Coursework on Generative Adversarial Networks

Alexis Othonos Koral Hassan
Imperial College London
{ao4818, kbh15}@ic.ac.uk

Abstract

The performance of the DCGAN and CGAN are measured with regards to different parameters, such as architecture, normalisation, number of epochs and the ratio of G and D training. The CGANs were evaluated based on the inception score of their generated images. In CGAN the small depth architecture showed the best results with inception score of 0.9654. The classifier was then retrained with generated data but showed poor results with accuracy dropping to 0.98.

1. Dataset

The MNIST dataset [1] is used for training and testing purposes. The dataset contains grayscale images of handwritten digits. Each image is 28x28 pixels. Some exemplar datapoints are shown in Appendix A.

The dataset contains $N_{train} = 60000$ training images and $N_{test} = 10000$ testing images. In some tests, the training set is further split into training and validation sets with a ratio of five-to-one. That is, the training set is made up of randomly sampled (without replacement) 50,000 images, while the validation is made up of the rest 10,000 images.

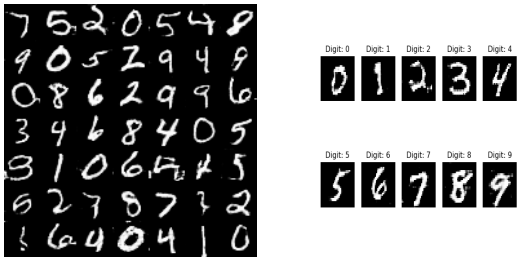
1.1. Preprocessing

In the original dataset, each pixel has a value between 0 and 255. The values are normalised between -1 and 1 in order to make the training process easier.

$$Pixel_{norm} = \frac{2 \times Pixel_{unnorm}}{250} - 1$$

2. DCGAN & CGAN

We first use Deep Convolutional Generative Adversarial Networks (DCGAN's) for generating images. The architectures of the Generator and Discriminator models can be seen in Appendix B. The architectures are used for both DCGAN and CGAN training for comparison. The discriminator model is shown in Figure 11, and 3 different generator architectures shown in Figure 12.



(a) (b)

Figure 1: Images generated from the best performing architecture for (a) DCGAN and (b) CGAN

2.1. Depth

The binary cross-entropy of DCGAN losses with the 3 different generators after 20 epochs is shown in Table 1. Medium depth achieves the lowest loss for both the discriminator and the generator, with outputs shown in Figure 1. The resulting generated images can be observed in Appendix C. The above results show that the architecture is very important to the success of the network. Small architectures may fail to converge to a good solution, while larger ones may require a lot of training be acceptable. On the contrary, the CGAN network performs better with a smaller architecture. This may be due the labels introducing additional constraints to the generator. The generator then needs fewer weights to store as the intraclass correlation is smaller than the interclass correlation. That is the same digit samples should look similar when compared to different digits. Therefore, the CGAN can "reach" a conclusion faster with the more specific samples. The comparison of the CGAN models mad by evaluating their closeness to the real data and is further analysed in the Inception Score section.

For the reasons explained above, the medium depth model will be chosen for the DCGAN tests while the small model is used for the CGAN.

2.2. Number of Epochs

We ran the network for 100 epochs and recorded the model weights for the DCGAN. Figure 2 shows the loss

	Small Depth	Medium Depth	Large Depth
Discriminant	5.5	0.31	1.0
Generator	10.1	1.3	16

Table 1: Loss of different generator models.

of both the discriminator and generator. Appendix D shows the images produced at various epochs. We can see that by epoch 20 the generated images already look believable to a human.

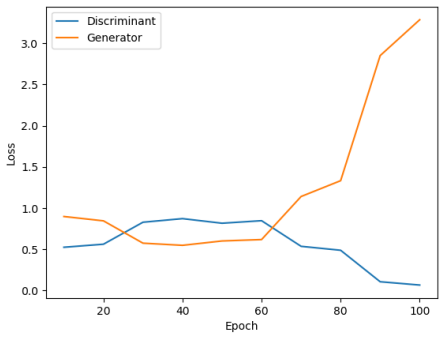


Figure 2: Loss varying as further epochs are computed.

2.3. G to D Training Ratio

We see that the discriminator initially outpaces the generator. This might be due to the fact that at every iteration, the discriminator technically trains twice as much as the generator; first on the authentic images batch, and then again on the generated images batch. To balance the generator and discriminator, we ran another training up to 50 epochs where we trained G twice at each iteration as well. The results of this can be seen in Figure 3.

We see the quality of the images generated with the different generator to discriminator training ratios in Appendix E. It does not make a significant difference relative to the extra computation time and it seems that our original method of training the generator once for every batch is better.

2.4. Batch Normalisation

The importance of batch normalisation can be demonstrated in Figure 4 for a DCGAN and a CGAN network. It can be observed that both, DCGAN and CGAN have failed to converge when the normalisation layers are removed. The poor performance is possibly due to the networks being unable to ”adapt” to the high valued outputs of the convolutional layers. This means that the network would require a lot more training epochs to adapt the weights to the bias

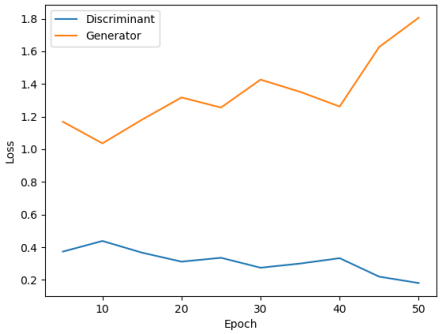
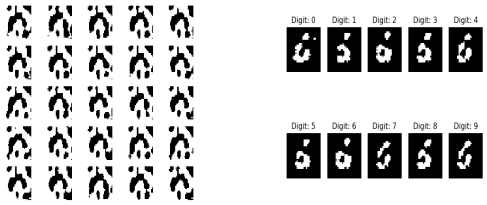


Figure 3: Loss varying as further epochs are computed. Generator trained twice per iteration.



(a) (b)

Figure 4: Results without batch normalisation for (a) DCGAN and (b) CGAN

appropriately or fail to converge entirely, as is the case in the figure.

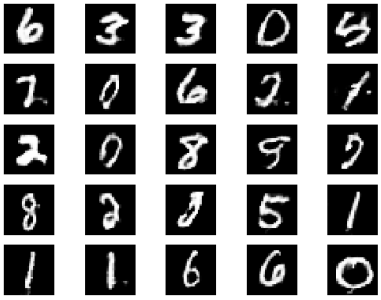


Figure 5: DCGAN generated images demonstrating mode collapse.

2.5. Mode Collapse

The DCGAN might be creating realistic sample instances after a sufficient amount of training, however, these

instances could be relatively close to each other or may be generate a disproportionate amount of specific digits, as can be seen in Figure 5. This issue is addressed with the CGAN, where the labels are used to ensure that interclass images (different digits) are not identical. This however doesn't mean that the intraclass (same digit) images have much variation. With the inception score, the performance of the CGAN generated images is evaluated in regards to how recognisable the generated images are when compared to real data. This is still not addressing the 'closeness' of intraclass images. This is further analysed in the Conlcusions and Discussion section.

3. Inception Score

A classifier is trained with real data for 12 epochs and can achieve a test accuracy of 0.992, as seen in Figure 6. The classifier is then used to evaluate generated samples of different CGAN models. The percentage of accurate predictions of the classifier can be used to evaluate the performance of the CGAN models. In the following test, the inception scores of the three architectures - mentioned above - are evaluated, and the results can be seen in Table 2.

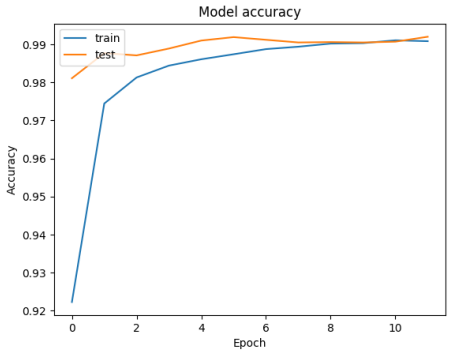


Figure 6: Train history of classifier

	Small Depth	Medium Depth	Large Depth
Accuracy	0.9654	0.9469	0.28

Table 2: Inception score of different CGAN architectures

As can be seen on Table 2, the small depth CGAN is the most recognisable by the classifier, followed closely by the medium depth architecture. The largest architecture has poor performance probably due to small number of epochs and a poor layer design. The resulting images can be seen in Appendix F

Another method to evaluate the performance

4. CGAN Classifier

A possible way to increase the performance of a classifier is to incorporate a number of CGAN generated images in the training set.

To test this, a classifier was trained with a data set of 50,000 images with increasing percentages of generated images. All classifiers were trained for the same amount of epochs and the test was repeated many times, averaging the results to reduce variance. Additionally, the generated images were sampled from a pre-traind CGAN for each test. As can be seen in Figure 7, the performance of the classifier drops when as the ratio of generated images increases. This might be due to the quality of images that are generated, however, even for a fully generated dataset, the accuracy of the classifier is around 0.95 which is relatively good.

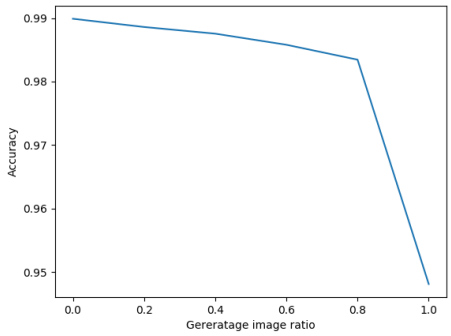


Figure 7: Classifier accuracy for ratio of generated vs real images.

Ideally the training set of real images should not decrease. Instead, the training set should be reinforced with the generated images to create an overall better classifier. In the following test, a pre-trained classifier is trained again for 5 epochs and is compared with a pre-trained classifier that is trained with fully generated dataset. As can be seen in Figure 8, the validation accuracy of the generated data set retrained classifier, however, the training accuracy is increasing sharply as can be observed in Figure 9. The conclusion from these results is that the generated images have poor inception score, meaning that they are all closely related, and as such the classifier can be quickly retrained to overfit on those examples without the ability to generalise on the real data.

5. Conclusion and Discussion

The DCGAN and CGAN are useful in computer vision applications. The GANs approximate the distributions that are implied the data, as they may self evident, due to the high dimensionality of images. If the GAN is trained cor-

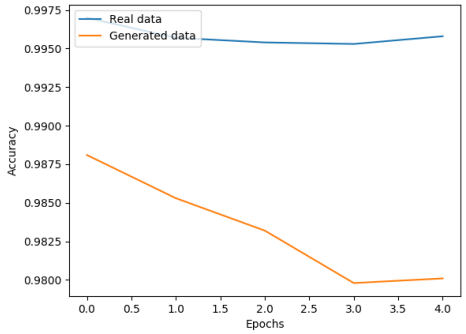


Figure 8: Validation accuracy for classifier re-training

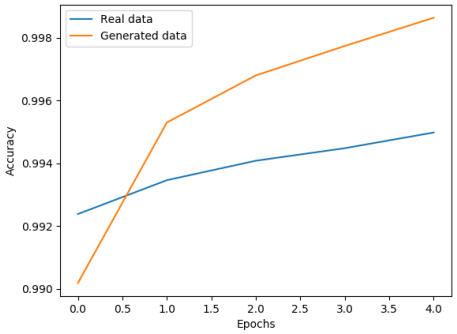


Figure 9: Training accuracy for classifier re-training

rectly, it can be used to train a classifier with a more representative distribution of the data, allowing the classifier to generalise more effectively. Unfortunately, in the above experiments, the improvement of a classifier was not achieved. During the classifier retraining, the generated data set resulted in model over-fitting. This means that the generated images were closely correlated. This probably is caused due to the small architecture of the CGAN. However, when comparing results from the CGAN generated images, the smaller architecture results would look better. Evidently, looking at generated samples is not an appropriate way to judge inception scores. For this reason, it is concluded that a deeper, and better designed architecture, trained for more epochs, is required for the CGAN to work effectively.

The inception score may provide an idea of the CGAN performance, in regards to how 'close' the generated images are to the real data. A different measure is proposed: the Nearest Neighbour of a generated image can be found, or the Mean Square Error between the generated image and the real images of the same class. This will reveal if the generated images are 'too close' to the real data. The proposed measure can be used in conjunction with the initial inception score measurement so that an image can be judged based on how different it is to the real data while still maintaining a good recognisability.

References

[1] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

6. Appendix

Appendix A. Images from Training Set

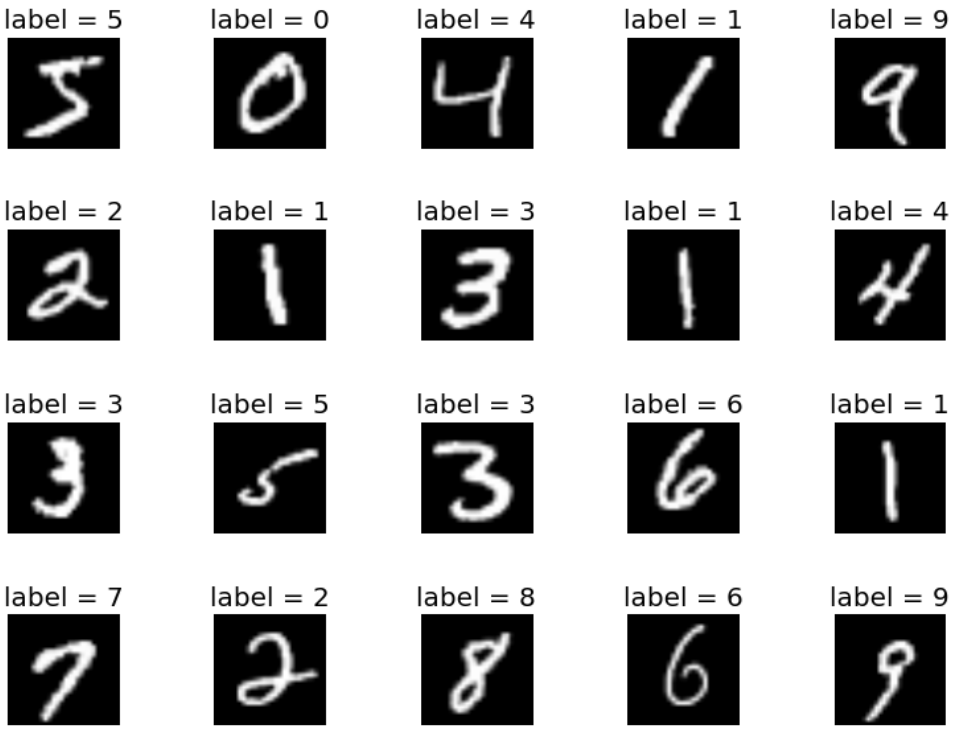


Figure 10: MNIST images and their corresponding labels.

Appendix B. DCGAN Model Architectures

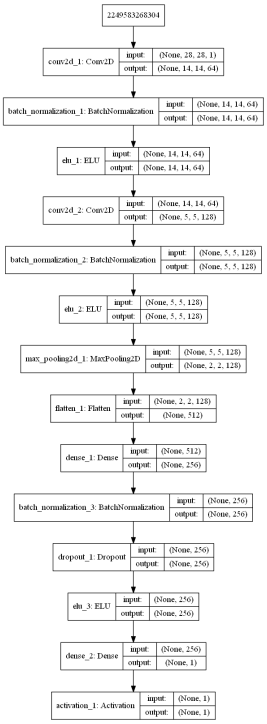
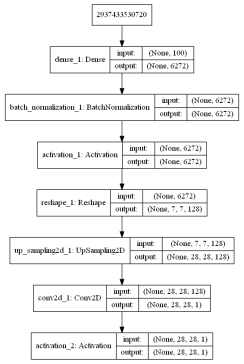
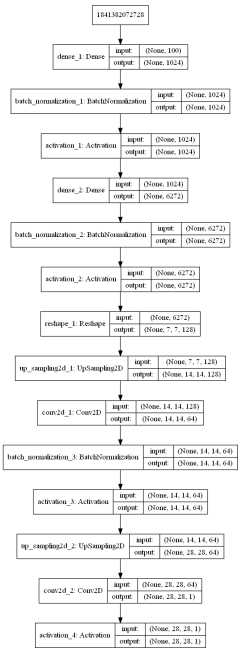


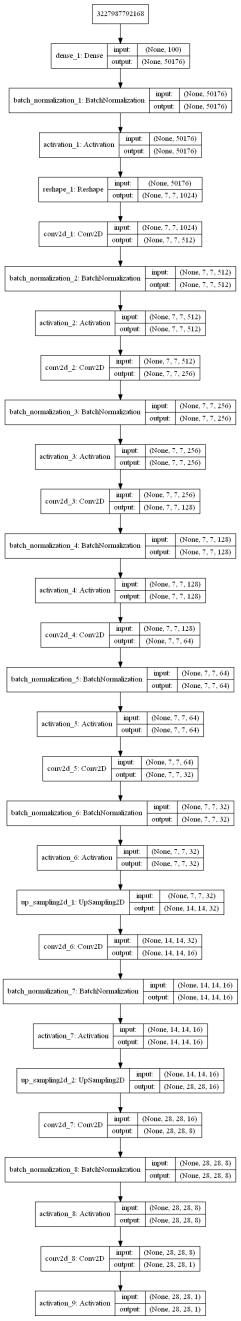
Figure 11: Discriminator Architecture



(a) Small Depth Network



(b) Medium Depth Network



(c) Large Depth Network

Figure 12: Generator architectures.

Appendix C. Images Generated with Different Generator Depths

(a) Small Depth Network

(b) Medium Depth Network

(c) Large Depth Network

Figure 13: Generated images.

Appendix D. Images Generated at Different Epochs

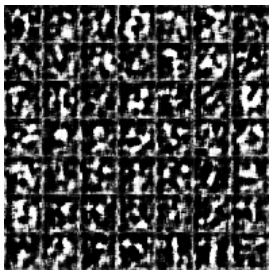


Figure 14: Image at 1st epoch.

Figure 15: Image at 20th epoch.

Figure 16: Image at 40th epoch.

Figure 17: Image at 60th epoch.

Figure 18: Image at 80th epoch.

Figure 19: Image at 100th epoch.

Appendix E. Images Generated at Different Epochs while Training the Generator Twice per Iteration

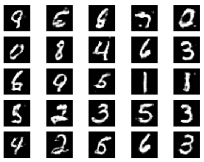


Figure 20: Image at 20th epoch.

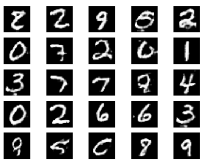


Figure 21: Image at 30th epoch.

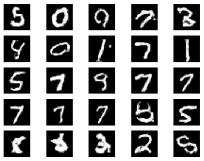


Figure 22: Image at 40th epoch.

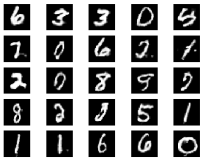
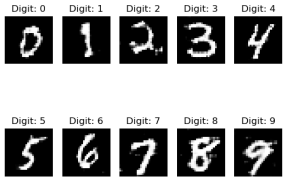
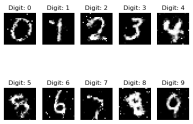


Figure 23: Image at 50th epoch.

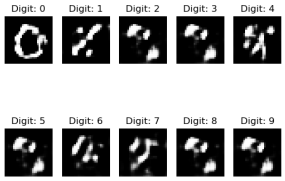
Appendix F. CGAN images from different architectures



(a) Small Depth Network



(b) Medium Depth Network



(c) Large Depth Network

Figure 24: Generated images.