the mirror loss is dropped). The original colors are no longer retained in pixel arts (see the color of Mario dress in Box B). In addition, without the mirror loss, we cannot hold the reversibility of the whole network since our training data are unpaired. That makes some uncontrollable artifacts such as in box C. Fig 10(d) shows results of the third configuration, we remove the GAN loss while retaining all the other losses. We can see that the results are no longer pixelized and more similar to the original images. This is because without the adversarial loss, the model fails to reject results that do not look like pixel art. As we mentioned in Section. 3.2.1, our network generates multi-scale results to the improve the generalization and representation abilities of our model. In Loss4, we let GridNet outputs only one result instead of three, while the whole network uses all losses. Without the multi-scale training, we lose much more details of the image such as the right face of the man in box D. This demonstrates the effectiveness the multi-scale training on detail preserving and representation ability. Boxes E and F in Fig 10(f) show the results with all losses as well as the multi-scale training. These results preserve perceptually



(a) Input (b) Content-adaptive (1/6) (c) Content-adaptive (1/8) (d) Image abstraction (1/6) (e) Image abstraction (1/8) (f) Ours (network output) (g) Ours (1/6) (voting) (h) Ours (1/8) (voting)

Fig. 8. Comparisons to other methods under two scaling factors, 1/6 and 1/8. (©Nintendo Co., Ltd., and ©taskyamaura/DeviantArt.)



(a) Input (b) CycleGAN (c) "GridNet alone" (d) "PixelNet alone" (e) Complete model

Fig. 9. Comparison to alternative CNN models. All results are direct network output. (Mario ©Nintendo Co., Ltd., photo from the public domain.)

important local details without hurting the global structure. In the same time, amplify the edges and retain some interesting effects such as the checker board in box F.

4.4 Comparisons with Manual Pixel Arts

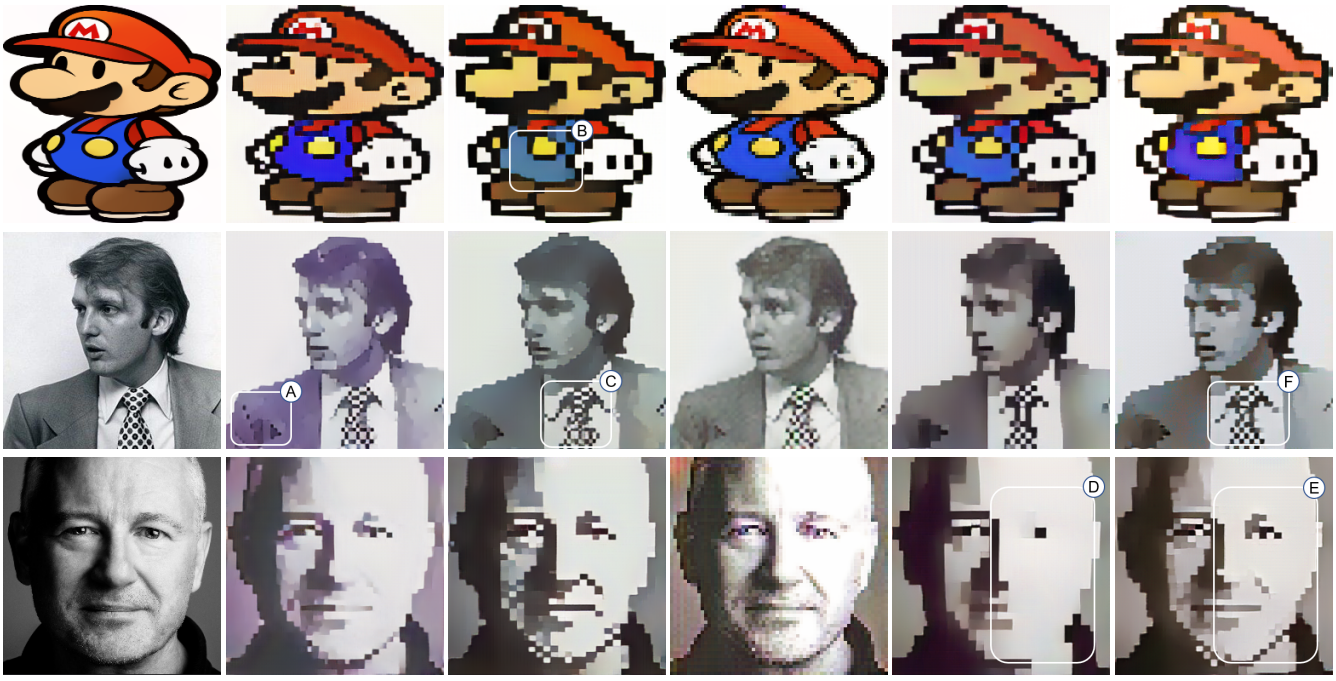
We also conduct an experiment (Fig. 11) to compare our results to manual pixel arts drawn by artists. Two manual pixel arts are drawn (Fig 11(c)). We can see that artists may modify the overall structure and even the color to suit their own style. They also pay less attention to the conformity to the original input. In contrast, our results strongly conform to the input (Fig 11(b)). Our results can retain the perceptually important elements such as eyes and the pixelized appearance. On the other hand, our model can create new content or style, as artists do.

4.5 Depixelization

Due to our bi-directional training design, a side effect of our method is depixelization. With the well trained network, we can feed a pixel art (Fig. 12(a)) to the DepixelNet, to reconstruct a smooth cartoon image (Fig. 12(c)). Fig 12(b)&(c) compare [Kopf and Lischinski 2011] results and ours. Our results look rougher than that of [Kopf and Lischinski 2011] in general. Note that our output is raster image while that of [Kopf and Lischinski 2011] is vector output. Generating vector graphics is beyond the scope of this paper. Nevertheless, our network can still generate visually pleasant results with clear edges and a smooth gradient.

4.6 Performance

We implemented our network using the deep learning framework PyTorch on Ubuntu 16.04. The whole training process took around



(a) Input (b) Loss1 (w/o Gradient) (c) Loss2 (w/o Mirror) (d) Loss3 (w/o GAN) (e) Loss4 (w/o Multi-scale) (f) Loss5 (all)

Fig. 10. Impact of Losses. From left to right, we show the input images and the synthesized images with various loss configurations in Table 1. (Mario ©Nintendo Co., Ltd., photos from the public domain.)

twelve hours on a single NVIDIA GeForce GTX 1080 Ti with an Intel Core i7-8700 CPU at 3.20GHz. During the testing phase, it takes around 0.6s to generate one pixel art on average.

4.7 Limitations

Unlike the image downscaling methods, our method is not able to generate pixel arts with arbitrary resolution, as our pixelized appearance is always 1/6 of the resolution of the input. Nevertheless, a workaround is to first downsample the input image to 6x of target pixel art resolution, before feeding to our network, as our network can take arbitrary size image as input. However, such preprocessing downsample scheme may remove details and/or even introduce unwanted artifacts.

Just like other GAN-based methods, adversarial losses may introduce unpredictable artifacts (Fig. 13(d)). It cannot guarantee that the colors of output are exactly the same as the original image (Fig. 13(b)).

5 CONCLUSION

In this paper, we propose a cascaded network for unsupervised pixelization. Its unsupervised nature relieves the requirement for preparing paired pixel art training data. Paired pixel art data is practically infeasible as artists may sometimes not follow the input as demonstrated in Fig 11. Three subnetworks are specially designed to serve for different purposes. Mirror losses are proposed to hold the reversibility of our network. Extensive experiments show that our method can generate visually pleasant pixel arts and outperform the existing image downscaling methods. The flexibility and simplicity of our network design allow us to generate pixel art efficiently. In the future, a natural extension is to create a temporal-consistent pixel art sequence for a video input.

ACKNOWLEDGMENTS

We thank all the reviewers for their helpful suggestions and constructive comments. In addition, we thank [Kopf et al. 2013] for providing us the test images in their user study and supplementary materials.

This project is supported by the National Natural Science Foundation of China (No. 61472145 and No. 61702194), Shenzhen Science and Technology Program (No.JCYJ20160429190300857), RGC General Research Fund (No. CUHK14217516 and No. CUHK14200915), the Special Fund of Science and Technology Research and Development on Application From Guangdong Province (SF-STRDA-GD) (No. 2016B010127003), the Guangzhou Key Industrial Technology Research fund (No. 201802010036), and the Guangdong Natural Science Foundation (No. 2017A030312008).

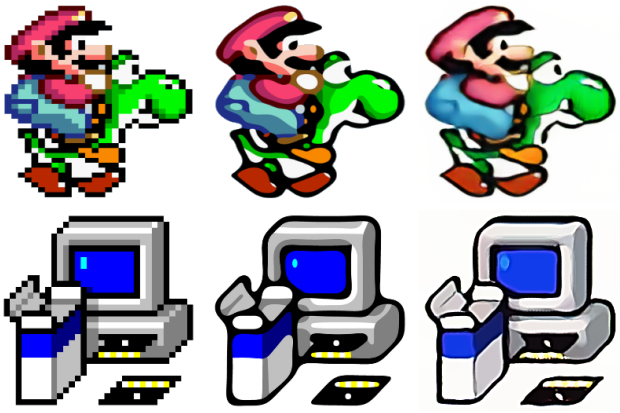
REFERENCES

Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. 2012. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on pattern analysis and machine intelligence* 34, 11 (2012), 2274–2282.

Qifeng Chen and Vladlen Koltun. 2017. Photographic image synthesis with cascaded refinement networks. In *The IEEE International Conference on Computer Vision*

(a) Original (b) Ours (Network output) (c) Manual pixel arts

Fig. 11. Comparison with real pixel arts drawn by artists. (Pixel arts in (c) are drawn by ©Vixels, photos from the public domain.)



(a) (b) (c)

Fig. 12. Depixelization results. (a) Input image. (b) [Kopf and Lischinski 2011] (c) Our results. (©Nintendo Co., Ltd.)

(a) Input (b) Ours (Network output) (c) Input (d) Ours (Network output)

Fig. 13. Limitations of our method. (b) shows the colors change in our result. (d) shows artifacts in our result. (Input image (c) © Nintendo Co., Ltd.)

(ICCV), Vol. 1.
Mark AZ Dippé and Erling Henry Wold. 1985. Antialiasing through stochastic sampling. *ACM Siggraph Computer Graphics* 19, 3 (1985), 69–78.
Timothy Gerstner, Doug DeCarlo, Marc Alexa, Adam Finkelstein, Yotam Gingold, and Andrew Nealen. 2012. Pixelated image abstraction. In *Proceedings of the Symposium on Non-Photorealistic Animation and Rendering*. Eurographics Association, 29–36.
Adele Goldberg and Robert Flegal. 1982. ACM president's letter: Pixel Art. *Commun. ACM* 25, 12 (1982), 861–862.
Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. 2672–2680.
Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.
Tiffany C Inglis and Craig S Kaplan. 2012. Pixelating vector line art. In *Proceedings of the Symposium on Non-Photorealistic Animation and Rendering*. Eurographics Association, 21–28.
Tiffany C Inglis, Daniel Vogel, and Craig S Kaplan. 2013. Rasterizing and antialiasing vector line art in the pixel art style. In *proceedings of the symposium on non-photorealistic animation and rendering*. ACM, 25–32.
Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. 2017. Image-to-image translation with conditional adversarial networks. *arXiv preprint* (2017).
Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
Johannes Kopf and Dani Lischinski. 2011. Depixelizing pixel art. In *ACM Transactions on graphics (TOG)*, Vol. 30. ACM, 99.
Johannes Kopf, Ariel Shamir, and Pieter Peers. 2013. Content-adaptive image downscaling. *ACM Transactions on Graphics (TOG)* 32, 6 (2013), 173.
Ming-Hsun Kuo, Yong-Liang Yang, and Hung-Kuo Chu. 2016. Feature-Aware Pixel Art Animation. In *Computer Graphics Forum*, Vol. 35. Wiley Online Library, 411–420.
Jonathan Long, Evan Shelhamer, and Trevor Darrell. 2015. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 3431–3440.
Mehdi Mirza and Simon Osindero. 2014. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784* (2014).
Augustus Odena, Christopher Olah, and Jonathon Shlens. 2016. Conditional image synthesis with auxiliary classifier gans. *arXiv preprint arXiv:1610.09585* (2016).
A Cengiz Öztireli and Markus Gross. 2015. Perceptually based downscaling of images. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 77.
Claude Elwood Shannon. 1949. Communication in the presence of noise. *Proceedings of the IRE* 37, 1 (1949), 10–21.
Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Josh Susskind, Wenda Wang, and Russ Webb. 2017. Learning from simulated and unsupervised images through adversarial training. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 3. 6.
Nicolas Weber, Michael Waechter, Sandra C Amend, Stefan Guthe, and Michael Goesele. 2016. Rapid, detail-preserving image downscaling. *ACM Transactions on Graphics (TOG)* 35, 6 (2016), 205.
Saining Xie and Zhuowen Tu. 2015. Holistically-nested edge detection. In *Proceedings of the IEEE international conference on computer vision*. 1395–1403.
Zili Yi, Hao Zhang, Ping Tan, and Minglun Gong. 2017. Dualgan: Unsupervised dual learning for image-to-image translation. *arXiv preprint* (2017).
Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint arXiv:1703.10593* (2017).