CS-E4890 Deep Learning - Project ProGAN

Miika Kanerva, 474021
Jack Nurminen, 648569

GitHub Repository

1. Problem and data description

In this project we decided to compare the performance of two different Generative Adversarial Network (GAN) models. The first model is a Progressive GAN (ProGAN) and the second is a Deep Convolutional GAN (DCGAN). To train the models, we decided to use the CelebA dataset consisting of more than 200 thousand celebrity images. The dataset was used because it is widely used in research, which presented the possibility of comparing our models to existing ones.

We construct a GAN model with progressive growing capabilities and train it and compare the results to the DCGAN model found in the DCGAN tutorial [3]. The results are then validated through subjective inspection of the generated images of the two models.

2. Methods

As presented in the first section, we decided to implement ProGAN and use the readily available model from DCGAN tutorial and compare the results generated by the networks. Both of these models are extensions of a GAN model, which is made of two parts; a generator and a discriminator. The generator attempts to create fake images that closely resemble the training data. The discriminator then tries to identify these fake images. If the generator is ‘perfect’ the discriminator should always guess whether the image is fake or not.

GANs are trained by showing the discriminator fake images generated by the generator and real images from the dataset, one after another. The discriminator maps the image to the labels 1 (if the image was generated) and 0 (if it was a real image). From this the loss can be calculated and gradients can be propagated backwards. For the real images weights are only updated for the discriminator whereas for the generated images the gradients are propagated through the discriminator and the generator. This way the discriminator becomes increasingly accurate in classifying real images from fakes and the generator learns to generate increasingly realistic images, in an attempt to fool the discriminator.

2.1 Progressive GAN

The motivation to use ProGAN as one of our models came from the paper from Karras et al. [1]. The idea of the progressive GAN is that you start with low-resolution images and increase the resolution with added layers. Starting the training with smaller images allows the network to discover the large-scale structure of the image distribution and then during the later layers shift the focus to the details.

The benefits of using ProGAN model is that it is more stable, because there is less class information on lower layers. In addition, the training time is reduced when compared to a normal GAN, where the training can be 2-6 faster [1]. To reduce the training time, we decided to give the first layer input size of 8x8 and double the size during each layer until the size of 64x64 is reached (instead of progressing to sizes such as 1024x1024). As the parameter count increases, so should also the number of epochs as there is more to train. Therefore we double the number of epochs after each layer addition, starting from 2 epochs for the training of 8x8 images.

A generator block in our network consists of three layers. First, a ConvTranspose2d is applied, followed by a BatchNorm2d. Lastly, there is a LeakyReLU activation function, where the leakiness was set to 0.2 for all layers [1][2]. In addition, after the three layers there is a ToRGB block that projects feature vectors to RGB colors. The block applies Conv2d and tanh activation.

The discriminator block is the mirrored version of the generator, which means that it starts with a FromRGB block, that does the reverse of ToRBG. After this the same three layers that generator block applied are applied in reverse, now using a Conv2d layer instead of ConvTranspose2d. The network is trained first with the small resolution with these blocks and then the result is passed on to the next layer. This is illustrated below in Figure 1, where for us the last layer would be one of size 64x64.

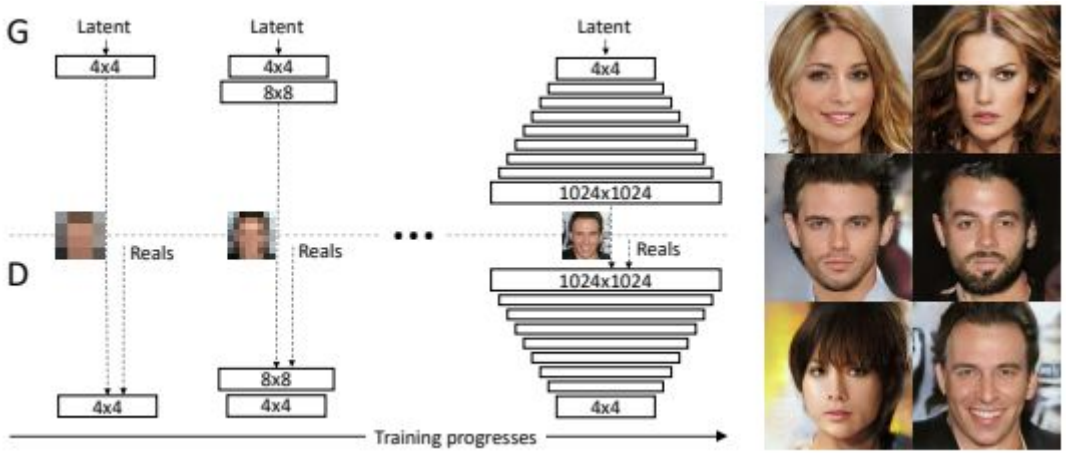

Figure 1: *Training progression [1]*

2.2 Deep Convolutional GAN

Though the main focus in the project was on ProGAN, we decided to compare its performance to a DCGAN. Whereas the ProGAN was trained incrementally, the DCGAN was trained with all the layers from the start. The DCGANs generator consists of five convolutional layers with batch normalization and the discriminator is a mirror image of this, as with ProGAN.

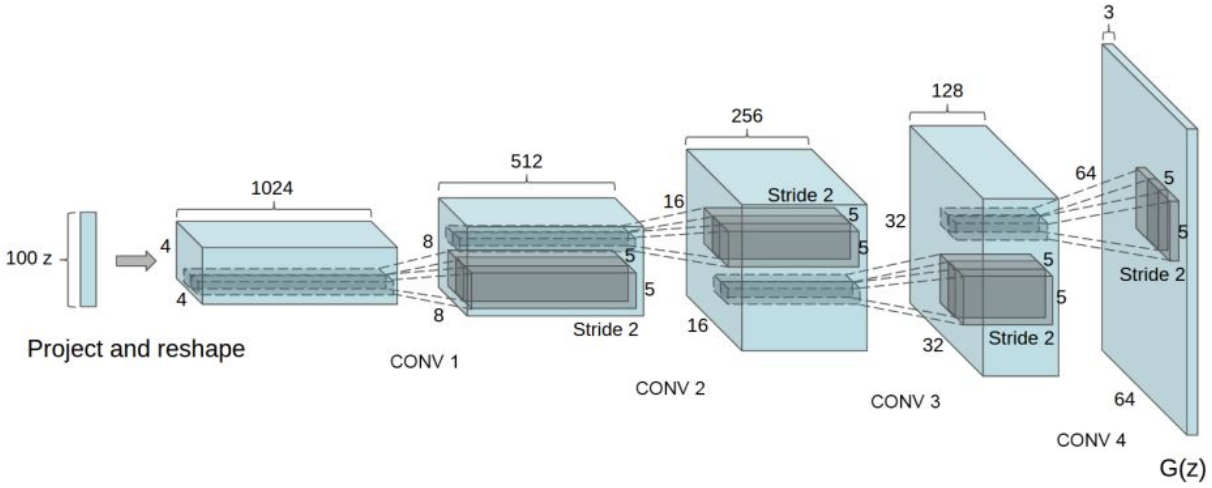

Figure 2: *DCGAN generator architecture [3]*

3. Results

The images generated by ProGAN after each training phase is shown below. Figure 7 shows the losses from the training of ProGAN and the layer additions are clearly visible in the chart as spikes. The images generated by DCGAN are shown in figure 8.

ProGAN

Figure 3: *8x8 images, 2 epochs*

Figure 4: *16x16 images, 4 epochs*

Figure 5: *32x32 images, 8 epochs*

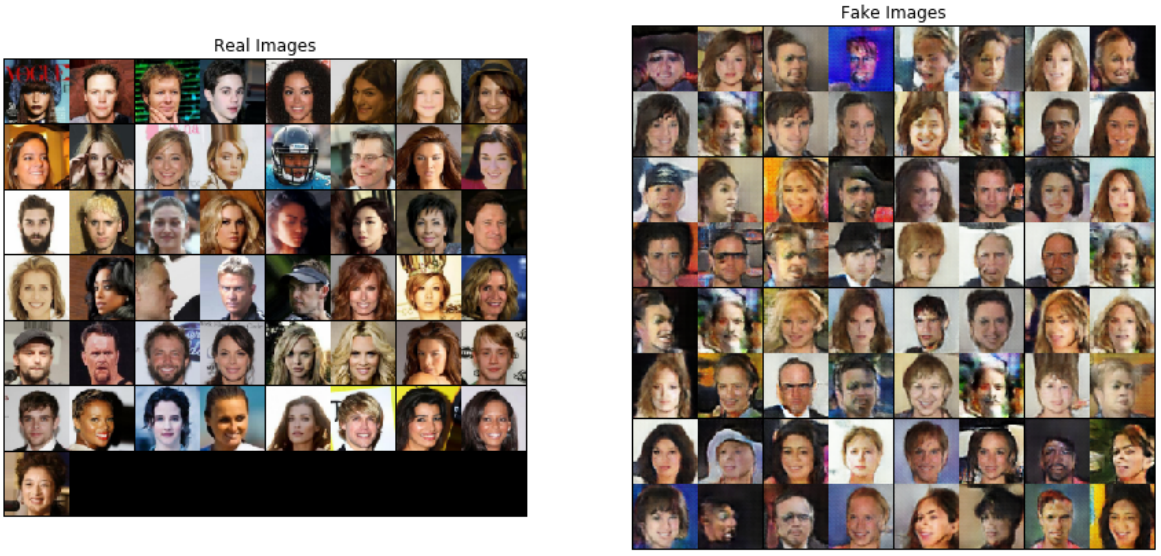

Figure 6: *64x64 images, 16 epochs*

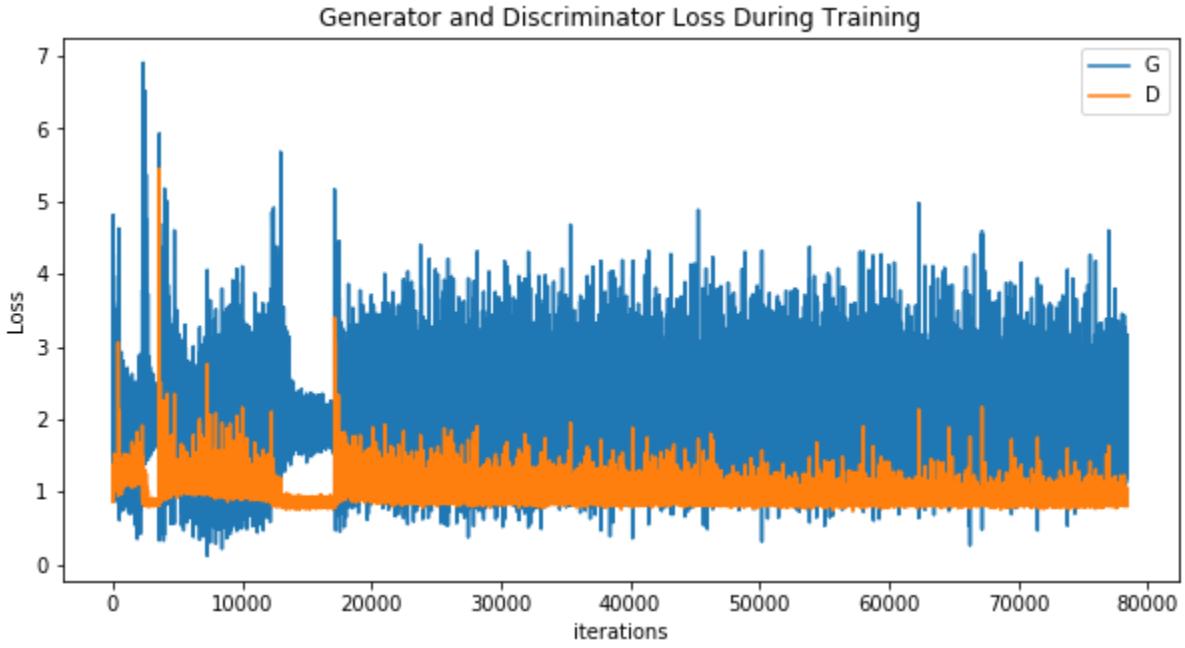

Figure 7: *ProGAN loss*

DCGAN

Figure 8: *64x64 images, 8 epochs*

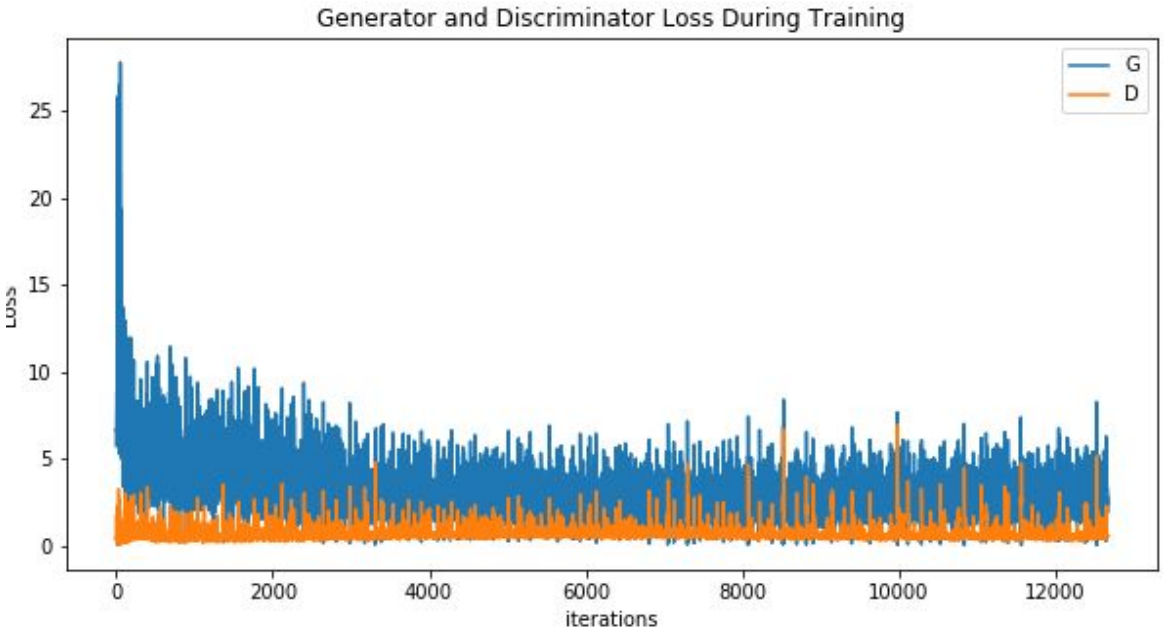

Figure 9: *DCGAN loss*

4. Conclusions

As the results above show, by training the a GAN model in a progressive manner, the resulting images seem somewhat more blurry, although exact measurement of the image quality is hard. However, the images from the ProGAN seem to have more variation amongst them whereas the images generated by the DCGAN all have a very similar look to them. A common problem with training GANs is a phenomena called *mode collapse* where the generator learns to generate only one mode of the data, resulting in very similar output. Symptoms of this are visible in the images from the DCGAN. Due to limited computational resources, the epochs and resolutions where kept low and it is possible that if more time was spent on training the models, the progressive growing method of training GANs would yield higher quality images compared to the DCGAN.

5. References

1. Karras, Tero, et al. "Progressive growing of gans for improved quality, stability, and variation." *arXiv preprint arXiv:1710.10196*(2017).
2. Radford, Alec, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." *arXiv preprint arXiv:1511.06434* (2015).
3. Inkawhich, Nathan. “DCGAN Tutorial.” Link