Training a DCGAN on little data within few epochs

Jelmer Jansen s4480848
Laura Tigchelaar s1013029

January 2018

Abstract

Generative Adversarial Networks are neural networks that are trained to generate data based on their training data. We used a Deep Convolutional Generative Adversarial Network to generate images of the faces of cats, using a combination of techniques that showed good results in previous research. Our results suggest further investigation into the size and parameters of DCGANs for image generation with a small dataset within few epochs.

1 Introduction

Ever since their introduction in 2014 [5], Generative Adversarial Networks (GANs) have gained more and more interest in the field of neural networks. In fact, the introduction of GANs has been considered 'the most important breakthrough in deep learning in recent times' [10].

GANs are networks that consist of two separate, collaborating networks: a generator and a discriminator. The discriminator learns to distinguish generated data from actual data, while the generator learns to turn noise into some form of the desired data. The goal of GANs is to train a generator network that produces samples that humans cannot distinguish from the original data.

The latest successful development concerning GANs is the introduction of Deep Convolutional Generative Adversarial Networks (DCGANs), as proposed by Radford and Metz [11]. A deep neural network has more than one hidden layer, whereas a convolutional network contains at least one convolutional layer. DCGANs, therefore, usually have several convolutional layers.

GANs have been applied to multiple domains so far, starting with training on the simple MNIST dataset [9,12], but now also in emotion classification [13], which can be used by both robotics and mental health care, and in medical care [7]. Thusfar, we have seen GANs applied to images of animals within CIFAR-10 and ImageNet [12], but not to images of cats. To make a start on this specific domain, we applied a DCGAN of our own architecture to a dataset of solely faces of cats.

In this paper, we explore the architecture and parameter settings that are necessary for a DCGAN that generates recognizable images based on a relatively small dataset, within relatively few epochs.

2 Related Work

Recent papers focus mostly on improving GAN results by adding more layers [1], adjusting parameter settings [8], or adding information [2]. All these GANs have produced photorealistic results, but took much time, computing power and/or a large dataset.

Our work tries to reproduce and improve results achieved by Horsley and Preze-Liebana [6], who used a DCGAN to automatically generate unique sprites, combined with the work of Radford and Metz [11], who used a DCGAN to generate images of bedrooms.

Most imagery datasets contain more than 10.000 images, such as the CIFAR-10, which has 60.000 images in ten different categories, or the LSUN bedrooms dataset, which has a little over 3 million images of bedrooms. Horsley and Preze-Liebana [6] showed that generation based on a smaller dataset is also possible, as they used datasets with 1.210, 36 and 517 images, with which they needed 15.000, 5000 and 10.000 epochs, respectively, to achieve results of a certain quality.

The architecture used by Radford and Metz [11] achieved very good results, even after only five epochs of training, but their dataset was extremely large and a GPU was needed for fast processing.

To improve DCGAN efficiency, we propose an architecture that generates recognizable features using a small dataset and few epochs.

3 Dataset

The dataset we used was an altered combination of Microsofts Kaggle/ASSIRA Dataset (as used in [4] and The Oxford-IIIT Pet Dataset. From both datasets, only the images of cats were used. This made up a total of 19.893 images of cats (12.500 from Kaggle/ASSIRA Dataset, 7.393 from Oxford-IIIT Pet Dataset). We used the built-in catface-recognition function from opencv (version 3.3.1) to select all catfaces. We then removed all duplicates and images that did not contain a catface, and reduced image size to 28 by 28 pixels for every image. This resulted in 7.061 images (6.495 from Kaggle/ASSIRA, 566 from Oxford-IIIT Pet Dataset) of just the faces of the cats, all of equal size.

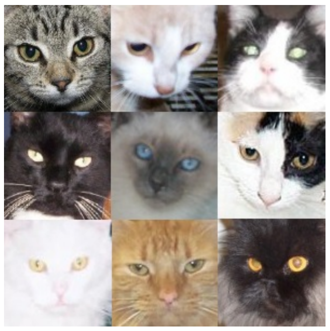

Figure 1: Dataset sample.

4 Architecture

For the network we used Python 2.7.13 with chainer package version 2.1.0.

4.1 Basics of Convolutional Networks

A convolutional neural network is a neural network with at least one convolutional hidden layer [3]. Convolutional layers take the output of the previous layer as a 2-dimensional matrix. The output of the layer is computed by sliding a smaller "filter" or "kernel" matrix over the input. The function of this filter is to calculate the sum of element multiplications between the filter and the matching part of the input matrix. Each time this multiplication is calculated, the result becomes a new field in the output matrix. How fast the filter moves is known as the 'stride'. For example, in a convolutional layer with a stride of 2, the filter moves two fields to the right after each calculation [3]. The number of filters used per layer is known as the "depth" of the layer. Deconvolutional layers work in a similar manner, using a transposed form of convolution to process the input in such a way that the effect of convolution is reversed.

4.2 Discriminator

Our discriminator makes use of three convolutional layers, with a batch normalization layer directly after each of them. The output of each of the batch normalization layers is followed by a rectified linear unit (ReLU) activation function. Through the repeated convolutions, the images are brought down from a size of 28 by 28 to 14 by 14, then to 7 by 7 and finally to 4 by 4. We use a linear layer to convert the output from a 4 by 4 format to a single value. The number of copies processed during each layer increases from 32, initally, to 64 and then 128 through the network, before being reduced to 1 in the final layer.

We initially implemented minibatch discrimination, but this slowed the network down considerably. Since we did not experience any issues with mode collapse, we did not end up using minibatch discrimination in subsequent training sessions.

4.3 Generator

Our generator has three deconvolutional layers, with a batch normalization layer directly after each one of them, followed by a rectified linear unit (ReLU) activation function. For the intermediate layers, we used the same sizes as for the discriminator, but in reverse. We also added a linear layer before the first deconvolutional layer, to convert the 100x1 input noise to a 4 by 4 format. We planned on removing this layer during training, but this had a major negative effect on the network's performance. The number of copies here started at 128 after the first deconvolutional layer, then going from 64 to 32 before being reduced to 3 in the final layer (one copy for each colour).

For training the generator, we also implemented feature matching, as described in Salimans et al. [12]. We therefore took the activation values of the first batch normalization layer in the discriminator, and the last batch normalization layer in the generator, then calculated the mean squared error, and added the result to the loss of the generator.

5 Experiment

When creating the networks, we experimented with various parameters, layer sizes and numbers of units. After training several times with the standard training parameters, we switched to a learning rate of 0.0005 and a β_1 value of 0.5, as specified by Radford et al. [11], for both networks, which gave significantly better results. We tried various deconvolution shapes for the generator, finally settling on the layer sizes of 4, 7, 14 and 28 instead of a slimmer network of 4, 6, 10 and 28. For the network breadths (number of copies), we started out with a consistent breadth of 10 for all layers, tried various combinations, and finally settled on a smaller version of the traditional DCGAN funnel structure introduced by Radford et al. [11]. The final number of copies we ended up using for the generator were 128, 64 and 32.

For our final results, we ran the networks for 100 epochs with 6000 samples from our dataset, using Adam optimizers and a batch size of 32.

6 Results

We were able to generate the following images (see Figure 2.). Although they are clearly distinguishable as cats, they also have a very artificial quality to them and would not be confused with the real images. However, we did manage to reproduce the original database's variety of colour among the different cats.

There are several possible reasons for the relatively high level of noise in the images. First, we did not use cuda and thus did not have enough processing power to run the network for large numbers of epochs. Another possible reason is the complexity of cat faces compared to those of humans, since we partially based our network on GANs which generate human faces. When looking at the final images, one can see that the GAN generalized the basic features of the faces pretty well, such as the eyes, the snout and the forehead patterns, but they seem to have trouble with the more advanced patterns of the fur. As such, it is possible that GANs need some new structural adjustment to properly generate these textures. On the other hand, Salimans et al. [12], who had trouble generating the body structure of animals, were still able to generate the specific textures of animal fur.

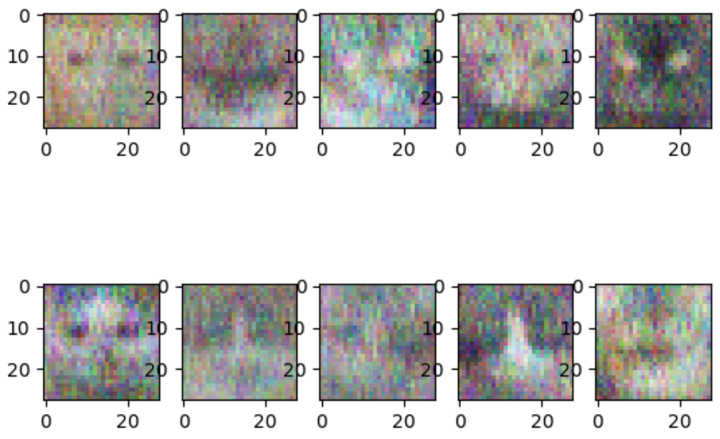

Figure 2: A sample of images from the generator.

7 Conclusion and Future Work

Despite our best efforts, we were unable to produce realistic images with the GAN techniques we used. However, for our relatively simple network, small number of epochs and small dataset, our image quality was still fairly decent.

In the future, research within this specific niche could try running our network on the dataset we created with more impressive hardware to run it for more epochs, since in the past DCGANs have generated realistic images from relatively small datasets when trained for a large number of epochs. Another option would be to increase the sizes of the generator and the discriminator, or to use a larger dataset. Besides this, since the network seems to have trouble generating the more detailed textures within the training images, any structural advancements within the networks making this easier would be a great addition to the field.

References

[1] Bachman, P. An Architecture for Deep, Hierarchical Generative Models. *NIPS*, 2017.

[2] Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I. and Abbeel, P. InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets. *NIPS*, 2016.

[3] Ciregan, D., Meier, U., Schmidhuber, J. Multi-column deep neural networks for image classification. *Computer Vision and Pattern Recognition*, *IEEE*: pp. 3642-3649, 2012.

[4] Elson, J., Douceur, J.R., Howell, J., and Saul, J. Asirra: A CAPTCHA that exploits interest-aligned manual image categorization. Microsoft, 2007.

[5] Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y. Generative Adversarial Nets. *NIPS*, 2014.

[6] Horsley, L. and Perez-Liebana, D. Building an automatic sprite generator with deep convolutional generative adversarial networks. *IEEE*, 2017.

[7] Hwang, U., Choi, S., and Yoon, S. Disease Prediction from Electronic Health Records Using Generative Adversarial Networks. *arXiv*, 2018

[8] Marchesi, M. Megapixel size image creation using generative adversarial networks. *arXiv*, 2017.

[9] Nowozin, S., Cseke, B., and Tomioka, R. f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization. *NIPS*, 2016.

[10] Pattanayak, S. Advanced Neural Networks. In: Pro Deep Learning with TensorFlow. Apress, Berkeley, CA, 2017.

[11] Radford, A., and Metz, L. Unsupervised Representational Learning with Deep Convolutional Generative Adversarial Networks. *arXiv*, 2016.

[12] Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. Improved Techniques for Training GANs. *NIPS*, 2016.

[13] Zhu, X., Liu, Y., Qin, Z., and Li, J. Data Augmentation in Emotion Classification Using Generative Adversarial Networks. *arXiv*, 2017.