# Generating Dog Images with Generative Adversarial Networks

Ioannis Theodosiou , Gabriele Morello

January 2024

## 1 Problem Statement

The purpose of this project was the training and deployment of a model able to generate dog images from random noise. Generating new data is particularly useful when training new models and provide us with more insights about the underlying distribution.

## 2 Dataset

We used the Stanford dogs Dataset[2]2w for the purpose of this project. It 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. Specifically it contained:

- Number of categories: 120
- Number of images: 20,580
- Annotations: Class labels, Bounding boxes

. For a Generation task like ours we only made use of the dog images and ignored any class labels and bounding boxes. The images though did not have much consistency regarding the position and the size of the dogs while there were also major color differences in the background of similar dogs that opposed a problem to our generation task.

## 3 Agorithm-Mehtod

The method we used to generate dog images was with a Generative Adversarial Network (GANS) [1]. Such networks are a class of deep learning models that consists of two networks competing against each other like in the context of a game. The first network is called the Generator and is responsible of trying to generate realistic images from an noise input vector and model the distribution $P(X|Y)$ where X are the features and Y is the class. The generated image is then passed to the discriminator along with a stream of images from the ground-truth data set (real dogs).T he discriminator models $P(Y|X)$ and we expect the discriminator to accurately classify the ground-truth images as real and the generated images as fake. These two models exist in a double feedback loop. The discriminator is in a feedback loop with the ground-truth data set and the generated images, while the generator is in a feedback loop with the discriminator. Generator is improving by trying to generate realistic images to decept the discriminator while the discriminator consistently tries to recognize the synthetic images.

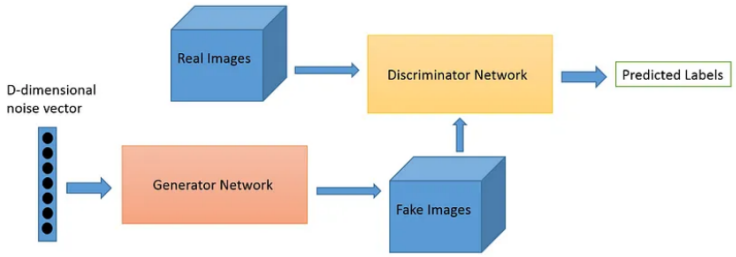


Figure 1: GAN structure

In our GAN architecture we used DCGANs, a variant of the vanila GAN that uses convolutional layers instead of fully connected layers. This has proven to give better results when dealing with images.

# 4 Implementation Details

## 4.1 Feature Pipeline

In this part we take care of the data preprocessing part were the code runs on CPU. We perform the following:

- image resizing to 64x64 pixels
- image center cropping
- random horizontal flip
- random rotations and change of brightness
- pixel value normalization

## 4.2 Training Pipeline

The training pipeline can also be run for more efficiency on GPU. For our training we used the GPU provided by google colab and we saved our model on google drive before pushing it to HuggingFace. The training pipeline consists of the the following implementation details.

1. We defined a DCGAN as Generator network with 5 blocks of convolutional 2d transpose, batch normalization and Relu activation function and a Descriminator network with 5 blocks of Convolutional Layers with batch normalization and Leaky Relu activation function.
2. For our training hyperparameters we used a $lr = 0.0003$ and an Adam optimizer with beta values = (0.5 , 0.999) for both discriminator and generator networks and trained for 100 epochs.
3. We used The binary cross entropy loss for both network training.

# 5 Evaluation Metrics

To evaluate our results we computed losses and accuracies for both the generator and discriminator, the generator shows a loss of 0.5474 which means the the generated images are moderately challenging for the discriminator to distinguish from real. The discriminator's higher loss of 0.9649 and high real accuracy (96.875%) suggests it excels in identifying real images but low fake accuracy (31.25%) means that it struggles to accurately classify fake images which is exactly the goal of a GAN network. We are very satisfied with these results as they reflect a properly trained GAN where the discriminator recognizes real images as real with a very high accuracy and the generator can produce a substantial rate of images (almost 70%) that can fool the discriminator.

# 6 Results

Below we provide a sample of some generated dog images from our Generator Network with input random noise. We can see the although the dogs are not perfect we can clearly distinguish their key characteristics.

# 7 How to run

1. Run on Colab on CPU the dogs_cpu.ipynb notebook to extract images and preprocess pictures
2. Run on Colab on GPU the dogs_gpu.ipynb notebook to train the model and get checkpoints
3. Run on Colab on CPU the dogs_inference_and_eval.ipynb notebook to evaluate the model and see some results

We did an Huggingface Space to generate images on demand accessible here.

# References


[1] Ian J. Goodfellow. Generative adversarial nets. *Universite de Montreal*, 2014.

[2] Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Li Fei-Fei. Stanford dogs dataset. *Stanford University*.


Figure 2: Generated Dogs