Texture-aware Multi-resolution Image Inpainting

Mohamed Abbas Hedjazi, Yakup Genc

Abstract—Recent GAN-based inpainting methods have shown remarkable performance using multi-stage networks and/or contextual attention modules (CAM). However, these models require heavy computational resources and may fail to restore realistic texture details. This is mainly due to their training approaches and loss functions. Furthermore, GANs are hard to train on high-resolution images leading to unstable models and poor performance. Inspired by these observations, we propose a novel multi-resolution generators architecture allowing stable training and increased performance. Specifically, our training schema optimizes the parameters of four successive generators such that higher resolution generators exploit the inpainted images produced by lower resolution generators. To restore fine-grained textures, we present a new LBP-based loss function that minimizes the difference between the generated and ground truth textures. We conduct our experiments on Places2 and CelebHQ datasets, and we report qualitative and quantitative results against the state-of-the-art methods. Results show that the computationally efficient model achieves competitive performance.

Index Terms—Image inpainting, deep learning, generative adversarial networks.

I. INTRODUCTION

IMAGE inpainting has attracted significant interest from the computer vision and graphics communities. It synthesizes plausible contents to fill in the missing regions or to remove unwanted objects from an image. It can be utilized in a wide range of applications, including image editing [1], image compression [2], image restoration [3], diminished reality [4] and augmented reality [5].

Infilling is a fundamental part of human vision. Vertebrate eyes do not cover the whole visual field due to a blind spot where optic nerves leave the eye. This spot does not contain any photo-receptor cells and does not contribute to the information flow of the scene. However, our brains use the information from the peripheral area, such as texture, geometry and semantics to fill the gap [6].

Prior approaches in computer vision solve the inpainting problem by extracting low-level features, matching and pasting patches [7], [8], [9]. These methods generate realistic textures in images with simple structures or small holes but usually present critical failures for images with non-repetitive patterns, such as faces and complex scenes.

Like many other fields in computer vision, image inpainting also took its share with the advancements in deep learning. Recently, deep generative-based methods [10], [11], [12] address the problems of traditional inpainting using generative adversarial networks (GANs) [13]. The latter demonstrates a powerful tool to fill in the corrupted image with plausible alternative contents by learning high-level features from large-scale datasets. However, most of the current GAN-based inpainting techniques suffer from problems related to structure preservation and unrealistic texture generation, which usually leads to blurry and geometrically distorted results. Recent studies employ the contextual attention mechanism (CAM) to borrow information from the surrounding parts [14]. CAM still fails to ensure feature continuities [15] and requires heavy computational resources.

Other approaches divide the inpainting task into multiple stages, such that the early stages reconstruct the image structure represented in the edge [16], the contour [17] and the segmentation labels [18]. The later stages generally use reconstructed information to generate the final image. However, the performance of these multi-stage approaches is strongly related to the contour/edge/segmentation labels prediction stages. Also, they require expensive computational resources since they optimize the parameters of two or more networks. Another bottleneck that drastically increases the model capacity is training on high-resolution images which involves big models with a large number of parameters. Consequently, the training becomes slower and enforces smaller batch sizes due to computational and memory resources constraints [19].

To the best of our knowledge, this manuscript introduces the first study that presents a deep generative-based multi-resolution image inpainting framework. Our approach is composed of four successive generators filling in four different resolutions. Particularly, the training starts with lower-resolution images and progressively doubles their size such that their corresponding generators can exploit the previously inpainted images (see Fig. 1). This speeds up the training and improves stability since training GANs on low-resolution images is shown to be easier and converge faster [19].

Most of the current GAN-based inpainting methods are coarse-to-fine architectures [1], [16], [21]. Generally, they divide the problem into two subtasks: the coarse pass that predicts the initial image from the corrupted one. The refinement pass improves and sharpens the output of the coarse network to generate realistic textures using complex mechanisms such as CAM [14], [1]. We drop the refinement module after the target resolution since it drastically increases the network size. We remedy the lack of this refinement stage by a new texture-based loss function. This loss utilizes Local-binary-patterns (LBP) [22], which are non-parametric texture descriptors used well in computer vision tasks [23]. We minimize the distance between the ground truth LBP and the predicted one to enforce fine-grained textures. Hence, our approach does not require less computational resources since it neither performs complex modules nor uses the refinement network.

We conduct our qualitative and quantitative experiments on conventional inpainting datasets Places2 [24] and CelebHQ [19]. The experiment results show that our efficient and effective model can generate plausible inpainting results with realistic textures and achieve competitive results against the current state-of-the-art methods. We summarize our contribu-
Fig. 1. Overview of our network architecture, where we have four progressive resolutions (32, 64, 128 and 256). Each one has a specific PatchGAN discriminator that learns how to tell apart real from generated images, and a generator that exploits the previously inpainted resolutions to fill in the corrupted image.

...tions as follows:

1) We present a new image inpainting architecture that employs progressive multi-resolution generators to stabilize training and improve the performance.

2) We introduce a new LBP-based loss function to constrain the inpainting task and to ensure realistic texture details.

3) We present a new generative-based approach that reduces the network parameters without affecting the inpainting performance. We show competitive qualitative and quantitative results against current state-of-the-art methods.

The remainder of this paper is organized as follows. Section II reviews the related work. Section III presents the preliminaries while Section IV explains our approach in detail. Section V describes the experimental evaluation whose results are discussed in Section VI providing quantitative and qualitative comparisons. Finally, Section VII presents the conclusions and directions for future work.

II. RELATED WORK

Many and diverse image inpainting approaches are proposed in the literature. They can be classified into two major categories: traditional and deep learning approaches. Traditional methods employ either diffusion-based or patch-based techniques. Diffusion-based techniques fill in the holes by propagating the appearance of the neighborhood region to them [25]. Therefore, they may fail to generate meaningful structures for large or complex holes since only surrounding pixels of missing regions contribute to the inpainting process. In contrast, patch-based image inpainting can fill in relatively larger holes with realistic textures by searching and copying the best matching patches [7]. However, this iterative operation is expensive in terms of both memory and time. To overcome this limitation, [8] generalizes the previous algorithm and speeds up the inpainting applications. Furthermore, patch-based methods extract only low-level features. Consequently, they can not understand the semantic structure of the image resulting in lower performances in many cases, such as images of crowded scenes.

Recently, learning-based methods benefit from the fast improvements of deep neural networks (DNNs) and GANs [13] to learn the image semantic from large-scale datasets. These methods directly predict the missing pixel values using encoder-decoder architectures. Context encoders [10] is one of the first attempts that fill in a square hole in the center of the image using adversarial learning. The method suffers from obvious artifacts and exhibits blurriness. It was improved by [11] using two discriminators to ensure global and local image consistency. A postprocessing step using [26] followed by [27] is required to guarantee the color coherency around square holes. [14] replaces the postprocessing step by attaching the coarse network to another refinement network, which employs the contextual attention mechanism (CAM) to improve the quality of the coarse image. This method enhances the semantic consistency since it searches for a collection of surrounding background patches with the highest similarity score to the coarse image. However, it does not ensure pixel continuities since it is trained using rectangular regions. This was addressed by [15] that can handle free form masks by adding a coherent semantic attention layer to the refinement network. However, this method is time-consuming since it performs complex operations requiring...
heavy computational resources. Another approach [28] handles irregular masks and addresses the artifacts problem using an automatic mask updating mechanism of the partial convolution layers that eliminate substituting pixels and use only valid pixels. Moreover, [12] achieves competitive results using a fusion block that generates a flexible alpha composition map to combine corrupted and non-corrupted pixels.

Other methods utilize multi-stage architectures to reduce the complexity of the inpainting problem by providing additional information to the model. [18] is a two-stage architecture that predicts the segmentation labels to generate plausible images of foreground objects. [17] is a three-stage architecture that uses the contour information to preserve both foreground and background object boundaries. In another two-stage architecture [16], the edges are predicted to supervise the model prediction and recover the image structure. [21] adds appearance flow to a second stage to establish long-term corrections between masked and contextual regions. The hand-drawn sketches and gated layers generate plausible images using free from masks in [1]. All of these coarse-to-fine methods involve models with a large number of parameters to be optimized. Reducing the size of the model without affecting the quality of the generated images is desirable [29], [30]. In this work, we present an efficient and effective inpainting method that reduces the training time without using complex or expensive mechanisms such as CAM or large models.

III. PRELIMINARIES

A. Generative Adversarial Networks

Introduced in [13], GANs have shown huge success in image synthesis and have been adopted for modeling complex computer vision problems, including video generation [31], text generation [32], and image-to-image translation [20]. Although GANs are rapidly improving and building a new state-of-the-art in these tasks, they are still hard to train since they optimize the parameters of two neural networks independently in a minimax game. The first network is a generator that produces new samples similar to the real data. The discriminator network is optimized to distinguish between fake and real data. The loss function is defined as follows:

$$\min_{G} \max_{D} E_{x \sim P_{data}(x)}[\log(D(x))] + E_{z \sim P_{z}(z)}[\log(1 - D(G(z)))]$$

(1)

Where \( z \) is a random vector sampled from a Gaussian distribution, \( x \) is a real data sample, \( G(\cdot) \) is the generator network, and \( D(\cdot) \) is the discriminator network.

B. Local Binary Patterns

LBP is a nonparametric image operator that transforms an image into an array representing the local structure of the image by comparing each pixel with its adjacent pixels [22]. LBP is a robust descriptor that can summarize the most important texture information in an image. Also, it shows computational simplicity and good performance in many computer vision and image processing applications [23]. An example of a 3 × 3 LBP operator is shown in Fig. 2. LBP iterates over each pixel in a grayscale image to check the values of the surrounding 3 × 3 patch, whether they are smaller than the center pixel or not. The resulting binary number is converted to a decimal number and placed in the corresponding position in the LBP image.

IV. Approach

A. Multi-resolution-based Inpainting

Training GANs on high-resolution images is a hard optimization problem that involves a large number of parameters. [19] starts to produce low-resolution images from a latent vector in the first stage. After that, it progressively adds layers to the generator and the discriminator to increase the image resolution. However, this framework is not suitable for image-to-image translation applications since they require a high-resolution image as input. To overcome this problem, we train an encoder-decoder generator on a low-resolution image for many epochs to robustly produce images with a very close distribution to the original one. As the training progresses, we use the pretrained generators as a starting point for the generator of the next higher resolution. Using this strategy helps the later one to exploit the filled-in regions of the previous lower resolution images to complete the missing details with correct structures. In contrast, training GANs for the image inpainting task on high-resolution images is hard to stabilize, which may affect the performance of the model. We can explain this by the fact that, during training, the discriminator keeps rejecting most of the generated images, since the ground truth image contains fine-grained texture details which are very difficult for the generator to produce [19], [33].

B. Architecture

As described in Fig. 1 the training starts with the 32 × 32 resolution images. We channel-wise concatenate the corrupted image and the mask to feed them to their specific-resolution generator. The output and the ground truth images are then
we transform the problem into matrix multiplication operations using a fixed weight convolution layer. Thus, it does not add learnable parameters to our full model. Note that we only use the LBP loss in the last resolution ($256 \times 256$). We base our implementation on [35].

**Algorithm 1:** The LBP layer pseudo-code

**Input:** Gray-scale image  
**Output:** LBP image  
**Function** $\text{LBPLayer}$:

- $\text{Conv} =$ 2D convolution layer.  
- Initialize the parameters to: $\text{in}_{\text{channels}} = 1$, $\text{out}_{\text{channels}} = 8$, $\text{kernel} = 3$, $\text{stride} = 1$, $\text{dilation} = 1$, $\text{bias} = \text{False}$.  
- Initialize the kernels to zeros.  
- Initialize the center of the kernels to -1.  
- Initialize the remaining values to 1 in position: 0, 1, 2, 3, 4, 5, 6 and 7 for each kernel, respectively.  
- $\text{codes} =$ list of 8 values initialized to 1, 2, 4, 8, 16, 32, 64, 128.  
- $\text{ReLU} =$ Rectified Linear Unit activation function.  
- $\text{result} =$ $\text{Conv}(\text{input})$.  
- $\text{result} =$ $\text{ReLU}(\text{result})$.  
- $\text{result} =$ $\text{result} + \text{codes}$.  
- $\text{result} =$ $\text{result}.\text{sum}(\text{dim} = 1)$.  
- $\text{result} =$ $\text{result}/255$.  
- $\text{return}$ $\text{result}$

V. EXPERIMENTAL EVALUATION

A. Datasets

We conduct our experiments using two conventional image inpainting datasets. The first one is Places2 [24] that has more than 1.8M images and 400 scene categories, such as bedrooms, streets, etc. Although the Places2 dataset was created for a classification task, it became a popular image inpainting dataset since it has a vast natural scene variation. We use the original train and test split for the Places2 dataset. To further enrich our experiments, we evaluate our method on CelebHQ [19], which is a challenging dataset that has 30K cropped face images selected from the CelebA [36], it has a large pose and background variations. We use the same training and test split of the CelebA dataset. Since users of image inpainting applications usually want to edit or remove arbitrary shapes in the scenes, we use irregular mask sizes [37] during training. In test time, we classify the mask images based on the ratio between the hole size and the entire image size into four categories (10-20%, 20-30%, 30-40%, and 40-50%).

B. Implementation Details

In this part, we describe our training procedure and the hyper-parameter settings. We use Pytorch [38] to implement the proposed method using CUDA v10.1 and cuDNN v7.6.4. We use Adam optimizer [39] with hyper-parameters $\alpha = 0.5$...
and $\beta = 0.99$, respectively. We set the batch size to 32, and we fix the learning rates to $10^{-4}$ for the generators and the discriminators. We use spectral normalization [40] in all the convolution layers of the discriminator. The details of the architectures illustrated in Fig. 1 are described in Appendix A and Appendix B for the discriminators and the generators, respectively. We freeze the weights of the previous networks when training the generator and the discriminator of the current resolution.

C. Loss Functions

Let $I_{n \times n}$ and $M_{n \times n}$ be the ground truth image and the mask, where $n$ is the size of a square image. Also, let $G_{n \times n}(.)$ be a generator network that produces an image $O_{n \times n}$. Let $Gray(.)$ be a function that transforms a color image into a grayscale image. Let $LBP(.,.)$ be a differentiable LBP layer that takes a grayscale image and outputs the LBP image. The output image for various resolutions can be obtained using Equations 2-5

\begin{equation}
O_{32 \times 32} = G_{32 \times 32}(I_{32 \times 32} * M_{32 \times 32}, M_{32 \times 32}) \tag{2}
\end{equation}

\begin{equation}
O_{64 \times 64} = G_{64 \times 64}(I_{64 \times 64} * M_{64 \times 64}, M_{64 \times 64}, O_{32 \times 32}) \tag{3}
\end{equation}

\begin{equation}
O_{128 \times 128} = G_{128 \times 128}(I_{128 \times 128} * M_{128 \times 128}, M_{128 \times 128}, O_{32 \times 32}, O_{64 \times 64}) \tag{4}
\end{equation}

\begin{equation}
O_{256 \times 256} = G_{256 \times 256}(I_{256 \times 256} * M_{256 \times 256}, M_{256 \times 256}, O_{32 \times 32}, O_{64 \times 64}, O_{128 \times 128}) \tag{5}
\end{equation}

\textbf{L1 loss:} we measure the error between the ground truth image and the predicted image for each resolution as defined in Eq. 6

\begin{equation}
L_{rec} = ||O_{n \times n} - I_{n \times n}||_1 \tag{6}
\end{equation}

\textbf{Adversarial loss:} we optimize the LSGAN [41] adversarial loss for each resolution as defined in Eq. 7 and Eq. 8 respectively.

\begin{equation}
L_{dis} = E[(D(I_{n \times n}) - 1)^2] + E[D(O_{n \times n})^2] \tag{7}
\end{equation}

\begin{equation}
L_{adv} = E[(D(O_{n \times n}) - 1)^2] \tag{8}
\end{equation}

\textbf{Texture loss:} we use the LBP differentiable layer to calculate the loss between the ground truth texture and the generated $256 \times 256$ image texture, see Eq. 9

\begin{equation}
L_{texture} = ||LBP(Gray(O_{fine})) - LBP(Gray(I_{g}))||_1 \tag{9}
\end{equation}

\textbf{Overall loss:} we use a weighted sum of the reconstruction, the adversarial and the texture loss. We give a weight of $\lambda_{adv} = 0.1$, $\lambda_{rec} = 1$ and $\lambda_{texture} = 10$ for the adversarial loss, the reconstruction loss and the texture loss, respectively. The overall loss is defined in Eq. 10

\begin{equation}
L_{overall} = \lambda_{adv} * L_{adv} + \lambda_{rec} * L_{rec} + \lambda_{texture} * L_{texture} \tag{10}
\end{equation}

VI. RESULTS

We qualitatively and quantitatively compare our full model with current state-of-the-art methods, including contextual attention (CA) [14], edge connect (EC) [16], deep fusion network (DFNet) [12], and gated convolution (GC) [1]. We select these approaches for two main reasons. The first one is that they have the pretrained models, which ensure a fair comparison and save both time and computational resources. The second reason is that they achieve competitive results using different approaches. We use the original train and test splits for Places2 [24] and CelebHQ [19] datasets.

A. Qualitative Comparison

We qualitatively compare our approach with the selected state-of-the-art methods on two datasets. Seen from figure Fig. 4, CA [14] generates significant artifacts leading to mispresent structures. EC [16] produces better results since it predicts edges to recover the global structure of the image, but obvious visual artifacts still appear in the masked regions. While DFNet [12] generates plausible and smooth images with global image consistency using fusion blocks, it still exhibits observable color discrepancies. GC [1] produces realistic images due to the gated convolution layers and the refinement network, but it uses a large network (4.1M parameters). Our proposed method generates plausible images with fine-grained textures with a smaller number of parameters (3M). To further evaluate the proposed method, we report qualitative results on the CelebHQ dataset. Seen from figure Fig. 5 the images produced by CA show visually poor performance. GC generates realistic images but still shows irrelevance between the background and the parts of the hole. Our method shows the most natural faces without using large models or complex mechanisms such as CAM. We can explain this by the fact that our stable multi-resolution generators produce visually realistic images with global structure consistencies. Meanwhile, the proposed LBP-based loss function both improves and sharpens the texture generated image.

B. Quantitative Comparison

Generally, the image generation task lacks descriptive metrics. Nevertheless, to quantify the performance of the proposed approach, we use three well-know assessment metrics, including mean absolute error (MAE), peak signal-to-noise ratio (PSNR), and structural similarity index (SSIM) following works of [15], [21]. For a fair comparison, we use the same masks and test splits of the two datasets. Table 1 lists the evaluation results on the Places2 dataset. We can see that CA [14] shows the worse performances in the three metrics on different mask sizes. EC [16] exhibits better results since it predicts the edges to supervise the image structure generation. The scores of DFNet [12] and GC [1] are better and very
Fig. 4. Example cases of qualitative comparison between our model with state-of-the-art methods using irregular hole inpainting on Places2 test set. From left to right, the ground truth image, the corrupted image and the results of CA [14], EC [16], DFNet [12], GC [1], and our model are presented.

Fig. 5. Four sample results for qualitative comparison of the proposed model and the state-of-the-art methods on CelebHQ test data. From left to right, the ground truth image, the corrupted image and the results of CA [14], GC [1], and our model are shown.

close to each other. We can explain this by the fact that DFNet employs a fusion block, and GC employs gated layers as well as a refinement network. Our approach achieves competitive results compared to the mentioned state-of-the-art methods without using the refinement network. This is because multi-resolution generators help to stabilize the training, and the novel LBP-loss improves the performance by constraining the prediction. Table II reports the quantitative comparison of CelebHQ. Our proposed method outperforms CA, which shows significantly lower performance. Also, it achieves comparable results compared to GC that has a larger number of network parameters.

C. Ablation Study on LBP Loss

To analyze the contribution of our proposed LBP loss function to the full approach, we implement two settings of the model, and we show qualitative and quantitative results for each version on the CelebHQ dataset [19]. The first employs only the proposed architecture, while the second adds the LBP loss function to constrain the prediction. We believe that the LBP can describe the texture of the image since the filter comparison operations keep the most meaningful pixels. Table III indicates that the LBP loss improves the performance and correlates very well with the metrics. Also, we can see from Fig. 6 that our additional LBP layer restores the image texture and provide realistic images. Note that the images of the first version are plausible and have semantic
Fig. 6. Ablation studies show two results from our model with and without the LBP loss on CelebHQ dataset. From left to right, the first two columns show the ground truth and LBP images followed by the input image, output without LBP loss and with LBP loss and the predicted LBP image.

**TABLE I**

**QUANTITATIVE EVALUATION ON PLACES2 DATASET WITH CA [16], EC [10], DFN [12], GC [1] AND OUR MODEL. (FOR MAE LOWER IS BETTER, FOR SSIM+ AND PSNR+ HIGHER IS BETTER). THE BEST SCORES ARE INDICATED IN BOLD.**

| Mask | CA    | EC    | DFN   | GC    | Ours    |
|------|-------|-------|-------|-------|---------|
| 10-20% | 0.019 | 0.013 | 0.010 | 0.011 | 0.009   |
| 20-30% | 0.033 | 0.022 | 0.019 | 0.018 | 0.016   |
| 30-40% | 0.048 | 0.031 | 0.028 | 0.026 | 0.024   |
| 40-50% | 0.075 | 0.053 | 0.045 | 0.045 | 0.042   |
| 10-20% | 0.922 | 0.947 | 0.965 | 0.969 | 0.971   |
| 20-30% | 0.861 | 0.913 | 0.936 | 0.942 | 0.946   |
| 30-40% | 0.795 | 0.879 | 0.901 | 0.909 | 0.916   |
| 40-50% | 0.660 | 0.762 | 0.803 | 0.810 | 0.816   |
| 10-20% | 26.31 | 27.88 | 29.51 | 30.10 | 30.62   |
| 20-30% | 23.07 | 25.51 | 26.73 | 27.13 | 27.71   |
| 30-40% | 20.91 | 23.96 | 24.87 | 25.07 | 25.74   |
| 40-50% | 18.27 | 20.80 | 22.03 | 21.78 | 22.55   |

**TABLE II**

**QUANTITATIVE EVALUATION ON CELEBHQ DATASET WITH CA [14], GC [1] AND OUR MODEL. (FOR MAE LOWER IS BETTER, FOR SSIM+ AND PSNR+ HIGHER IS BETTER). THE BEST SCORES ARE INDICATED IN BOLD.**

| Mask | CA    | GC    | Ours    |
|------|-------|-------|---------|
| MAE  | 10-20% | 0.014 | 0.009   |
| 20-30% | 0.024 | 0.014 | 0.010   |
| 30-40% | 0.033 | 0.020 | 0.015   |
| 40-50% | 0.052 | 0.031 | 0.024   |
| SSIM+  | 10-20% | 0.995 | 0.982   |
| 20-30% | 0.918 | 0.968 | 0.979   |
| 30-40% | 0.881 | 0.940 | 0.967   |
| 40-50% | 0.796 | 0.905 | 0.924   |
| PSNR   | 10-20% | 28.35 | 32.53   |
| 20-30% | 25.54 | 29.73 | 31.79   |
| 30-40% | 23.58 | 27.80 | 29.81   |
| 40-50% | 21.03 | 25.06 | 26.64   |

consistency, which proves the effectiveness of our proposed multi-resolution generators.

**D. Interactive Editing**

Our method allows users to remove unwanted objects by interactively drawing the input masks. At the same time, it can...
robustly recover the corrupted parts without artifacts. In both cases, the generated images have realistic texture and global semantic consistency. Some results of the interactive inpainting are provided in Fig. 7. Our approach robustly removes the glasses and face accessories around complex textured objects such as eyes and hair in the CeleB-HQ dataset. Further, it provides plausible images on the Places2 dataset that includes crowded scenes.

VII. Conclusion

In this study, we propose both an effective and efficient end-to-end GAN-based framework for the image inpainting task. Our approach employs successive generators for progressive resolutions such that the generators of higher resolutions benefit from the previously inpainted images by the generators of lower resolutions. We show that the proposed architecture plays an important role in stabilizing the training of the model. We demonstrate that the LBP loss function can help to restore the image structure and generate fine-grained textures. Quantitative and qualitative results on Places2 and CelebA-HQ datasets show the competitiveness of our method compared to several state-of-the-art models. For the next step, we are planning to adapt our architecture and loss function to other image-to-image translation tasks, including image super-resolution, image denoising, and image deblurring.

Appendix A

Discriminator

Table IV shows the architecture of the PatchGAN discriminator [20] where: \( n = 24 \) for the \( 32 \times 32 \) and the \( 64 \times 64 \) discriminators, and \( n = 48 \) for the \( 128 \times 128 \) and the \( 128 \times 128 \) discriminators. We use a slope of 0.2 in the LeakyReLU activation function. We use Spectral Normalization [40] in the convolution layers where: bias = False. We initialize the weights using a Gaussian distribution with gain = 0.02.

### Table IV

| Layer | Dim | Kernel | Stride | Padding | Activation |
|-------|-----|--------|--------|---------|------------|
| Conv2D | \( n \) | \( 4 \times 4 \) | 2 | 1 | LeakyReLU |
| Conv2D | \( n \times 2 \) | \( 4 \times 4 \) | 2 | 1 | LeakyReLU |
| Conv2D | \( n \times 4 \) | \( 4 \times 4 \) | 2 | 1 | LeakyReLU |
| Conv2D | 1 | \( 4 \times 4 \) | 1 | 1 | - |

Appendix B

Generators

For all the generators defined in Table V [21] Table VI [22] Table VII and Table VIII [23] we use the same weight initialization method used in the discriminator. TConv2D refers to the ConvTranspose2d layer in Pytorch [38]. The Gray function used in Algorithm 1 is done as follows: \( Gray(r, g, b) = 0.299 * r + 0.587 * g + 0.110 * b \) where \( r, g, \) and \( b \) are the red, green and blue colors, respectively.

### Table V

| Layer | Dim | Kernel | Stride | Padding | Activation |
|-------|-----|--------|--------|---------|------------|
| Conv2D | 24 | \( 3 \times 3 \) | 1 | 1 | ReLU |
| Conv2D | 48 | \( 4 \times 4 \) | 2 | 1 | ReLU |
| Conv2D | 48 | \( 4 \times 4 \) | 2 | 1 | ReLU |
| Conv2D | 96 | \( 3 \times 3 \) | 1 | 1 | ReLU |
| Conv2D | 96 | \( 3 \times 3 \) | 1 | 1 | ReLU |
| Conv2D | 96 | \( 3 \times 3 \) | 1 | 1 | ReLU |
| Conv2D | 96 | \( 3 \times 3 \) | 1 | 1 | ReLU |
| Conv2D | 96 | \( 3 \times 3 \) | 1 | 1 | ReLU |
| TConv2D | 48 | \( 4 \times 4 \) | 2 | 1 | ReLU |
| TConv2D | 24 | \( 4 \times 4 \) | 2 | 1 | ReLU |
| Conv2D | 3 | \( 3 \times 3 \) | 1 | 1 | Tanh |

### Table VI

| Block | Layer | Dim | Kernel | Stride | Padding | Activation |
|-------|-------|-----|--------|--------|---------|------------|
| 1     | Conv2D | 24 | \( 3 \times 3 \) | 1 | 1 | ReLU |
| 2     | Conv2D | 48 | \( 4 \times 4 \) | 2 | 1 | ReLU |
| 3     | Conv2D | 48 | \( 4 \times 4 \) | 2 | 1 | ReLU |

### Table VII

| Block | Layer | Dim | Kernel | Stride | Padding | Activation |
|-------|-------|-----|--------|--------|---------|------------|
| 1     | Conv2D | 24 | \( 3 \times 3 \) | 1 | 1 | ReLU |
| 2     | Conv2D | 48 | \( 4 \times 4 \) | 2 | 1 | ReLU |
| 3     | Conv2D | 48 | \( 4 \times 4 \) | 2 | 1 | ReLU |
| 4     | Conv2D | 96 | \( 3 \times 3 \) | 1 | 1 | ReLU |

### Table VIII

| Block | Layer | Dim | Kernel | Stride | Padding | Activation |
|-------|-------|-----|--------|--------|---------|------------|
| 1     | Conv2D | 24 | \( 3 \times 3 \) | 1 | 1 | ReLU |
| 2     | Conv2D | 48 | \( 4 \times 4 \) | 2 | 1 | ReLU |
| 3     | Conv2D | 48 | \( 4 \times 4 \) | 2 | 1 | ReLU |

Gray function: \( Gray(r, g, b) = 0.299 * r + 0.587 * g + 0.110 * b \)
TABLE VIII
ARCHITECTURE OF THE 256 × 256 GENERATOR NETWORK

| Block | Layer | Dim | Kernel | Stride | Padding | Activation |
|-------|-------|-----|--------|--------|---------|------------|
| 1     | Conv2D | 24  | 3 × 3  | 1      | 1       | ReLU       |
|       | Conv2D | 48  | 4 × 4  | 2      | 1       | ReLU       |
| 2     | Conv2D | 48  | 4 × 4  | 2      | 1       | ReLU       |
|       | Conv2D | 48  | 3 × 3  | 1      | 1       | ReLU       |
| 3     | Conv2D | 24  | 3 × 3  | 1      | 1       | ReLU       |
|       | Conv2D | 48  | 3 × 3  | 1      | 1       | ReLU       |
| 4     | Conv2D | 24  | 3 × 3  | 1      | 1       | ReLU       |
|       | Conv2D | 48  | 3 × 3  | 1      | 1       | ReLU       |
| 5     | Conv2D | 96  | 3 × 3  | 1      | 1       | ReLU       |
|       | Conv2D | 96  | 3 × 3  | 1      | 1       | ReLU       |
|       | Conv2D | 96  | 3 × 3  | 1      | 1       | ReLU       |
|       | Conv2D | 96  | 3 × 3  | 1      | 1       | ReLU       |
|       | TConv2D | 48  | 4 × 4  | 2      | 1       | ReLU       |
|       | TConv2D | 24  | 4 × 4  | 2      | 1       | ReLU       |
|       | Conv2D | 3   | 3 × 3  | 1      | 1       | Tanh       |

APPENDIX C
LEARNING CURVES

We show the training curves of our four generators and discriminators. The loss curves show a stable training that reflects the visual quality of the generated images. Fig. 8 shows the loss values of the generators and the discriminators. In Fig. 9 we show the reconstruction loss values. During training, we use masks that cover 30-40% of the image. The successful exploitation of previously inpainted low-resolution images leads to fast convergence.

Fig. 8. The GAN losses of the generators and the discriminators showing a stable training on the four resolutions.

![GAN losses](image)

Fig. 9. The image reconstruction of the four resolutions and the LBP texture reconstruction loss of the target resolution (256 × 256).

![Image reconstruction](image)

REFERENCES

[1] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. Huang, “Free-form image inpainting with gated convolution,” IEEE International Conference on Computer Vision (ICCV), vol. 2019-October, pp. 4470–4479, 2019.

[2] Y. Gao, G. Cheung, T. Maugpy, P. Frossard, and J. Liang, “Encoder-driven inpainting strategy in multiview video compression,” IEEE Transactions on Image Processing, vol. 25, no. 1, pp. 134–149, 2016.

[3] H. Yamauchi, J. Haber, and H. P. Seidel, “Image restoration using multiresolution texture synthesis and image inpainting,” in Proceedings of Computer Graphics International Conference, CGI, 2003, pp. 120–125.

[4] N. Kawai, T. Sato, and N. Yokoya, “Diminished reality based on image inpainting considering background geometry,” IEEE Transactions on Visualisation and Computer Graphics, vol. 22, no. 3, pp. 1236–1247, 2015.

[5] N. Kawai, M. Yamasaki, T. Sato, and N. Yokoya, “Diminished reality for AR marker hiding based on image inpainting with reflection of luminance changes,” ITE Transactions on Media Technology and Applications, vol. 1, no. 4, pp. 343–353, 2013.

[6] S. Anstis, “Visual filling-in,” Current Biology, vol. 20, no. 16, pp. R664–R666, 2010.

[7] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman, “PatchMatch,” in ACM SIGGRAPH, 2009.

[8] C. Barnes, E. Shechtman, D. B. Goldman, and A. Finkelstein, “The generalized PatchMatch correspondence algorithm,” in Lecture Notes in Computer Science, 2010, vol. 6313 LNCS, no. PART 3, pp. 29–43.

[9] K. He and J. Sun, “Image completion approaches using the statistics of similar patches,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 36, no. 12, pp. 2423–2435, 2014.

[10] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, “Context Encoders: Feature Learning by Inpainting,” IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2536–2544, 2016.

[11] S. Iizuka, E. Simo-Serra, and H. Ishikawa, “Globally and locally consistent image completion,” ACM Transactions on Graphics, vol. 36, no. 4, pp. 1–14, 2017.

[12] X. Hong, P. Xiong, R. Ji, and H. Fan, “Deep fusion network for image completion,” in Proceedings of the 27th ACM International Conference on Multimedia, 2019, pp. 2033–2042.

[13] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” Advances in Neural Information Processing Systems, vol. 3, no. January, pp. 2672–2680, 2014.

[14] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang, “Generative Image Inpainting with Contextual Attention,” IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 5505–5514, 2018.

[15] H. Liu, B. Jiang, Y. Xiao, and C. Yang, “Coherent semantic attention...
for image inpainting,” in IEEE International Conference on Computer Vision (ICCV), Oct. 2019.

[16] K. Nazeri, E. Ng, T. Joseph, F. Qureshi, and M. Ebrahimim, “Edgeconnect: Structure guided image inpainting using edge prediction,” in International Conference on Computer Vision Workshop (ICCVW), 2019, pp. 3265–3274.

[17] W. Xiong, J. Yu, Z. Lin, J. Yang, X. Lu, C. Barnes, and J. Luo, “Foreground-aware image inpainting,” in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Jun. 2019.

[18] Y. Song, C. Yang, Y. Shen, P. Wang, Q. Huang, and C.-C. J. Kuo, “Spg-net: Segmentation prediction and guidance network for image inpainting,” in BMVC, 2018.

[19] T. Karras, T. Aila, S. Laine, and J. Lehtinen, “A style-based generator architecture for generative adversarial networks,” in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 4401–4410.

[20] P. Isola, J. Y. Zhu, T. Zhou, and A. A. Efros, “Image-to-image translation with conditional adversarial networks,” in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), vol. 2017-January, Jul. 2017, pp. 5967–5976.

[21] Y. Ren, X. Yu, R. Zhang, T. H. Li, S. Liu, and G. Li, “Structureflow: Image inpainting via structure-aware appearance flow,” in IEEE International Conference on Computer Vision (ICCV), vol. 2019-October, 2019, pp. 181–190.

[22] T. Ahonen, A. Hadid, and M. Pietikainen, “Face description with local binary patterns: Application to face recognition,” IEEE Transactions on pattern analysis and machine intelligence, vol. 28, no. 12, pp. 2037–2041, 2006.

[23] D. Huang, C. Shan, M. Ardabilian, Y. Wang, and L. Chen, “Local binary patterns and its application to facial image analysis: a survey,” IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), vol. 41, no. 6, pp. 765–781, 2011.

[24] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, “Places: A 10 million image database for scene recognition,” IEEE transactions on pattern analysis and machine intelligence, vol. 40, no. 6, pp. 1452–1464, 2017.

[25] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, “Image inpainting,” in Proceedings of the ACM SIGGRAPH Conference on Computer Graphics. New York, New York, USA: ACM Press, 2000, pp. 417–424.

[26] A. Telea, “An image inpainting technique based on the fast marching method,” Journal of graphics tools, vol. 9, no. 1, pp. 23–34, 2004.

[27] P. Pérez, M. Gangnet, and A. Blake, “Poisson image editing,” in ACM SIGGRAPH 2003 Papers, 2003, pp. 313–318.

[28] G. Liu, F. A. Reda, K. J. Shih, T. C. Wang, A. Tao, and B. Catanzaro, “Image Inpainting for Irregular Holes Using Partial Convolutions,” Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 11215 LNCS, pp. 89–105, 2018.

[29] Y.-G. Shin, M.-C. Sagong, Y.-J. Yeo, S.-W. Kim, and S.-J. Ko, “Pepsi++: Fast and lightweight network for image inpainting,” IEEE Transactions on Neural Networks and Learning Systems, 2020.

[30] H. Liu, B. Jiang, W. Huang, and C. Yang, “One-stage inpainting with bilateral attention and pyramid filling block,” arXiv preprint arXiv:1912.08642, 2019.

[31] S. Tulyakov, M.-Y. Liu, X. Yang, and J. Kautz, “Mocogan: Decomposing motion and content for video generation,” in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 1526–1535.

[32] W. Fedus, I. Goodfellow, and A. M. Dai, “Maskgan: Better text generation via filling in the _,” arXiv preprint arXiv:1801.07736, 2018.

[33] A. Odena, C. Olah, and J. Shlens, “Conditional image synthesis with auxiliary classifier gans,” in International conference on machine learning, 2017, pp. 2642–2651.

[34] J. Canny, “A computational approach to edge detection,” IEEE Transactions on pattern analysis and machine intelligence, no. 6, pp. 679–698, 1986.

[35] J. Canny, “A computational approach to edge detection,” IEEE Transactions on pattern analysis and machine intelligence, no. 6, pp. 679–698, 1986.

[36] D. Patel, “LOCAL BINARY PATTERNS, access time (10-03-2020).” [Online]. Available: https://devanginiblog.wordpress.com/2016/06/03/local-binary-patterns-lbp/.

[37] Z. Liu, P. Luo, X. Wang, and X. Tang, “Large-scale celebrity faces attributes (celeba) dataset,” Retrieved August, vol. 15, 2018.

[38] K. Iskakov, “Quick Draw Irregular Mask Dataset, access time (20-02-2020).” [Online]. Available: https://github.com/karfly/qd-imd.

[39] D. Patel, “LOCAL BINARY PATTERNS, access time (10-03-2020).” [Online]. Available: https://devanginiblog.wordpress.com/2016/06/03/local-binary-patterns-lbp/.

[40] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, “Spectral normalization for generative adversarial networks,” arXiv preprint arXiv:1802.05957, 2018.

[41] X. Mao, Q. Li, H. Xie, R. Y. K. Lau, Z. Wang, and S. P. Smolley, “Least squares generative adversarial networks,” in IEEE International Conference on Computer Vision (ICCV), 2017, pp. 2813–2821.

[42] A. L. Maas, A. Y. Hannun, and A. V. Ng, “Rectifier nonlinearities improve neural network acoustic models,” in International Conference On Machine Learning (ICML), vol. 30, no. 1, 2013, p. 3.