Journal of Science and Technology
ISSN: 2362-1443 (Print) 3059-9490 (Online)
2024, Vol. 4, No. 2 (Special Issue), 105-110

Image Reprocessing of Occluded Object

Aakrit Dongol
Pulchowk Campus, IOE, TU
Lalitpur, Nepal
077bct002.aakrit@pcampus.edu.np

Biraj Kumar Karanjit
Pulchowk Campus, IOE, TU
Lalitpur, Nepal
077bct020.biraj@pcampus.edu.np

Krishala Prajapati
Pulchowk Campus, IOE, TU
Lalitpur, Nepal
077bct038.krishala@pcampus.edu.np

Nishchal Acharya
Pulchowk Campus, IOE, TU
Lalitpur, Nepal
nishchal4feb@pcampus.edu.np

Santosh Giri*
Pulchowk Campus, IOE, TU
Lalitpur, Nepal
santoshgiri@pcampus.edu.np

*corresponding author

Article History:
Received : 29 July 2024
Revised: 8 August 2024
Accepted: 18 Nov 2024

Keywords— Occlusion, Image Reconstruction, Convolutional Neural Networks (CNNs), Inpainting, Partial Convolution, Mean Opinion Score (MOS), Image Restoration

Abstract— **The occlusion in images, where parts of an object in an image are hidden or obstructed by other elements, often hampers their interpretability and utility in various applications. This paper aims to address the challenge of occlusion using a learning approach. This paper proposes a technique that uses Convolutional Neural Networks (CNNs) to inpaint (reconstruct) missing or occluded regions within images with the help of sur- rounding information of the occluded area. The model discussed in the paper was trained on the Place4Net dataset, which focuses on different scenes such as urban, rural, indoor, and outdoor environments. Our method aims to accurately restore occluded regions, thereby recovering the visual information lost due to occlusion. The typical convolution operation, which covers all the pixels of the image, is slightly modified and is termed Partial Convolution, where features are extracted from the desired (non- occluded) regions. The effectiveness of our method was evaluated through user studies, where participants provided feedback on the reconstructed images. The model in this paper was tested on the old images that were collected from the people in the vicinity, and the collected feedback, quantified using a Mean Opinion Score (MOS), indicated that the majority of participants found the restored images to be of good visual quality, demonstrating the utility and robustness of our approach.**

I. Introduction

Accurately identifying and interpreting objects within the images is important in applications such as autonomous systems and surveillance, which utilize computer vision and image processing. The main challenge in these applications is the presence of occlusion, where the objects are partially or entirely obscured by different factors or other objects present in the images. Overcoming occlusion is crucial for the successful deployment of technologies. So, this paper emerges from the necessity to address this challenge and enhance the effectiveness of object recognition in complex visual scenarios.

In Nepal, the landscape of research in image reconstruction remains relatively sparse. However, for different tasks, individuals rely on software applications like Adobe and its tools (Photoshop, GIMP, etc) to address the occlusion challenges. For occlusion removal, Photoshop utilizes the techniques such as ”inpainting” or ”content-aware fill”, while GIMP uses ”Poisson Editing”. These tools rely on traditional methods for filling in the missing parts. They cannot handle scenarios involving complex or significant occlusion which makes them ineffective leading to a need for a better approach.

In the modern era, various GAN-based models have been developed for tackling the challenges of occlusion but the Partial-CNN approach that we will be utilizing in our paper seems best among all. Non-GAN-based models (Partial CNN models) tend to produce inpainted images with better structural coherence and context preservation. They are often more effective at handling complex textures, occlusions, and irregular patterns. This approach frequently yields superior preservation of original features and intricate details in the resultant image.

Hence, this paper tries to assist in the image reprocessing of occluded objects using deep learning techniques.

II. RELATED WORK

In the last decade, various approaches for removing occlusion from an image have been developed. A simple approach where no machine learning is used has been developed which moves the pixel information from neighboring pixels to the target area using a distance field technique [3] for image inpainting. However, this approach could only handle small occluded objects with simple textures. A method called PatchMatch [11] was developed which improves the algorithm by using faster patch searching algorithms like in patch-based methods such as [12]. However, for real-time applications, this approach was still not fast enough and the patch selected is also random and does not match with the surroundings.

To overcome the shortcomings of traditional approaches, deep learning approaches have been developed. In the deepest learning related to images, CNN(Convolutional Neural Network) based models are widely used due to their ability to capture features in an image. In these methods, holes are initialized with values like a mean of pixel values of ImageNet [13] and these values are passed through a convolutional network. Content Encoders [14] take the 128x128 image with a 64x64 hole and convert it into a low dimensional feature map and then using a decoder, the feature map is converted to a 64x64 sized image. Yang et al. [4] take the result from Encoders(latent space) as input and then use the information extracted from the latent space of non-hole(valid) regions to fill the invalid regions. Song et al. [15] utilize a network that refines a blurry initial hole-filling result and then it is iteratively replaced with the patches from the closest regions that are not damaged (or do not have holes) in the feature space. This iterative approach helps to improve the quality of the filled-in regions. Li et al. [16] and Izuka et al. [5] add global and local discriminators to the Content Encoders and then Iizuka et al. [5] apply Possion blending for post-processing. The post-processing of Iizuka et al. [5] was replaced by a refinement network that uses contextual attention layers in Yu et al. [17].

Yeh et al. [18] analyze the latent space to find the closest encoding to the corrupted image, and the result is then used as the condition for the output of a hole-filling generator. Based on Ulyanov et al. [19], a corrupted image can be completed by the network based on the structure of its generative network, and it does not require further external dataset training. However, in this method, every image requires a separate set of hyper-parameters.

In this research, we aimed to produce satisfying results in a single feed-forward pass. We applied masked and re- weighted convolution operations. Harley et al. [6] also utilized this method for semantic segmentation with a soft attention mask. Sparsity-invariant CNN with re-weighted convolution and max-pooling was utilized by Uhrig et al. [10] for updating masks for depth completion. Ren et al. [9] used Shepard convolutional layers for image inpainting, where the same kernel is applied for both feature and mask convolutions.

III. METHODOLOGY

A. Data collection and Pre-processing

To train and validate the image inpainting model, a dataset containing the original images and corresponding binary masks(to highlight the occluded region of the original image) was used. The original images were collected from Place4Net and the binary masks were created by generating black-filled lines, circles, and ellipses of random sizes at random positions on a white background using the OpenCV library in Python, For lines, the size of each line was set to 20 pixels and lengths were randomly set within the range of image window i.e. 512X512. For circles, the center(h, k) was set randomly within the image window size i.e. (512X512) and the radius was randomly set within the range of 30 to 50 pixels in OpenCV using ’randint’ function of Python. For an ellipse, center(h,k), major axis length(a), and minor axis length(b) were randomly set within the range of the image window i.e. 512X512. Each of the circle generation and line generation was done for two iterations and the ellipse generation process was done for nine iterations. This basically means the algorithm to generate masks generates two circles, two lines, and nine ellipses in each binary mask image. The dataset collected from Places4Net consisted of scene diversity (different types of scenes such as urban, rural, indoor, outdoor, etc.) to ensure detailed inpainting results. The Place4Net dataset had 24 classes. Since this paper does not deal with image classification, the images were selected randomly from each class, for training and validation purposes.

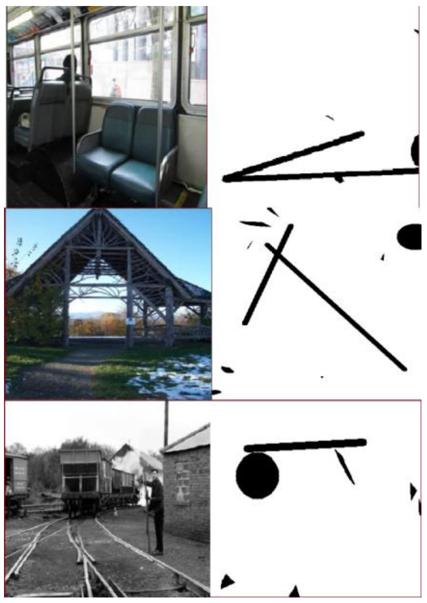
Fig. 1. Training Dataset(The left panel depicts the original image, while the right panel illustrates the corresponding binary mask)

Here, the original image is the image to be reconstructed and the binary mask points out the occluded region. The occluded region is represented by black colored pixels (represented by 0 value in a jpeg image) and the valid pixels (represented by 255 value in a jpeg image) are represented by white pixels.

The model is designed to take images with a height and width of 512 pixels each. Moreover, all the images were converted to tensor format, which is a multi-dimensional array used to represent data(images in this paper) in a format that can be easily processed by a deep learning model. Therefore, as part of image pre-processing, both the original images and the binary masks were resized to the appropriate dimensions of (512 pixels x 512 pixels).

B. *Implementation of Model and Training*

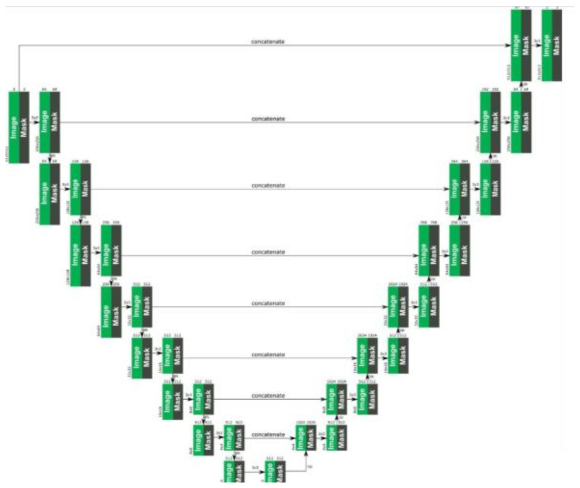

Fig. 2. Diagarm of UNET Architecture

The Convolutional Neural Network architecture used was U-Net architecture [8]. which was mainly developed biomedical image segmentation of biomedical image data.

A UNet architecture is a fully convolutional network(FCN). FCN is a type of neural network that consists of convolutional layers without fully connected layers. It consists of 2 parts. They are the contracting path and the expanding path.

The contracting path consists of encoder layers and the expanding path consists of decoder layers. The encoder layer consists of a convolution process that captures contextual information within the image, reduces the spatial dimension, and increases the depth or color channels. The decoder layer consists of a transpose convolution process (upsampling, skip connection, and normal convolution process) which increases the spatial dimension and reduces the depth or color channels.

Skip connection is used to concatenate the feature extracted from the contracting path to the expanding path. In deep FCNs, the feature extracted in the encoder layer might get lost while image reconstruction in the decoder layer, can degrade the performance of the model. Hence, to tackle this problem, the feature extracted in each encoder layer is concatenated with the inputs of the decoder layer.

In this paper, a small change was made to the UNet architecture. Instead of performing a normal convolution process in each convolution layer, partial convolution was performed. The partial convolution works as follows:

$$x_0 = \begin{cases} W^T (X \odot M) \frac{sum(pixels)}{sum(M)} + B & \text{if } sum(M) > 0 \\ 0 & \text{otherwise} \end{cases} \qquad (1)$$

where,

- W is the weight of the convolutional filter(kernel values) and B is the corresponding bias.
- X is the original image containing the damage which needs to be reconstructed.
- M is the binary mask of the corresponding original image.
- $\odot$ represents element-wise multiplication
- pixel is a tensor with the same shape as M but with all
- elements (pixel elements) being 1
- $\frac{sum(1)}{sum(M)}$ represent the scaling factor
- x_0 is the output
- W^T is transpose of W

Here, instead of performing convolution on the original image(X), the convolution is performed on the masked image created by performing element-wise multiplication between M and X ($X \odot M$).

Partial convolution enables the feature extraction of the valid regions only without extracting the features of the occluded regions, which is not needed. This results in improved efficiency of the inpainting algorithm.

Moreover, there is a scaling operation performed on the output of the partial convolution with the scaling factor of sum(1)/ sum(M). If a convolution of a certain layer has more invalid or masked regions, then the output might result near the value of 0. So, to tackle this problem, we multiply the output of convolution by sum(1)/sum(M). Here, sum(1) is constant but sum(M) decreases when the invalid or masked region in the convolution process increases. This helps to adjust or compensate the output where a significant portion of the input($X \odot M$) is masked.

After each partial convolution process on the masked image, some invalid regions of the image will be filled. Hence, the mask for the next partial convolution (next layer) needs to be updated. It is updated as follows:

$$m_0 = \begin{cases} 1 & \text{if } \sum M > 0 \\ 0 & \text{otherwise} \end{cases} \qquad (2)$$

where,

m_0 represent updates mask

The Neural Network consists of 16 layers, where 8 are of encoder and the remaining 8 are of decoder region. The details of each layer are discussed in Table I.

In the contracting path, the encoder layers use convolution operations. The convolution process contains specific stride values such that the spatial dimension of the output decreases, without the need of pooling layers. Here, stride value means the step size taken by the kernel during the operation layers in both directions of the 2D image. In the initial layer, the kernel size is 7x7. As the spatial dimensions are reduced, the kernel size of smaller

TABLE I. PARTIAL CONVOLUTION UNET ARCHITECTURE

Layer				HyperParameters		
	Process	**Kernel Size**	**Output Channels**	**Stride Value/ UpFactor**	**Batch Normalization**	**Activation Function**
1	PConv1	7X7	64	2	No	ReLU
2	PConv2	5X5	128	2	Yes	ReLU
3	PConv3	5X5	256	2	Yes	ReLU
4	PConv4	3X3	512	2	Yes	ReLU
5	PConv5	3X3	512	2	Yes	ReLU
6	PConv6	3X3	512	2	Yes	ReLU
7	PConv7	3X3	512	2	Yes	ReLU
8	PConv8	3X3	512	2	Yes	ReLU
9	Upsample		512	2	-	-
	Concat(PConv7)		512+512		-	-
	PConv9	3X3	512	1	Yes	LeakyReLU(0.2)
10	Upsample		512	2	-	-
	Concat(PConv6)		512+512		-	-
	PConv10	3X3	512	1	Yes	LeakyReLU(0.2)
11	Upsample		512	2	-	-
	Concat(PConv5)		512+512		-	-
	PConv11	3X3	512	1	Yes	LeakyReLU(0.2)
12	Upsample		512	2	-	-
	Concat(PConv4)		512+512		-	-
	PConv12	3X3	512	1	Yes	LeakyReLU(0.2)
13	Upsample		512	2	-	-
	Concat(PConv3)		512 + 256		-	-
	PConv13	3X3	256	1	Yes	LeakyReLU(0.2)
14	Upsample		256	2	-	-
	Concat(PConv2)		256 + 128		-	-
	PConv14	3X3	128	1	Yes	LeakyReLU(0.2)
15	Upsample		128	2	-	-
	Concat(PConv1)		128 + 64		-	-
	PConv15	3X3	64	1	Yes	LeakyReLU(0.2)
16	Upsample		64	2	-	-
	Concat(Input)		64 + 3		-	-
	PConv16	3X3	3	1	No	

dimensions is used(5x5 and 3x3), as shown in Table I. This ensures that at each layer, the kernel size is smaller than the spatial dimensions of the corresponding input. Additionally, batch normalization (BN) is used in the hidden layers and the ReLU activation function is used for activation. BN normalizes input values of the hidden layers using its mean and variance to prevent Internal Covariate shifts. When the input(not normalized) for the hidden layers changes, hidden layers try to adapt to the new distribution which makes the convergence process inefficient and slow. Thus, applying the BN process to the hidden layers helps prevent the Internal Covariate Shift. ReLU activation is explained below:

$$\text{output} = \begin{cases} \text{input} & \text{if input} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

In this paper, batch normalization is done before activation.

In the expanding path, the decoder layers use transpose convolution. The transpose convolution consists of UpSampling (to increase the spatial dimension of output), skip connections (to retain the features extracted in encoder layers), and partial convolution(to decrease the number of color channels or depth). Similar to the contracting path, BN is used in the hidden layers before the Leaky ReLU activation function with a negative slope of 0.2. Leaky ReLU is explained below:

$$\text{output} = \begin{cases} \text{input} & \text{if input} > 0 \\ \text{input} \times 0.2 & \text{otherwise} \end{cases} \tag{4}$$

IV. EXPERIMENTAL SETUP

The model was implemented using PyTorch deep learning library and it was trained on V100 GPU and A100 GPU of Google Colab Pro with memory 16GiB.

The weights of the model were adjusted using the Adam optimizer, targeting only the parameters that require gradients. This approach ensures that the optimizer updates the convolutional kernels for the image (partial convolution) without changing or updating the kernels for the calculation of sum(M).

Here, 5 types of losses were incorporated according to [1]. They are hole loss, valid loss, perceptual loss, style loss, and total variational loss.

- Hole Loss: Mean Absolute Error of the restored occluded pixels.

$$HoleLoss = (1 - M) \odot (Iout - Igt) \tag{5}$$

- Valid Loss: Mean Absolute Error of the restored non- occluded pixels.

$$ValidLoss = (M) \odot (Iout - Igt) \tag{6}$$

- Perceptual Loss: Mean Absolute Error between Output Image and Ground Truth after Feature Extraction through pre-trained VGG-16 Net.

$$PerceptualLoss = FeatureMap(Iout) - FeatureMap(Igt) \tag{7}$$

- Style Loss: Mean Absolute Error between Output Image and Ground Truth after Feature Extraction and computing Gram Matrix.

$$\begin{aligned} StyleLoss &= GramMat(FeatureMap(I_{out})) \\ &- GramMat(FeatureMap(I_{gt})) \end{aligned} \tag{8}$$

- Total Variational Loss: Mean Absolute Error in the region of 1-pixel dilation.

$$\begin{aligned} TotalVariationalLoss &= (Iout(i,j+1) - Iout(i,j)) \\ &+ (Iout(i+1,j) - Iout(i,j)) \end{aligned} \tag{9}$$

where,

M represents the values of mask pixels.

Igt and Iout represent ground truth and output images respectively.

i,j are all the corresponding pixels of the output and ground-truth images.

The final loss was computed as the weighted sum of each loss. The final loss according to [1] is :

$$TotalLoss = ValidLoss + 6 * HoleLoss + 0.05 * PerceptualLoss + 120 * StyleLoss + 0.1 * TotalVariationalLoss \tag{10}$$

42000 images from the dataset were used for training and 4200 datasets were used for validation. The parameters used for training the model are epochs(55), batch size(7), and learning rate(0.0002).

V. RESULTS

A. *Training and Validation*

If a convolution of a certain layer has more invalid or masked regions, then the output might result near the value of 0. So, to tackle this problem, we multiply the output of convolution by sum(1)/sum(M).

The model was trained on a dataset containing 42000 images (original and its corresponding masked image) and validated using 4200 images. The graph showing the loss during training and validation for each epoch is shown in the figure given below.

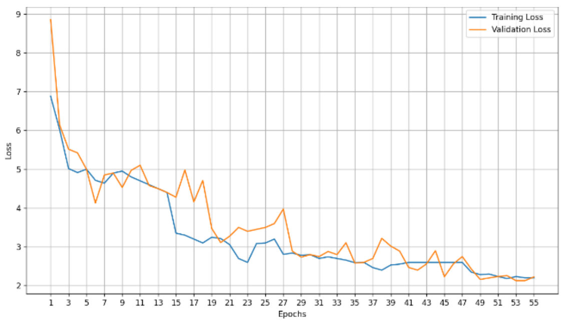

Fig. 3. Fig. 3. Training and Validation losses over Epochs

From the graph, the following things are concluded:

- A small change was made to the architecture of the model during training(around 14-15 epochs). The outputs of the model were scaled by sum(pixels)/sum(M). As mentioned in the methodology section, this is done to overcome the problem of the output image containing a value near 0 in the case of a heavily masked region. So to overcome this we multiplied it with sum(pixels)/sum(M) in equation 1 to scale the result accordingly. The sudden spike in the validation loss curve might be due to this small modification of the model done during the training process.
- With the significant decrement of the training loss at the end, this suggests that the model has not under-fitted.
- The training loss and validation loss were nearly the same at the end of the training. This suggests that the model has not over-fitted.

B. *Testing*

To test our model, 59 sample images were used, 10 local images were obtained from people living nearby, and the remaining 49 images were obtained from the Internet. Evaluation of the test data set was done by receiving feedback from people at 5 levels. These are: Excellent, Very Good, Good, Needs Improvement, and Poor. These grades were converted to a numerical grade out of 5. The grades of each data set were then averaged. No loss function or precision function is used in the evaluation because ground truth images are required for the calculation of these metrics. Since the images collected for testing were not identical to their ground-truth counterparts, a feedback-based approach was proposed.

The tables showing the feedback we obtained on the

TABLE II. FEEDBACK ON LOCAL DATASETS

USER	FEEDBACK
User 1	Overall, the system seems to perform well, but to me, it seems output becomes hits or miss when there is ink-like smudge on the image like seen in sample 3 and sample 9.
User 2	Good! Needs improvement for Sample 9 as the damages are not removed completely.
User 3	Impressive! Improve the application to make it work for more damaged images like Sample 9.
User 4	The outputs are impressive. But still requires improvement for a few images.
User 5	Great work! Some sample images require improvement, work on them.
User 6	Most of the images have been processed very properly and most of the creases have been removed. Sample 9 needs further processing to remove the blemishes.
User 7	Large occlusions were not removed. But it can be improved. Overall the outputs were good for small sized occlusions.
User 8	Good overall Needs improvement in software to improve the output of sample 8 and sample 9.
User 9	Everything seems good. Sample 9 is more damaged compared to other images so need to improve the software to restore that image

tested local dataset (overall system) from 10 users and the average rating on each local dataset (sample images) are given below in Table II and Table III respectively:

TABLE III. AVERAGE OF RATING OBTAINED ON EACH SAMPLE

SAMPLE NO.	AVERAGE RATING OF 10 USERS
1	4.55
2	4.5
3	4.35
4	4.7
5	4.55
6	4.7
7	4
8	4.2
9	2.6
10	3.85

VI. CONCLUSION

In this paper, we have discussed the challenges that arise in image processing due to the presence of occluded objects within the images and to overcome these challenges, the partial convolution-based UNET model is trained successfully using supervised learning to handle the images with occluded objects. The model described in the paper has learned from the dataset, without overfitting and underfitting. Additionally, the feedback-based evaluation of the testing datasets is quite impressive. With the development of a user-friendly interface and deep learning model, the developed application can re- construct the images by removing the occluded object. This application demonstrates high effectiveness in reconstructing images with occluded objects of small size, specifically when the occluded region constitutes less than 20 percent of the total image area. However, it does not guarantee excellent output for images where the occluded region exceeds 40 percent of the total image area, indicating a limitation in handling larger occlusions.

REFERENCES

[1] Guilin Liu. et al. Image inpainting for irregular holes using partial convolutions. NVIDIA Corporation, page 23, 2018.

[2] Nermin Mohamed Fawzy Salem. A survey on various image inpainting techniques. Future University, 2:19, 2021.

[3] Telea A. An image inpainting technique based on the fast marching method. Journal of graphics tools, 2004.

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

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

[6] Derpanis-K.G. Kokkinos I. Harley, A.W. Segmentation-aware convolu- tional networks using local attention masks. IEEE International Confer- ence on Computer Vision (ICCV), 2:7, 2017.

[7] Weights & Biases. Introduction to image inpainting with deep learning, 2023.

[8] O. Ronneberger, P. Fischer, and T. Brox, ”U-Net: Convolutional Net- works for Biomedical Image Segmentation,” presented at the MICCAI, 2015.

[9] Xu-L. Yan Q. Sun W. Ren, J.S. Shepard convolutional neural networks, 2015.

[10] Schneider-N. Schneider L. Franke U. Brox T. Geiger A. Uhrig, J. Sparsity invariant cnns, 2017.

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

[12] Essa-I. Bobick A. Kwatra N. Kwatra, V. Texture optimization for example-based synthesis. In: ACM Transactions on Graphics (ToG)., 24:795–802, 2005.

[13] Deng-J. Su H. Krause J. Satheesh S. Ma S. Huang Z. Karpathy A. Khosla

[14] A. Bernstein M. Berg A.C. Fei-Fei L. Russakovsky, O. Imagenet large scale visual recognition challenge. International Journal of Computer Vision (IJCV), 115:211–252, 2015.

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

[16] Yang-C. Lin Z. Li H. Huang Q. Kuo C.C.J. Song, Y. ”image in- painting using multi-scale feature image translation. arXiv preprint arXiv:1711.08590, 2017.

[17] Liu-S. Yang J. Yang M.H. Li, Y. Generative face completion. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 1:3, 2017.

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

[19] Chen-C. Lim T.Y. Hasegawa-Johnson M. Do M.N. Yeh, R. Semantic image inpainting with perceptual and contextual losses. arXiv preprint arXiv:1607.07539, 2016.

[20] Vedaldi-A. Lempitsky V. Ulyanov, D. Deep image prior. arXiv preprint arXiv:1711.10925, 2017.