Image Inpainting

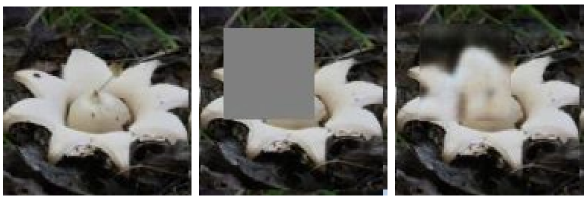
Vaibhav Rawat
(vrawat279@tamu.edu)

Shubham Bhargava
(shbhmbhrgv@tamu.edu)

Texas A&M University, College Station

I. INTRODUCTION

In our modern world, we come across a variety of images daily. The multitude of diversity involved in these images is huge. Yet, when we come across an image with a missing or obscured region that we have not seen before, we are still able to imagine its exact content. The way we picture its content is by analysing the image to find the surrounding regions and complete the image coherently. But the most important thing that helps us in making sense of the context is the fact that images are highly structured. This structure helps in extrapolating the recognizable contents of the image to complete the image. This is the motivation behind our project to apply machine learning algorithms to train our system on the image content so that similar to humans they can learn the context of the image. Then use this context to make predictions for the missing parts of the image that come close to the actual image. We use a CNN based Generative Adversarial Network trained using reconstruction and adversarial loss for the filling of missing regions.



(a) Original (b) Cut Region (c) Generated Image

Fig. 1: An example of original image and its generated image for its cropped version in part (b).

II. LITERATURE REVIEW

Images are used in a lot of domains such as Computer Graphics, Computer Vision, Robotics etc. Inpainting of images is relevant when dealing with real world data. A lot of previous work has been done and multiple solutions proposed from different domains to fill the noisy images.

Efros et al. [1] proposed a non-parametric approach for Texture Synthesis. Texture is generated pixel by pixel from an initial seed. A single pixel is chosen for synthesis using all pixels present and previously generated in a square window around p, considering the probability distributions. It is good at capturing statistical processes. Major problem with this approach is its tendency to output incorrect texture if the prediction of intermediate textures fall into wrong search space.

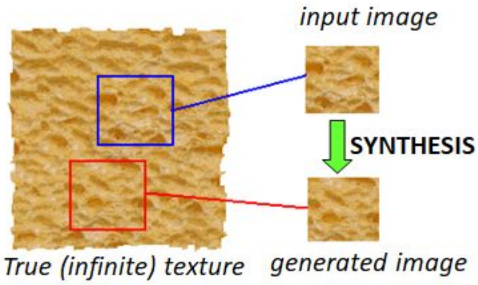


Fig. 2: Example of Texture Synthesis

In Computer Graphics, another approach of scene completion by Hayes et al. [2] is used. This approach relies on a huge database of images from the web. The algorithm works by patching up images by finding semantically similar image regions in the database. It relies on the fact that space of semantically differentiable scenes is not that large. Main drawback is that this approach is successfully in filling holes left by whole objects, but struggles when partial objects are removed from the image.

Recent work done by Kushagr et al [3] explored architectures such as CNN with a Sigmoid Euclidean Loss and a simplified PixelRNN for the purpose of Image Inpainting.

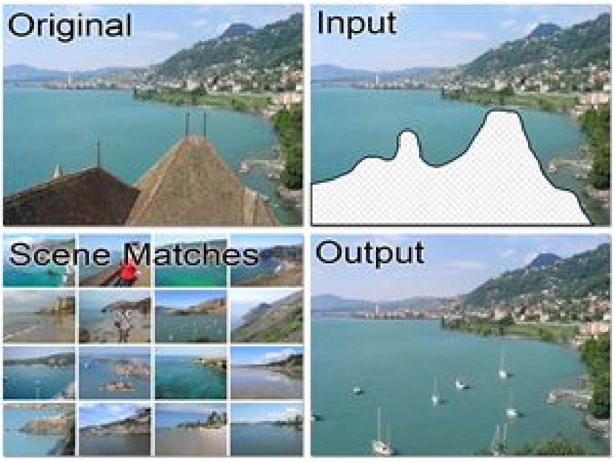


Fig. 3: Scene Completion

III. PROBLEM FORMULATION

Previous works employ different algorithms involving surrounding pixels, probability distributions, matching against database of images etc.

Most of the above approaches like Texture Synthesis involve local semantic methods which do not scale well as the missing regions of image become too large. Method learning the context of the entire image through semantically meaningful representations is needed for predicting the missing content. It will also helps in avoiding to go through millions of images and a lot of computation to predict missing region like scene completion approach.

Another important thing that needs to be considered is that there are multiple ways to coherently fill the missing region. We need an approach which generates the missing region as close to the original image through a learning based approach involving the original image.

IV. PROPOSED SOLUTION

With above considerations, we propose a unsupervised learning approach using Convolutional Neural layers with the following networks:

A. Encoder - Decoder Network:

Main idea behind this layer is to take an input image with missing regions and use the encoder part of the network to generate equivalent feature representation. Decoder layer then takes this feature representation learned from encoder network to produce the missing image content.

Encoder - Our network works on input image of size 128x128. It consists of five convolution layer networks each with a kernel size of 4X4 and stride of 2 to computer feature representations. To propagate information from one corner of the feature map to the another, fully connected layers are used where all activations are connected to each other.

Decoder - Encoder features are connected to decoder features using a fully connected layer. Decoder network produces pixels of the image using feature representations by passing them through five up-convolutional layers. Upsampling of the feature in a non-linear manner is performed until the original size of the image is reached.

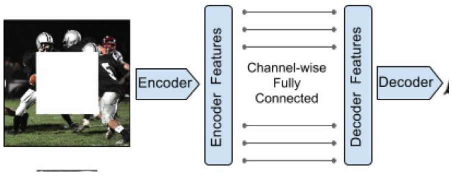


Fig. 4: Encoder Decoder network

B. Adversarial Network:

Generative Adversarial Networks architecture is used. Both our Generative network of Encoder-Decoder for reconstruction of images, and Discriminator Network learn in parallel. Discriminator Network is first given a batch of original input images with no cropping, which score them. Reconstructed images from the Encoder-Decoder network are then supplied to the Discriminator network for a score. Adversarial Loss is calculated using this repeated schedule to train both the networks.

Following loss functions are used to train our network:

Reconstruction Loss :

Normalized L2 distance between the input image and the reconstructed image is used for calculating the reconstruction loss associated with the image. It helps in context-based capturing structure and coherence of the missing region. It generates a blurry solution because it predicts mean of the distribution to minimize pixel-wise error and misses any high frequency detail.

Adversarial Loss:

Adversarial Loss is generated based on the scores of the original and reconstructed image and is used to train our Generative and Discriminator networks. It helps in capturing high frequency details and make a prediction look real by picking a modes particular to original image.

Our joint loss is calculated as follows:

$$\mathcal{L} = \lambda_{rec}\mathcal{L}_{rec} + \lambda_{adv}\mathcal{L}_{adv}.$$

V. DATA DESCRIPTION

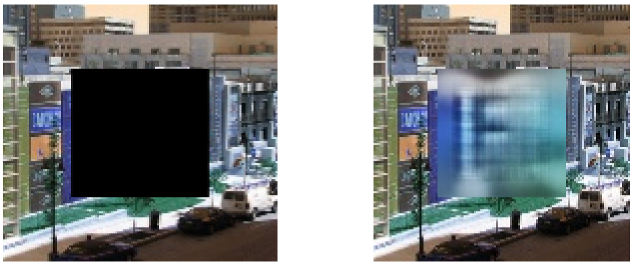
Our algorithm works with any input image of size 128x128. For this project we experimented with a lot of different image datasets such as ImageNet, Paris Street View, and Large Scale Screen Understanding (LSUN).

A region of 32x32 is cropped from the original image before passing it to the Encoder Decoder Network for image inpainting. 32x32 is a considered a large missing region for the purpose of inpainting. We tried with both center cropping and random cropping of regions of images to evaluate our algorithm. Different number of epochs were run with images to get more insights about Inpainting.

VI. RESULTS AND DISCUSSION

For most of the cropped images, after 1000 epochs our model was able to predict the missing regions very close to the original image. Although results improved with further epochs, 1000 epochs were good enough to get visually clear results in most of the cases.

Another interesting result that we observed was related to the surrounding pixels of the missing region We saw that images with non - uniform surrounding around the missing region were harder to inpaint. They took significantly more number of epochs for good results.

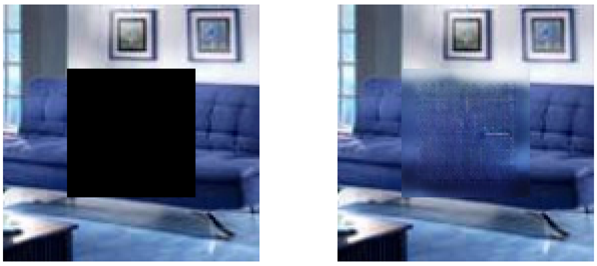

(a)

(b)

Fig. 5: Generated image with non-uniform surroundings

In the above figure nearby building has a variety of colors. Also the context around the missing region is quite varying with different types of buildings. The results for 1000 epochs clearly show that it is harder to inpaint this image.

In contrast the following image has quite homogeneous surroundings and with just first 170 epochs results are very promising.


(a)

(b)

Fig. 6: Generated image with uniform surroundings

In our experiments, we also observed that images with center cropped regions are much harder to inpaint than the images with random cropping. Generally images tend to have most of the context at the center of the image and so if the region is cropped at the center it is harder to learn the context from the missing image and fill in the cropped region.

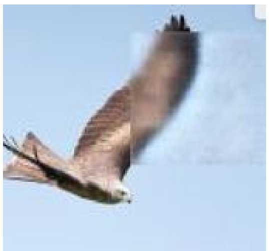


Fig. 7: Generated Image with randomly cropped region

Above image shows the results of applying our algorithm for filling an image of a bird which was randomly cropped. We got the above results in just 200 epochs.

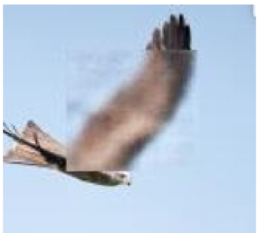


Fig. 8: Generated image with center cropped region

For the same image when we cropped the image from the center it took 500 epochs to produce the above image. Results for the random cropping were better even with less epochs.

VII. CONCLUSION

The model involving Convolutional layers for our Encoder-Decoder networks is very effective in reconstructing large missing parts of the image by propagation information across feature maps. Adversarial discriminator network helps in capturing high frequency details and make a prediction look similar to real image.

Our approach is not very effective when images have non-uniform context or surrounding around the missing regions. It also takes longer time to inpaint images with missing center regions.

VIII. FUTURE WORK

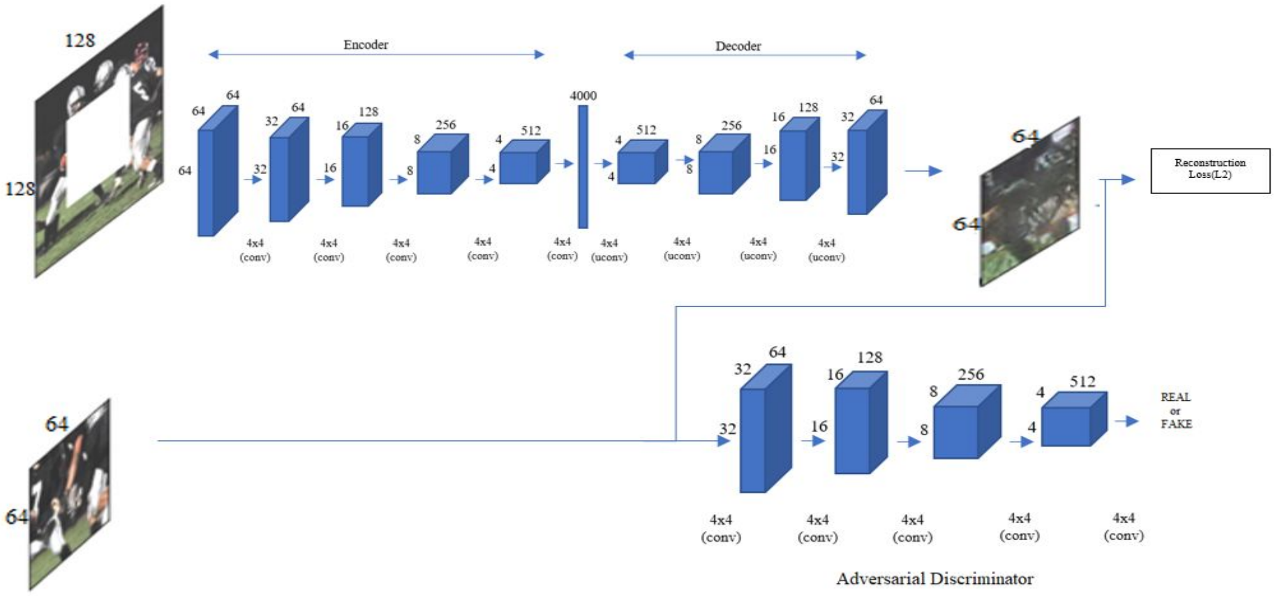
We plan to test our algorithm with other types of noisy images. We will test it against images that are entirely blurred, cropped from multiple regions and see if this can work up to be a general solution for multiple scenarios.

We also plan to use a channel wise fully connected layer instead of a fully connected layer in our Encoder - Decoder network which may reduce computations for similar results.

VIII. REFERENCES

[1] A. Efros and T. K. Leung. Texture synthesis by nonparametric sampling. In ICCV, 1999.

[2] J. Hays and A. A. Efros. Scene completion using millions of photographs. SIGGRAPH, 2007.



Layout of our Model Architecture

Following are some of the results from our Model trained with 300 epochs: