EXPLORING STACKED GENERATIVE ADVERSARIAL NETWORKS

Rupesh Deshmukh rddeshmu@iu.edu
Lucas Franz lufranz@iu.edu
Xin Tu xintu@iu.edu

ABSTRACT

Traditional Generative Adversarial Networks, or GANs, are the leading synthetic image generation model, but struggle to satisfy resolution expectations. Two GAN adaptations have been developed to address this issue, the Stacked GAN and Progressive GAN. This paper explores the benefits of the Stacked GAN and tests a previously unexplored adaptation in which the stacked architecture facilitates backpropagation through the generative models. The resulting experiments show that the Stacked GAN outperforms the 128x128 pixel Single GAN for convolution and deep convolution models. Full backpropagation does not show further improvement, either due to counterproductive updates or limited training due to complexity.

1. INTRODUCTION

As the power of deep learning networks is applied to generating synthetic images, the use cases of these synthetic images are exponentially growing. The leading deep learning network for this task has been the Generative Adversarial Network or GAN for short. This network includes a model responsible for generating an image, the generator, and a model responsible for discriminating between the real or fake origin of the image, the discriminator. The generator takes a latent input randomly sampled from a normal distribution, while the discriminator takes a full image as input. The discriminator is trained using an equal number of synthetic and real images, learning to appropriately classify the image's origin. The generator in turn learns how to confuse the discriminator.

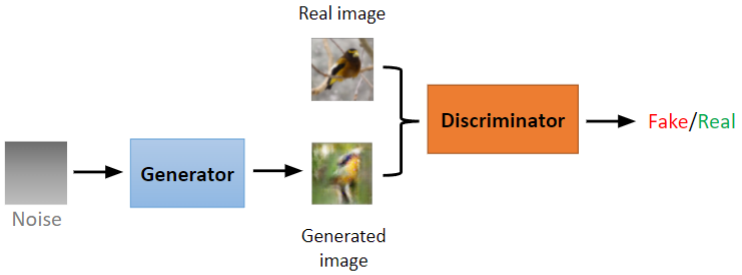


Figure 1: Traditional GAN

Training data plays a large role in the quality of synthetic images. Too few examples and the generator overfits, creating identical images and defeating its synthetic purpose. Requiring a high-resolution image can overwhelm the generator as there are too many possible pixel adjustments to account for. Having a non-centered target object can have the same effect, confusing the generator as to where best to focus its improvements.

While the first and third concerns can be addressed by applying additional attention to detail regarding the training data, the high-resolution difficulties remain for the traditional GAN architecture. Two differing approaches have emerged to address the high-resolution issue, the

Stacked GAN, and the Progressive GAN. Both rely on the same conceptual approach, beginning at a lower resolution and up-sampling to reach the desired resolution. The following experiments demonstrate this process using the Stacked GAN implementation, investigate their effect on varied layer types, and test if the Stacked network can be improved by allowing full backpropagation.

2 State-of-the-art Work on Stacked GANs

Stacked GANs are designed using a minimum of 2 Generative Adversarial Networks, each with its own generator and discriminator. The 'Stacked' adaptation combines a lower resolution GAN with a high-resolution GAN, using the low-resolution output as the input for the higher resolution model. The low-resolution network is trained first. Once complete, the weights are held static, and a larger network is connected with trainable weights. The network from latent input to now higher resolution output is then trained.

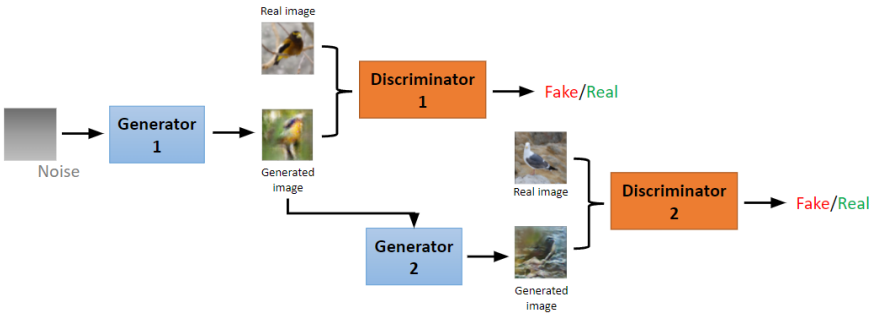


Figure 2: Stacked GAN

In our main reference article [1], the authors use a double Stacked GAN in coordination with a conditioning input to produce impressive images correlating to a user's input. The first stage creates basic low resolution images based on the text description by the user. The second stage generates high resolution images by identifying the defects in the output of Stage-I. They use a novel Conditioning Augmentation technique that encourages smoothness in the latent conditioning manifold.

Progressive GANs [2] are another popular architecture that improve the network by progressively adding new layers to both the generator and the discriminator to improve the quality of the generated image. Although they do not use stacks explicitly, the mechanism is similar and relies on additions of neural network layers to improve the GAN's performance. This paper along with some others [3][5] provided great insights to inform the direction of our project.

3 Implementation and Experiments

The goal of the experiment is to explore the advantages of Stacked GAN architectures over the traditional GAN. To do so, the network layer types, implementations, and structures are held as constant as possible, leaving the resulting differences dependent on whether a stacked network is utilized or not. Full backpropagation is easily testable by altering the trainable attribute of generator 1 layers previously held constant.

3.1 Dataset and Preprocessing

The experiments make use of the CUB 200 dataset. This dataset consists of 11,788 images of varying species of birds. The most important perk to this dataset is the consistent centering of

the target object in the image. This dataset is further helpful when testing high-resolution detail as birds by nature have many complex and diverse features. To further improve the dataset three preprocessing steps are applied. Firstly, the CUB 200 dataset images are accompanied by bounding box coordinates which are used to crop the images, focusing on the bird and further removing noise. Images are then resized to the correlating training sizes. The last step of preprocessing is to normalize the data, scaling values from 0-255 to 0-1.

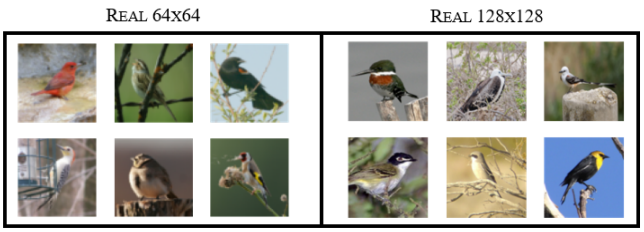


Figure 3: Training Images

3.2 Loss and Optimization

All models make use of the Binary Cross-Entropy loss function but in slightly different ways. The generator loss is based on whether a generated image was classified as real or not by the discriminator, accruing a higher loss per image that is identified as generated. The discriminator loss is based upon the correct classification of an image's label, suffering an equal loss for identifying a real image as generated or generated as real. Using these loss functions, the models are optimized using the same optimization function, either Adam or RMSprop, depending on the layer composition.

3.3 Baseline GAN

The Baseline GAN is a Traditional GAN, 1 generator and 1 discriminator, targeting synthetic images of 128x128 pixels (64x64 pixels for layer comparison).

3.3.1 Dense Architecture

The dense architecture consisted of three hidden dense layers in the generator and two hidden dense layers in the discriminator. Each layer was passed through a leaky-relu activation function, and the output was normalized using batch normalization. Dropout layers helped the performance of the discriminator. Considering the complexity of the problem, 256, 512, and 1024 nodes were used respectively in the generator, whereas 256 and 64 nodes were used in the layers of the discriminator.

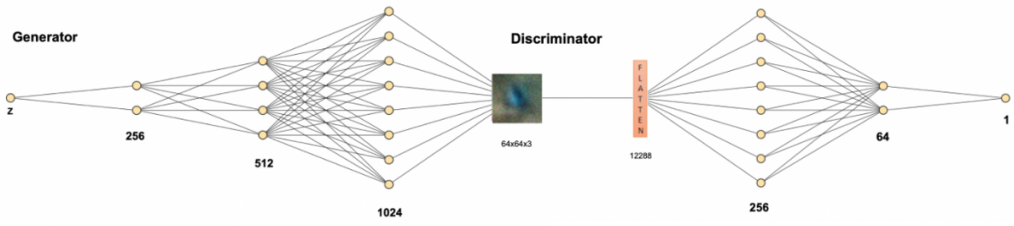


Figure 4: Dense Baseline GAN Architecture

3.3.2 Convolution Architecture

The convolution architecture makes use of 5 convolution layers and 1 convolution transpose layer. The convolution layers process different filters of the fed forward image, using the convolution transpose layer to up-sample the image size. Convolution layers use 256 filters of size 3x3 pixels and strides of 1x1. Padding the sides of the images allows the convolution layers to hold the image size constant.

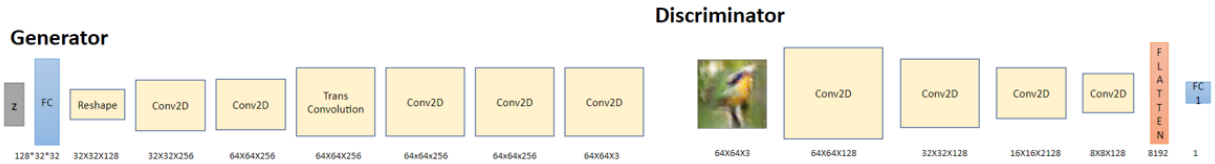


Figure 5: Convolution Baseline GAN

3.3.3 Deep Convolution Architecture

The architecture of Deep Convolution layers in this paper consists of five layers: one fully connected layer and five transposed convolution layers. Starting with the Dense layer, random noise is taken as input. Then the Conv2DTranspose layers can produce upsampling data from this random noise seed. The number of filters in the convolution layers are 256, 128, 64, and 3 respectively. Kernel size is 5 and strides size is 2. LeakyReLU activation is used for these layers and tanh is used in the last layer.

Figure 6: Deep Convolution Baseline GAN

3.4 Stacked GAN

The model architecture of StackGAN consists of the following components: stage I generator, stage I discriminator, stage II generator, and stage II discriminator. Using a random noise as an input, the stage I generator produces low resolution (64X64) images. Using these 64X64 images as inputs, the stage II generator can generate high resolution (128X128) images.

In line with typical Stacked GAN architecture, once trained, the 64x64 pixel stage 1 GAN's layer weights are held constant during stage 2 training. The Stacked GAN models are composed with similar layer architecture as their baseline, with the stage 2 generator differing in that it does not take a latent input, but instead uses a 64x64x3 image input.

3.5 Stacked Backpropagating GAN

In the Stacked GAN, stage 1 generator layer weights are held constant by setting their respective 'trainable' attributes to false. In the Stacked Backpropagating GAN these layers 'trainable' attribute is set to true. This allows the loss experienced by the stage 2 GAN's output to be backpropagated through the entire stage 1 and stage 2 generator network, updating all applicable weights.

4 Experiment Results

Because the GAN architecture creates an evolving environment, it is difficult to outline a solitary metric that will hold true when comparing different model outputs. The easiest solution is to use the "eye test" and determine the model quality by human perception. Beginning from easiest to most difficult thresholds, looking for the outline of an object, looking for the shape of the outline to resemble the target object, looking at the fill of the object, and finally looking for anatomic qualities can be comparable tests between the models.

4.1 Layer Comparison: Dense, Conv, DConv

Comparing Baseline GAN models the differing layer performance becomes readily apparent. Dense layers are able to identify the appropriate object shape but fail to detail within the object. The 128x128 baseline shows little to no difference compared to the 64x64 model. The convolution layers perform better than the dense layers, clearly developing the object and beginning to detail different aspects of it. The 128x128 model shows incremental improvement, including more detail in the forming features. The deep convolution layers demonstrated the best baseline performance. The 64x64 model shows the most identifiable birds and the 128x128 model shows further variance and detail.

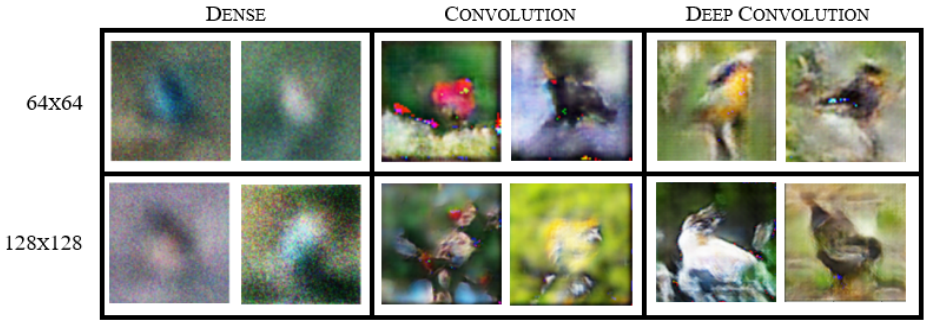


Figure 7: Baseline Results

4.2 Stacked GAN vs GAN at 128x128

When comparing the layers of the Stacked GAN and Baseline GAN the same performance order remains the same. The Stacked dense model fails to show improvement over the baseline. An argument can be made that there is a slight improvement in object shape as more variance can be seen. The stacked convolution model does show a noticeable improvement over the baseline. The stacked images show a clear object shape, further differentiating anatomic features. The image coloring is also of note, showing wider variance while following anatomic color patterns. The stacked Deep Convolution model also shows improvement over the baseline in respect to the object outline and background. The object shape is further refined and better details the bird's posture in the image while the background is much smoother, improving upon the noisy baseline background.

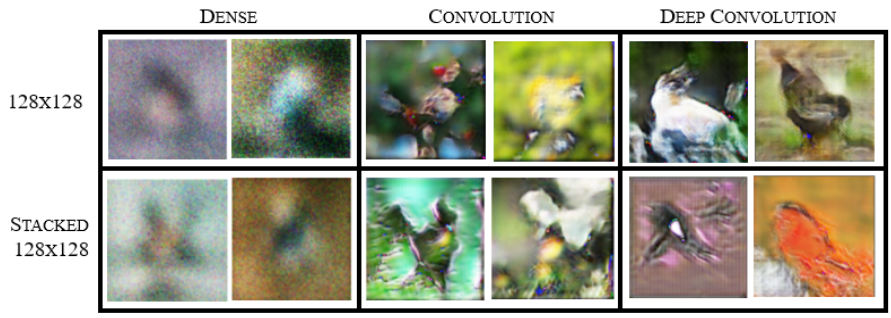


Figure 8: Stacked Results

4.3 STACKED BACKPROPAGATING GAN AT 128X128

As the Stacked models show equal or improved performance over the baseline, Stacked Backpropagated models are compared to their stacked predecessors. Again, in the dense layers, there is no noticeable change. Looking at the convolution layers, the Backpropagated model shows a decrease in performance, specifically in the detail applied to filling the identified object. The Deep Convolution model suffered from the same decreased performance when backpropagation was instituted.

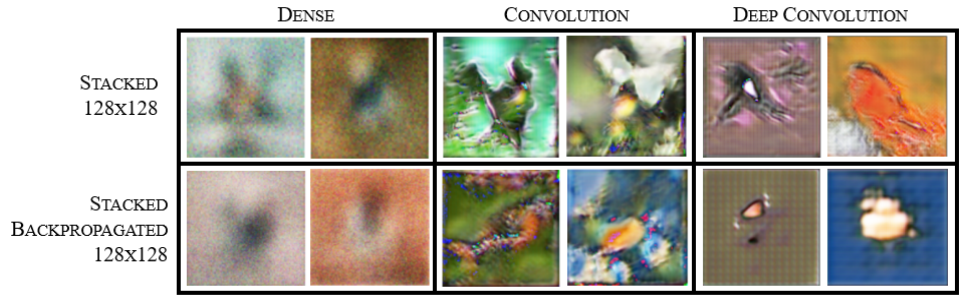


Figure 9: Backpropagated Results

5. CONCLUSION

In conclusion, these experiments demonstrate that Stacked GAN models do outperform traditional Single GAN models, but this increased performance is dependent on layer type. Dense layers as constructed for this experiment show a relatively low ceiling in synthetic image generation, only capable of producing a blurry shape resembling a bird. Convolution layers are able to add more detail to generated birds, with stacked models able to better develop and pattern anatomic features. Deep Convolution layers show the best performance, consistently producing diverse birds at a detailed quality comparable to human painting, although not achieving photo-realism.

When trying to improve the Stacked results by altering the architecture to allow for full backpropagation the experiments failed to demonstrate further improvement and instead showed a decrement. This result may be caused by generator 1 weight updates being counterproductive to generator 2 weight adjustments also applied in corresponding updates. Another cause may be attributed to the excessive running times and computation cost by including additional millions of parameters as this experiment's allotted computation power was not capable of applying extensive training to these models.

7. References

[1] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris Metaxas. “StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks”. IEEE Int. Conf. Computer Vision, 2017.

[2] Tero Karras, Timo Aila, Samuli Laine, Jaakko Lehtinen. “Progressive Growing of GANs for Improved Quality, Stability, and Variation”. Int. Conf. Learning Representations, 2018.

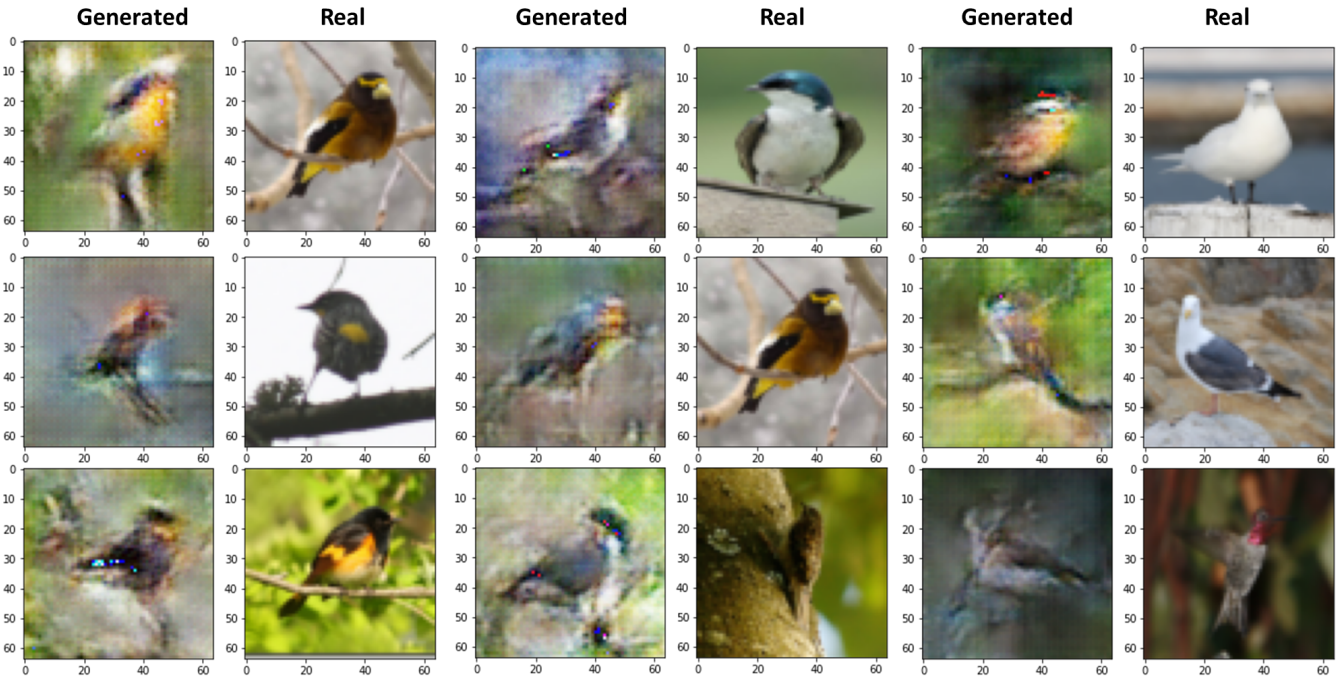
[3] Yong-Goo Shin, Yoon-Jae Yeo, Sung-Jea Ko. “Simple yet Effective Way for Improving the Performance of GANs". IEEE Transactions on Neural Networks and Learning Systems, 2019.

[4] J. Yu, X. Xu, F. Gao, S. Shi, M. Wang, D. Tao, et al., "Toward realistic face photo-sketch synthesis via composition-aided GANs", IEEE Trans. Cybern., Mar. 2020.

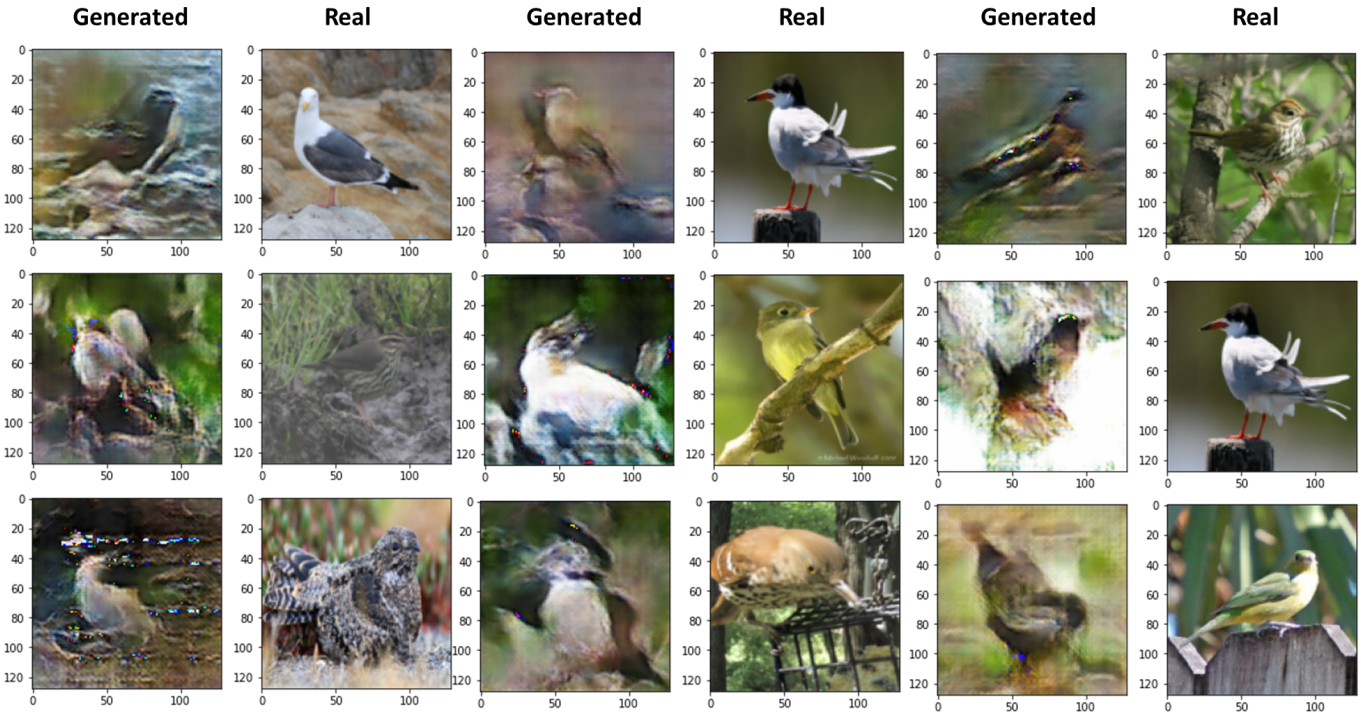
[5] S. Lin, L. Chen, Q. Zou and W. Tian, "High-resolution driving scene synthesis using stacked conditional GANs and spectral normalization", Proc. IEEE Int. Conf. Multimedia Expo., pp. 1330-1335, Jul. 2019.

8. APPENDIX

64X64 images generated by DCGAN



128X128 images generated by DCGAN



128X128 images generated by DC StackGAN