Image Inpainting with Context Encoders

Samveed Desai
Electrical and Computer Engineering
A59005733

Sambaran Ghosal
Electrical and Computer Engineering
A59005052

Abstract

In this project, we attempt to implement a pixel-prediction based approach: Context Encoder for the problem of image inpainting. Context Encoders are convolutional neural networks that are used to generate particular image regions, based on the surroundings of the image. The surroundings provide the necessary context to the model to learn better features such that it can produce efficient and more realistic estimates for the missing image region. We first attempt to solve the inpainting task by using a Variational Autoencoder with just the standalone pixel-to-pixel based reconstruction loss, using the base architectures of VGG, AlexNet and ResNetFor. Next, we train the context encoder, along with the same VGG, AlexNet and ResNet based architectures for the encoder-decoder pipeline using a combination of the reconstruction loss along with an adversarial loss. Qualitatively, we compare the results from the different model architectures across different losses and also perform, some analysis corresponding to the hyperparameters used during training. Finally, we can conclusively confirm that context encoders can be used for semantic image inpainting tasks as they tend to generate more sharper and clear images.

1 Introduction

Image inpainting is defined as the task of reconstructing damaged or missing parts of an image, so as to present a complete image. This is an important problem in the area of computer vision and has it's applications in many areas including image restoration and image editing. The main motivation behind solving the problem of image inpainting is that the real world is full of diverse yet highly structured visuals. We as humans have the capability to interpret the structure in the image even if some portions of the image are occluded or missing. Here, we try to explore whether some deep learning based, computer vision algorithms are able to recreate this ability. Through our approach, we show that it is possible to learn and predict the structure within natural images using convolutional neural networks (CNN's), which are the state of the art models used for image understanding tasks.

We define the problem here as follows: Given an image with a missing region, we need our convolutional neural network based model to 'fill' in the missing pixel values. This model is called a context encoder, as it consists of an encoder which takes as input the context of an image (the area surrounding the missing/damaged portion) and creates a latent representation out of it and a decoder which takes in the latent features and tries to generate the missing/damaged image content.

The barebone architecture of the context encoder is closely related to autoencoders as it shares the same encoder-decoder pipeline. The autoencoder architecture is straightforward: it takes an input image and tries to reconstruct it from the decoder after passing the input through a low dimensional, bottleneck layer. The main objective here is to generate a compact feature representation of the input image. One main problem of the vanilla autoencoder architecture is that it compresses the image content without learning a semantically meaningful representation. The fundamental problem with autoencoders for generation, is that the latent space they convert their inputs to may not be continuous, or allow easy interpolation. Based on this observation, we try to use variational autoencoders(VAE) to tackle this problem. Rather than creating a fixed, latent representation as in the

case of autoencoders, variational autoencoders provides a way to describe each latent attribute using a probability distribution. To generate the missing portion in an image, we randomly sample from each latent state distribution to generate a vector as input for our decoder model. This probabilistic manner of representation and continuous laten space allows to create different variations on an input image, thus suiting our purpose. Furthermore, our context encoder varies in a way that it fills in missing areas from the image using a deeper semantic understanding of the scene and the ability to synthesize high-level features over large spatial extents.

We train our context encoder in a end-to-end unsupervised learning framework. The task of generating a clear and sharp image for the missing portion is multi-modal as there could be multiple ways in which the missing area could be filled as well depending on the context. For this purpose, we use two sets of losses: a reconstruction loss and an adversarial loss for the training of the context encoder. The reconstruction (L2) loss captures the overall structure of the missing region in relation to the context while the adversarial loss allows to pick particular features from the distribution, thus allowing sharper images to be generated. When we use the reconstruction loss with the variational autoencoders, we see that we get blurry results whereas when we use the context encoders with the additional adversarial loss, we see much sharper predictions.

We also perform some analysis on the size of our latent feature representation corresponding to both the variational autoencoders and the context encoders and see that as the latent feature size increases, the overall generated image is better. Overall, we can see that context encoder gives reasonable results for the task of semantic hole-filling.

2 Related Work

The work on using convolutional neural network based, context encoders has sub-problems corresponding to the area of unsupervised learning and generation of natural images. We will go over the related work in each of these fields, thus leading up to our paper.

Unsupervised Learning For quite some time, CNN's have been trained on large image classification datasets with appropriate labels to learn features, which could also be transferred and used across other different tasks. Our aim is to see if we can learn such generalizable features without any labels. Some prior works in the area of unsupervised learning are autoencoders[3]. Another variation of autoencoders, denoising autoencoders[4] reconstruct the image from regional corruptions, such that the encoding learnt is robust to noise. Along the lines of variational autoencoders, context encoders require an extensive semantic information of the overall image to try and fill the missing/damaged portions of the image, which could be considered as spatially large corruption of the input.

Image Generation In the recent years, there has been a strong push in the area of image generation. One of the first works to present an simple yet effective model pertaining to the task was Generative Adversarial Networks (GAN's)[5]. As stated earlier, we also implemented VAE's[6] for our use case. One of the main reasons was that VAE's give a continuous latent space as the encoding output, which allows the decoder to create variations of the input image. Along these lines, Radford et al. [7] proposed new convolutional architectures and optimization hyperparameters for GAN's [5] producing encouraging results. We train our VAE's[6] with a pixel-to-pixel based, L2 reconstruction loss and our context encoders with a combination of the reconstruction loss and an adversarial loss, in an attempt to generate better results for the task of image inpainting.

Previous works including van den Oord et al[8], Dosovitskiy et al [9] and Rifai et al [10] show that CNN's can learn to generate novel images of certain object categories (ImageNet labels, chairs and faces respectively) but they heavily rely on the labels corresponding to the input images present in their respective datasets. In our work, we see that context encoders can be implemented on any unlabeled image dataset and learn to generate missing/damaged portions based on the surrounding context.

Inpainting and hole-filling Given that the missing/damaged region in our images is too large, classical inpainting [11, 12] and texture synthesis [13, 14] approaches can't handle our hole-filling task as they generally handle this task at a local, non-semantic level. Based on the field of computer graphics, filling in large holes is typically done via scene completion [15], involving a cut-paste formulation using nearest neighbors from a large dataset of images. However, according to analysis, scene completion is normally used to fill in holes which were formed by removing whole objects.

This does not suit our application as the missing/damaged regions in our image are arbitrary i.e it could represent partially occluded objects or even completely random scenes. Also, the nearest neighbour approach which we defined now, relies on a manually defined distance metric which is not transferable across different scenarios. We can see that, with the adversarial loss, the context encoder can be used to inpaint semantically accurate content, which performs much better than the nearest neighbour inpainting methods.

3 Method

In this section, we study the two main architectures we used to do Image Inpainting : Variational Autoencoders and Generative Adversarial Networks. Both of these architectures take in the input image which has a portion of it masked, and try to fill the masked patch of the images. The input image in both the architectures is passed through an encoder-decoder pipeline. The encoder takes the masked image and learns some feature representation of the image in latent dimension, which is then fed as input to the decoder which is responsible for reconstructing the patched out portion of the image. The encoder-decoder pipeline is discussed below.

3.1 Encoder

Our vanilla encoder is derived partially from the AlexNet architecture[16]. Given an input of 256 $\times$ 256 $\times$ 3, we use a combination of convolutions, leaky rectified linear units (Leaky ReLU) activation functions and batchnorms six times to compute a 4 $\times$ 4 $\times$ 512 dimensional feature representation. The height and width along with the number of channels after each convolutional layer set, which are used to achieve this representation are different from AlexNet as they allow for more effective computations. The context encoder network is trained for context predictions, as there are no image labels and we initialize the model with random weights.

If the encoder architecture only contains convolutional layers, there will be no information flow between one corner of a particular feature map to another. This is because the receptive field of the CNN connects all the channel information together but does not directly connect the information present in all locations within a particular channel. To enable this form of information transfer, we use a bottleneck layer. Our latent feature representation from the encoder 4 $\times$ 4 $\times$ 512 gets converted to 1 $\times$ 1 $\times$ 4000. This bottleneck layer allows to propagate information within activations of each feature map. This 4000 dimensional vector represents the feature representation of input image (with the missing region).

3.2 Decoder

We now discuss the second half of the architecture, the decoder. The decoder is used to generate the missing parts of the image using the latent feature representation from the encoder. The bottleneck is a bridge between the encoder and decoder. The decoder is a series of a set of up-convolutional layers (using transposed convolutions), batchnorms and ReLU activation functions. The main reason behind this is that the series of upconvolutions and non-linearities comprise a non-linear weighted upsampling of the feature produced by the encoder until we roughly reach the original target size i.e the missing image size which in this case is 64 $\times$ 64 $\times$ 3. The overall architecture of the encoder-decoder pipeline is defined below in the Figure 1.

Given the above encoder-decoder pipeline, the main differences between the Variational Autoencoder and Generative Adversarial Network(GAN) is described in the sub-sections below.

3.3 Variational Autoencoder

The VAE architecture first compresses the input image of size 256 * 256 * 3 using the encoder architecture into a 1 * 1 * 4000 vector. This vector represents the compressed learned features of the input image. Contrary to a traditional autoencoder that learns to represent the input image into a single point in the latent space, the variational autoencoder tries to project the input image x to a distribution over the latent vectors according to a distribution $p(z|x)$. This helps regularize the latent feature representation of the images such that input images are not always projected to a single latent vector. This helps in adding variation qualities when passed to the decoder. Next, we sample a

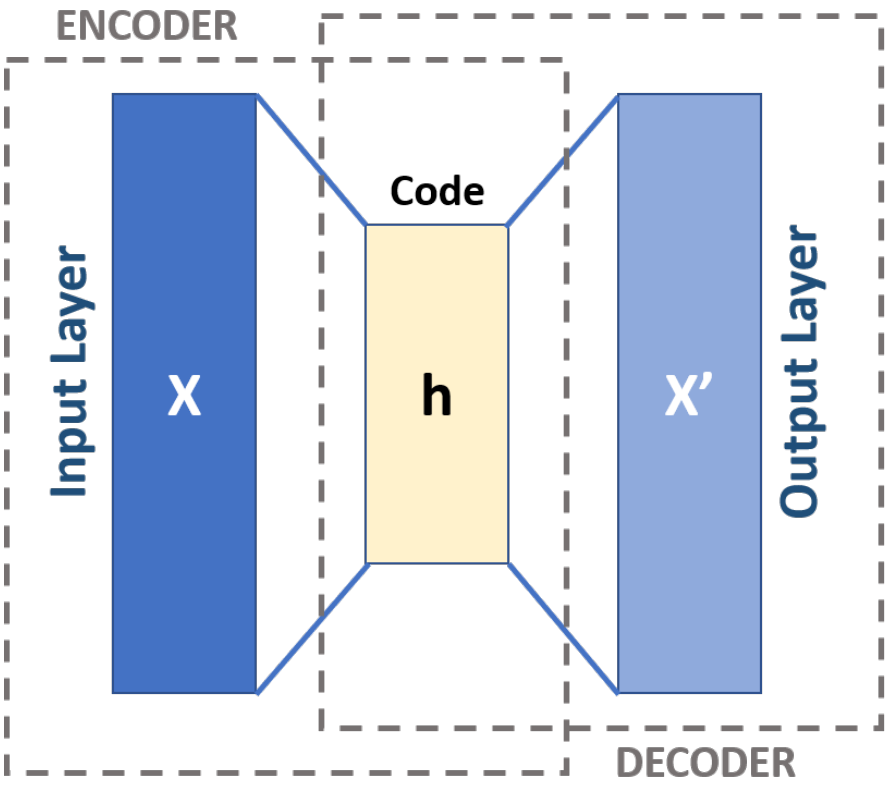

Figure 1: General implementation of Encoder-Decoder Architecture

latent vector z from the distribution $z \sim p(z|x)$ and this is then passed to the decoder architecture for reconstruction.

In practice, the encoded latent feature distribution is chosen as a normal distribution $\mathcal{N}(\mu, \sigma)$. The latent feature distribution is desired to be a standard normal distribution which enables both local and global latent space distribution regularization. We implement two separate networks that take as input the latent dimension feature vector and produce the mean and covariance of the latent dimension. These are considered as the parameters of the standard gaussian distribution representation of the latent features.

For the reconstruction loss, we use a normal L2 loss between the output of the decoder and the original missing portion of the image. Besides, this, to train the mean and variance network of the VAE, we use the Kullback-Leiberg Divergence loss between the generated encoder latent space distribution and the standard normal distribution. The KL-divergence loss between two distributions $p(x), q(x)$ is given as

$$KL[p||q] = \int_{-\infty}^{\infty} p(x) log \frac{q(x)}{p(x)} \tag{1}$$

For the VAE, p(x) is the gaussian distribution generated by the encoder and q(x) is the standard normal distribution $\mathcal{N}(0, I)$. The total loss for the VAE is then defined as the sum of the reconstruction loss and the KL-divergence loss between the generated gaussian distribution and the standard normal distribution.

$$\mathcal{L}(x) = ||x - D(z)||_2 + KL[\mathcal{N}(\mu, \Sigma)||\mathcal{N}(0, I)] \tag{2}$$

where z is sampled from the encoder latent space distribution $z \sim p(z|x) = \mathcal{N}(\mu, \Sigma)$. The total loss is then passed backward throughout the VAE architecture.

3.3.1 Generative Adversarial Networks

This network was first introduced by Goodfellow et al and provided a new framework for estimating generative models via an adversarial process. We have a generative model G that captures the overall data distribution and an adversarial discriminative model D which computes the probability that a sample came from the training data rather than G. G and D are defined as deep neural networks where G: Z $\rightarrow$ X maps samples from a random distribution Z to data distribution X. The training procedure is a minmax game wherein the adversarial discriminator D takes in both the predictions from the ground truth samples and generator G and tries to distinguish between them (the output is a probability, from the sigmoid function) whereas main aim of G is to produce samples that appear as real as possible, so as to confuse D. The main objective for the discriminator is the logistic likelihood which checks whether the input is a real sample or a predicted one:

$$\min_G \max_D \mathbb{E}_{x \in \mathcal{X}}[\log(D(x))] + \mathbb{E}_{z \in \mathcal{Z}}[\log(1 - D(G(z)))] \tag{3}$$

We adapt this framework for our context prediction task. In our problem setting, the adversarial loss L_{adv} is defined as:

$$\mathcal{L}_{adv} = \max_D \mathbb{E}_{x \in \mathcal{X}}[\log(D(x)) + \log(1 - D(F((1 - \hat{M}) \odot x)))] \tag{4}$$

where $\hat{M} \odot x$ is the mask applied on the input image x. We use a normalized masked L2 distance as our reconstruction loss function L_{rec},

$$\mathcal{L}_{rec}(x) = \|\hat{M} \odot (x - F((1 - \hat{M}) \odot x))\|_2^2 \tag{5}$$

where $\odot$ us the element wise product operation.

The joint loss, which is a weighted combination of the reconstruction and adversarial loss, is defined as: $L_{total} = \lambda_{rec} L_{rec} + \lambda_{adv} L_{adv}$. The figure for the overall architecture is shown below.

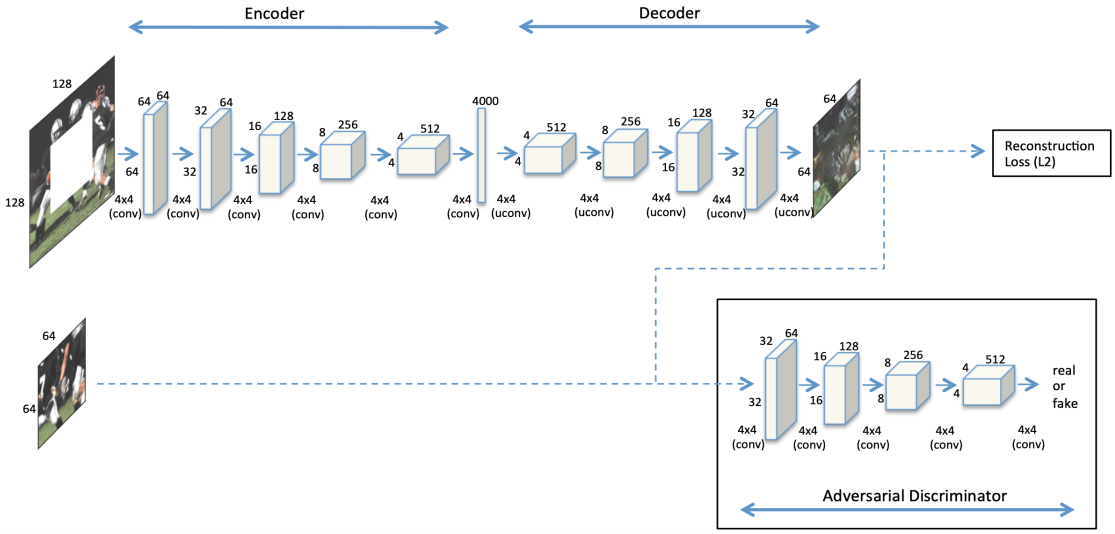

Figure 2: Generative Adversarial Network Architecture

4 Experiments

4.1 Dataset

The dataset used for our experiments is the Places dataset. Places consists of 2.5 million images of over 205 scenes. Each image is 256*256 color image, in jpg format. For our experiments, we masked out the central 64 * 64 portion of the image and saved these images as our training data. The total training data was 5000 images whereas the test data was 2000 images. These were chosen in accordance with the compute resources. The batch size used during training and test is 32. These

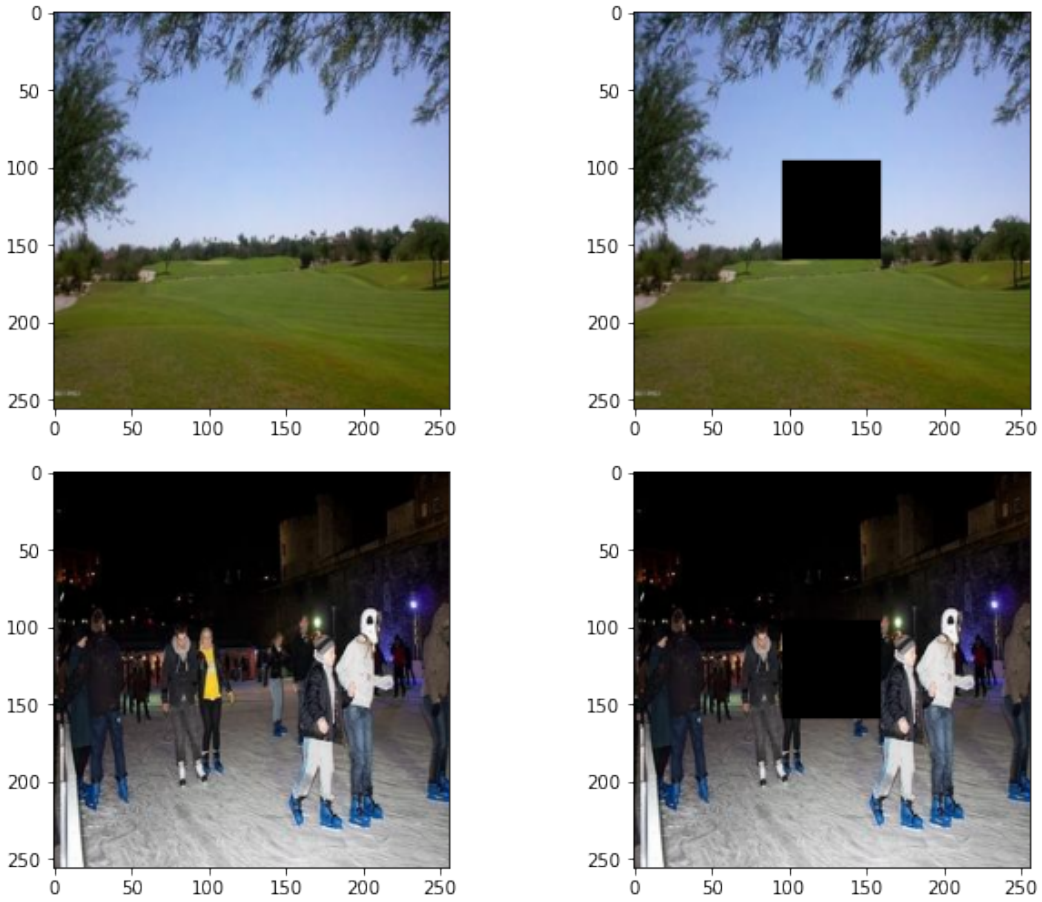

Figure 3: Places dataset and corresponding training images

images are then passed to the different context encoders considered by us. Some of the images from our dataset and the corresponding training images are shown below :

For our first set of experiments, we try to compare the performance of Variational Autoencoder and Generative Adversarial Networks for the Image Inpainting using different encoder architectures. For each experiment, the bottleneck of the encoder architecture was set to 4000, and the networks are trained for 100 epochs on a set of 5000 training images from the Places dataset mentioned above. The networks are evaluated on the basis of reconstruction loss between the generated image and the original image, averaged over all the batches present in the training and test set.

4.2 Variational Autoencoder

We first attempted to do Image Inpainting using the Variational Autoencoder architecture discussed above. We considered different encoder architectures for the VAE namely AlexNet, VGG and ResNet. The performance metrics for the mentioned networks is shown in Table 1. From the reported metrics,

Table 1: Comparison between different encoder architectures in VAE

Network	Training Loss	Test Loss
AlexNet	1894.29	1882.63
VGG	1747.96	1784.43
ResNet	1725.11	1769.02

we can osberve that VAE with ResNet and VGG architectures perform better than the simple AlexNet architecture on both the training and test set. We expected this kind of observation since AlexNet

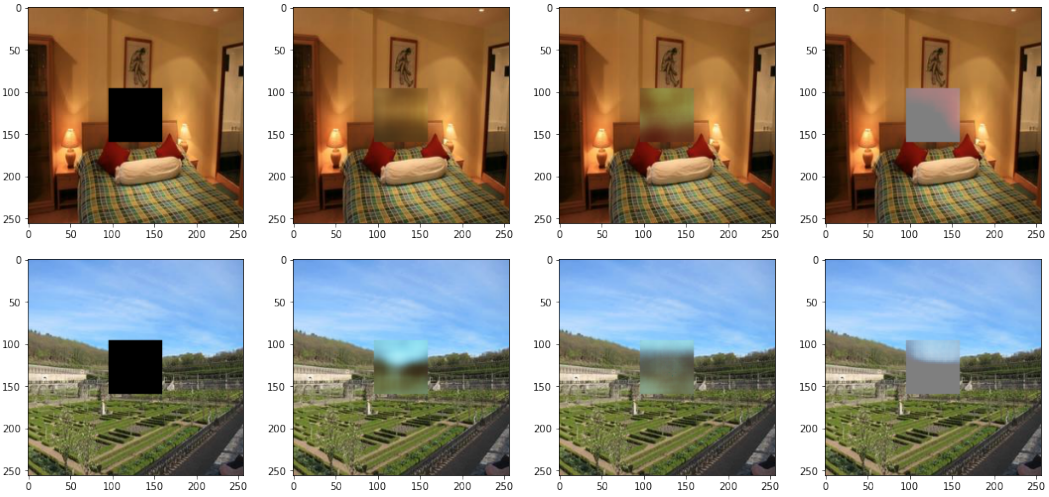

Figure 4: Reconstruction of masked portion using Variational Autoencoder. Column1 : original, Column2 : VAE VGG, Column3 : VAE ResNet, Column4 : VAE AlexNet

is a shallow network whereas VGG and ResNet are deeper networks. Also AlexNet starts with a large kernel size which leads to loss of information from the image very soon whereas VGG and ResNet start with smaller kernels which captures information at a greater level than AlexNet. We had expected ResNet encoder structure to give a better peformance than VGG due to the fact that ResNet enables better backflow of gradients through skip connections but we observe that for the Image Inpainting task, both VGG and ResNet perform very similarly.

From the reconstructed images shown if Figure (4), we observe that reconstructed images of VGG and ResNet encoder architecture are better than using AlexNet, but the reconstructed patch is blurry. One of the reasons we believe that ResNet is not able to perform better than VGG is because ResNet has much more parameters compared to VGG and since we are training the networks only using 5000 training images, there is a high possibility of underfitting.

4.3 Generative Adversarial Networks

Using the GAN architecture, we again experimented by considering two different architectures for the encoder of the generator network of the GAN : AlexNet and ResNet. Using the same training conditions and performance metric mentioned above, the results obtained are reported in Table 2. The reconstructed images are shown in Figure (5). We can observe that in row 1 column2, the reconstruction looks better than what we had observed from the reconstruction using VAE. The second row second column also is better than the corresponding one in VAE but it has a green jitter that may be present because of insufficient training. The reconstruction using ResNet does not give good results possibily because of underfitting because of training with less data which is evident from the training and test loss shown in Table 2

From the reconstructions shown in Figure (4) and Figure (5), we observe that VAE are able to fill the patch in a more accurate sense but the reconstruction is blurry and not very clear, whereas the reconstruction using GAN is not super close to what was present in the original image but it learns to fill the patch more smoothly and clearly as compared to VAE.

Table 2: Comparison between different encoder architectures in GAN

Network	Training Loss	Test Loss
AlexNet	1798.83	1768.69
ResNet	3030.98	3038.22

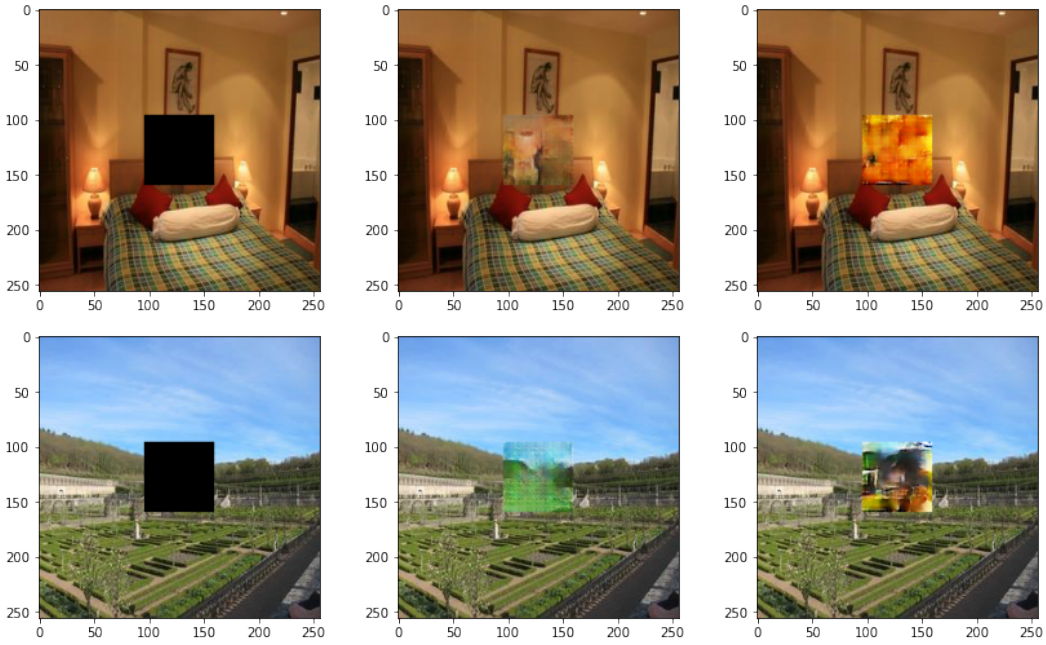

Figure 5: Reconstruction of masked portion using GAN. Column1 : original, Column2 : GAN AlexNet, Column3 : GAN ResNet

4.4 Effect of bottleneck size on the performance

One of the important parameters in the VAE and GAN architecture is the bottleneck size that the encoder compresses the input image to. A larger size of the bottleneck would be able to represent more features of the input image, whereas a smaller bottleneck size would lead to losing some features of the input image and this may harm the reconstruction performance. To study this behaviour, we implemented VAE with ResNet and GAN with ResNet and considered bottleneck sizes 500, 2000 and 4000. The networks were trained on 5000 training images from Places for 30 epochs and the performance metrics are reported in Table (3)

Table 3: Comparison between bottleneck size in reconstruction

	Bottleneck Size		
	500	2000	4000
VAE VGG	1817.82	1784.43	1808.84
VAE ResNet	1819.92	1813.5658	1818.61
GAN AlexNet	2101	2237.55	1813.56
GAN ResNet	5470.18	4704.27	3038.22

From the table above, we observe that in all cases, the bottleneck size of 500 performed the worst. This supports our claim that a larger size of the latent dimension is better since it is able to capture more information from the input image. Comparing bottleneck size of 2000 and 4000, we observe that a bottleneck size of 4000 is better for GAN's whereas a bottleneck size of 2000 works better for VAE. This shows that the bottleneck size is heavily dependent on the network type and problem type.

4.5 Reconstructed Images

From the results in Table 1 and Table 2, we see that VGG encoder architecture gives the best reconstruction performance with Variational Autoencoder and AlexNet architecture with GAN. To distinguish between the reconstruction performance of both, we present a series of reconstructed images and compare them side by side. The results are shown in Figure (6). Figure (7) shows some scenarios of reconstruction failure in both VAE and GAN.

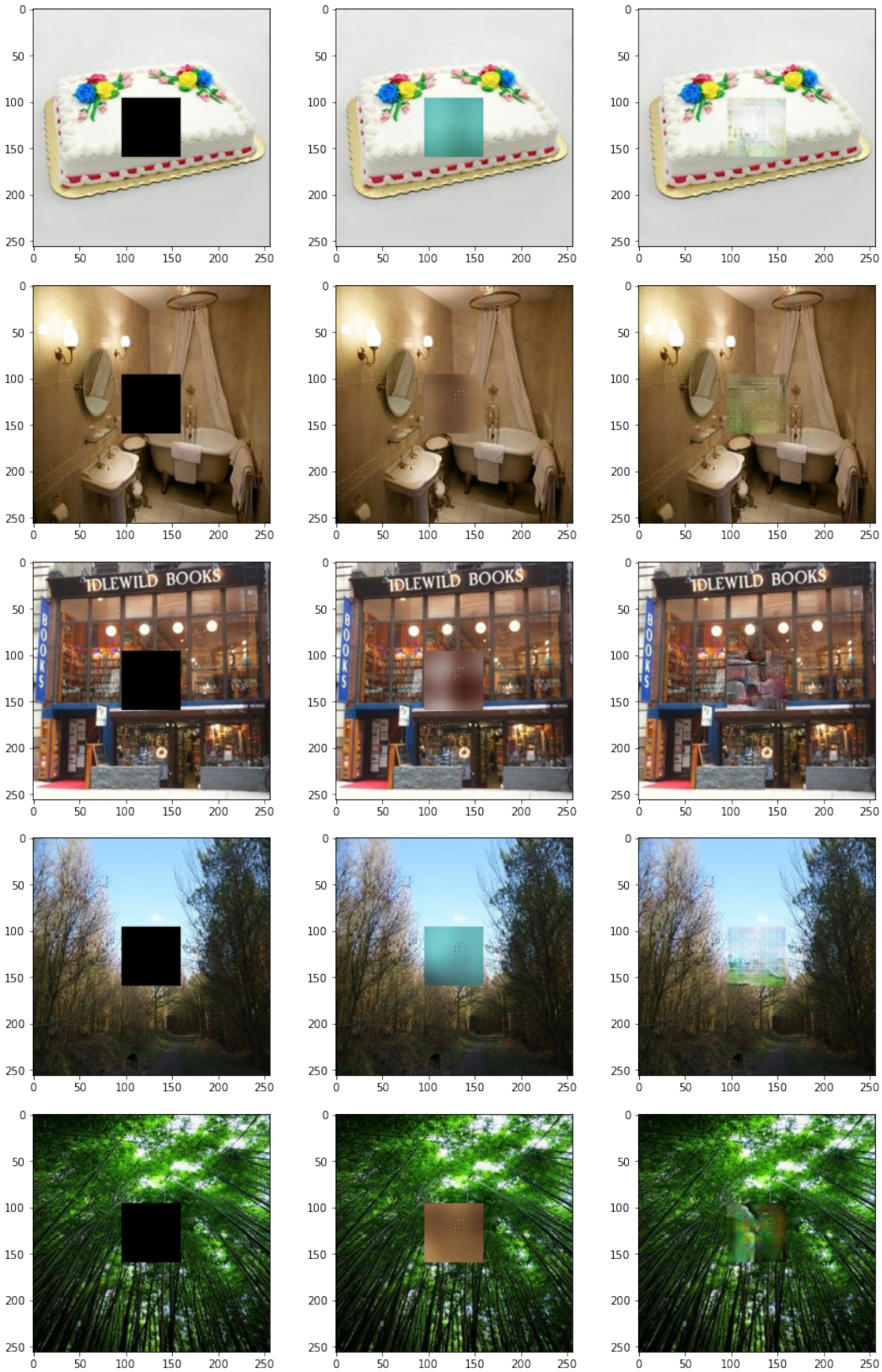

Figure 6: Comparison of reconstructed images using Variational Autoencoder and Generative Adversarial Network. Column 1 : Original Image, Column 2 : VAE, Column 3 : GAN

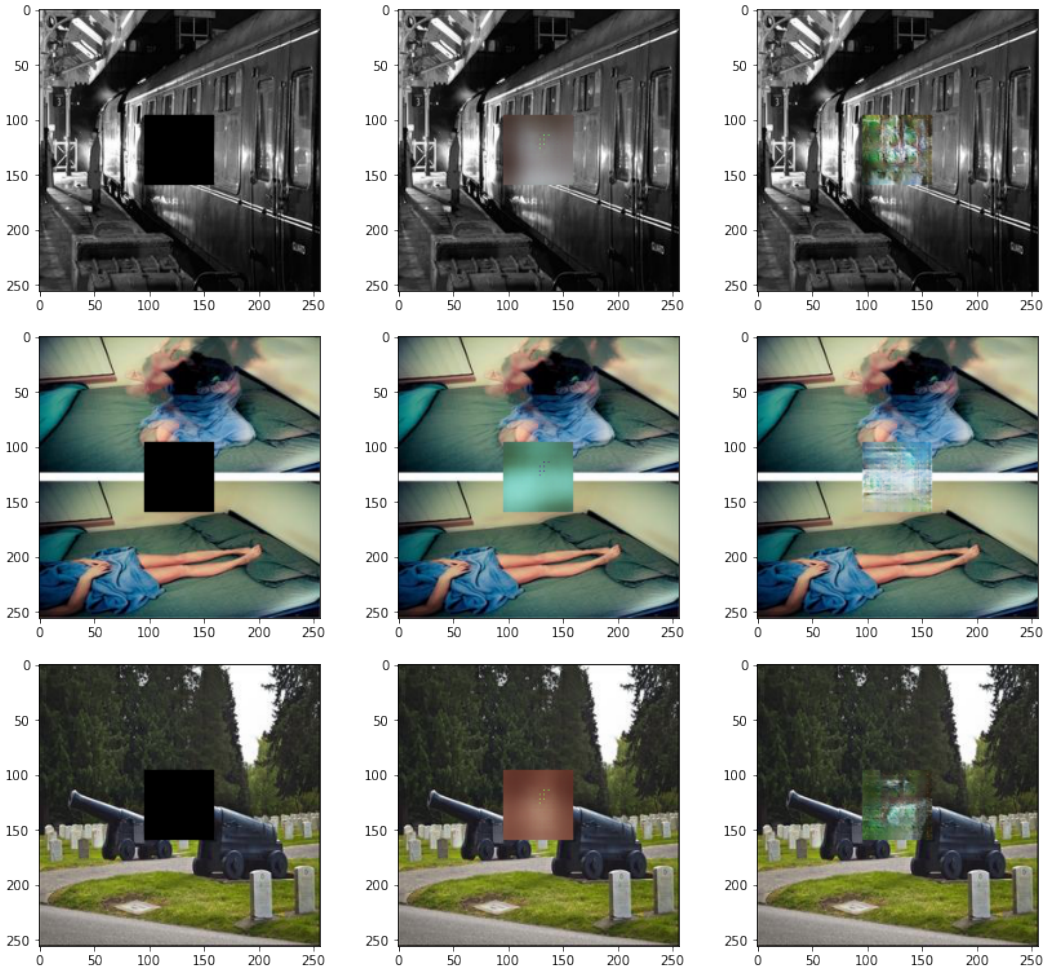

Figure 7: Some failure case images using both VAE and GAN. Column 1 : Original Image, Column 2 : VAE, Column 3 : GAN

5 Conclusion

In conclusion, we implemented two architectures for Image Inpainting, Variational Autoencoder and Generative Adversarial Networks. We experimented with several different encoder architectures and size of latent dimension to see the effect in performance of the reconstruction. From our preliminary experiments, we can conclude that the latent dimension size is highly dependent on the type of architecture we implement. It is better to have a larger latent space compared to having very small latent dimension, but as our experiments suggest, we need some fine tuning for the latent dimension size depending on the exact architecture we intend to implement.

We saw that AlexNet architecture in Variational Autoencoder and ResNet architecture in GAN's did not give appropriate results. We conclude that it is possible that due to less training data and epochs, there was underfitting in these networks and hence they were not able to generalize well to the test set.

We strongly believe that if we train the networks with larger sized training data and for more epochs, they will generalize better to the task of Image Inpainting. In general, we saw that the reconstructions are very sharp and clear obtained from GAN's although they may not exactly represent the original image pixels, whereas the reconstructions from VAE were closer in resemblance to what was present in the original image but the reconstructions turned out to be super blurry. Hence we conclude that for reconstruction tasks, including a term for adversarial loss often improves the performance than compared to what can be achieved only using reconstruction loss.

6 Future Work

In the future, we would like to use additional compute resources and train our model for a larger number of epochs, both for the VAE and GAN based architectures and compare the performance with our current setup. Another study to try out is to add extra data while training and testing our model. This would allow us to see the influence of larger data on the accuracy/loss as well as quality of the output. We could also experiment with changing our encoder architectures and adding more layers to the network. The weight factors corresponding to the reconstruction and adversarial loss can be varied to see is putting more weightage on adversarial loss will improve the overall performance.

7 Acknowledgement

We would like to thank Professor Xiaolong Wang and the ECE285 teaching staff for their continuous support throughout the quarter and their feedbacks that helped us choose this topic for our project and provide us with proper materials needed for executing this. We would also like to extend our sincere thanks to Datahub, UCSD for the computational resources without which we would not have been able to train our methods and evaluate our method performances.

References

1. D. Pathak, P. Krahenbuhl, J. Donahue, T. Darell, A. Efros, Context Encoders: Feature Learning by Inpainting, CVPR 2016

2. J. Jordan, Variational autoencoders, 2018

3. G. E. Hinton and R. R. Salakhutdinov. Reducing the dimen- sionality of data with neural networks. Science, 2006.

4. P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising au- toencoders. In ICML, 2008

5. I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Gen- erative adversarial nets. In NIPS, 2014

6. D. P. Kingma and M. Welling. Auto-encoding variational bayes. ICLR, 2014

7. A. Radford, L. Metz, and S. Chintala. Unsupervised repre- sentation learning with deep convolutional generative adver- sarial networks. ICLR, 2016

8. A. van der Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, K, Kavukcuoglu, Conditional Image Generation with PixelCNN Decoders, NIPS 2016

9. A. Dosovitskiy, J. T. Springenberg, and T. Brox. Learning to generate chairs with convolutional neural networks. CVPR, 2015

10. S. Rifai, Y. Bengio, A. Courville, P. Vincent, and M. Mirza. Disentangling factors of variation for facial expression recognition. In ECCV, 2012

11. M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester. Image inpainting. In Computer graphics and interactive techniques, 2000

12. S. Osher, M. Burger, D. Goldfarb, J. Xu, and W. Yin. An iterative regularization method for total variation-based image restoration. Multiscale Modeling Simulation, 2005

13. C. Barnes, E. Shechtman, A. Finkelstein, and D. Goldman, Patchmatch: A randomized correspondence algorithm for structural image editing. ACM Transactions on Graphics, 2009

14. L. Gatys, A. Ecker, M. Bethge, Texture Synthesis Using Convolutional Neural Networks, NIPS 2015

15. J. Hays and A. A. Efros. Scene completion using millions of photographs. SIGGRAPH, 2007

16. A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In NIPS, 2012.