# Generating Photorealistic Images of Dogs using Generative Adversarial Networks (GANs)

Individual Final Report

George Washington University

Columbian College of Arts & Sciences

DATS 6303 Deep Learning

Final Project - Group 7

Name: Ei Tanaka

Date: December 8, 2023

## 1. Introduction

Our project aims to generate realistic images of dogs using Generative Adversarial Networks (GANs). The fascinating diversity and complexity of dog breeds present a unique challenge for synthetic image generation, making this project innovative and a significant step forward in AI-generated art and data enhancement for machine learning. We utilize Deep Convolutional Generative Adversarial Networks (DCGANs) and Unit2D diffusers to create square images of dog breeds. Our primary goal is to improve the realism of these generated images. To achieve this, we experiment with different loss functions, adjust hyper-parameters, and add more convolutional layers to our models.

## 2. Description of Individual Work

I was responsible for the following tasks:

- Data Preprocessing
- DCGAN Model Building and Training

## 3. Data Preprocessing

To facilitate ease and reproducibility in handling the Stanford Dogs dataset, the StanfordDogImage class was created. This class simplifies fetching, verifying, and preparing the dataset containing an extensive collection of dog images. These images are divided into training and testing sets. Utilizing the existing bounding box data, the class offers the functionality to crop images, focusing specifically on the dogs. Additionally, it transforms the images into a tensor format compatible with PyTorch and assesses the dataset's breed distribution, which is crucial for ensuring a balanced approach in model training.

## 4. DCGAN

The Deep Convolutional Generative Adversarial Network (DCGAN) represents a substantial advancement in deep learning, specifically in generative models. Developed as an extension of the original Generative Adversarial Networks (GANs) introduced by Ian Goodfellow and colleagues in 2014. DCGANs incorporate the principles of convolutional neural networks (CNNs) into the GAN architecture, making them particularly adept for tasks related to image data. A notable innovation in DCGANs is convolutional layers in both the generator and discriminator, enhancing their suitability for image-related tasks. Additionally, DCGANs address the challenges of training traditional GANs, which were prone to model collapse, leading to limited diversity in the generated samples. DCGANs introduce several architectural constraints to overcome these challenges, resulting in more stable training and higher-quality outputs. Furthermore, DCGANs have demonstrated their ability in unsupervised feature learning, making them valuable in applications such as image recognition and classification.

## 4.1 The Structure of DCGAN

In terms of architecture, DCGANs make essential modifications: they employ stridden convolutions in the discriminator to reduce spatial dimensions of images and use fractionally-stridden convolutions, or deconvolutions, in the generator for upscaling the input noise into a full-sized image. Both the generator and discriminator incorporate batch normalization, stabilizing the training process. Leaky ReLU activation functions, particularly in the discriminator, prevent the dying ReLU problem and promote gradient flow during training.

The training algorithm of DCGANs also deviates from traditional GANs. Training alternates between updating the discriminator, which learns to differentiate between authentic and generated images, and the generator, trained to produce images increasingly indistinguishable from actual images. Commonly, DCGANs employ a binary cross-entropy loss function to measure the discrepancy between the discriminator's predictions and the actual labels of images, whether real or fake. The discriminator (D) and generator (G) loss functions can be represented mathematically, showing the intricate relationship between these two components in the learning process. A typical DCGAN architecture comprises multiple layers of convolution for the discriminator and deconvolution for the generator, underlining the complexity and sophistication of this model in handling image data.

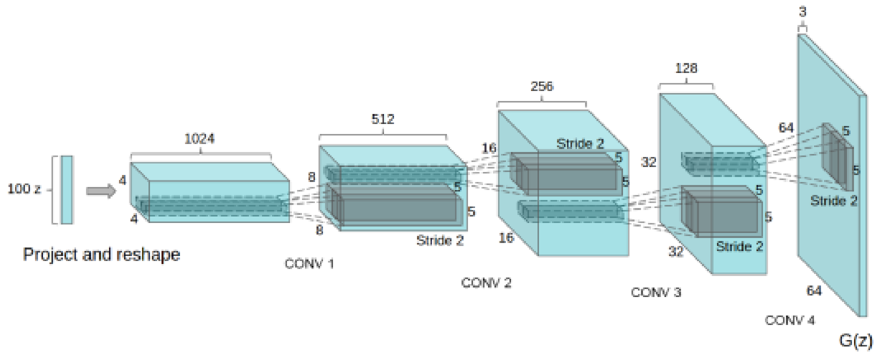

Fig-1: DCGAN Generator Architecture

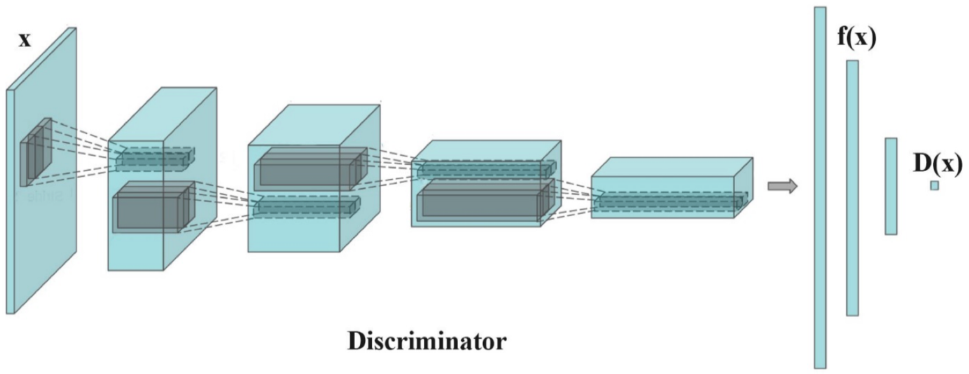

Fig-2: DCGAN Discriminator Architecture

The cost function for DCGAN is the same as GAN.

$$min_D max_G V(G, D) = \mathbb{E}_x \log\big(D(x, y)\big) + \mathbb{E}_z \log\big(1 - D(G(z, y_z), y_z)\big)$$

## 4.2 Training

First, to train the Generator in a DCGAN, ensure the Discriminator's weights remain fixed. Begin by inputting random noise into the Generator to produce fake images. Then, feed these fake images into the already-trained Discriminator to evaluate the likelihood of them being perceived as accurate. During the training of the Generator, the goal is to have the input noise classified as class 1. Therefore, use the target label 1 to compute the loss and gradients, which will update the Generator's weights.

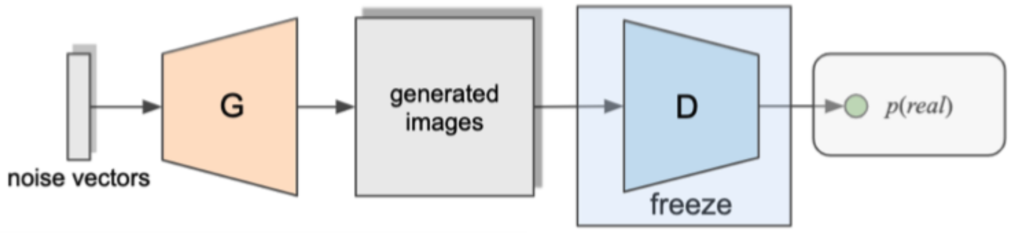

Fig-3: Training Generator

Training the Discriminator for a DCGAN involves several key steps. All weights within this network are initialized. The initialized Generator generates a batch of fake images from random noise inputs. Next, real images, randomly selected from the original dataset, are labeled as class 1, while the fake images produced by the Generator are labeled as class 0. Lastly, the Discriminator is trained to differentiate between these real and fake images, during which its weights are updated accordingly.

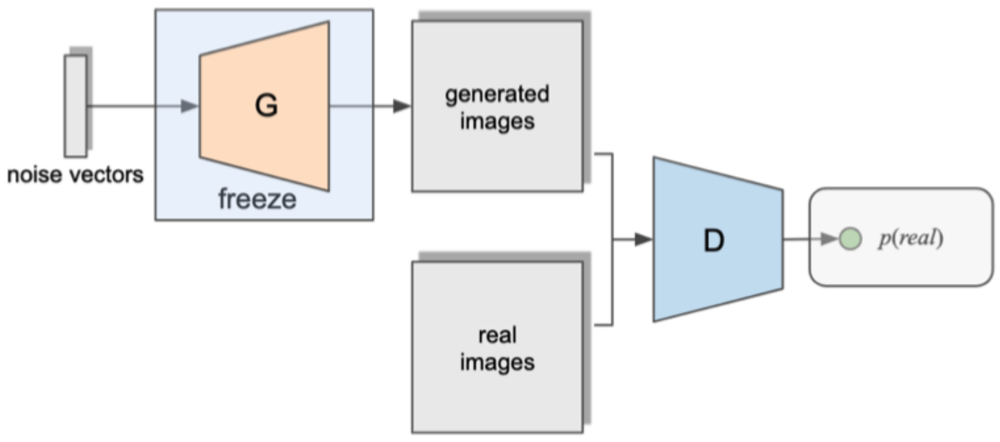

Fig-4: Training Discriminator

## Train Details

- Preprocessing, scaled to a range of tanh activation [-1, 1]
- Images will be preprocessed to match the input requirements of the network. Since our generator uses a Tanh activation function in the output layer, images will be scaled to the range of [-1, 1] to align with the Tanh output.
- Batch_Size = 64
- The size of mini-batches is set at 64. This size is often chosen based on computational constraints (like memory limits) and how well it fits the model's learning dynamics.
- Weight Initialization: Weights of convolution layers will be initialized from a normal distribution with a mean of 0.0 and a standard deviation of 0.02, which is a common practice in training deep neural networks to prevent models from being stuck at the start of training.
- Learning Rate: An initial learning rate of 0.001 with a Beta value of 0.5 for the Adam optimizer is used. The learning rate might need to be adjusted based on the model's convergence rate during training.
- Non-saturating activation functions(ReLU, LeakyReLu): The saturating loss function cannot provide sufficient gradient for G to learn. It also comes with a vanishing gradient problem in this minimax game. Implementing non-saturating activation allows the passing of negatives with a small value.

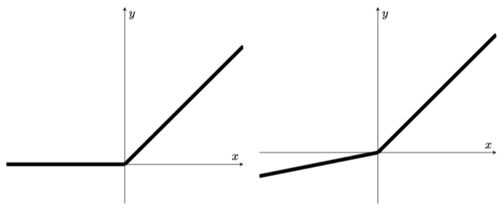

Fig-5: ReLU vs LeakyReLU

## 5. Results

### 5.1 Model Construction

Here is the final architecture summary.

| Generator | Discriminator |
|---|---|
| Input Noise Vector $z \in R^{100}$ | Input RGB Image $\in R^{3*64*64}$ |
| ConvTranspose2D, 64*8, stride=1 + BN + ReLU | Conv2D, 64 + LReLU |
| ConvTranspose2D, 64*4, stride=2 +BN + ReLU | Conv2D, 64*2 + BN + LReLU |
| ConvTranspose2D, 64*2, stride=2 + BN + ReLU | Conv2D, 64*4 + BN + LReLU |
| ConvTranspose2D, 64, stride=2 + BN + ReLU | Conv2D, 64*8 + BN + LReLU |
| ConvTranspose2D, 3 + Tanh | Conv2D, 1 + Sigmoid |
| Output Fake Images $\in R^{3*64*64}$ | Output Probability $\in R^{1}$ |

### 5.2 Model Performance

In the context of DCGAN compared to a simple GAN with an MLP structure, it's observed that a finely tuned deep convolutional structure in DCGANs can discern more intricate details in images. However, DCGANs may perform worse than a basic GAN with MLP without proper tuning, sometimes producing images with block-like artifacts. Even after addressing these issues, DCGANs can still struggle with generating images that closely resemble the target, indicating a misdirection in the learning process of both the Discriminator and Generator. Despite some improvements in image quality, DCGANs are not immune to the mode collapse issue. The tuning process for DCGANs, involving two deep CNNs, is significantly more complex and sensitive to changes than basic GANs, often requiring extensive effort and experimentation. Solutions from other studies or projects may not always be directly applicable or effective in enhancing the performance of the GAN model in specific projects.

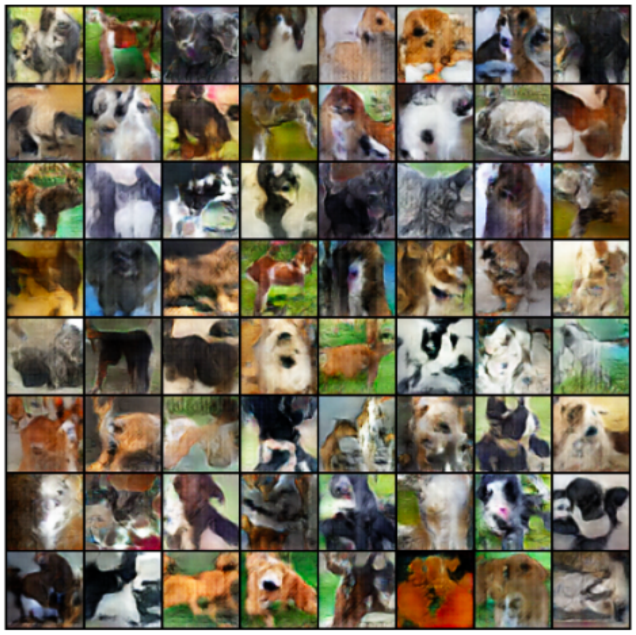
Fig-13: Generated Dog Images after 100 epoch

## 5.3 Problems

In the context of utilizing DCGAN for a dog classification task, one of the primary challenges encountered is the limited variety in the Generator's outputs, often leading to mode collapse. This phenomenon, especially prevalent in its partial form, poses a significant hurdle. To counter this, techniques such as Conditional GAN (CGAN) are introduced, where the model is influenced by specific labels, in this case, the various dog breeds in the dataset. Additionally, GANs can face other issues like training instability and difficulty in balancing the training of the Discriminator and Generator, requiring careful tuning and experimentation to achieve optimal performance.

# 6. Summary and Conclusion

In conclusion, this project successfully explored the generation of photorealistic dog images using DCGANs, highlighting the complexity and potential of GANs in image synthesis. Despite challenges like mode collapse and the need for precise tuning, the project demonstrated the effectiveness of DCGANs in producing diverse and realistic images. The use of Conditional GANs to address partial mode collapses, and the careful balance in training the Discriminator and Generator underscore the nuanced approach required in deep learning projects. This endeavor advances the field of AI-generated art and contributes valuable insights into the capabilities and limitations of GANs in real-world applications.

## 7. Calculates

The percentage of the code that I found or copied from the internet is 80%.

## References

A. Radford, L. Metz, S. Chintala, Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks (2015)

Adorni, G., Boelter, F., & Lambertenghi, S. C. (2023). Investigating GANsformer: A Replication Study of a State-of-the-Art Image Generation Model. arXiv preprint arXiv:2303.08577.

Background: What is a Generative Model? (n.d.). Google Developers. https://developers.google.com/machine-learning/gan/generative

DCGAN Tutorial — PyTorch Tutorials 1.6.0 documentation. (n.d.). Pytorch.org. https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html

Ian Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, S. Ozair, Y. Bengio, Generative Adversarial Networks (2014)

Khosla, A., Jayadevaprakash, N., Yao, B., & Li, F.-F. (n.d.). Novel Dataset for Fine-Grained Image Categorization: Stanford Dogs. Retrieved April 27, 2022, from http://people.csail.mit.edu/khosla/papers/fgvc2011.pdf

Grigoryev, T., Voynov, A., & Babenko, A. (2022). When, why, and which pretrained GANs are useful?. arXiv preprint arXiv:2202.08937.

Radford, A., Metz, L., & Chintala, S. (2016). Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. ArXiv:1511.06434 [Cs]. https://arxiv.org/abs/1511.06434v2

Sha, M. A. (2021, January 5). Training Generative Adversarial Networks (GANs) for DOG using PyTorch. Medium. https://medium.com/@mirjanalisha/training-generative-adversarial-networks-gans-for-dog-using-pytorch-26d15ec73d1b

Stanford Dogs dataset for Fine-Grained Visual Categorization. (n.d.). Vision.stanford.edu. http://vision.stanford.edu/aditya86/ImageNetDogs/

Stanford-dogs/data/stanford_dogs_data.py at master · zrsmithson/Stanford-dogs. (n.d.). GitHub. Retrieved December 9, 2023, from https://github.com/zrsmithson/Stanford-dogs/blob/master/data/stanford_dogs_data.py