Exploring GANs for Dog Images Generation

Ayush Sharma

Abstract

Generative AI has captured the imagination of everyone since the launch of ChatGPT 1.5 years ago. Leading LMs today have multi-modal abilities. Stable diffusion and DALL-E are able to create stunning images and have already transformed many industries. Getting inspired from these developments I did this project to test the generative capabilities of GANs using the Stanford dogs dataset.

1 Introduction

While searching for a well annotated dataset that I can use to train my GANs, I came across this Kaggle competition: Kaggle Competition Link.
This Kaggle competition helped me get started and the discussions posted by the participants gave me inspiration for the various approaches I could try to improve my model. These improvements led to the images produced by my model getting significantly better as can be seen below.
I started with a simple GAN and then subsequently moved to BigGAN with conditional batch normalization, attention mechanism, spectral normalization etc highlighted in the improvements section. Through these I was able to overcome the problems of artifacts and mode collapse.
I was restricted by the maximum run-time of Kaggle notebooks and the GPU resources available for Kaggle notebooks for the number of epochs I could train the models for. I believe that the results will get better with additional training. Sample generated images along with the hyper-parameters which resulted in those images have been added to this report and link to my notebook has also been provided at the end.

2 Dataset

The Stanford Dogs dataset contains images of 120 breeds of dogs from around the world. This dataset has been built using images and annotation from ImageNet for the task of fine-grained image categorization. The dataset contains 20,580 images with annotations being of class labels and bounding boxes. The number of training images per class is varied from 1 to 100.
Link to the dataset: Dataset Link.

Figure 1: Sample image from Stanford Dogs dataset

3 Pre-processing

- Cropping : Bounding boxes for each image are loaded from XML file containing annotations. The images have diverse backgrounds and angles. Using the bounding boxes, I crop the image bringing the focus just to the dog.

- Resizing : After cropping, the image is resized to a uniform size (64x64 pixels in this case) to maintain consistency. This resizing is essential for training neural networks, which typically require input data of a consistent shape.

- Optional transformations : During the improvement process I performed some additional transformations. This includes transformations like random horizontal flipping, normalization, etc., which are standard practices in data augmentation to enhance model robustness and prevent over-fitting.

4 Models used : GANs and Big GANs

4.1 GANs

Generative Adversarial Networks (GANs) [2] are a sophisticated framework in machine learning, often used for generating new data that mimics the distribution of real data. Introduced by Ian Goodfellow and his colleagues in 2014, GANs operate through a dynamic rivalry between two distinct neural networks: a generator and a discriminator.

The generator network aims to manufacture fake data that is so plausible that it can pass as real data. It learns to map from a latent space to the data space, improving its ability to forge data as training progresses. Conversely, the discriminator network evaluates the authenticity of both the incoming real data and the synthetic data produced by the generator. It's trained to perfect its accuracy in discerning genuine data from counterfeits.

Training GANs involves a min-max game where the discriminator tries to maximize its ability to correctly classify data as real or fake, while the generator strives to minimize the discriminator's ability to detect its fabrications. This adversarial process continues until the generator produces data indistinguishable from actual data, leading to an equilibrium where the discriminator is only as good as random guessing.

GANs have demonstrated significant success in various applications such as photo-realistic image generation, art creation, and even simulating 3D environments. They are also pivotal in semi-supervised learning scenarios, data augmentation, and domain adaptation, showcasing their versatility and power in the AI field.

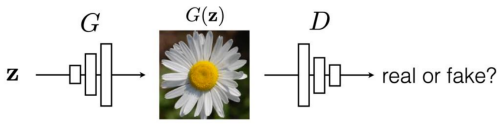

Figure 2: Example Working of GANs

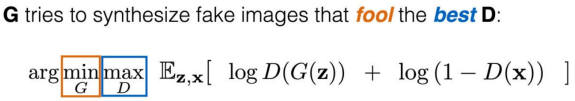

Figure 3: GAN Loss Function

4.2 BigGANs

BigGANs [1], or Big Generative Adversarial Networks, are a scale-up of the traditional GAN architecture designed to produce high-resolution, high-fidelity images. Introduced by researchers at DeepMind, BigGANs apply the concept of scaling up both the model size and the batch size to enhance the quality of the generated images. The core principle behind BigGANs involves increasing the depth and width of both the generator and discriminator networks, which allows the model to learn more complex and detailed data distributions.

An important aspect of BigGANs is the use of class-conditional generation techniques. By conditioning the generation process on class labels, BigGANs not only generate visually appealing images but also ensure that these images are relevant to specific classes, enhancing both the diversity and fidelity of the outputs. Furthermore, BigGANs incorporate techniques like orthogonal regularization and Truncation Trick. Orthogonal regularization helps in stabilizing the training process by maintaining the orthogonality of the weight matrices, preventing the collapse of different features into indistinguishable outputs. The Truncation Trick, on the other hand, involves modifying the input noise distribution to trade off between image diversity and image fidelity. By truncating the noise vector, BigGANs can generate more realistic and detailed images, albeit at the cost of reduced variability.

The introduction of BigGANs has pushed the boundaries of what's possible with generative models, significantly improving the realism and detail of generated images, particularly in tasks requiring complex image synthesis like photo-

realistic landscapes and detailed object images. However, the computational cost associated with training such large models is substantial, making them less accessible for individual researchers and smaller institutions.

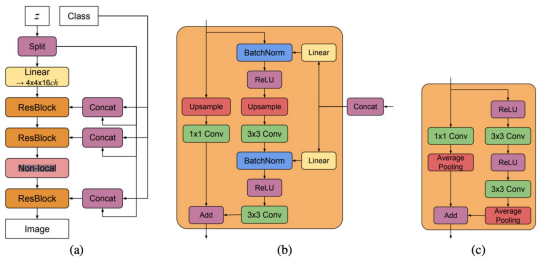

Figure 4: Typical BigGAN Architecture

5 Improvements

5.1 Removing Artificats

Artifacts in Generative Adversarial Networks (GANs) refer to imperfections or inconsistencies in the generated images that make them look unnatural or obviously synthetic. These artifacts can manifest as distortions, blurred edges, unrealistic textures, or repetitive patterns that do not appear in genuine images. The presence of such artifacts is primarily due to the generator network not learning the data distribution accurately enough, often resulting from issues in the training process such as insufficient model capacity, inadequate training data, or poor convergence between the generator and discriminator.
I used following techniques to overcome the problem of artifacts.

- Improved Network Architecture: Employing deeper or more sophisticated network architectures can enhance the learning capabilities of GANs. Architectures like BigGANs use large-scale networks conditioned on class labels, which help in generating more detailed and coherent images.

- Regularization Techniques: Implementing regularization methods such as spectral normalization or gradient penalty helps stabilize the training process. These techniques constrain the weights or the gradients, preventing the model from overfitting to specific features in the training data that could lead to artifacts.

5.2 Conditional Batch Normalization

This technique is employed in the form of ConditionalNorm classes within the generator. Conditional Batch Normalization allows the model to modify the normalization parameters based on additional information (such as class labels). This is particularly useful in generating images that are conditional on specific input classes, enhancing the diversity and specificity of the generated outputs.

5.3 Attention Mechanism

The Attention class within both the generator and discriminator introduces self-attention to the GAN architecture. This mechanism allows the networks to focus on relevant parts of the image more effectively, improving the detail and quality of generated images. Attention mechanisms have been shown to enhance the learning of dependencies in data, particularly in complex image datasets.

5.4 Spectral Normalization

Used in various convolutional and linear layers (spectral_norm), this technique stabilizes the training of the discriminator by constraining the Lipschitz constant of the layer functions. Spectral normalization helps in controlling the explosion of gradients during training, leading to a more stable GAN training process and often better image quality.

5.5 Residual Blocks

Both the generator and discriminator utilize residual blocks (ResBlock_G and ResBlock_D), which help in building deeper networks by allowing gradients to flow through the network more effectively. This is crucial for learning detailed and complex patterns in images, as residual blocks help in avoiding the vanishing gradient problem in deep networks.

5.6 Exponential Moving Average (EMA)

While not detailed in the provided segments, the mention of an EMA strategy indicates a technique often used to stabilize and improve the performance of the generator. By keeping a moving average of model weights over training epochs, EMA can help in producing smoother and more visually appealing results.

5.7 Leaky ReLU Activation

This activation function is used throughout the architecture, allowing for a small gradient when the unit is not active. Leaky ReLU can help prevent issues of ”dying neurons” in a network, contributing to a more robust learning process.

6 Hyperparameters

Below are the hyperparameters for best results in each of the three approaches.

Hyper-parameter	Description	Value
`seed`	Seed for randomness control	2019
`ComputeLB`	Flag to compute leaderboard score	True
`DogsOnly`	Flag to process only dog images	True
`BATCH_SIZE`	Batch size for training	32
`NUM_WORKERS`	Number of worker threads for data loading	4
`EMA`	Exponential Moving Average for model weights	False
`LABEL_NOISE`	Use label noise during training	False
`LABEL_NOISE_PROB`	Probability of label noise	0.1
`lr`	Learning rate for training	0.5 initially
`epochs`	Number of training epochs	Dynamic, starts at 10
`batch_size`	Training batch size	32 (from BATCH_SIZE)
`MEAN1, MEAN2, MEAN3`	Mean values for image normalization	0.5, 0.5, 0.5
`STD1, STD2, STD3`	Standard deviation values for image normalization	0.5, 0.5, 0.5
`img_size`	Image size for processing	64

Figure 5: Hyperparameters: First (Initial) Approach

Hyperparameter	Value	Description
`nz` (Noise Dimension)	100	Size of the latent vector/input noise dimension.
`nfeats` (Features in Generator)	32	Number of feature maps in the first layer of G.
`nchannels` (Output Channels)	3	Number of channels in the output image (RGB).
`lr` (Learning Rate for D)	0.0002	Learning rate for the Discriminator.
`lr_d` (Learning Rate for G)	0.0002	Learning rate for the Generator.
`beta1`	0.5	Beta1 hyperparameter for Adam optimizer.
`epochs`	1000	Number of training epochs.
`batch_size`	Dynamic	Number of samples per batch (set during training).
`real_label`	0.7	Label smoothing value for real images.
`fake_label`	0.0	Label value for fake images.
`alpha` (PixelwiseNorm & MinibatchStdDev)	1e-8	Numerical stability parameter in normalization.
`lr_scheduler` intervals	epochs//200	Interval for learning rate scheduler adjustments.

Figure 6: Hyperparameters: Second (Improved) Approach

Hyperparameter	Value	Description
`seed`	2019	Seed for random number generators to ensure reproducibility.
`BATCH_SIZE`	32	Number of samples in each training batch.
`NUM_WORKERS`	4	Number of worker processes for loading data.
`EMA`	False	Whether Exponential Moving Average is used for model weights.
`LABEL_NOISE`	False	Whether label noise is introduced during training.
`LABEL_NOISE_PROB`	0.1	Probability with which to flip labels if label noise is introduced.
`img_size`	64	Dimension (width and height) to which all images are resized.
`MEAN1, MEAN2, MEAN3`	0.5, 0.5, 0.5	Mean values for normalization of image channels (RGB).
`STD1, STD2, STD3`	0.5, 0.5, 0.5	Standard deviation values for normalization of image channels.
`lr_G`	0.0003	Learning rate for the Generator.
`lr_D`	0.0003	Learning rate for the Discriminator.
`beta1`	0.0	Beta1 hyperparameter for the Adam optimizer.
`beta2`	0.999	Beta2 hyperparameter for the Adam optimizer.
`nz` (Noise Dimension)	120	Size of the latent vector/input noise dimension.
`epochs`	250	Number of training epochs.
`n_ite_D`	1	Number of iterations for updating the Discriminator per GAN training loop.

Figure 7: Hyperparameters: Third (Final) Approach

7 Runtime

Runtimes were large due to limited resources on Kaggle.

- First (initial) approach took 4 hours to run.
- Second (Improved) approach took around 12+ hours.
- Third (Final) approach also took around 12+ hours.

8 Results

Below are the resultant images from my three approaches.

Figure 8: First (Initial) Approach - Epoch 1/9

9 Limitations

9.1 GANs

Despite their impressive capabilities, Generative Adversarial Networks (GANs) come with notable limitations. One major challenge is their training stability; GANs are notoriously difficult to

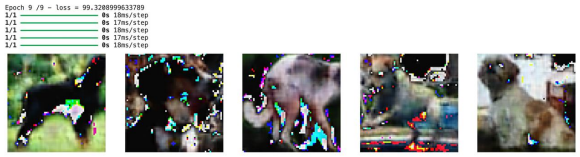

Figure 9: First (Initial) Approach - Epoch 9/9

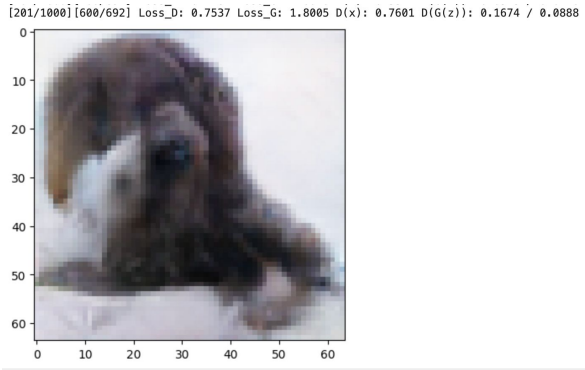

Figure 10: Second (Improved) Approach - Epoch 201/750

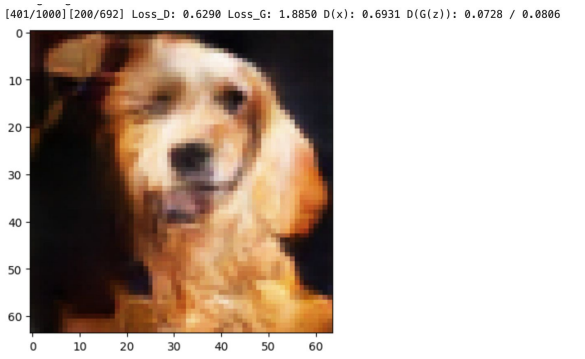

Figure 11: Second (Improved) Approach - Epoch 401/750

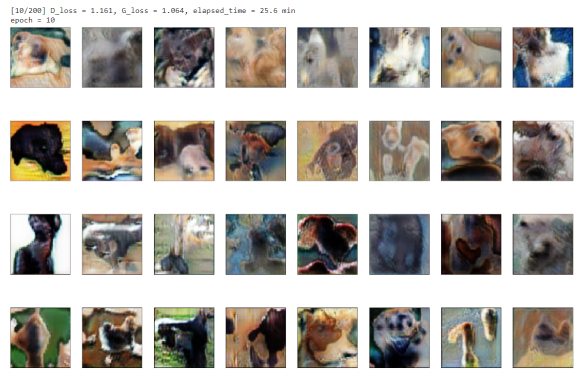

Figure 12: Third (Final) Approach - Epoch 10/200

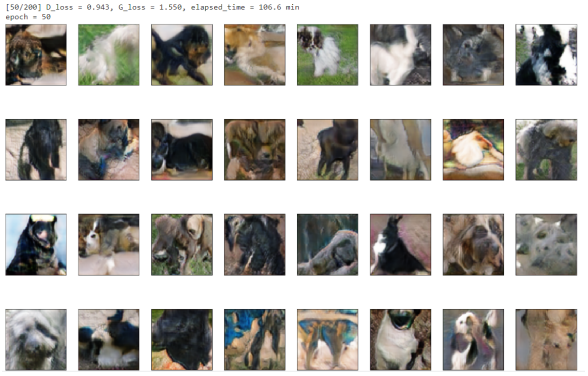

Figure 13: Third (Final) Approach - Epoch 50/200

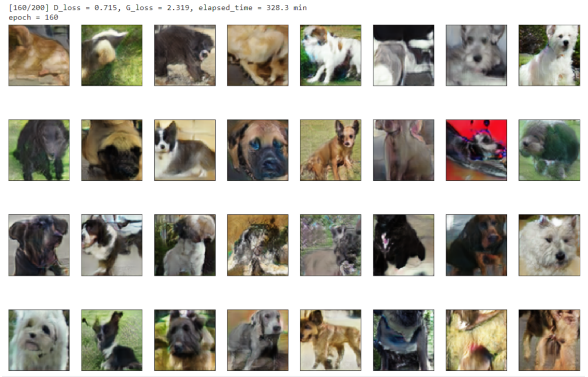

Figure 14: Third (Final) Approach - Epoch 160/200

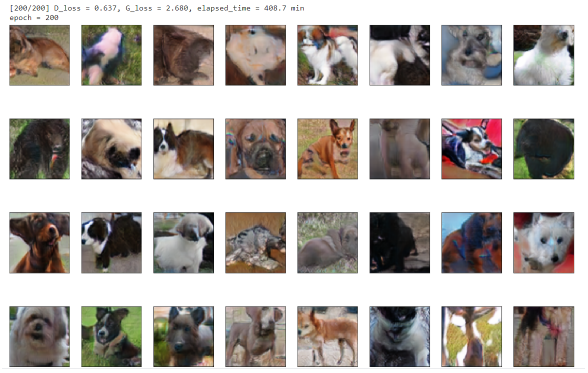

Figure 15: Third (Final) Approach - Epoch 200/200

train effectively without the networks oscillating or failing to converge. This instability often arises from the adversarial setup itself, where the generator and discriminator can end up in a continuous loop of one-upmanship without finding a stable solution. Another issue is mode collapse, where the generator learns to produce only a limited variety of outputs, ignoring significant portions of the input space and thus reducing the diversity of the generated data. Furthermore, GANs require substantial computational resources and large datasets, which can be a barrier in resource-constrained environments. Additionally, evaluating the quality of the generated data can be subjective and lacks a definitive, consistent metric, complicating the assessment of GAN performance. These limitations highlight the need for ongoing research to refine their architecture and training methodologies to harness their full potential while mitigating these downsides.

9.2 Kaggle and Limited Resources

I did my project on Kaggle and the maximum duration for which I can get free resources/GPUs is 12 hours, so the number of epochs for training were limited due to this limited time. Also, due to dataset size and unavailability of free cloud resources, I was forced to use Kaggle.

10 Future Work: Diffusion Models

I aim to use Diffusion Model as my future work for generating dog images. While Generative Adversarial Networks (GANs) excel in generating high-resolution images, they often face challenges in image quality when compared to newer techniques like diffusion models.

Diffusion models [3], which gradually construct images through a process of adding and removing noise, have demonstrated a superior ability to generate more coherent and high-fidelity images across a variety of tasks. GANs, on the other hand, can sometimes produce artifacts or inconsistencies in the images due to their adversarial training dynamics. These issues often manifest as distortions or unnatural features, which can detract from the realism of the generated images.

Moreover, GANs can struggle with ensuring global coherence in generated images, meaning that while parts of an image might look realistic, the overall image composition might not be plausible or harmoniously structured. Diffusion models, benefiting from their iterative refinement process, tend to be better at capturing and maintaining complex textural and contextual relationships within images, leading to more realistic and visually appealing results. This advantage makes diffusion models particularly effective in tasks requiring high levels of detail and consistency, such as in medical imaging or high-resolution landscape generation.

11 Conclusion

Through this project, I was able to show that GANs can be trained to generate decent quality images. The resultant images can definitely be even better with increase in number of training epochs.

12 Link to My Work

Notebook/Code Link: Code Link.

References

[1] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis, 2019.

[2] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[3] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020.