Image Inpainting with Semantic Context using Deep Generative Models

Garima Gupta
ggupta22@uic.edu

Sai Teja Karnati
skarna3@uic.edu

Shubham Singh
ssing57@uic.edu

Wangfei Wang
wwang75@uic.edu

Abstract

Semantic inpainting is a task where a missing regions of images need to be filled based on the available visual data. It is very challenging especially when the missing regions are large. Here we show the implementation of the model proposed by Yeh et al. [1], where we first generated missing content by utilizing a trained deep convolutional generative adversarial network (DCGAN), and then searched the closest encoding of the corrupt image in the latent space using both the context loss and prior loss. The closest encoding was then passed to the generator used in DCGAN to generate the final inpainted results. We evaluated this model on three datasets, CelebA [2], DTD [3] and Pets [4]. While not showing equivalently good results as the original paper, we did find that this method generated relatively realistic images. We found that this method worked the best on the CelebA dataset and worked better with smaller missing region. We finally compared this method with another similar method Context Encoder (CE) [5], and found that CE tended to generate overly smooth images, but it performed pretty well according to our experiments.

1 Introduction

Image inpainting refers to the task of filling in the missing parts of an image by inferring arbitrary large missing regions based on image semantics [1]. The task is challenging because missing pixels, in many cases, large missing regions, need to be inferred by the surrounding image pixels and high-leveled semantic context. Many methods were proposed to recover the missing regions.

Classical methods are mostly based on single image inpainting that use either local or non-local information. For example, total variation based methods such as the method described by Afonso et al. [6] deconvolutes and reconstructs from compressive observations using total variation regularizations with the advantages of taking into account the smoothness of images. However this type of methods only works well with small missing holes. Many other methods were proposed for filling the missing holes by using similar information on the same image [7, 8, 9, 10]. However, these methods require similar pixels, structures or patches that are not flexible enough for real life applications.

To tackle the task of inpainting for relatively large missing regions, non-local methods were also proposed by searching the external data. This kind of methods such as cutting and pasting semantically similar patch or fetching information from the internet[11, 12] have a major caveat that when test set is dramatically different from the training set, they would fail immediately.

Recently, a semantic inpainting approach called Context Encoder (CE) was proposed by Pathak et al. [5], which is the closest to the method using in our project. Therefore, we will compare

our method with CE in later Section 6. We will also describe briefly how CE works in Section 2. The major difference between the method proposed in Yeh et al. [1] and CE [5] is that deep convolutional generative adversarial network (DCGAN) used in Yeh et al. [1] does not need to train on the corrupted images.

In this proposal, we aim to implement the semantic inpainting method proposed by Yeh et. al [1] which utilized a trained DCGAN model, and then inferred the missing regions by conditioning on the available data. To inpaint the missing regions, we implemented the model where we searched the "closest" encoding of the corrupted image using the context and prior loss, and then we inferred the missing content by passing this encoding through the generative model in DCGAN [1].

2 Related Work

2.1 Generative Adversarial Network (GAN)

Deep generative models were once not favored due to the difficulties of approximating probabilistic computations from maximum likelihood estimation [13]. But with the advent of generative adversarial network (GAN), it overcame these difficulties and leveraged the benefits of piecewise linear units in the generative context, which generally endow the model with a well-behaved gradient [13]. GAN employs a framework corresponding to a minimax two-player game, which it trains a generator, G, and a discriminator D simultaneously [13]. GAN aims to optimize the following value function $V(G, D)$:

$$\min_G \max_D V(G, D) = \mathbb{E}_{\mathbf{h} \sim p_{\text{data}(\mathbf{h})}}[log(D(\mathbf{h}))] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{Z}}(\mathbf{z})}[log(1 - D(G(\mathbf{z})))], \tag{1}$$

where $\mathbf{h}$ is the sample sampled following p_{data} distribution and $\mathbf{z}$ is the random encoding in the latent space; G maps the random vector $\mathbf{z}$ to the image space, and D maps an input image to a likelihood. GAN has many promising applications so far, such as generating realistic photos [14], image-to-image translations [15], photograph editing [16] and image inpainting [1, 5]. When GAN is applied in inpainting, it needs to be constrained by the provided corrupted images; otherwise, it would generate unrealistic images with high probability [1].

2.2 Context Encoder

Convolutional neural networks (CNNs) for ImageNet classification [17] over millions of examples in a supervised setting has great success of generalizations across tasks [5]. However, learning semantic and generalizable features from unlabeled raw images is challenging. Autoencoders became popular approaches to tackle this problem. For example, denoising autoencoders are able to reconstruct the image from local corruptions, and make encoding robust to corruptions [5]. The CE model was proposed by Pathak et al. [5] by using a similar architecture as autoencoders and can be viewed as a denoising autoencoder. The model is a CNN trained to generate missing regions by conditioning on its surroundings. Therefore, unlike DCGAN, training CE takes corrupted images as input.

3 Model

3.1 DCGAN

DCGAN extends the GAN framework by also borrowing the power of deep convolutional neural network. The Generator (G) in DCGAN was implemented as Figure 1. Taking $\mathbf{z}$ from the latent space as the input, we implemented four convolutional layers for G. The first three layers performs a transposed convolution after each step, followed by a batch normalization and ReLu activation. The last convolutional layer used transposed convolution and tanh as activation function. The generated images have the dimension $64 \times 64 \times 3$. The Discriminator (D) takes image with dimension $64 \times 64 \times 3$, and maps it to a scalar probability. D consists of strided convolution, batch normalization and LeakyReLU activations. In our project, we trained our DCGAN model with uncorrupted data.

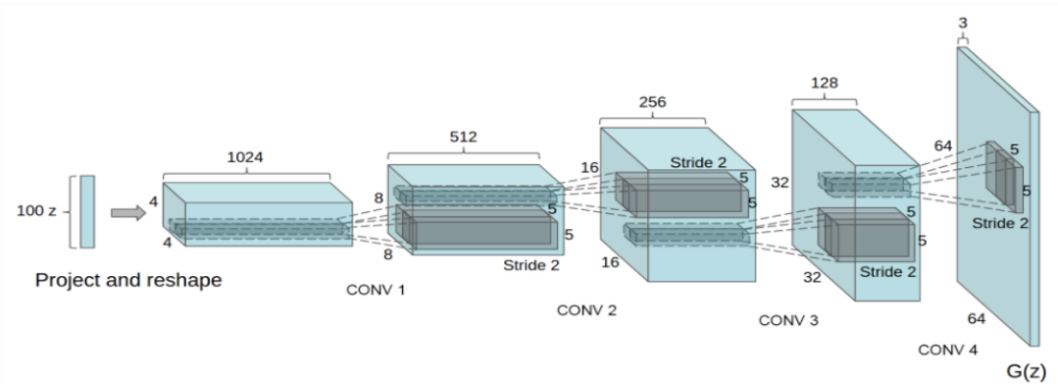


Figure 1: Architecture for generator in DCGAN [18].

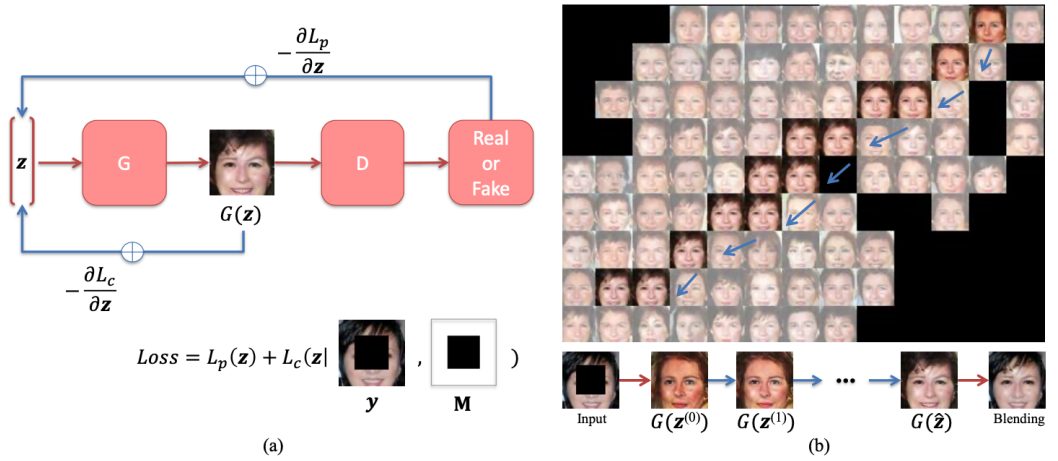


Figure 2: Inpainting framework. (a) After training a DCGAN model, $\mathbf{z}$ on the latent image manifold was iteratively updated based on the designed loss function. (b) Visualization of the latent manifold using t-SNE [19] on the 2D space, the "closest" encoding $\hat{\mathbf{z}}$ was updated using backpropogation, and the final result was generated [1]. M is the mask, $\mathbf{y}$ is the corrupted image.

3.2 Semantic Inpainting

After we trained DCGAN model, we implemented the semantic inpainting model that was proposed by Yeh et al. [1] (Figure 2). In the first step, $\mathbf{z}$ was updated using backpropogation until the "closest" encoding $\hat{\mathbf{z}}$ was reached by minimizing the loss function. Then the generated $G(\hat{\mathbf{z}})$ was used to generate the final inpainting result.

3.2.1 Search the "closest" encoding $\hat{z}$

The main idea is that the reconstruction of corrupted images is constrained to the manifold. Therefore, the model is based on the hypothesis that if G is efficient in its representation, then the image that is not sampled from p_{data} should not be learned [1]. Shown in Figure 2, the "closest" encoding $\hat{\mathbf{z}}$ is updated by optimizing the loss:

$$\hat{\mathbf{z}} = \underset{\mathbf{z}}{\operatorname{argmin}}\{\mathcal{L}_c(\mathbf{z}|\mathbf{y}, \mathbf{M}) + \mathcal{L}_p(\mathbf{z})\} \tag{2}$$

where $\mathcal{L}_c$ is the context loss, $\mathcal{L}_p$ is the prior loss, M denotes the mask, and $\mathbf{y}$ represents the corrupted image.

Context loss compares the difference between the corrupted images and the generated image (G($\mathbf{z}$)).

$$L_c(\mathbf{z}|\mathbf{y}, \mathbf{M}) = \|\mathbf{W} \odot (G(\mathbf{z}) - \mathbf{y})\|_1 \tag{3}$$

where $\mathbf{W}$ is the importance weighting term. To capture the hypothesis that the closer the uncorrupted part of the image is to the mask, the more important those pixels are for inpainting, an importance weighting term W was used. W is taking the form:

$$\mathbf{W}_i = \begin{cases} \sum_{j \in \mathcal{N}(i)} \frac{1-\mathbf{M}_j}{|\mathcal{N}_i|} & \text{if } \mathbf{M}_i \neq 0 \\ 0 & \text{if } \mathbf{M}_i = 0 \end{cases} \tag{4}$$

where $\mathbf{W}_i$ is the importance weight at pixel location i, $\mathcal{N}_i$ is the set of neighbors of pixel i, and $|\mathcal{N}_i|$ represents the cardinality of $\mathcal{N}_i$.

Prior loss takes the same form as that in the GAN loss for training the discriminator which penalizes the unrealistic images. Instead of penalizing pixel-wise differences like context loss, prior loss penalizes based on high-level image feature representations:

$$\mathcal{L}_p(z) = \lambda log(1 - D(G(\mathbf{z}))) \tag{5}$$

where λ is a parameter that balances the two losses L_c and L_p. Yeh et al. [1] demonstrated that $\mathcal{L}_p$ is critical to generate perceptually plausible result.

3.2.2 Inpainting

Using the defined context and prior loss described in equation (3) and (5), the closest $\hat{\mathbf{z}}$ in the latent space was generated using backpropogation (Figure 2 (b)). We then generated $G(\hat{\mathbf{z}})$ using the generator G. The final inpainting result was obtained by overlaying the uncorrupted pixels from the input.

4 Datasets and Masks

In this section, we discuss the image datasets and masks that we use to evaluate our approach. We use the popular CelebA dataset [2], also used in [1] for the image inpainting task. To evaluate the transferability of our approach, we also conduct the experiments on the Describable Textures Dataset (DTD), released by Cimpoi et al. [3] and Oxford-IIIT Pet Dataset (Pets) [4].

We used the `Align&Cropped Images` from the CelebA dataset, as the images were centered and cropped to fill the visible area. We believed that the homogeneity in the images would help in learning effective representation while training the DCGAN model. Although, DTD and Pets datasets contain less number of images as compared to the CelebA dataset, the images within the dataset vary significantly. The diversity within the dataset and evaluation on images from different domain can shed some light into the robustness of our approach.

The details for all the datasets could be found in Table 1.

Dataset	Description	Number of Images
Large-scale CelebFaces Attributes (CelebA) Dataset	A large-scale face attributes dataset containing celebrity images.	202,599
Describable Textures Dataset (DTD)	A collection of textural images in the wild, inspired by the perceptual properties of textures.	5,640
The Oxford-IIIT Pet Dataset (Pets)	A dataset consisting of pet images that have a large variations in scale, pose and lighting.	7,349

Table 1: Table describing the datasets used in the experiments.

We resized all the images to a size of 64×64, and we created a 20×20 rectangular mask placed at the center of the image to generate the corrupted images. We created a smaller mask which is 10×10 in size and a larger one of size 30×30 (referred to as *small square* and *large square* masks, respectively hereinafter). Then, we created a circular mask and a mask that contains 2×2 rectangles placed randomly.

5 Experiments

We trained all our models on an NVIDIA Tesla K80. We train the DCGAN model for 15 epochs and update z to find the closest embedding for 3000 iterations. Due to the lack of an effective quantitative measure that encapsulates the effectiveness of the inpainting results, we turn to analysing the results visually. It also gives the readers an opportunity to analyse the results subjectively.

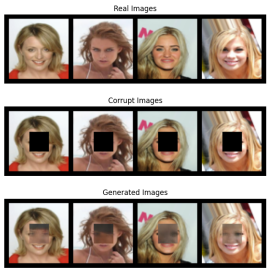


Figure 3: Inpainting results on the CelebA dataset.

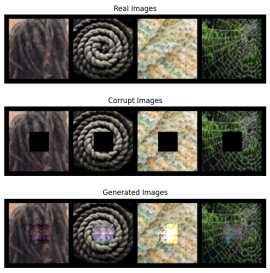


Figure 4: Inpainting results on DTD.

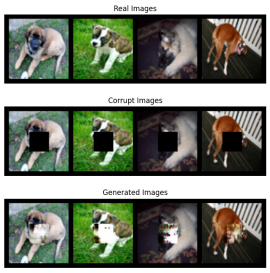


Figure 5: Inpainting results on the Pets dataset.

5.1 Effect of the Image Domain

Figures 3 to 5 show the results on CelebA, DTD and Pets datasets with corrupted images using the regular square mask. We observe that the generated images on the CelebA dataset are closer to the real images, as compared to the generated images on DTD and the Pets dataset. A few of the causes for the difference in the generated images can be attributed to the less number of images in the other two datasets and more diversity within the dataset. As a result, the trained generator from the DCGAN model fails to generate effective image representations.

5.2 Effect of the Mask

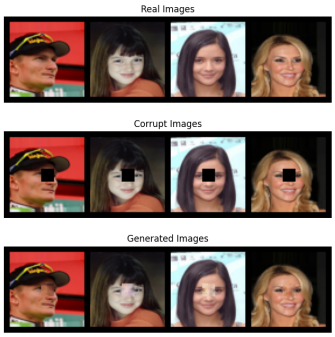
Having seen the best results on the CelebA dataset, we proceeded to see the effect of the different mask shapes only on the images from the CelebA dataset. Figure 6 shows how the generated images change with the shape of the masks.

As seen in the Figure 6a, we get better results with a smaller missing region. Smaller missing regions allow the model to have more image data which it can use to learn the image representation and generate a more effective encoding. On the contrary, Figure 6b shows that increasing the amount of missing regions generate relatively poor results.

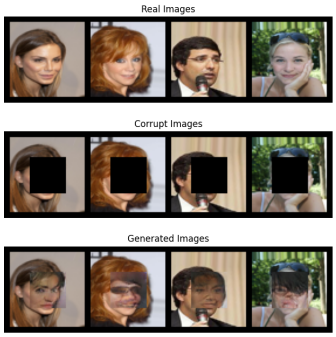
We create more corrupted images using a circular mask, located at the center of the image. We see that the results in Figure 6c are better than in the Figure 6b, but very similar to Figure 3. The similarity on the performance can be caused by the similar size of the mask, as the diameter of the circle is equal to the side of the square in Figure 3. The best overall performance of our approach is shown in Figure 6d where the mask is randomly generated by placing 2×2 rectangles. We generate 10 such tiny rectangles, such that the amount of missing region is same as 20×20 rectangular mask. The results highlight one of the core insights into our approach. Even though the absolute missing region are same, the discontinuity of the missing regions majorly affect the inpainting results. Therefore, we believe our approach would work most effectively, when the image has multiple smaller corrupted parts, as compared to one bigger corrupted region.

6 Benchmark with Context Encoders

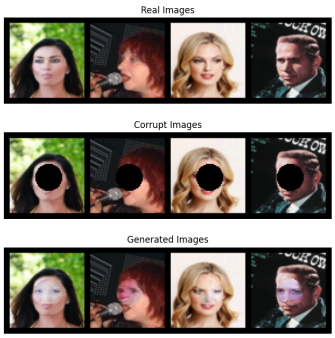
Context encoders are CNNs that predict missing parts of a scene from the surroundings. The overall architecture is a simple encoder-decoder pipeline. The encoder takes an input image with missing regions and produces a latent feature representation of that image. The decoder takes this feature



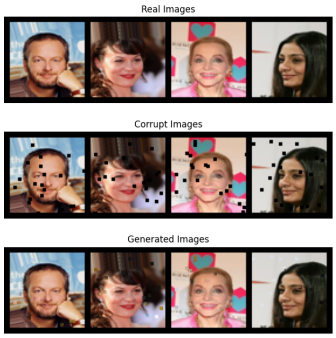
(a) Results on corrupted images generated using the small square mask.



(b) Results on corrupted images generated using the large square mask.



(c) Results on corrupted images generated using the circular mask.



(d) Results on corrupted images generated using the randomly placed 2×2 rectangles as mask.

Figure 6: Inpainting results showing the effect of mask shapes.

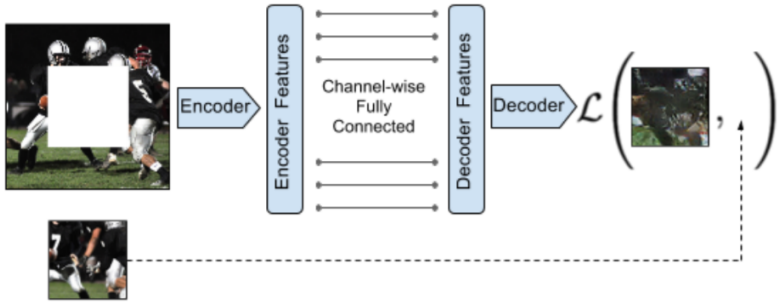


Figure 7: Context encoder framework [5].

representation and produces the missing image content. The authors [5] found it important to connect the encoder and the decoder through a channel-wise fully-connected layer, which allows each unit in the decoder to reason about the entire image content, shown in Figure 7. It uses a joint loss function which minimizes both reconstruction and adversarial loss.

We compare our model with those obtained from CE, which was the popular state-of-the-art method for semantic inpainting. Masks were used to train the CE. We applied all the different masks and

Figure 8: CE initial results after 25 iterations.

retrained the CE model with our dataset. In contrast, our method can be applied to arbitrary masks without retraining the network, which is a huge advantage. Results can be seen in Figure 8.

(a) Random real CelebA Samples.

(b) Cropped samples with central region mask.

(c) Reconstructed samples using CE.

Figure 9: CE final results on CelebA dataset.

7 Conclusion

In this project, we implemented a model for semantic inpainting. Experimental results demonstrated that unlike the original papers [1], we failed to generate realistic images when the mask holes are large. We found that the method worked the best on the CelebA dataset. We compared the method on different mask shapes and sizes and found that the method worked very well on randomly placed small masks. CE generated overly smooth images. But in our experiments, the method worked pretty good. We acknowledge that with the recent development of this area, more advanced approaches may work better for semantic inpainting for large missing areas, such as contextual attention model proposed by Yu et al. [20]. To improve the performance of the DCGAN model, the future work may involve efficient hyperparameter tuning and testing on other large image datasets.

References

[1] 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. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017-Janua:6882–6890, 2017. doi: 10.1109/CVPR.2017.728.

[2] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[3] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, , and A. Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[4] O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. V. Jawahar. Cats and dogs. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[5] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. Context Encoders: Feature Learning by Inpainting. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016-Decem:2536–2544, 2016. ISSN 10636919. doi: 10.1109/CVPR.2016.278.

[6] Manya V. Afonso, José M. Bioucas-Dias, and Mrio A.T. Figueiredo. An augmented lagrangian approach to the constrained optimization formulation of imaging inverse problems. *IEEE Transactions on Image Processing*, 20(3):681–695, 2011. ISSN 10577149. doi: 10.1109/TIP.2010.2076294.

[7] Alexei A. Efros and Thomas K. Leung. Texture synthesis by non-parametric sampling. *Proceedings of the IEEE International Conference on Computer Vision*, 2(September):1033–1038, 1999. doi: 10.1109/iccv.1999.790383.

[8] Kaiming He and Jian Sun. Statistics of patch offsets for image completion. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7573 LNCS(PART 2):16–29, 2012. ISSN 03029743. doi: 10.1007/978-3-642-33709-3_2.

[9] Yao Hu, Debing Zhang, Jieping Ye, Xuelong Li, and Xiaofei He. Fast and accurate matrix completion via truncated nuclear norm regularization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(9):2117–2130, 2013. ISSN 01628828. doi: 10.1109/TPAMI.2012.271.

[10] Jia Bin Huang, Sing Bing Kang, Narendra Ahuja, and Johannes Kopf. Image completion using planar structure guidance. *ACM Transactions on Graphics*, 33(4):1–10, 2014. ISSN 15577333. doi: 10.1145/2601097.2601205.

[11] Oliver Whyte, Josef Sivic, and Andrew Zisserman. Get out of my picture! Internet-based inpainting. *British Machine Vision Conference, BMVC 2009 - Proceedings*, 2009. doi: 10.5244/C.23.116.

[12] James Hays and Alexei A. Efros. Scene completion using millions of photographs. *ACM Transactions on Graphics*, 26(3):1–7, 2007. ISSN 07300301. doi: 10.1145/1276377.1276382.

[13] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in Neural Information Processing Systems*, 3(January):2672–2680, 2014. ISSN 10495258.

[14] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GaN training for high fidelity natural image synthesis. *7th International Conference on Learning Representations, ICLR 2019*, pages 1–35, 2019.

[15] Phillip Isola, Jun Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017-January:5967–5976, 2017. doi: 10.1109/CVPR.2017.632.

[16] Guim Perarnau, Joost van de Weijer, Bogdan Raducanu, and Jose M. Álvarez. Invertible Conditional GANs for image editing. (Figure 1):1–9, 2016. URL http://arxiv.org/abs/1611.06355.

[17] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[18] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, pages 1–16, 2016.

[19] Laurens van der Maaten and Geoffrey Hinton. Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[20] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S. Huang. Generative Image Inpainting with Contextual Attention. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 5505–5514, 2018. ISSN 10636919. doi: 10.1109/CVPR.2018.00577.