EEC 206 Final Project Report: Image Inpainting Using Partial Convolutions

Hui-Ying Siao, Melody Tao, Vivienne Chiang, Yuexin Li

Abstract—**Image inpainting, or the act of filling up holes in an image, can be used for a variety of tasks. It can be used in image editing to get rid of certain objects or to clean up an image. There are many approaches to image inpainting, but many are related to filling images with rectangular holes. For our project, we try to apply a pre-existing technique that uses partial convolutions, which can be applied to images that contain irregularly shaped holes. We add our own modifications to the current technique and test it on a data set based off the ICME 2019 Challenge data set. Our results indicate that the technique does work on irregular holes reasonably well.**

Index Terms—**partial convolution, image inpainting, deep learning**

I. Introduction

Image inpainting, the method of filling in holes or "damages" to an image, can be used in many fields, such as in image editing. For example, it can be used in omitting passersby in security footage, or erasing objects in an image to make the image cleaner. Nowadays, a lot of image inpainting techniques are being developed using machine learning and deep learning. This allows us to do image editing with better results. However, most neural networks are focused on improving images that contain very limited types of holes and heavily rely on expensive post-processing [5]. For example, Pathak et al. [1] and Yang et al. [2] focus on images that are damaged with a fixed hole size. In addition, these holes are squares that are placed in the middle of an image. On the other hand, Iizuka et al. [3] and Yu et al. [4] did account for irregularly sized holes, but they did not provide an extensive test. Their test data sets were small, and therefore, their results may not be reliable, where the test size was 51 images.

In this project, we try to implement a technique that uses partial convolutions. The method is originally introduced by Guilin et al. [5], where the authors introduce a way to fill in irregularly sized holes with a much larger training and test set. Our project is to try and re-implement their work, but instead of using the occlusion/dis-occlusion mask technique from [6] to generate the masks, we opted for more simple techniques. We will be using the 2019 ICME Challenge data set for training our model.

Our paper is structured as follows: in Section II, we will go over previous approaches, their contributions, and why we chose to implement this method. In Section III, we will provide an overview of the algorithm and modifications to it that we made to make it work for our project. In Section IV, we will discuss the results of our project and provide both objective metrics and images to demonstrate our technique. Lastly, in Section V, we will conclude our final report.

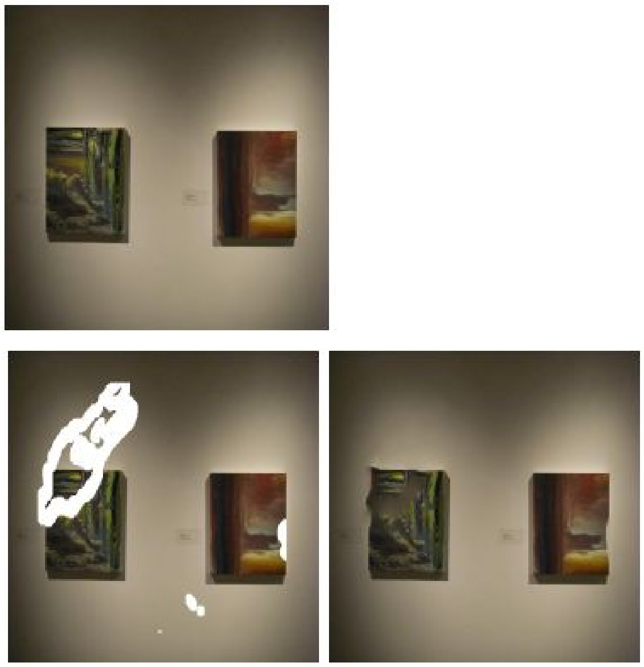


Figure 1. The top image shows the ground truth. The bottom left shows the damaged image, while the bottom right is the output of PatchMatch [5].

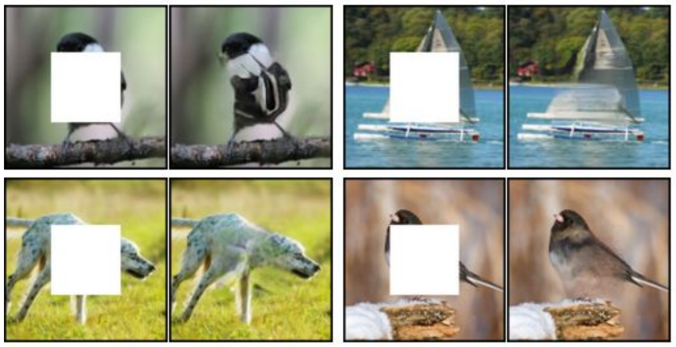


Figure 2. Method used by Pathak et al. [1] and Yang et al. [2], where the hole is at the center of the image and is a fixed square size.

II. Background

Previous work on image inpainting includes both non-learning based techniques along with learning based techniques. One state-of-the-art non-learning based technique is called PatchMatch [11], which finds correspondences between regions of an image using a nearest-neighbor field (NNF). It is an iterative technique that searches for regions of pixels that would best fill the hole. This results in smooth pictures near the holes, but it does not care that the pixels "make sense". In Figure 1, the upper left shows the ground truth, while the bottom left shows the damaged image, then followed by the PatchMatch output on the bottom right [5]. The PatchMatch approach takes pixels from the wall to fill in the hole for the green painting as well.

Approaches that are learning based include image inpainting techniques that use masks generated with fixed hole sizes and shapes. This is shown in Figure 2, where Pathak et al. [1] use masks that contain square holes that are 64x64 in size and placed at the center of a 128x128 image. Other techniques include Content Encoders [12], which also use square center holes. In Content Encoders, an encoder and decoder structure is used, where the image with the hole is embedded into the low dimensional frequency space, then is decoded to an image to half the size.

However, as mentioned previously, damages to images are not normally known. They can have various sizes, shapes, and forms, and thus, a center square hole is not enough to test. Consequently, Guilin et al. introduced the concept of partial convolutions, which is a technique that works well on irregular hole inpainting. Some examples of masks Guilin et al. used in training their model is shown in Figure 3, where the masks are irregular streaks, scribbles, and dots in the image. After many training iterations, the results are shown on the right-hand side of each image with holes. Based on the results, it would be safe to say that at a glance, the image would most likely fool the average viewer.

Thus, we try to implement this technique using the ICME 2019 Inpainting Grand Challenge data set, while applying masks that differ from the original masks used in Guilin et al.'s paper.

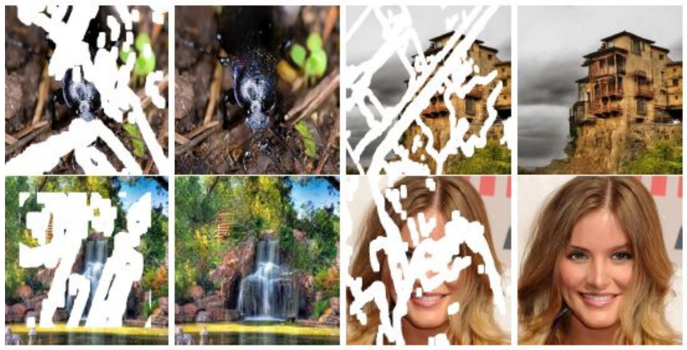


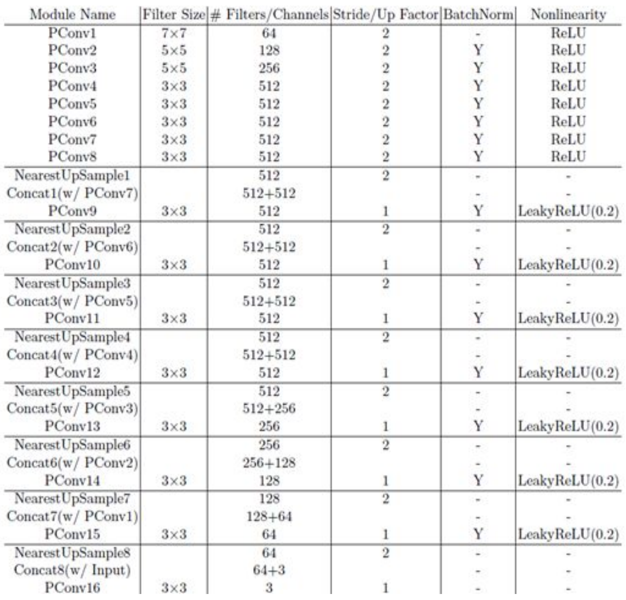
Figure 3. Masks used by Guilin et al. in the original implementation of the partial convolution technique [5].

III. Method

In this section, we will provide an overview of the partial convolution technique introduced by Guilin et al. [5]. Then, we will explain metrics used to analyze the performance of this technique. Lastly, we will describe the datasets we created to test this technique and the differences compared to the original by Guilin et al.

A. *Partial Convolution Technique*

The Network Design uses a UNet-like architecture [13], replacing all convolutional layers with partial convolutional layers and using nearest neighbor up-sampling in the decoding stage [5]. Figure 4 shows details of the network architecture. PConv means a partial convolutional layer with the specified filter size, number of filters, and stride. PConv1 to PConv8 are in the encoder stage, whereas PConv9 to PConv16 are in the decoder stage. The BatchNorm column shows whether this partial convolutional layer is followed by a Batch Normalization layer. The Nonlinearity column indicates the nonlinearity layer used in this project, which is either ReLU or LeakyReLU.



Module Name	Filter Size	# Filters/Channels	Stride/Up Factor	BatchNorm	Nonlinearity
PConv1	7×7	64	2	-	ReLU
PConv2	5×5	128	2	Y	ReLU
PConv3	5×5	256	2	Y	ReLU
PConv4	3×3	512	2	Y	ReLU
PConv5	3×3	512	2	Y	ReLU
PConv6	3×3	512	2	Y	ReLU
PConv7	3×3	512	2	Y	ReLU
PConv8	3×3	512	2	Y	ReLU
NearestUpSample1		512	2	-	-
Concat1(w/ PConv7)		512+512		-	-
PConv9	3×3	512	1	Y	LeakyReLU(0.2)
NearestUpSample2		512	2	-	-
Concat2(w/ PConv6)		512+512		-	-
PConv10	3×3	512	1	Y	LeakyReLU(0.2)
NearestUpSample3		512	2	-	-
Concat3(w/ PConv5)		512+512		-	-
PConv11	3×3	512	1	Y	LeakyReLU(0.2)
NearestUpSample4		512	2	-	-
Concat4(w/ PConv4)		512+512		-	-
PConv12	3×3	512	1	Y	LeakyReLU(0.2)
NearestUpSample5		512	2	-	-
Concat5(w/ PConv3)		512+256		-	-
PConv13	3×3	256	1	Y	LeakyReLU(0.2)
NearestUpSample6		256	2	-	-
Concat6(w/ PConv2)		256+128		-	-
PConv14	3×3	128	1	Y	LeakyReLU(0.2)
NearestUpSample7		128	2	-	-
Concat7(w/ PConv1)		128+64		-	-
PConv15	3×3	64	1	Y	LeakyReLU(0.2)
NearestUpSample8		64	2	-	-
Concat8(w/ Input)		64+3		-	-
PConv16	3×3	3	1	-	-

Figure 4. Partial Convolution Neural Network Structure

Instead of using typical padding for convolution operations, using partial convolution with appropriate masking at image boundaries can ensure the inpainted content at the image border will not be affected by invalid values outside of images. The image inpainting includes partial convolution and mask updating operations, both of which are implemented in the Partial Convolutional Layer [5].

Let **W** be the convolution filter weights for the convolution filter and b be the corresponding bias. **X** are the pixel values for the current convolution sliding window and **M** is the corresponding binary mask. The partial convolution at every location is done as follows:

$$x' = \begin{cases} \mathbf{W}^T(\mathbf{X} \odot \mathbf{M})\frac{\mathrm{sum}(\mathbf{1})}{\mathrm{sum}(\mathbf{M})} + b, & \text{if } \mathrm{sum}(\mathbf{M}) > 0 \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

where $\odot$ means element-wise multiplication, **1** has the same shape as **M** but all of its elements are being 1, and sum(**1**)/sum(**M**) is a scaling factor. The scaling factor sum(**1**)/sum(**M**) applies appropriate scaling to adjust the varying amount of valid inputs.

After each partial convolution operation, the mask is updated as follows:

$$m' = \begin{cases} 1, & \text{if sum}(\mathbf{M}) > 0 \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

If the input has any valid pixels, after successive applications of the partial convolution operations, the mask would eventually be updated to all ones.

B. Metrics

$$\text{MSE} = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i,j) - K(i,j)]^2 \quad (3)$$

$$PSNR = 10 \log_{10}\left(\frac{MAX^2_I}{MSE}\right) \quad (4)$$

$$SSIM(x,y) = \frac{(2\mu_x \mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \quad (5)$$

$$L_{TV} = \sum_{(i,j) \in R} \frac{\| I_{comp}^{i,j+1} - I_{comp}^{i,j} \|_1}{N_{I_{comp}}} + \sum_{(i,j) \in R, (i+1,j) \in R} \frac{\| I_{comp}^{i+1,j} - I_{comp}^{i,j} \|_1}{N_{I_{comp}}} \quad (6)$$

$$L_{perceptual} = \sum_{p=0}^{P-1} \frac{\left\| \Psi_p^{I_{out}} - \Psi_p^{I_{gt}} \right\|_1}{N_{\Psi_p^{I_{gt}}}} + \sum_{p=0}^{P-1} \frac{\left\| \Psi_p^{I_{comp}} - \Psi_p^{I_{gt}} \right\|_1}{N_{\Psi_p^{I_{gt}}}} \quad (7)$$

$$L_{style} = \sum_{p=0}^{P-1} \frac{1}{C_p C_p} \left\| K_p (\Psi_p^{I_{comp}})^T (\Psi_p^{I_{comp}}) - (\Psi_p^{I_{gt}})^T (\Psi_p^{I_{gt}}) \right\|_1 \quad (8)$$

To evaluate the performance of our model we selected several metrics, namely PSNR, SSIM and the loss functions provided by Liu's paper [5]. Their algorithms are shown in equations 4 - 8.

The first metric that we implemented is the PSNR, which is the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation. This is a classic metric used to measure the quality of reconstruction of lossy compression codecs. Acceptable values for wireless transmission quality loss are considered to be about 20 dB to 25 dB while the higher the better. For our model we get 19dB, which is an acceptable value for generating convincingly similar output images.

The other metrics we used to complement the deficiency of PSNR is SSIM. SSIM stands for structural similarity index and is a more complex test aimed to determine perceivable difference between two images, while also incorporating important perceptual phenomena, including both luminance masking and contrast masking terms. While PSNR pays more attention towards the absolute error, SSIM describes the structural information between neighboring pixels. The higher the SSIM value, the better that the image is reconstructed.

The last three metrics we implemented are the *total variation* metrics, abbreviated as *TV* as shown above, the *perceptual* metrics and the *style* metrics. The *total variation* metric is the sum of the absolute differences for neighboring pixel-values in the input images. The perceptual loss function works by summing all the squared errors between all the pixels and taking the mean. The style loss is defined as the root mean square difference between the two style matrices and it makes sure that the correlation of activations in *all* the layers are similar between the style image and the generated image.

C. Data Sets

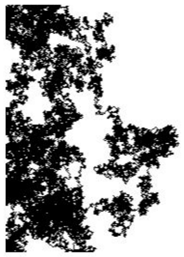
The main differences between our work and Guilin et al. is that we use different data sets and different training. We also constrained the image size to 256x256, both due to computational cost, and because we believe it was a good size to crop the ground truth images provided. We have created several data sets from the ICME 2019 data. In the original challenge, there are 1541 ground truth images and a validation set. The validation set contains two folders - one for error concealment (EC), the other for object removal (OR). Both datasets are structured in the same way, but the masks differ. For a dataset, there are 70 distinct images, but each image had a subset of ways it was cropped. The total number of files are shown in Table 1. The validation test set contains damaged images that already have holes, as well as 70 ground truth images.

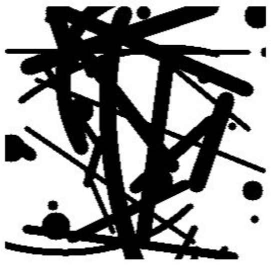
Table 1. Number of Images Per ICME Data Sets

Training (Ground Truth)	Validation (OR) (Damaged)	Validation (EC) (Damaged)
1541	4900	5600

For the training set, since we are provided 1541 ground truth images, we wanted to expand our dataset to make our neural network (NN) more robust. Thus, we applied augmentation to these ground truth images, where we used a dataset containing 7700 images for training. In Figure 5, the images show the augmentation we implemented for the training dataset in our experiment, including image random flipping, color jittering, resizing, and random image rotation. Since we were not provided masks, we also generated masks using three techniques.

In Figure 5 **(**a), the mask is generated using a random walk technique, where we start at a random pixel and randomly choose neighboring pixels to move to. The walk pattern is the part in black. This method is used to create holes in the image that are more finely detailed and are not solid blocks. The second type of mask is shown in Figure 5 **(**b), where we generated masks using random lines and circles and ellipses. Lastly, in Figure 5 (c), we have a mask that contains random rectangles that can overlap to also create irregular shaped holes. We generate 7700 random masks using these three techniques, where all masks are equally likely.


(a)


(b)

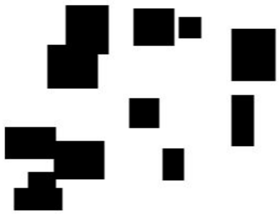

(c)

Figure 5. (a) Random walk type mask, as used by Shinagawa in [8]. (b) Random lines and shapes as used by Gruber in [9]. (c) Randomly generated rectangles technique.

We also decided to test our implementation on different types of damaged images. In addition to the ICME 2019 validation set of masks, we added in the previously three mentioned masks. These masks are completely independent from the training data set. For validation, we tested on 1750 randomly selected images, which is 16.67% of the original ICME validation test set of 10,500 damaged images.

IV. Results

In Figure 6, the image shows what the augmented dataset contains, which are randomly flipped, rotated, or slightly discolored versions of the ground truth images. In Figures 7 and 8, we compared the training process after 100 epochs and 150 epochs, where the results indicate that the training process with the epochs after 150 show a more continuous image impairment.

In order to compare the quality of the image impairment, Table 2 provided a list of loss values, PSNR and SSIM, to show the evaluation metrics of the image inpainting results. Our results indicate that the PSNR after 150 epochs is quite low compared to a good PSNR value, which is above 30dB. However, PSNR is known to be a worse quality metric compared to SSIM, where $0 < \text{SSIM} < 1$. Based on the average SSIM results in Table 2, a good SSIM value is close to 1. As mentioned earlier, the higher the SSIM value, the better that the image is reconstructed. For this metric we get the result of 0.93, which further proves the reliability of our model. For TV metrics, we get the value of 0.68 and this shows that the transition between the holes and the surroundings are smooth enough for human perception. For the perceptual metric and the style metric, we get the value of 0.05 and 2.45, respectively, which shows that the output image is similar enough to the ground truth image after image inpainting.

In Figure 9, we also added the results of testing on one of the images from the test set provided on Canvas from the "Final Test Images.zip".

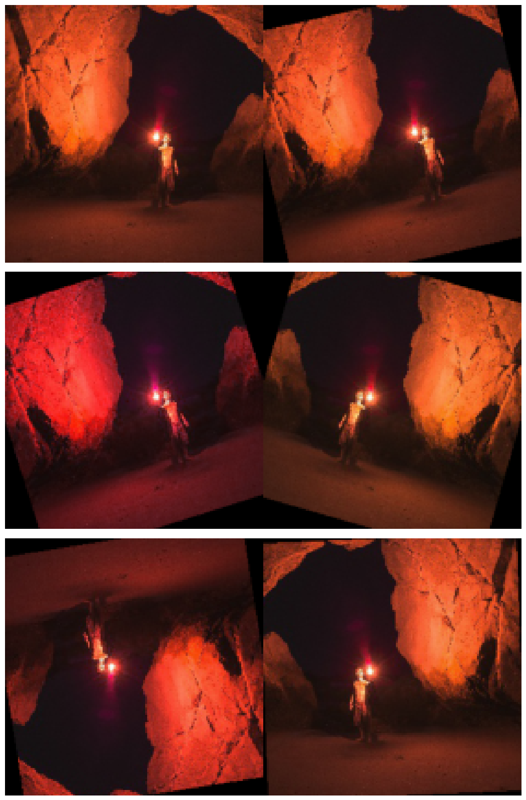


Figure 6. Image augmentation for the training dataset, including random flipping, color jittering, resizing, and image random rotation.

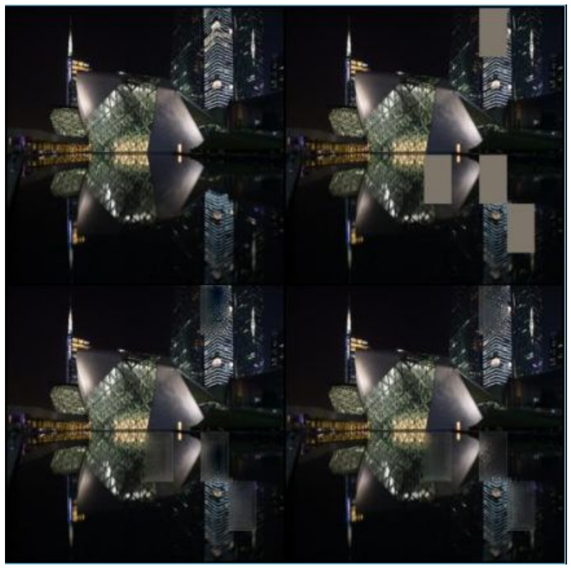


Figure 7. Image result using mask and image from the ICME Challenge validation dataset (EC). Bottom left is after 100 epochs, while bottom right is after 150 epochs.

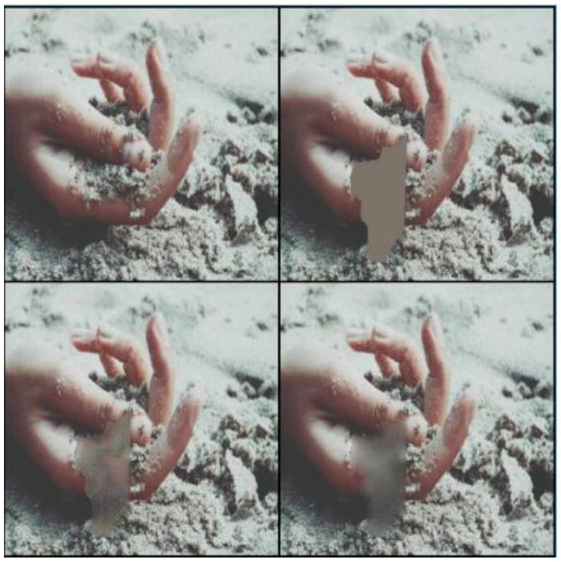


Figure 8. Image result using mask and image from the ICME Challenge validation dataset (OR). Bottom left is the output after 100 epochs, while bottom right is after 150 epochs.

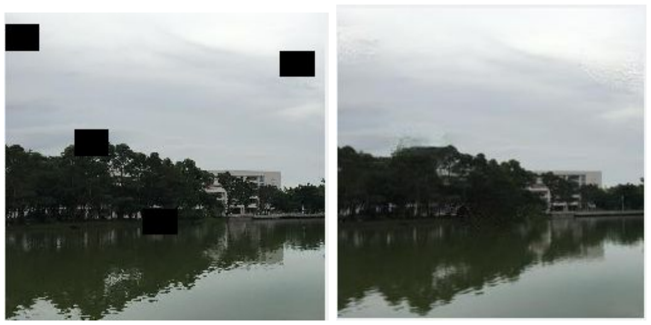


Figure 9. Image result from the test set provided from Canvas from "Final Test Images.zip".

Table 2. Performance Metrics Averaged over 1750 Random Images

L_{TV}	0.678908
$L_{perceptual}$	0.057417
L_{style}	2.450222
L_{total}	295.4533
PSNR	19.37373
SSIM	0.927627

V. Conclusion

In this work, we implemented partial convolutional layers to replace convolutional layers in a U-Net like architecture. By generating more irregular shapes of masks and adding them to our training process, the results have shown that the image inpainting can be applied on irregular shapes of holes. The high value of SSIM values indicate the high similarity between the impaired images and the original images. With the image augmentation for both the training dataset and the masks, the image impairment has shown a significant improvement compared to the original given dataset.

We could improve upon this model by training it for more epochs or switching the neural network model. For object removal tasks, we could try training on more irregularly shaped holes, where we could use a model tracking or object tracking algorithm to help generate masks of an object to remove. Then, we could see how well the results are with this technique. In addition, we could train our model on additional datasets to improve the performance of the model and test the model. Since the test set only had 70 ground truth images, the results may not be entirely accurate.

References

[1] Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., Efros, A.A.: Context encoders: Feature learning by inpainting. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2536{2544 (2016)

[2] Yang, C., Lu, X., Lin, Z., Shechtman, E., Wang, O., Li, H.: High-resolution image inpainting using multi-scale neural patch synthesis. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). vol. 1, p. 3 (2017).

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

[4] Yu, J., Lin, Z., Yang, J., Shen, X., Lu, X., Huang, T.S.: Generative image inpainting with contextual attention. arXiv preprint arXiv:1801.07892 (2018).

[5] Liu, Guilin & Reda, Fitsum & Shih, Kevin & Wang, Ting-Chun & Tao, Andrew & Catanzaro, Bryan. (2018). Image Inpainting for Irregular Holes Using Partial Convolutions.

[6] Sundaram, N., Brox, T., Keutzer, K.: Dense point trajectories by gpu-accelerated large displacement optical flow. In: European conference on computer vision. pp. 438{451. Springer (2010).

[7] Wei, Bob, Inpainting Partial Convolution, https://github.com/bobqywei/inpainting-partial-conv . (Accessed: May 15, 2020).

[8] Shinagawa, Seitaro, Chainer Partial Convolution, https://github.com/SeitaroShinagawa/chainer-partial_convolution_image_inpainting/blob/master/ . (Accessed: June 6, 2020).

[9] Gruber, Mathias, PConv-Keras, PConv-Keras/util.py at master · MathiasGruber/PConv-Keras · GitHub. (Accessed: June 6, 2020).

[10] Li, Sunner, P-Conv, https://github.com/SunnerLi/P-Conv. (Accessed: June 6, 2020).

[11] Barnes, C., Shechtman, E., Finkelstein, A., Goldman, D.B.: Patchmatch: A randomized correspondence algorithm for structural image editing. ACM Transactions on Graphics-TOG 28(3), 24 (2009).

[12] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z.,Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV) 115(3), 211{252 (2015). https://doi.org/10.1007/s11263-015-0816-y

[13] Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedi-cal image segmentation. In: International Conference on Medical image computing and computer-assisted intervention. pp. 234{241. Springer (2015)

Hui-Ying Siao received her M.S. in physics from National Cheng Kung University, Taiwan. She is currently pursuing her master's degree in Electrical and Computer Engineering at University of California, Davis. Her current research focuses on deep learning assisted automatic detection on 2D materials.

Melody C. Tao received her B.S. in Computer Engineering from the University of California, Davis in 2019. She is currently finishing up her M.S. degree in Electrical and Computer Engineering from the University of California, Davis, in 2020.

Vivienne Chiang is a MS student in Computer Engineering at the University of California, Davis. She has a BS degree in Information and Computer Science from the University of California, Irvine.

Yuexin Li received her B.S. in Computer Engineering and Electrical Engineering from the University of California, Davis in 2020. She is now in her first quarter of M.S. degree in Electrical and Computer from the University of California, Davis.