Image Generation Using Deep Convolutional Generative Adversarial Networks

Harald Stiff, Erik von Keyserlingk, and Jacob Stuart

Royal Institute of Technology

Abstract. In this paper the problem of generating images using deep convolutional generative adversarial networks (DCGANs) is considered. We present the theory of GANs and the general model architecture present in DCGANs. Furthermore, the results of trained network using the MNIST, Oxford 102, and CAT datasets are presented showing both the quality and the diversity of the generated images. The presented models are able to produce diverse data and captures to varying degree the features present in the images used for training. Lastly it is shown that the generated images are distinguishable from the real dataset showing that the models have room for improvements.

Keywords: GAN, DCGAN, MNIST, CAT, Oxford 102

1 Introduction

Unsupervised learning is generally an interesting problem as the amount of available data is virtually infinite, unlike supervised learning where the availability of labelled data is a severe constraint. Commonly the goal is to learn hidden higher-level representations of the data that can be used for grouping or other analysis. For example, a description of an image such as "a cat" would be a higher-level representation of the raw pixel data that could be used to group images of cats together.

Once a transform between high dimensional data to a low dimensional description has been created the question shortly arise how the opposite transformation would look, going from a low dimensional description to high dimensional data. In terms of the example above the question is what the archetypical cat image would look like that represented the network's concept of a cat. This capability to go from description to image is useful if the description is learnt autonomously and lack a human understandable representation, for example, if the network learns that cats are a separate group of things but do not know that the name for the group is "cat". The resulting low-level data can also be used on it own as it is new data. If part of the description can be randomized an infinite amount of cat pictures can be produced. Further, if transform has both a concept of types, such as animal and colour, new images can be generated that satisfy both concepts but does not exist in real life such as "a blue cat". This could have applications in creating phantom sketches or similar.

The high-level description is not constrained to text but could also be sketches or other kinds of data, making it possible to automatically go from sketch to realistic image such as in the work by Wengling Chen and James Hays [3]. In this paper, we focus on applying the General adversarial network (GAN) approach to creating new images that look like existing images in a limited domain.

The GAN consist of to parts, a generator which generates images from a comparably small input vector, and a discriminator which tries to discriminate the generated images from genuine images taken from the domain. Usually the input vector simply consists of noise.The discriminator is trained by simply generating images using the generator and then applying the discriminator to a mix of generated and genuine images. Training the generator, on the other hand, is more complex as the generator on its own lack a target functions to evaluate its accuracy against. This problem is solved by feeding the generated image to the discriminator and trying to optimize the generator as to deceive the discriminator, the result of the deception must then be backpropagated through the discriminator to the generator weights in order to improve them.

At first, the randomly initiated generator only produce noise from the input noise vector which the discriminator easily distinguishes from the real images. But as the discriminator creates a clearer model of what real images look like the generator can be optimized to fit this model. Formally the generator tries to minimize the accuracy of discriminators judgment of image source while the discriminator tries to maximize the same accuracy.

2 Background

Many other techniques for learning high-level representation of data in order to recreate new instances of the data from the high-level representation has been proposed, for example, restricted Boltzmann machines, auto-encoders and deep belief networks. These have been applied to a wide variety of applications such as speech shown by Donahue et al. in [5] and images presented by Radford et al. [10].

Auto-encoders, as used in Cai et al. [2], is the approach most similar to the general adversarial network approach and has shown success on similar problems such as image generation. In this technique, the data is fed through a network where each layer has fewer nodes than the previous until a layer is reached which consists of only a few nodes. From this layer on each subsequent layer consists of more nodes until the final layer has the same dimension as the original data. The network is then trained with the goal of producing an output identical to the given input. This means the network must first scale the data to a much lower dimensional representation and then again back up. The trained network can then be split in the middle and input feed directly to the middle layer with fewest neurons, the input will then be upscaled to data similar to the one used to train the network. The auto-encoder could be considered to have two parts. First, an encoder that tries to generator a sparse description of the data, at a glans similar to the GAN discriminator which could be argued reduces complex data to

a binary judgment of authenticity. Second is a decoder which tries to generate a copy of something from very sparse information, this is equivalent to the forger in a GAN network. A restricted Boltzmann machine is a simpler and older version of the auto-encoder where the same single layer network is used both as the lower dimension encoder and the decoder by running it in reverse. A deep belief network is a network created by stacking multiple restricted Boltzmann machines. While the restricted Boltzmann machine on their own can generate impressive results on some types of data the increase in available computational capacity has swayed the focus to approaches such as Deep belief networks, GAN and auto-encoders that can be scaled to take advantage of the increased capacity. An adjacent problem to generating entirely new data from noise is to fill in gaps in existing data or increasing its dimension by for example adding colour to a grey-scale image. In the image case of this application, the generator should generate a manipulated image based on an input image. While the discriminator tries to discriminate images manipulated by the generator from real images. This is a form of supervised learning but one where a large number of real and manipulated samples pairs can often be generated if one direction of the image transform can be performed. In the case of colourizing an image, the transform is trivial in the direction from colour to grey-scale meaning that a large number of images both correctly colourized and to be colourized can be generated. Other examples of one directional image transform where the opposite transform can be approximated by a GAN or auto-encoder is adding noise or removing parts of the image. These have obvious useful applications in the real world. Several more creative one directional transform such as from sketch to image has been implemented using GANs by Isola et al. in [7].

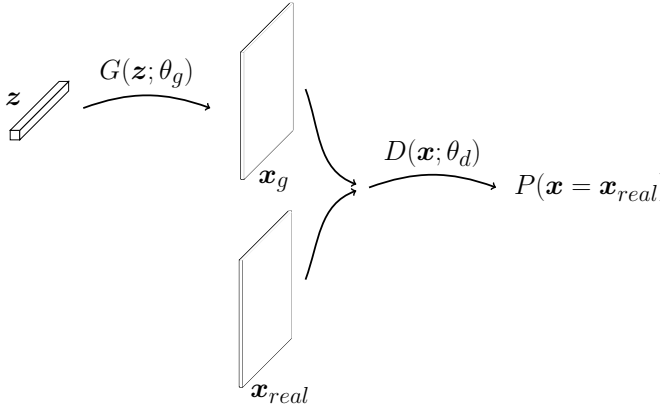

Fig. 1: A representation of the GAN model used in this paper. The generator G maps the input noise $\boldsymbol{z}$ to a tensor $\boldsymbol{x}_g$. The discriminator D maps a tensor $\boldsymbol{x}$ to a probability score of it coming from real data.

3 Approach

A GAN as outlined in Figure 1 is a combined model of both a generator model G and a discriminator model D. Noise generated by the distribution $p_{\boldsymbol{z}}(\boldsymbol{z})$ is used as an input to the multilayer perceptron function $G(\boldsymbol{z};\theta_g)$ with parameters θ_g that maps the input to the data space. Similarly the discriminator function $D(\boldsymbol{x};\theta_d)$ is a multilayer perceptron function with parameters θ_d which outputs a scalar probability of $\boldsymbol{x}$ coming from real data rather than being generated by G. Both the generator and the discriminator are trained with contradicting goals. D is trained to output small values for data generated by G and large values for real data, thus distinguishing between real images and generated images. The generator however is trained to create images that gets classified as real images by the discriminator. The training of the entire GAN model is summarized by the min-max optimization problem by Goodfellow et al. [6]

$$\min_G \max_D \mathbb{E}_{\boldsymbol{x}\sim p_{data}(\boldsymbol{x})}[\log(D(\boldsymbol{x}))] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[1 - \log D(G(\boldsymbol{z}))] \tag{1}$$

where we are tuning θ_d to maximize the expected value of $\log D(\boldsymbol{x})$ and $1 - \log D(G(\boldsymbol{z}))$ as well as tuning θ_g to minimize the expected value of $1 - D(G(\boldsymbol{z}))$ where $\boldsymbol{x}$ is generated from the true data distribution $p_{data}(\boldsymbol{x})$ and $\boldsymbol{z}$ is generated from a noise distribution $p_{\boldsymbol{z}}(\boldsymbol{z})$.

3.1 Deep Convolutional Generative Adversarial Networks

Several problems arise from training GANs. For instance, one common problem is that the discriminator becomes too good in comparison to the generator. Moreover the generator could learn a flaw from the discriminator and only output a very homogeneous distribution that does not cover all of the features present in the input data. It is shown by Berthelot et al. [1] that the loss of the generator and discriminator ideally should be equal to maintain a balanced effort between the models. To achieve this we use Deep Convolutional GANs (DCGANs) presented by Radford et al. [11] that utilizes the same structure as regular GANs but with several constraints in the model architecture to ensure stability in the training phase. The following constraints are present in the DCGANs models:

- Replace fully connected and pooling layers with fractional strided convolutions (generator) and strided convolutions (discriminator).
- Use batch normalization in all of the layers except in the generator's output and in the discriminator's input.
- Use the ReLU activation function for all of the generator's layers except its output which uses a tanh function.
- Use the LeakyReLU activation function for all of the discriminator's layers except its output which uses a sigmoid function.

3.2 Datasets

The networks of this report has been trained on the gray scaled MNIST dataset made by Yann LeCun [12] of handwritten digits (65000 28×28 images, 10 classes), the Oxford 102 dataset of flowers by Nilsback and Zisserman [9] (8100 [reshaped] 128×128×3 images, 103 classes) and the CAT datset of cats (9998 [reshaped] 128×128×3 images) by Crawford [4].

3.3 Training Scheme

In contrast to training traditional models such as convolutional neural networks for image classification tasks there are two models to train for GANs models. The performance is very dependent of how much D and G are trained and the order of which they are trained. Neither D or G can be trained too much in comparison to one another before the other model is trained. The training scheme used in this paper involves G and D being trained on one batch $\mathcal{B}$ of the full dataset $\mathcal{D}$ each time. When training G, the parameters of the discriminator θ_d are freezed making sure that G is improving rather than D getting worse.

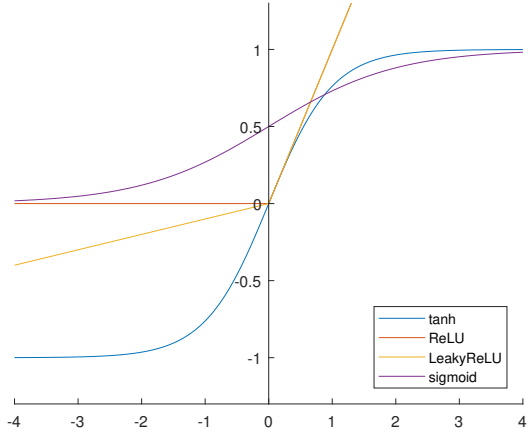

Fig. 2: A plot of different activation functions.

3.4 Model Architecture

The general model architectures used in this paper are presented in the upcoming sections. The DCGAN constraints are mostly followed but not all recommendations are applied to the models.

Discriminator Consider the discriminator model in Figure 3 used for images of size 128 × 128 × 3 like the images of flowers [9] and the images of cats [4]. From the input tensor the discriminator down scales its channel size and doubles the amount of channels through strided convolutions with kernels of size 5×5. This is repeated three times until a $8 \times 8 \times 128$ sized tensor is created. From the $8 \times 8 \times 128$ tensor it is reshaped to a scalar of size 1 with a fully connected layer. Between layers one to six the LeakyReLu activation function is used, displayed in Figure 2. In the last layer the sigmoid function is used, the function is also displayed in Figure 2.

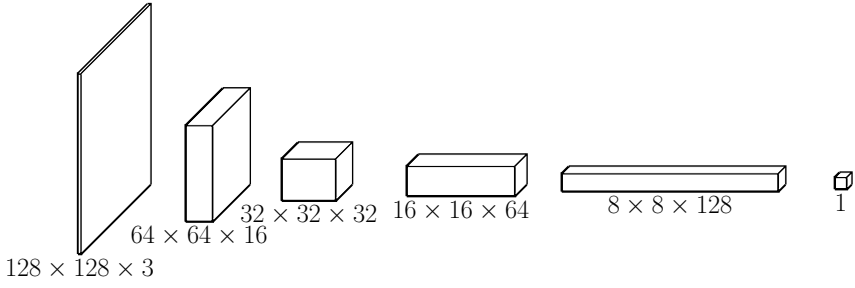

Fig. 3: Architecture of the discriminator model.

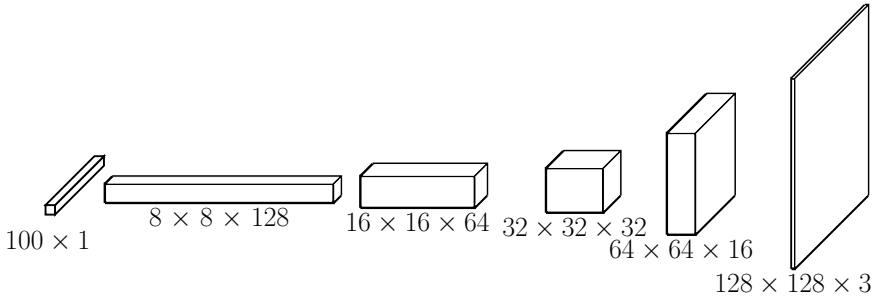

Fig. 4: Architecture of the generator model.

Generator Consider the generator model used for generating images of size 128 × 128 × 3 in Figure 4. The input used is uniform white noise generated with the mean value of 0 between -1 and 1. The noise is a vector of length 100 and through matrix multiplication and reshaping, the input noise vector of the generator gets transformed to a 8 × 8 × 128 tensor. This tensor is passed

through four convolutional layer network with kernels of size 5×5. After each convolution, the number of channels is reduced by half and the height and width of each channel are doubled in size through upsampling. This is repeated until the channel size has reached the same size as the data the generator wants to replicate. Between all layers in the generator the activation function used is LeakyReLU.

4 Results

In this section generated images of trained DCGANs are shown and compared to real images from the dataset that the networks were trained on. Furthermore, an analysis of how the images continuously change with the input is performed. All of the models were trained on the general architecture design shown in Figures 3-4 using the Adam optimizer presented by Kingma and Ba [8]. For images of size $128 \times 128 \times 3$ the exact same architectures are used, for smaller images layers are removed until the channel size matches the channel size of the input.

4.1 MNIST

A DCGAN was trained with images from the MNIST dataset using 25 epochs. A sample of the generator's output distribution of the trained DCGAN network is shown in Figures 5-6.

Fig. 5: The generators output distribution is shown to the left of the white line and the true MNIST data is shown to the right of the white line.

While the generator is trained to generate images from noise useful information can be gathered by manually setting the input vector to control the output. In each row of Figure 6 we gradually update one element of the input vector to map how the generated images change continuously with the input.

Fig. 6: The small change in the latent vector shows how numbers slowly changes into something else.

4.2 Oxford 102 Flowers

With this dataset a DCGAN was trained with 40 epochs. Three images of the dataset are shown in Figure 7 and three generated images are shown in Figure 8. How the images change with the input is shown in Figure 9.

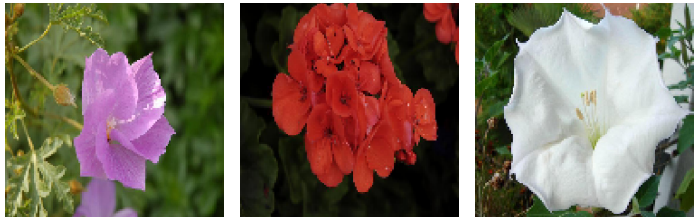

Fig. 7: Sample images from the Oxford 102 dataset.

Fig. 8: Pictures of flowers generated by the trained DCGAN.

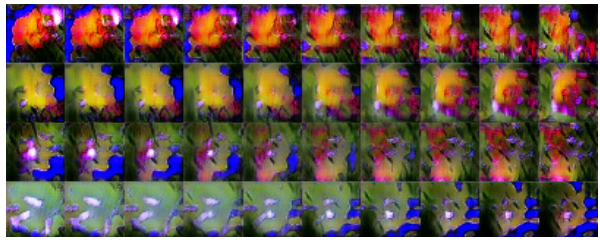

Fig. 9: In each column the input noise vecor is slightly altered changing the appearance of the generated image.

4.3 CAT

The DCGAN trained for the CAT dataset was trained with 55 epochs. Three sample images from the CAT dataset is shown in Figure 10. In Figures 11-12 images generated after 55 epochs and 60 epochs respectively.

Fig. 10: Pictures of cats from the CAT dataset.

5 Discussion

5.1 Quality and Diversity of the Generated Data

From the results, it is evident that the implemented DCGANs works well with generating handwritten digits. In Figure 5 the generated digits are very diverse capturing all of the classes in the real MNIST dataset. One can note that some of the generated digits are hard to classify. In some cases, these images can lie in the transitioning phase from one digit to another. This becomes clear in Figure 6 where we gradually altered the input noise vector to spot smooth changes in the generated images. For instance, in the second row of Figure 6 a non-classifiable generated image gets turned into a 9 after the input is altered.

Generation of colourful flowers worked reasonably well, in some of the images we cherry-picked such as in Figure 8 it is quite clear that the generator has

Fig. 11: Generated images after 55 epochs of training.

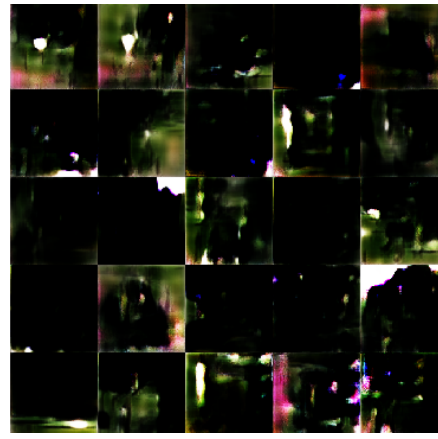

Fig. 12: Generated images after 60 epochs of training.

succeeded in creating images that look like flowers and plants. However, in comparison to the MNIST images, the generated flowers are nowhere near of what the real images look like. This is to no surprise since there are approximately 63 times more elements to generate and the diversity of the flower images is much wider in comparison to the handwritten digits. When performing an analysis of the latent space in Figure 9 it can be seen how the plant like objects both change colour and shape creating new flowers as the input is altered. With further analysis, it would be possible to steer the output image to a wanted plant.

5.2 Model Collapse

During training, a model might collapse. This means that the discriminator outperforms the generator, practically freezing the generator from making any improvements. An example of this is shown in Figures 11-12 were the model from epochs 55-60 goes from producing somewhat promising images to images that clearly are not showing any cats. In a model collapse, the generator produces images the discriminator is entirely certain are genuine images for all input vectors, from this state the generator cannot improve as all small changes in wights results in the same hundred percent success rate. The generated images are trivial to discern as fake but this is not possible for the discriminator for some unclear reason, probably due to the current discriminator being a local minima. One theory is that the total system of generator and discriminator is in an oscillating state. In this oscillating state, the discriminator oscillates between two local minima, in each, the generator has found a flaw allowing it to unhindered pass false images. When the generator changes to correct the flaw in one state it shifts to the other state.

5.3 Further Improvements

As already mentioned the problem of generating images of cats and flowers is much harder than the problem of generating handwritten digits. However, some improvements can be made quite easily. For instance, in the CAT dataset cats do not occupy the entire frames, a substantial amount of the images is of the background environment. This makes training hard because the generator has to learn both what the cats look like and the environment they are in. Images of better quality would probably be generated if the images were only zoomed in on the cats. Then the generator would only have to learn the appearance of the cats and nothing else. The images of roses had a bit larger success, even though the problem of having to generate a background environment is present the background mostly is of green leaves which reduces the background diversity a lot. This improvements would mean that the network would learn essential cat characteristics.

Another improvement that can be investigated is to use more kernels and deeper architectures in both the generator and the discriminator model. The poor quality of some of the images generated in this report might be improved by more complex models that can capture the real data distribution with more ease. Since the amount of data for generative models nearly is unlimited deep complex models can almost always be sufficiently trained.

Bibliography

[1] D. Berthelot, T. Schumm, and L. Metz. Began: Boundary equilibrium generative adversarial networks. *arXiv preprint arXiv:1703.10717*, 2017.

[2] L. Cai, H. Gao, and S. Ji. Multi-stage variational auto-encoders for coarse-to-fine image generation. *arXiv preprint arXiv:1705.07202*, 2017.

[3] W. Chen and J. Hays. Sketchygan: Towards diverse and realistic sketch to image synthesis. *CoRR*, abs/1801.02753, 2018. URL http://arxiv.org/abs/1801.02753.

[4] C. Crawford. Cat dataset. *https://www.kaggle.com/crawford/cat-dataset/data*, 2018.

[5] C. Donahue, J. McAuley, and M. Puckette. Synthesizing audio with generative adversarial networks. *arXiv preprint arXiv:1802.04208*, 2018.

[6] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[7] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint*, 2017.

[8] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[9] M.-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *Computer Vision, Graphics & Image Processing, 2008. ICVGIP'08. Sixth Indian Conference on*, pages 722–729. IEEE, 2008.

[10] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, abs/1511.06434, 2015. http://arxiv.org/abs/1511.06434.

[11] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[12] C. J. B. Yann LeCun, Corinna Cortes. The mnist database of handwritten digits. http://yann.lecun.com/exdb/mnist/.