Image Outpainting using Generative Adversarial Networks (GANs)

Chintan Acharya
Department of Computer and Information Science
University of Florida
chintan.acharya@ufl.edu

Lixandross Gernier
Department of Computer and Information Science
University of Florida
lgernier@ufl.edu

Abstract

Image outpainting is one of the topics of research that still has a lot to work on. It is a process of making an AI model learn to imagine what lies beyond the boundary of the image. On other hand, AI has successfully solved the problem of image inpainting, the process of filling the areas inside the image. The main difference between image outpainting and inpainting is that the inpainting model gets the context of the image as filling a patch in the image can get help from the neighboring portion of the patch. In image outpainting, the model must learn to imagine what lies beyond the boundaries of the image. This paper talks about solutions to the problem of image outpainting using GANs, generative adversarial networks. The paper proposes 3 GANs with different layouts and approaches, of which 2 GANs give promising results.

1. Introduction

Humans can imagine what will lie beyond the image limits when we look at an image. That is possible only because we are familiar with similar scenarios, and we can generate what will lie beyond the image boundaries. Looking at the objects at the edge of the picture, we can comprehend whether these objects would extend outside the image or would not be repeated or extended. For example, if we look at a picture of a mountain and the edge of that picture has another, we will know that the next mountain extends outside the image boundary. Furthermore, the context inside the image helps us imagine a similar thing. For a picture of a forest with trees of a different type, we can imagine that we will have similar random trees beyond the image boundary. This paper proposes solutions inspired by this capability of humans. It talks about using Generative adversarial networks to outpaint the image. A GAN or generative adversarial network is a deep learning approach to developing generative models. GANs consist of two deep-learning neural networks competing with each other to generate data with the same statistics as the data used for their training. The two neural networks are the generator and discriminator. A generator generates an image, and the discriminator checks how similar the generated image is to the original image. Both the generator and the discriminator adjust their trainable parameter and learn. Once the training is complete, that generator can generate data similar to the training set that never belong to the training set. In this paper, the first GAN is a simple DC-GAN that uses a generator and discriminator to outpaint an image. The images used to train these models are 256x256 RGB images. These images are cropped to 128x128 and padded with a constant pixel value to convert them back to 256x256. The model trains to fill those padded pixel values to match the original image. A second GAN uses one generator, one global discriminator, and four local discriminator models. The local discriminators work on extracting the knowledge from the edges of the image. The global discriminator learns the context from the image. The third GAN uses a single generator and discriminator but uses weighted adversarial loss.

The flow of the solution is similar for all 3 GANs. The first step is to pre-process the images. The pre-processing is different for different models. The second step is to build the model, and the third is to train the model. The fourth step is to test the model on separate data. The last step is to evaluate the training. Figure 1 illustrates the flow of the model training and development.

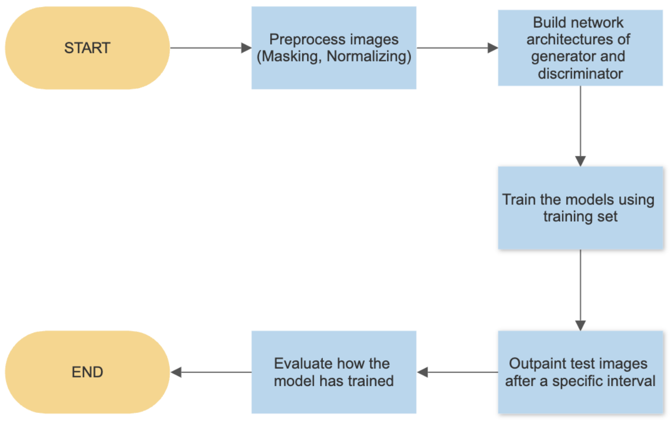

START
Preprocess images (Masking, Normalizing)
Build network architectures of generator and discriminator
Train the models using training set
Outpaint test images after a specific interval
Evaluate how the model has trained
END

FIGURE 1: Flow of the solution.

2. Related work

GANs are one of the most popular approaches for image outpainting. The paper by Ian Goodfellow and others [1], proposed the idea of GANs a machine learning framework that solely works on generating data using Deep learning. The statistics of the generated data are similar but not the same as the training data. This is perfectly aligned with the goal of this paper.

The paper [2] by Mark Sabini and Gili Rusak, talks about a similar approach to GAN 2 discussed in the introduction. It discusses the use of local discriminators for the edges. But this paper only outpaints the images on the sides and uses two local discriminators on each side. The loss of the discriminators is calculated by concatenating the output of all the discriminators and using the binary cross entropy loss. The paper displayed promising results due to the usage of local discriminators.

The third GAN model architecture and training was inspired by the paper [3]. Due to the lack of research into image outpainting the authors of the paper chose to look towards different image inpainting models to start as a base. They then added architectural ideas from working outpainting models and modified them with standard inpainting model techniques. This resulted in a model consisting of several down-convolution layers that halves the image at each layer. This is the part that is consistent with many other inpainting techniques. The paper then follows this up with several up-convolution layers that mirror the down-convolution layers resulting in the output being the same as the input. For the discriminator, they used down convolution layers that reduced the image to a 24 by 24 grid that then outputs a single value, The probability of the input being fake.

The paper used 3 loss functions. Adversarial loss, Generator loss, and discriminator loss. Generator loss was calculated by taking the absolute difference between the generator output and the target output. This is then multiplied by the adversarial loss which is the absolute difference between the Generators score when put to the discriminator and 1. Discriminator loss is a combination of real loos - fake loss where fake loss is the MSE between the discriminator output with the generated image as input and 0 and real loss is MSE between the discriminator output with the real image as input. They ran into a problem during training. Due to the nature of GAN, they are very hard to train because of the complexity of dealing with multiple losses and multiple models. The team saw that adversarial loss would converge at 1 which meant that the generator was not fooling the discriminator. They fixed this by applying a weight to the adversarial loss and varying the weight of it throughout training so the generator would be able to catch up to the discriminator. The training took over 40 hours and the results delivered were very promising.

Another related work was considered but not implemented due to the complexity of the task. In the paper [4] instead of attempting simple outpainting of a uniform border around an input image, they attempt to outpatient and modify more complex regions. Though the data set that they used is not as diverse as the places dataset, They used a Celeb face and a cityscape data set. Their models were not as generalized as the model previously stated but the results they produced on minimal input data are very impressive.

The reason that we did not implement this architecture is that the training was going to take too long and due to the architecture it would take more computing power because of the number of outputs that would need to be generated as opposed to the other paper. In this one, they would mask out just the nose, eyes, and mouth then would outpaint an entire face. We are going for the slightly less ambitious task of filling out uniform borders.

3. Experimental Setup

3.1 Dataset

This paper uses [6] MIT's Places365 dataset. It contains 256x256 images of different indoor and outdoor scenarios. The dataset has more than 2 million images. But due to the computational limitations of the hardware, the paper uses only the validation set of the dataset containing 36,500 images. For the second GAN, the training of the models was too computationally expensive, therefore we decreased the dataset by halving it and using about 20000 images out of the 36,500 images.

3.2 Technical Setup

The hardware that is used for training is Google Colab hardware. It provides GPU acceleration for faster training of neural networks. The paper used TensorFlow for model building and training.

4. Proposed Solutions

4.1 DC-GAN-1

4.1.1 Pre-processing

For DC-GAN-1, the 256x256x3 image is cropped to 128x128x3 image. The cropped image is padded on each side by 64 pixels with the pixel value as "100". The padded image is normalized between 0 to 1.

4.1.2 Model Architecture

DC-GAN is an extension of standard GAN. In DC-GAN the generator and discriminator models have convolutional layers and transposed convolutional layers respectively. DC-GAN mainly used LeakyReLU as the activation function for the layers. The generator model proposed has 6 2D Convolutional layers and Transposed Convolutional layers. It has a MaxPooling Layer between Convolutional layers. A batch normalization layer is added after each convolutional layer. The generator model uses the LeakyReLU activation function for each layer. It inputs a 256x256x3 image The discriminator model has 4 Convolutional Layers and ends with a dense layer with output as 1 unit.

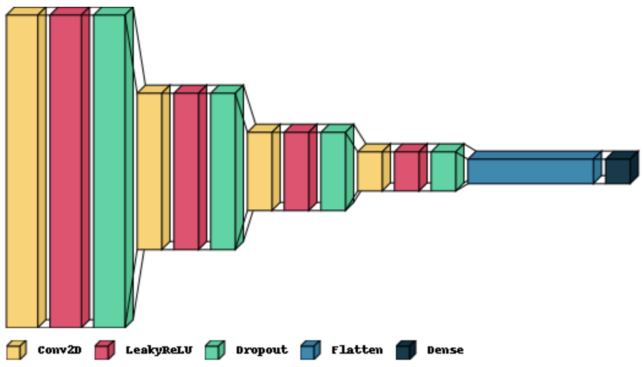


FIGURE 2: DCGAN-1 Generator Model

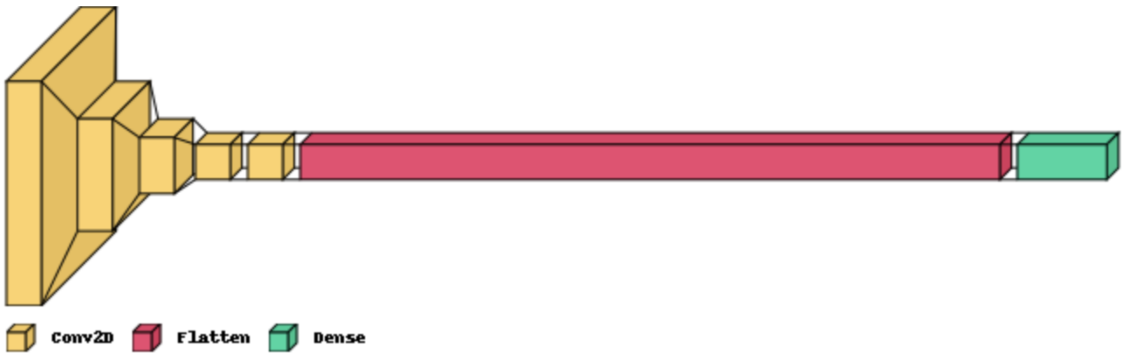


FIGURE 3: DCGAN-1 Discriminator Model

4.1.3 Model Training

Both Generator and discriminator for DCGAN 1 use binary cross entropy as the loss function for the training. For the generator, the loss function is applied to each pixel value of generated image and the ground truth image. The model is trained with a batch size of 32 images. This model was run for 350 epochs but didn't give a promising result.

4.2 GAN-2

4.2.1 Pre-processing

For GAN-2, the 256x256x3 image is cropped to a 128x128x3 image. The cropped image is padded on each side by 64 pixels with the pixel value as the mean value of all the pixels in the image. The padded image is normalized between 0 to 1. One more channel is added to the padded image. All the pixel values of this channel are 0 at the center 128x128 pixel and the rest are 1. Hence, the output of this preprocessing step is 256x256x4 image from 256x256x3 image. This is called masking. The mask is applied to make the center of the image correct to increase the accuracy of output.

4.2.2 Model Architecture

The GAN-2 uses one generator, one global discriminator, four local discriminators, and one concatenator. The generator inputs a 256x256x4 masked and preprocessed image and outputs a 256x256x3 outpainted image. The generator model comprises 8 convolutional layers with more than 2 million trainable parameters. The global discriminator inputs 256x256x3 images and output 512 units. The local discriminator for vertical sides inputs 256x64x3 images and outputs 512 units. The local discriminator for horizontal sides input 64x256x3 images and outputs 512 units. The concatenator takes the input of all the discriminators i.e.2560 units and outputs 1 unit i.e. the decision.

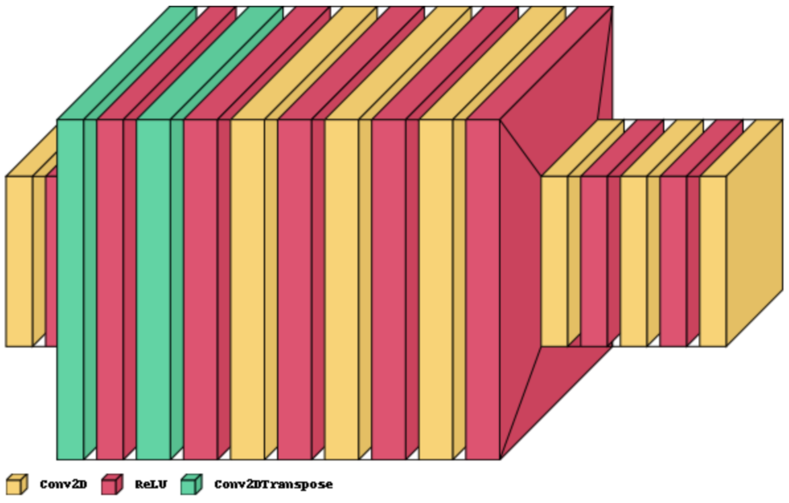


FIGURE 4: GAN-2 Generator Model

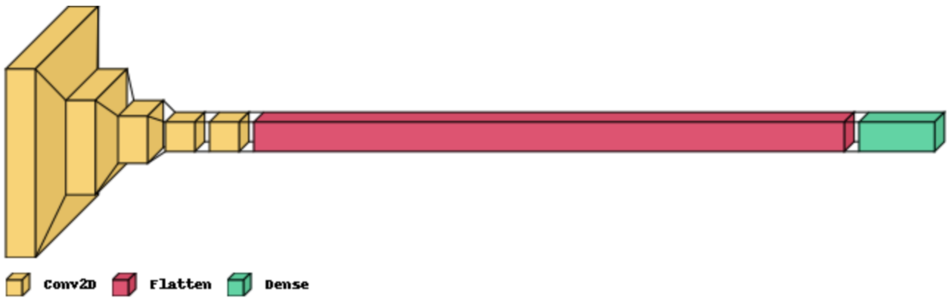


FIGURE 5: GAN-2 Global Discriminator Model

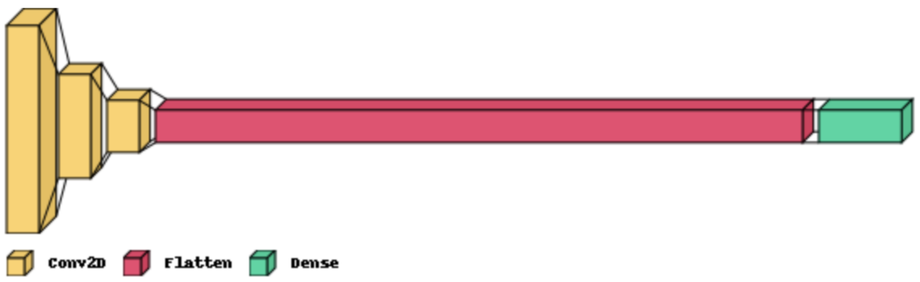


FIGURE 6: GAN-2 Local Vertical Discriminator Model

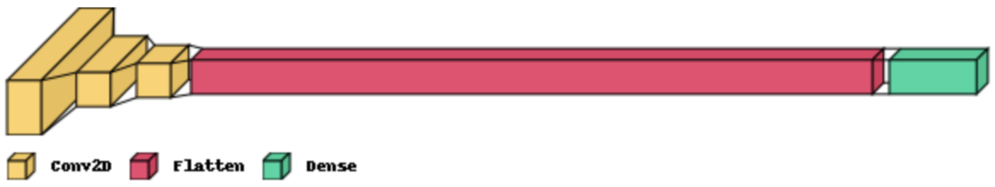


FIGURE 7: GAN-2 Local Horizontal Discriminator Model

The vertical local discriminator model is applied on the red box in Figure 8. The horizontal discriminator model is applied to the blue box in Figure 8.

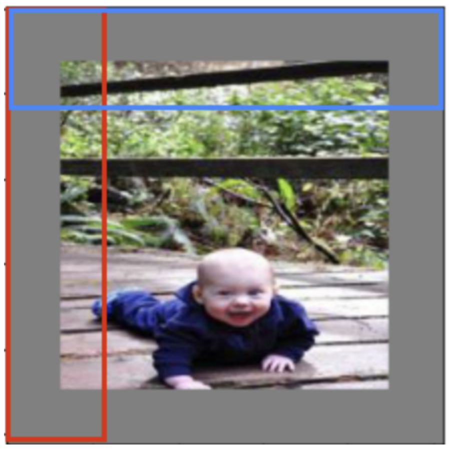


FIGURE 8: Local Discriminator applied on preprocessed image

4.2.3 Model Training

The model was trained in 3 phases for 80,000 iterations. Phase one had 30,000 iterations of generator training only. The second phase had 6000 iterations of just discriminator training. The third phase has 44,000 iterations of discriminator and generator training. Every iteration was done on 4 images. The loss function used for discriminators is the Binary cross-entropy function. The loss function is applied to the output of the concatenator. The loss function for a generator for the first phase is Mean Squared Error. The loss function for the generator for the third phase is as follows

$$Loss_{MSE} = \frac{\Sigma\left(Y_i - Y_{Mean}\right)^2}{n}$$

$$Loss_{PHASE3} = Loss_{MSE} - \alpha . log\,(D(Y_{Gen}))$$

Here, $D(Y_{Gen})$ is the discriminator output for the generated images. α is a constant value with value $1e^{-9}$.

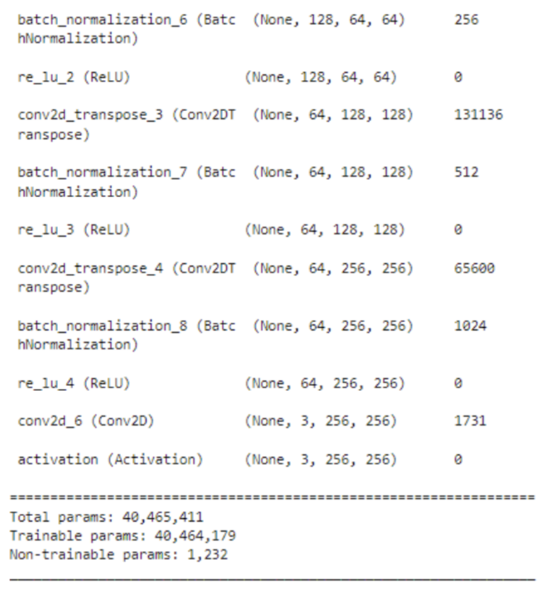
4.3 GAN-3

4.3.1 Pre-processing

The Third GAN model was built and trained using a very similar design to the one proposed in the third paper. For our input, we had images of size 192 by 192 that was cropped to be 128 by 128 with a white border around the picture. The goal was to fill out not just the white outline but the entire picture to make sure that style stayed consistent throughout the image.

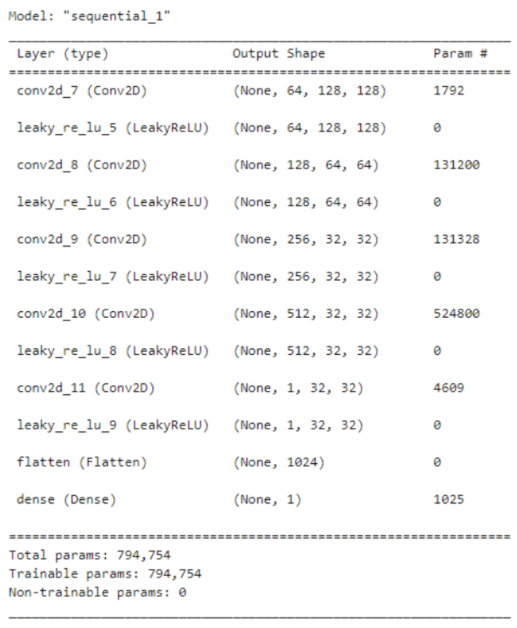
4.3.2 Model Architecture

For the generator, we used 6 down convolution layers that divided the size of the input image into half into each layer. Each down convolution layer was followed by a batch normalization and a Leaky Relu function. Throughout each layer, the number of feature maps is doubled. This is then followed by a dense convolution layer and 6 up convolution layers back to the original size of the input image. The up-convolution layers are not normalized but a Relu activation is applied after each layer. Then the last activation layer is a tanh function. The Discriminator is a lot simpler and consists of 6 layers 5 down convolution layers followed by a leakyRelu activation with the last layer consisting of a dense layer that produces one output variable.



```
 batch_normalization_6 (Batc  (None, 128, 64, 64)      256
 hNormalization)

 re_lu_2 (ReLU)              (None, 128, 64, 64)      0

 conv2d_transpose_3 (Conv2DT  (None, 64, 128, 128)     131136
 ranspose)

 batch_normalization_7 (Batc  (None, 64, 128, 128)     512
 hNormalization)

 re_lu_3 (ReLU)              (None, 64, 128, 128)     0

 conv2d_transpose_4 (Conv2DT  (None, 64, 256, 256)     65600
 ranspose)

 batch_normalization_8 (Batc  (None, 64, 256, 256)     1024
 hNormalization)

 re_lu_4 (ReLU)              (None, 64, 256, 256)     0

 conv2d_6 (Conv2D)           (None, 3, 256, 256)      1731

 activation (Activation)     (None, 3, 256, 256)      0

=================================================================
Total params: 40,465,411
Trainable params: 40,464,179
Non-trainable params: 1,232
_________________________________________________________________
```

FIGURE 9: GAN-3 Generator



```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_7 (Conv2D)           (None, 64, 128, 128)      1792

 leaky_re_lu_5 (LeakyReLU)   (None, 64, 128, 128)      0

 conv2d_8 (Conv2D)           (None, 128, 64, 64)       131200

 leaky_re_lu_6 (LeakyReLU)   (None, 128, 64, 64)       0

 conv2d_9 (Conv2D)           (None, 256, 32, 32)       131328

 leaky_re_lu_7 (LeakyReLU)   (None, 256, 32, 32)       0

 conv2d_10 (Conv2D)          (None, 512, 32, 32)       524800

 leaky_re_lu_8 (LeakyReLU)   (None, 512, 32, 32)       0

 conv2d_11 (Conv2D)          (None, 1, 32, 32)         4609

 leaky_re_lu_9 (LeakyReLU)   (None, 1, 32, 32)         0

 flatten (Flatten)           (None, 1024)              0

 dense (Dense)               (None, 1)                 1025

=================================================================
Total params: 794,754
Trainable params: 794,754
Non-trainable params: 0
_________________________________________________________________
```

FIGURE 10: GAN-3 Discriminator

4.2.3 Model Training

Due to time and hardware limitations, we are not able to train in the huge places place365 but we are able to train on a subset of them called val_256 Another more focussed data set was used for training and result comparisons and that was the forest and roads dataset. The training method was identical to the one used in the paper but the only difference is that instead of using L1 loss which was the mean of the difference between the two values we use MAE. This change only affects the Generative loss. We also used the technique of varying the adversarial weights throughout the epochs. After epochs 10, 30, and 60 we change the weight on the adversarial loss so that it affects the generator less and less that way if values start converging at 1 for adversarial loss the generator will still be able to improve and get adversarial loss back to a reasonable number. Both the deep nets use adam optimizers for training.

$$LOSS_{rec} = \frac{1}{n}\sum_{n}||x - G(x)||$$

$$LOSS_{adv} = \frac{1}{n}\sum_{n}||D(G(x)) - 1||$$

$$LOSS_{G} = \lambda_{adv} * L_{rec} + \lambda_{adv} * L_{adv}$$

$$LOSS_{D} = ||D(x) - 1||_{2}^{2} + ||D(G(x)) - 0||_{2}^{2}$$

After epochs 10 30 and 60, the weight of how adversarial loss affects backpropagation is changed to prevent generator loss from not decrease.

5. Results

5.1 DC-GAN

DC-GAN provided very generalized results and was not able to learn much. The output produced was general and not promising. We trained the model for 4 hrs not producing impressive results but giving an insight into what can we do to improve the training.

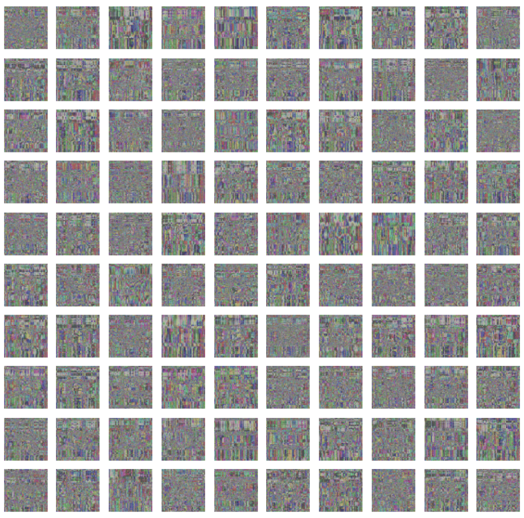


FIGURE 11: Output of GAN1

5.2 GAN-2

We trained this model for over 10 hrs and the result produced was convincing that the problem of image outpainting can be solved using local discriminators. If we use deeper networks and larger datasets like using the whole Places365 dataset of 2.1 million images can produce near-perfect results. The training and validation losses decreased over the iterations.

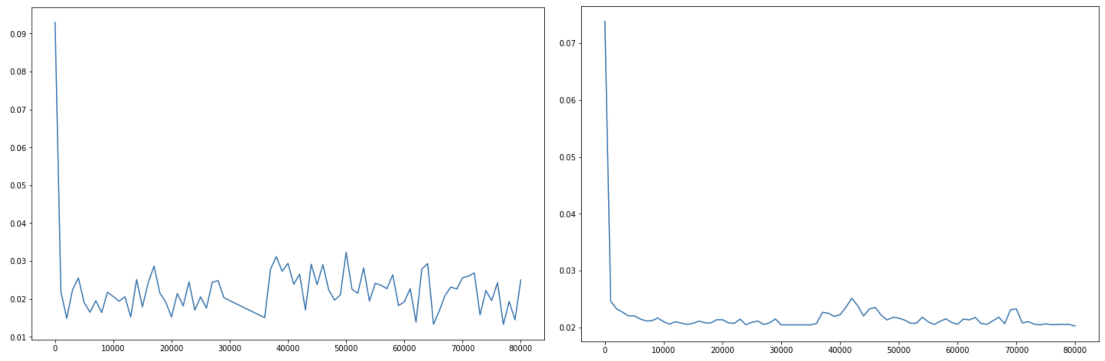
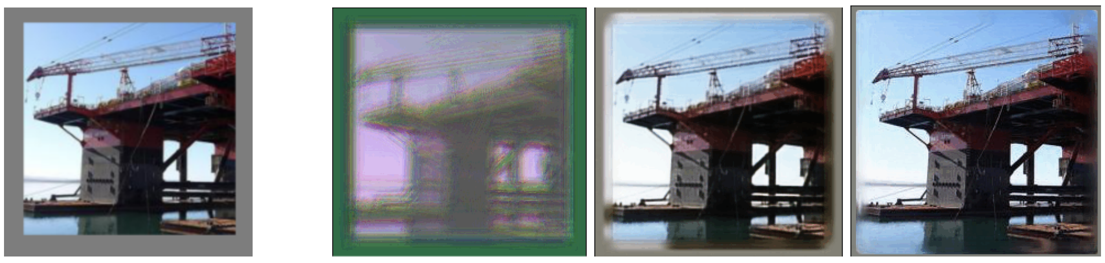


FIGURE 12: Train and validation losses vs Iterations.

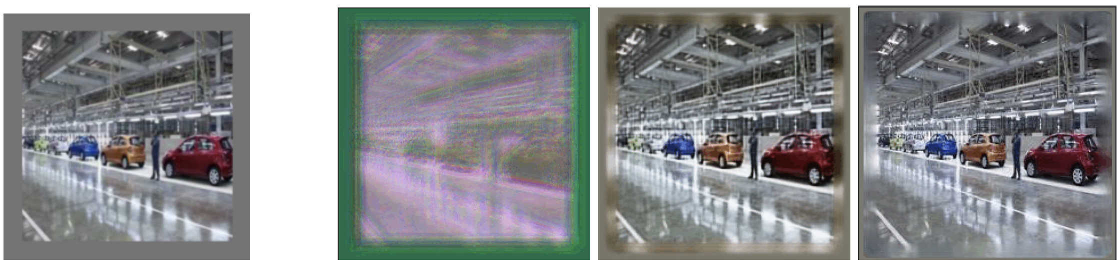
Figures 13 and 14 show the images of model learning.



Input

Training

FIGURE 13: GAN 3 training



Input

Training

FIGURE 14: GAN 3 training

5.3 GAN - 3

We trained this model for 5 hours and the results produced look decent but the performance from the more theme-focused forest and roads data set did produce better results when giving images similar to what it was trained on ie: trees and Forrest it produced more natural and believable results but The model trained on the other dataset generalized to unseen data better. One interesting thing that occurs when training on the forest and roads data set is the model can predict some building images if you give it a building as input. Both models also seem to struggle greatly when people are on the scene.

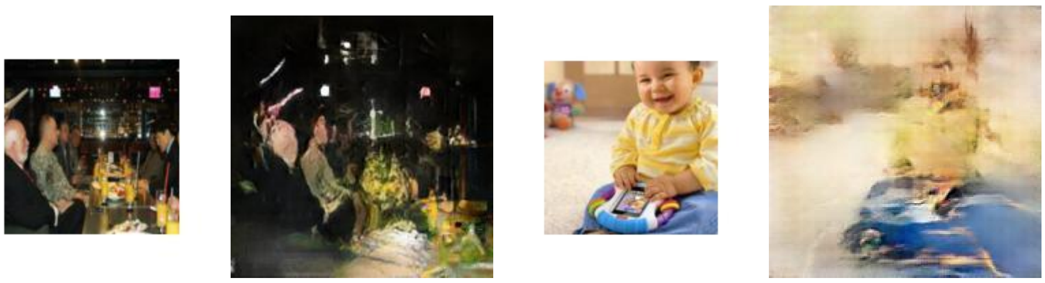


FIGURE 15: you can see that the val256-trained model(left) can handle people a lot better than the on trained Forrest-road trained model(right)

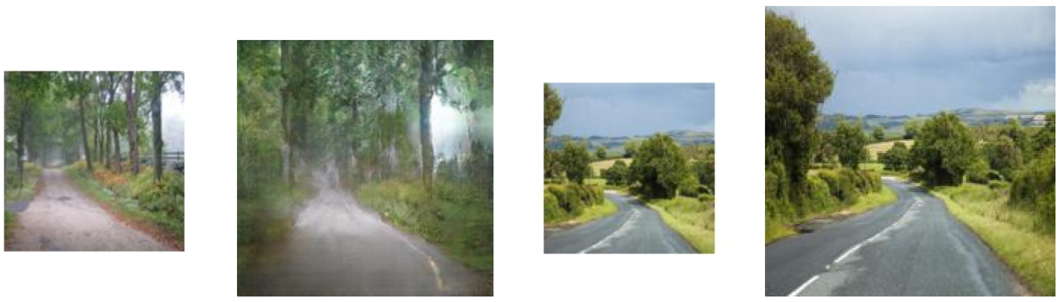


FIGURE 16: Results from Forrest-Road trained model

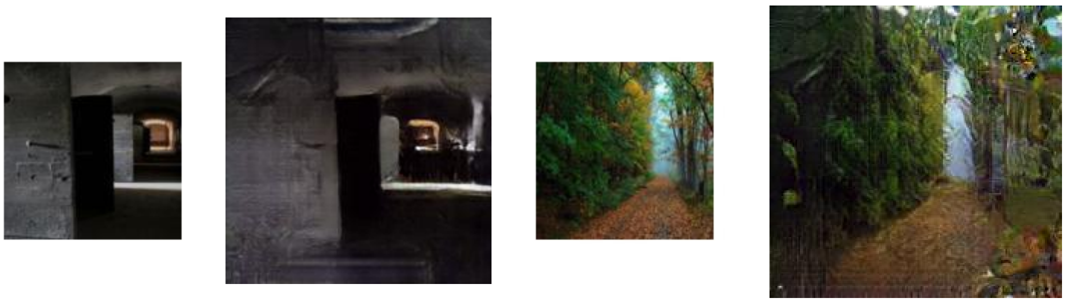


FIGURE 17: Results from Val256 trained model

From figures 15-17 you can see that Forrest and the roads trained model can produce very realistic results from inputs similar to that in the data set. You can also see how much better the Val 256 trained model is at being able to take in any input and produce a somewhat coherent output. The next logical step for improvement would be to train on the val_256 data set longer to see if we can retain the generality while having higher resolution outputs. Another idea is to incorporate the ideas proposed in the fourth paper and that would be to add a normalized diversification step and a regularization step That is stated by that paper to improve output diversification.

6. Discussion

The GAN with local discriminators and the GAN which takes into account adversarial loss while training provided the best and most promising results. With additional computation power and deeper networks, these two models maybe can produce near-perfect results. Especially if trained using the entirety of the places365 data set, the results would not only be realistic but also extremely diverse because of the variety in the larger data set. We may even be able to incorporate some transfer learning techniques to help us achieve what the [4] was trying to do. By using sparser input details say a couple of trees or a building as opposed to a square box with a good amount of detail we could probably generate some interesting scenes at the very least. This though would be more artistic than practical because it would allow the model to make more predictions about the region that needed to be shaded.

7. Conclusion

In this project, we successfully solved the problem of image outpainting using GANs. The GANs really proved to be true to their properties of generating data similar to the training set. The results can be improved using the whole dataset for training.

8. Authors' Contribution

Both team members had equal contributions to the project. CA was responsible for studying, developing, training, and evaluating the GAN using local discriminators. He also was responsible for developing the DG-GAN. LG was responsible for studying, developing, training, and evaluating the GAN that uses adversarial loss while training.

References

[1] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2020. Generative adversarial networks. Commun. ACM 63, 11 (November 2020), 139–144. https://doi.org/10.1145/3422622

[2] Sabini, M., & Rusak, G. (2018). Painting Outside the Box: Image Outpainting with GANs. *arXiv*. https://doi.org/10.48550/arXiv.1808.08483

[3] Van Hoorick, B. (2019). Image Outpainting and Harmonization using Generative Adversarial Networks. *arXiv*. https://doi.org/10.48550/arXiv.1912.10960

[4] Zhang, L., Wang, J., & Shi, J. (2019). Multimodal Image Outpainting With Regularized Normalized Diversification. *arXiv*. https://doi.org/10.48550/arXiv.1910.11481

[5] Generative adversarial network. (2022, December 3). In *Wikipedia*. https://en.wikipedia.org/wiki/Generative_adversarial_network

[6] Zhou, B., Lapedriza, A., Khosla, A., Oliva, A., & Torralba, A. (2017). Places: A 10 million Image Database for Scene Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.