GROUP 8 - IMAGE INPAINTING WITH GENERATIVE ADVERSARIAL NETWORK

Chia-Wei Hsieh $A59004366$*, Chin Lee* $A59005092$*, and Hsin-Chien Chang* $A59005404$

Department of Electrical and Computer Engineering at UC-San Diego

ABSTRACT

For the challenge of inpainting missing parts in an image, recent deep learning-based techniques have demonstrated promising results. In this paper, we designed a deep neural network to deal with the task of image inpainting. The goal of this task is for the model to gain whole image understanding and generate plausible results for missing parts in an image. Our model has a GAN structure with a generator and a discriminator. For the generator part, we designed architectures similar to the UNet and ResNet. For the discriminator, we used simple convolutional neural networks in combination with feed-forward neural nets. We trained our network with loss functions that incorporate pixel-wise reconstruction loss and adversarial loss. The model is trained on the COCO dataset provided by Microsoft. Results demonstrate that the models are capable of generating missing patches and provide useful applications.

***Index Terms*—** generative adversarial network, image generation, UNet, ResNet

1. INTRODUCTION

The task of filling empty pixels in a picture, also known as Image Inpainting, is crucial in computer vision. Its application includes object or text removal, automatic modifications of images and videos, and computational photography [1, 2, 3]. The most challenging aspect of image inpainting is creating visually realistic and semantically believable pixels patches for missing parts in the image. Early attempts sought to overcome the problem by direct inferencing from surrounding regions. The method typically copies or samples patches from the background to propagate through the missing hole on the image [1, 4]. However, they fail when encountering unique image contents such as complicated, non-repetitive structures (faces, objects). Therefore, many NN-related studies come out to deal with the task.

In this project, we designed a deep generative model for the image inpainting task. Following the general structure of GAN which consists of a generator and a discriminator. A generative convolutional network with an encoder-decoder structure can preserve high-level recognition and global structure of images, as well as achieving with low-level pixel synthesis. Therefore, we designed three kinds of auto-encoder models for the generator. One has a ResNet-like architecture [5], another is a UNet-based model, and the other is the combination of the previous two. Each model is then jointly trained with the discriminative networks to improve consistency between generated and original images. The input to our algorithm is an broken image. We then use a Generative Adversarial Network to output a predicted residual map which contains the missing part of the input. And the code and results are sharing with the course of ECE 285.

2. RELATED WORK

Existing image inpainting researches can be categorized into two different types. Traditional diffusion-based or patch-based approaches make up the first group. The second group uses statistical or machine learning strategies to handle the inpainting problem, such as training deep convolutional neural networks to forecast the missing pixels.

2.1. Diffusion and Patch-based Approaches

Traditional diffusion or patch-based techniques, for example, often apply various mathematical algorithms on surrounding pixels to transfer information to the missing regions [6, 4, 7]. Patch-based methods often search through the undamaged part of the image for matching replacement patches. Diffusion-based approaches usually smoothly propagate image content from the boundary to the interior of the missing regions. These methods are effective for uniform textures but have limitations when applied to more complex images such as portraits or photographs. A bidirectional patch similarity-based method [8] is proposed to tackle this issue, but the computation is expensive for patch similarities and its practical use is thus limited.

2.2. Deep Learning and GAN

Deep learning and GAN-based models show promising results in the image inpainting tasks. In particular, Autoencoders are widely used as the generators in this architecture. Different structures of Autoencoders also have crucial effects on texture synthesis as information flows differently through the information bottleneck [9, 10]. The framework usually consists of an encoder and a decoder. The encoder learns an abstract representation of a set of data, and the decoder tries to reconstruct the original input from that representation. The model is thus forced to learn the important properties in the input data. Various approaches such as Skip Connection and Residual Connections are implemented to improve the gradient flow and quality of the result [11, 12, 13]. Early works trained Convolutional Neural Networks to inpaint smaller regions [14, 15]. GAN (Generative Adversarial Networks) [16]

architectures provide a substantial performance boost to texture generation for CNNs. Various GAN-based architectures are proposed in the past few years. Context Encoders [17] experimented with generative loss and adversarial loss training on much larger missing patches. Its result shows incorporating the adversarial loss improves the overall model performance. Recent work by Iizuka et al. [18] incorporates both global and local context discriminators. The global discriminator ensures the entire image is coherent, while the local discriminator only focuses on the area around the generated pixels. Sharing the similar idea of utilizing both global and local information, Demir et al. [19] applied PatchGAN and global GAN (G-GAN) to the image inpainting task. PatchGan uses a sliding window for the discriminator to examine each patch of the image, ensuring the local continuity. The global discriminator is used to capture the holistic features in the generated image.

3. DATASET

Fig. 1: Example of (a) iconic object images, (b) iconic scene images, (c) non-iconic images, (d) resized coco images, and (e) randomly cropped part.

We obtain the images from Microsoft COCO(Common Objects in Context) dataset [20]. The dataset contains a wide variety of objects and scenes, including iconic object images, iconic scene images, and non-iconic image. As Fig. 1 shows, iconic object and scene images are easily distinguishable, such as an image of a dog, a car, or a forest or desert. Non-iconic images contain several objects in a more complex scene. For the image inpainting task, we generate random patches that can be as large as one-fourth of the original image. The train, validation, and testing datasets each have 118k, 50k, and 40k, instances. We resize the image to 224and set the mean and standard deviation as what PyTorch officially used for normalizing images. The benefit of matching the PyTorch official setting is that we can effectively use those official pre-trained models to design our network. Since the pictures in COCO dataset are complete without any missing, we randomize the cropped size and the start point to create our own broken images for model's input as shown in Fig. 1.

4. METHOD

The challenge of image inpainting lies in synthesizing realistic and semantically plausible pixels for the missing regions while maintaining coherency with the whole image. In this project, we adopt the architecture of generative adversarial network (GAN)[16] which consists of two networks: a generator and a discriminator.

4.1. Generator

Both skip connection from UNet[11] architecture and residual blocks from ResNet[5] pass features from previous layers to later ones and are widely used in the field of computer vision[21, 22, 23, 24, 25, 26]. Therefore, we experiment with both architectures and their combination for our generators.

4.1.1. UNet-based generator

Inspired by Ronneberger etc[11], we design a UNet-based network G_{Unet} for the task. As show in Fig.2, this model has four downsampling blocks as the encoder and corresponding up-sampling blocks as the decoder. Every downsampling block has two convolutional layers stacked, and each followed by a ReLU layer[27]. And the decoder is similar to the encoder except that the single sub-sampling layers are replaced with upsampling layers in each block. To fulfill the characteristic of UNet, several skip connections are established between encoder and decoder to incorporate previous information when upsampling.

4.1.2. ResNet-based generator

We also experiment with ResNet50 [12] as the encoder for our generator G_ResNet. Instead of letting every few stacked layers directly fit a desired underlying mapping, it adds an identity mapping capability and lets these layers fit a residual mapping. To be exact, denoting the desired underlying mapping as $H(x)$, ResNet lets the stacked nonlinear layers fit another mapping of $F(x) := H(x) - x$. The original mapping is recast into $F(x) + x$. The decoder architecture remains the same as the decoder of our UNet-based model.

4.1.3. Generator with the combination of ResNet and UNet

Additionally, we experiment with combining the UNet and ResNet $G_ResNet\&UNet$ to create a ResNet-based encoder with skip connections between encoder and decoder. It inherits the characteristics and benefits of two models. The performance comparison of above three are illustrated in Sec. **??**.

Broken image B cropped out a random size part from ground truth image gt and serves as input to create the task of image inpainting. To make our model G_{Unet} / G_{Resnet} focusing on generating the missing part, all generators predict the missing part as a residual map r. After the generation, we derive our final result by adding r with B. Let G as one of G_{Unet}, G_{resnet} and $G_{ResNet\&UNet}$. Our $output = G(B)+B$. To make the result approach to the ground truth gt, we employed a pixel-wise reconstruction loss to capture the global structural information and other low-frequency information. Also, we calculate the distance between our cropped out result $r^{'}$ with cropped out gt which labeled as $gt^{'}$:

$$L_{L_1} = \sum_{W}\sum_{H}||r + B - gt||_1 + \sum_{W'}\sum_{H'}||r^{'} - gt^{'}||_1 \quad (1)$$

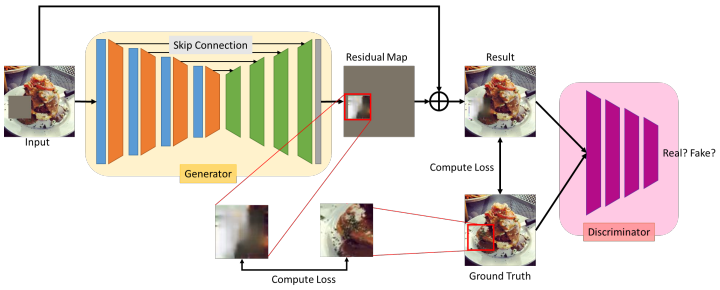

Fig. 2: Concept of training our model

where W and H represent the width and height of the transformed image, $W^{'}$ and $H^{'}$ represent the width and height of the missing part, and r represent $G(B)$.

Also, a binary cross-entropy loss is utilized for the adversarial loss is adopted to encourage G_{Unet} to produce a realistic image:

$$L_{GAN}^{G} = \underset{\mathbf{X}\in p_{gt}}{\mathbb{E}}[log(D(\mathbf{X}))] + \underset{\mathbf{Y}\in p_{r+b}}{\mathbb{E}}[log(1-D(\mathbf{Y}))] \quad (2)$$

where $\mathcal{E}$ is the expectation in binary cross-entropy loss, p_{gt} represents the ground-truth image set, p_{r+b} represents the set of our result, and D represents the discriminator that will be introduced in Sec.**??**. Combining these two, the total loss for our generator model G is

$$arg\min_{G} L_{L_1} + L_{GAN}^{G} \quad (3)$$

Our goal of G is to deceive D to discern our result as real image. Therefore, we try to maximize $log(D(G(y)))$ and minimize $-log(D(G(y)))$.

4.2. Discriminator

We design a discriminator D as a four-block convolution network. Each block includes one convolution layer, a BatchNorm layer, and a leackyRelu layer [28] . After four blocks, a linear layer is applied to output real or fake for the input image. The discriminator is trained to discriminate ground truth as the real image, labeling with 1, and generated result as the fake image labeling with 0. The GAN loss is calculated based on binary cross-entropy as following:

$$L_{GAN}^{D} = \max_{D} \underset{\mathbf{X}\in p_{gt}}{\mathbb{E}}[log(D(\mathbf{X}))] + \underset{\mathbf{Y}\in p_{b}}{\mathbb{E}}[log(1-D(G(\mathbf{Y})+\mathbf{Y}))] \quad (4)$$

where p_b represents the broken-image set as input data. With the above loss function, our discriminator is updated to distinguish the ground truth image and generated image correctly.

Lastly, our final loss function is

$$arg\min_{G_{Unet}} \max_{D} L_{GAN}(G_{Unet}, D) + L_{L_1} \quad (5)$$

5. EXPERIMENTS

5.1. Implementation Detail

We adopt the GAN[16] architecture for our image-inpainting model. Therefore, our network is composed of a generator and a discriminator. The implementation details of both models and the training process are as follow.

The UNet-based generator is designed with four downsampling convolution blocks and corresponding upsampling blocks. Each downsampling convolution block is composed of 2 unchanged size convolution layers and a single sub-sampling convolution layer. All layers contain 3 filters and are followed by ReLU layers. But the strides of former layers are set to 1, and later one is set to 2. The upsampling blocks are similar to the downsampling blocks but replaced the single sub-sampling convolution layer with a upsample layer in each block. Besides, skip connections are built between them to consider the previous detailed information when upsampling. Therefore, a list variable is adopted to store each output from downsampling blocks and later concatenate with the output from upsampling blocks respectively before inputting to upsampling blocks. Additionally, our output channel numbers are designed as 128, 256, 384, and 512 for downsampling blocks and designed as 128 for all upsampling blocks. Therefore, the input channel number of downsampling and upsampling match each other after concatenation. After these blocks, we built an additional upsampling block to generate our three channel RGB result based on the previous features. On other hands, the ResNet-based generator is very similar to the U-Net based generator since both are followed the structure of auto encoder. The encoder of ResNet-based generator is constructed with ResNet50 without average pooling and fully connected layer. The decoder is similar to the UNet-based model which built with four plus one upsampling blocks to generate the original size result image. And for the combination of the two models, we adopt all the setting of those two models.

For the discriminator, we designed a four-block convolution network. Each block includes one convolution layer, a BatchNorm layer, and a leackyRelu[28] layer. Each convolution layer is set with stride = 2 and contains 5 filters. After four blocks, a linear layer is adopted to output real or fake for the input image.

In the training process, we use Adam[29] as the optimizers of generator and discriminator with $\beta_1 = 0.5$ and $\beta_2 = 0.999$. Besides, the learning rate for gradient descent is set as 2e-4 which close to the recommendation of PyTorch for Adam optimizer, 1e-4. Furthermore, we didn't use mini-batch learning and updating our network for every single image. Because one of our losses is calculating each missing part $r^{'}$ compared with corresponding $gt^{'}$ has different size and start point.

5.2. Result

We focus on comparing the architectures of residual blocks and skip connections on the task of image inpainting. These two have similar concepts of passing previous information to the later layers and have been proven to be effective in several subjects of computer vision. The experiment of our three models is shown in Fig.3. Unfortunately, we are unable to run all the epochs and finish the experiments because of the high amount of trainable parameters in the models and

skip/residual connections. Therefore, we show and compare the testing results of each best model of three architectures in Fig.3 and compare them in the same epochs in quantitative evaluation.

5.2.1. Quantitative Evaluation

Because of the expensive computation of our models and the limited GPU resource, our models didn't finish training with the CoCo dataset that includes 118K images. Fortunately, some models still work well when testing after a few epochs even though they haven't obtained their local minimum loss.

We evaluate the quantitative performance of our models with the mean square loss(MSE). For the fairness of comparison, we calculate MSE for our three models in the same training epochs. Since we ran the UNet-based model much earlier than other models, the model has trained longer and reached better performance. Therefore, the best UNet-based model is also included in Table. 1 for further understanding of how good the model could be after a more completed training.

Table 1: Comparison of our three models and the best UNet-based model perform on 5k testing data.

	Mean Square Error	
	filling part	whole image
ResNet-based model	0.8423	0.1920
UNet-based model	0.4678	0.0563
ResNet&UNet-based model	0.5089	0.063
UNet-based model (longer)	0.4532	0.0548

As shown in Table.1, we can observe that the UNet-based model (MSE = 0.4678 for filling parts and 0.0563 for whole images) performs much better than the ResNet-based model(MSE = 0.8423 for filling parts and 0.1920 for whole images). To our surprise, it is even better than the model of the combination with ResNet and UNet which is labeled as ResNet&UNet-based model. However, because we don't have the chance to train them for a long time and the complexity of the UNet-based model is much smaller than the ResNet-based model, we cannot conclude that the UNet works better than ResNet. But we could say that UNet works quite well and converge fast even in a few of training epoch on the large dataset.

5.2.2. Qualitative Evaluation

As shown in Fig.3, we can observe that both the UNet-based model and the ResNet&UNet-based model complete the image in-painting task and got similar results. But the ResNet-based model only changed the color of the blocks near edges and failed to complete the task. We suspect that it's because more and more low-frequency information is lost in the keeping-deepening and shrinking network. It is therefore difficult to recover the missing part. Another reason might be it is too early in the training process and the result might differ if the training continues. In the following section, we will focus on the comparison between the UNet-based model and the ResNet&UNet-based model.

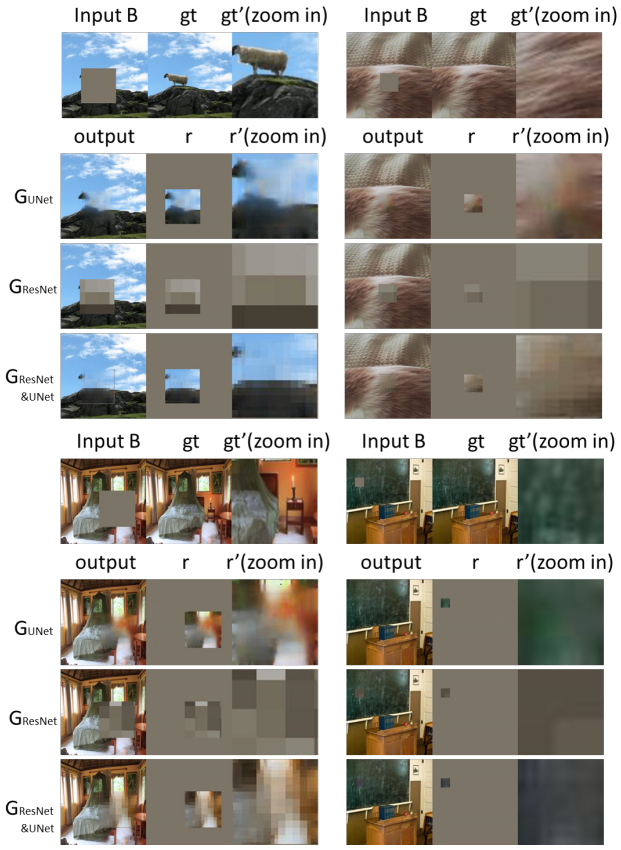

Fig. 3: Testing result of our three models with current best checkpoints respectively, where $r^{'}(zoomin)$ represents and $gt^{'}(zoomin)$ represent the close look of filling part of the prediction and the ground-truth missing part.

The left-upper block in the figure demonstrates one of the usages of image in-painting: removing unwanted objects and filling them with the surrounding scene. Our models successfully remove most of the sheep's body and refill it with the sky and hill. The right-upper block shows that the UNet-based model generates a better filling mask than the ResNet&UNet-based model, especially at the edges of the missing region. The lower two blocks show that the UNet-based model performs better for round or curve objects, while the ResNet&UNet-based model performs better on straight or sharp objects.

6. CONCLUSION

We experiment with three variants of GAN models on the task of image inpainting. Each model has a different encoder: UNet, ResNet, and the combination of the two. The result shows all three models are capable of generating missing patches in the image. Limited by the computing power and time, we did not finish our entire experiment. Even with limited epochs, the model shows promising results, and the difference in performance remains an interesting topic for future research.

7. CONTRIBUTIONS

Chia-Wei Hsieh designed and wrote the code of the models. She is also responsible for the training of $UNet$. **Chin Lee** carried out the training of $ResNet$ the data processing. **Hsin-Chien Chang** conducted the training of the $ResNet\&UNet$ and the data processing. The paper is written by all of the team members.

8. REPLIES TO CRITICAL REVIEWS

- Critiques by group 3

  1. Overall: The overall goal of the project is clear (to fix flawed images using DL). GAN is explained well. The generator architecture is explained clearly. The code was explained clearly.
     Our Response: Thanks.
  2. A suggestion is to mention some specific use cases where one might have a photo/image with a flaw and want to fix it. This was discussed briefly and broadly in the presentation, but a specific, practical example would help the viewer see why this is an important problem to solve.
     Our Response: See Sec. 1 and 2.
  3. Difference between Diffusion and Exemplar traditional methods not clear. From the definition on the slides, they seem like they could be the same thing.
     Our Response: See Sec. 2.1
  4. It wasn't mentioned how much of the COCO data was used to train your model. That would be nice for the viewer to know.
     Our Response: See Sec. 3 and 5.1
  5. What was the training/validation/testing data split used in training the model?
     Our Response: See Sec. 3
  6. How much data was used to train the model?
     Our Response: See Sec. 3 and 5.1
  7. Why did the team opt to use custom square patches to create "image flaw" data instead of the already available segmented areas from the COCO dataset? Does this have more applications or help train the model better?
     Our Response: We designed our model to train on fixing broken patches on the image. Different models can deal with different image inpainting tasks such as irregular shapes.
  8. It was mentioned the model will output if the image is real or fake, but wasn't explained how this is used in the model or project.
     Our Response: See Sec. 4.2, the result is computed with binary cross-entropy and added with generative loss for the training.
  9. It would be nice to see numerical results or a loss/accuracy training curve of the model so one can know how accurate the model is numerically speaking. This would also be useful to see how accurate this method is compared to other methods.
     Our Response: Quantitative results are in Sec. 5.2.
  10. Would be nice to know how many epochs the model was trained.
      Our Response: See Sec. 3 and 5.1. We didn't finish all the epochs in time.
  11. From the results, it seems the model only fixes image flaws where the flawed areas are square and solid colored? This is what the results shown seem to imply. May want to consider clarifying if this is the case or not or show a greater variety of output images.
      Our Response: See Sec. 5.2. The model is able to generate patches with ambiguous shape of desired output. Again, with more training epochs the model may produce better results.
  12. The filled in image areas from the outputs in the results section seem rather blurry, in some outputs more than others (as was mentioned in the presentation). Could add an explanation for why this is the case.
      Our Response: We didn't finish all the epochs in time. With more training epochs the model may produce better results.

- Critiques by group 20

  1. Overall, the presentation and code illustrations are informative and easy to follow. Also, the problem to solve is refreshing and interesting. In my opinion, Group 8 did a good job in explaining what the problem is and what dataset includes by giving informative images. The model and results are illustrated well with diagrams and input/output images.
     Our Response: Thanks
  2. The Literature part is vague and details of other people's work are not included. Also, references are not shown. Giving a big picture of what different approaches to this problem have been developed will make it more informative.
     Our Response: See Sec. 2
  3. For the Result section, it'll be better if the differences between the output and ground truth are

presented with numbers, as losses or accuracies. And a plot of training and validation accuracy will give the audience more information about the training Process.
Our Response: See Sec. 5.2

4. In the Future Work part, it was mentioned that the batch size can only be 1 so that the group would look into finding another model. It is not clear why larger batch sizes cannot be used. And it should be better if more details of the alternative were given since we are approaching the end of this quarter.
Our Response: The models are large with a high amount of trainable parameters. We used datahub for our training, and larger batch sizes may cause errors in the kernel.

- Critiques by group 36

1. The intro was clear and concise. The motivation behind reconstructing images using inpainting was made really clear.
Our Response: Thanks

2. Deep learning is said to be widely used to improve on inpainting, but neither how or why are very well described. Nor are the different deep learning methods described in the background slides, so it makes the choice for the deep learning methods seem arbitrary, or not well thought out. Perhaps explaining the different deep learning techniques would help the reader understand what is going on during the presentation
Our Response: See Sec. 1 and 2. Motivations and various deep learning model are discussed.

3. The differences between traditional computer vision methods are more inferred than stated, by simply explaining how traditional methods work. It would be nice if a deeper explanation of the deep learning was provided
Our Response: See Sec. 1 and 2

4. Traditional methods are mentioned, but there is no clear demonstration of their effectiveness. Although it is meaningful to say that deep learning does better, perhaps a numerical explanation of the differences or even a visual demonstration would be a more effective way of demonstrating the shortcomings of traditional methods
Our Response: We didn't focus on the performance differences between traditional models and deep learning models. We focused more on the three variants of generators we used in our experiments. Perhaps future research can focus on this topic.

5. The discriminator is said to provide scores for the images, but the method of determining the score is a bit unclear'
Our Response: See Sec. 4.2

6. It would be nice to see the numbers supporting the conclusions drawn. For example, the discriminator and decoder are said to get better with time as they try to outperform each other, but no numbers are given to demonstrate this concept. Similarly, the generated images are compared with the ground truth and are said to be blurry. While this is a valuable conclusion, it would also be nice to see the numerical accuracy of the model.
Our Response: See Sec. 5.2. Unfortunately, we didn't finish all the trainings in time, so the results remain blurry and unfinished.

7. A more detailed explanation of the model that is going to be implemented would be nice, along with the reasoning behind not being able to train with more than one batch
Our Response: See Sec. 4 for the details of the models. We could only train with a batch size of 1 because of the limitation of computing power.

9. REFERENCES

[1] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. Graph.*, 28(3):24, 2009.

[2] Anat Levin, Assaf Zomet, Shmuel Peleg, and Yair Weiss. Seamless image stitching in the gradient domain. In *European Conference on Computer Vision*, pages 377–389. Springer, 2004.

[3] Raymond A Yeh, Chen Chen, Teck Yian Lim, Alexander G Schwing, Mark Hasegawa-Johnson, and Minh N Do. Semantic image inpainting with deep generative models. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5485–5493, 2017.

[4] Marcelo Bertalmio, Guillermo Sapiro, Vincent Caselles, and Coloma Ballester. Image inpainting. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 417–424, 2000.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[6] Coloma Ballester, Marcelo Bertalmio, Vicent Caselles, Guillermo Sapiro, and Joan Verdera. Filling-in by joint interpolation of vector fields and gray levels. *IEEE*

transactions on image processing, 10(8):1200–1211, 2001.

[7] Alexei A Efros and Thomas K Leung. Texture synthesis by non-parametric sampling. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1033–1038. IEEE, 1999.

[8] Denis Simakov, Yaron Caspi, Eli Shechtman, and Michal Irani. Summarizing visual data using bidirectional similarity. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[9] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[10] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, Pierre-Antoine Manzagol, and Léon Bottou. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(12), 2010.

[11] O. Ronneberger, P.Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 9351 of *LNCS*, pages 234–241. Springer, 2015. (available on arXiv:1505.04597 [cs.CV]).

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[13] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[14] Rolf Köhler, Christian Schuler, Bernhard Schölkopf, and Stefan Harmeling. Mask-specific inpainting with deep neural networks. In *German conference on pattern recognition*, pages 523–534. Springer, 2014.

[15] Li Xu, Jimmy S Ren, Ce Liu, and Jiaya Jia. Deep convolutional neural network for image deconvolution. *Advances in neural information processing systems*, 27:1790–1798, 2014.

[16] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. In *Proceedings of the International Conference on Neural Information Processing Systems(NIPS)*, 2014.

[17] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2536–2544, 2016.

[18] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. Globally and locally consistent image completion. *ACM Transactions on Graphics (ToG)*, 36(4):1–14, 2017.

[19] Ugur Demir and Gozde Unal. Patch-based image inpainting with generative adversarial networks. *arXiv preprint arXiv:1803.07422*, 2018.

[20] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[21] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[22] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[23] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017.

[24] Fabian Isensee, Paul F Jaeger, Simon AA Kohl, Jens Petersen, and Klaus H Maier-Hein. nnu-net: a self-configuring method for deep learning-based biomedical image segmentation. *Nature methods*, 18(2):203–211, 2021.

[25] Zheng He, Xueli Wei, Kangli Zeng, Zhen Han, Qin Zou, and Zhongyuan Wang. Low-quality watermarked face inpainting with discriminative residual learning. In *Proceedings of the 2nd ACM International Conference on Multimedia in Asia*, pages 1–6, 2021.

[26] Rushi Lan, Haizhang Zou, Cheng Pang, Yanru Zhong, Zhenbing Liu, and Xiaonan Luo. Image denoising via deep residual convolutional neural networks. *Signal, Image and Video Processing*, 15(1):1–8, 2021.

[27] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.

[28] Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. Rectifir nonlinearities improve neural network acoustic models. In *ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.

[29] Diederik P. Kingma and Jimmy Ba. Adam:a method for stochastic optimization. In *The 3rd International Conference for Learning Representations*, 2015.