Selected Topics in Machine Learning Coursework 2

Gustavo Brito Rodríguez
Imperial College London
01198785
gb1216@ic.ac.uk

Alberto García Matachana
Imperial College London
01205901
ag4116@ic.ac.uk

Abstract

This report evaluates the change in qualitative performance of DCGANs and CGANs with respect to hyper-parameter tuning, varying architectures and training methods. It also further evaluates quantitatively the CGANs by computing the Inception Scores of the resulting algorithms.

1. Question 1

1.1. DCGAN

The first question consists in the training of a DCGAN with the MNIST data set and to perform a qualitative analysis of the results with respect to hyper-parameter tuning, varying architectures, Batch Normalisation (BN) and the balance of Generator against Discriminator [9].

The first DCGAN architecture was programmed with `Keras` from `Tensorflow`. The generator consists of one dense layer and three convolutional transpose layers to perform up-sampling and produce an image in $\mathbb{R}^{28\times28\times1}$ from a seed (random noise). The discriminator, consists of two convolutional layers with 30% dropout during training. Virtual Batch Normalisation (VBN) was performed after each layer in the generator, but not in the discriminator to avoid creating dependencies between the data samples.

The noise samples taken as input by the generator are independent samples, notwithstanding, regular BN, normally used in Deep Neural Networks, creates a dependency between the samples, causing the generated samples to be dependent of each other. VBN samples a reference batch before training and combines it with the current batch to compute the mean and variance. It is then used to normalise all the training batches and reduce the dependencies created with regular BN.

Furthermore, following the architecture guidelines for stable DCGANs expressed in [13], LeakyReLU activation was used in all discriminator layers and ReLU activation was used in the generator for all layers except for the output, which used Tanh. In this paper, it was found that using a bounded activation function allowed for faster training.

Initially, the optimiser used was ADAM , with the default hyper-parameter values of: 0.001 for the Learning Rate α, and 0.9 and 0.999 for the Momentum Terms β_1 β_2 respectively [7, 8].

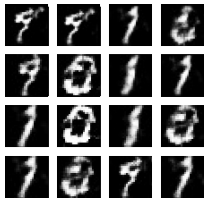


Figure 1: Images generated by DCGAN with learning rate $\alpha = 0.001$

The loss function used for the discriminator was the cross entropy loss between 1 and the discriminator output for the real images plus the cross entropy loss between 0 and the discriminator output for the generated images. This is equivalent to rewarding the discriminator for identifying real images as real and discerning fake images.

For the generator, the loss function was the cross entropy loss between 1 and the discriminator output for fake images. The generator was rewarded for having fake images being recognised as true. The DCGAN's discriminator and generator were trained in equal parts in an alternating manner.

Training the Vanilla DCGAN resulted in mode collapse, where the generator always generated the same image of every possible latent vector fed as input (Figure 1).

There are several approaches that have been adopted to address mode collapse. First, perform hyper-parameter tuning on the learning rates, of both generator and discriminator, as well as modifying the training ratios between the competing networks. Second, explore different model architectures, specifically, the Wasserstein GAN with gradient penalty (WGAN-GP). The WGAN has been chosen since it uses as loss function the Wasserstein-1 Distance, which is continuous and differentiable, meaning that the critic can be trained to optimality, at which point, mode collapse is impossible [2]. Furthermore, Wassertein Loss is also a good way to address the vanishing gradient, in which the critic is too good and will not provide any useful information to the generator. The WGAN-GP was chosen over the WGAN weight clipping since the former one enforces the Lipschitz constraint, which acts as a regularisation technique, in a much more sophisticated way [2, 5].

The images created by the vanilla WGAN generator from epochs 3 to 8 are shown in Figure 5. As can be seen, the quality of the images is much better in (5f) than in (5a), nevertheless, a qualitative improvement does not always happen every epoch. For example, (5b) and (5e)

produced worse images than the ones generated one epoch prior (5a and 5d, respectively). A probable cause for this instability in learning, is the fact that α and $\beta_{1,2}$ were assigned values which were too high. GANs commonly use smaller learning rates than other neural networks [2, 13].

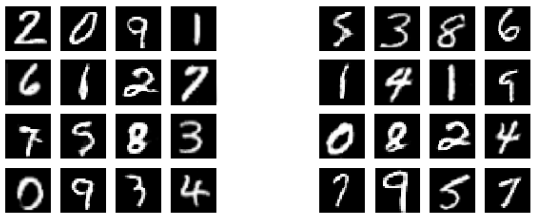
As expected, the vanilla WGAN-GP didn't suffer from mode collapse. However, it still produced images of very low quality (Figure 5), indicating that further improvement by hyper-parameter tuning can be achieved.

To address this, hyper-parameter tuning was performed on learning rates, of both generator and discriminator, as well as on training ratios, between the competing networks for both the DCGAN and WGAN. Table 2 summarises the hyper-parameter values tested for tuning both the models. It is important to note that training was carried out for a larger number than epochs shown, but, only the epoch corresponding to the best image quality is shown in the table.

The first update was to decrease the learning rates of both generator and discriminator from 0.001 to 0.0001 while keeping the training ratios symmetric, i.e., 1:1. This resulted in better images, as can be seen in Figures 6a and 6b, but, after a longer number of epochs compared to vanilla.

In the second update, the ratios of generator to discriminator training were altered while fixing learning rates of 0.0001. In both cases, the DCGAN performance worsened (as can be seen from Figures 7a and 7c). In the 1:5 case (1 generator iteration for 5 discriminator iterations), the network suffered of vanishing gradient, as the discriminator would always be able to discern fake images, and therefore, the generator was unable to produce high-quality images. For a 5:1 ratio, the network suffered of mode collapse, as the generator was able to quickly create images that would fool the discriminator and kept producing them repeatedly. The WGAN only improved in the case of the 1:5 ratio (Figure 7d), as the loss to the generator could then become an accurate estimate of the Earth-mover's distance [2].

The last update applies the Two Time-Scale Update Rule (TTUR) which consist on choosing different learning rates for the generator and discriminator.



(a) DCGAN (b) WGAN

Figure 2: Images produced by GANs with discriminator learning rate increased to 0.0004

The authors of [6] provided a mathematical proof of convergence to Nash equilibrium which demonstrates how using different learning rates can achieve state-of-the-art results. In their paper it can be found that it is beneficial when the generator has to make smaller steps to fool the discriminator. Hence, a higher learning rate was chosen for the discriminator 0.0004 and a lower for the generator 0.0001. Despite observing a higher loss for the generator during training, the image qualities did improve (Figures 2a and 2b).

2. Question 2

This question consists in performing qualitative comparisons of a Conditional GAN (CGAN) architecture in a similar fashion to that of Question 1. Unlike, a DCGAN, a CGAN trains using labels as part of the input to the competing networks.

The first architecture's discriminator, consisted of two convolutional layers and a fully connected layer. All of which, were followed by VBN (for the reasons explained in 1) and Leaky ReLU activation, except for the last layer which used a sigmoid activation.

The generator consisted of two fully connected layers and a convolutional layer. Once again, all layers were followed by VBN and ReLU activation, except for the last layer which used sigmoid activation.

For this architecture, the labels were used to train both the generator and discriminator as done in [12] and [4]. The labels are fed before up-sampling to the generator and to the discriminator as additional input.

In this vanilla architecture, the optimiser used was ADAM, and it was originally trained with the same default hyper parameters as in 1. This, again, lead to unstable learning and low quality images being generated (see Figure 8a).

Additionally, mode collapse can be seen in Figure 8a, as the 4th and 5th rows all were low-quality 5s. This is a common issue with CGAN architectures as they are unable to identify whether the generator has been trapped into a bad mode. To address this, a CGAN-mean-covariance balancing labeling (CGAN-MBL) approach, which attempts to improve on the convergence performance of CGANs and decrease mode collapse, was also trained and its qualitative performance was compared to that of the CGAN [10]. Table 3 summarises all the tests performed on both architectures for this Question.

CGAN-MBL uses the Mahalanobis distance of the generated image to the center of each class to assign labels to the unlabeled generated data [11, 10]. An image is assigned the label which corresponds to the shortest Mahalanobis distance to its centre. In mode collapse, some labels are assigned more (the majority) than others (the minority). To balance the majority and minority sizes, the algorithm attempts to balance them by producing more images in the minority.

Furthermore, [10] also introduced a Uniform Distribution to initialise weights that was proven to improve convergence performance. Randomly initialising the weights of a neural network is a widely used method for its simplicity [12, 4, 13, 5]. Nevertheless, oversized initial values cause nonlinear activation functions to be non-linear growth regimes, where the values of weights are unstable and undersized weights make gradient information disappear [10]. For that reason, the following uniform distribution was used to train the CGAN-MBL:

$$W_{ij} \sim U\Big[-\frac{\sqrt{6}}{\sqrt{h^i + h^{i+1}}}, \frac{\sqrt{6}}{\sqrt{h^i + h^{i+1}}}\Big] \qquad (1)$$

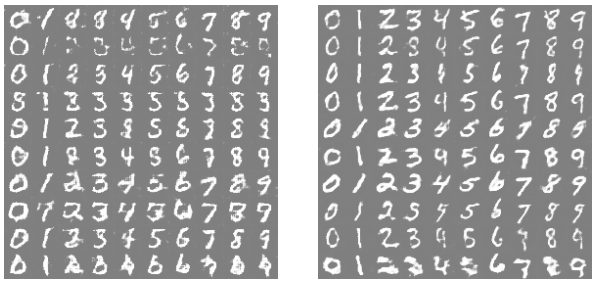
where W_{ij} is the weight corresponding to the jth neuron in the ith layer and h^n is the number of neurons in the nth layer.

The CGAN-MBL network was trained using the same core architecture as described for the regular CGAN and the first test used the same hyper-parameter values. The images generated by this network can be seen in Figure 8b and one can immediately appreciate the lack of mode collapse and increase in image quality. Moreover, one should note that this was also achieved in less epochs, as the change in weight initialisation allowed for faster convergence, as did in [10].

Given that a CGAN is, in effect, equal to a DCGAN, except for the use of labels, the networks were then trained with the values that yielded the best qualitative DCGAN performance in Question 1. This meant using lower value learning rates, and following the Two Time-Scale Update Rule explained in 1, the discriminator learning rate was set higher than that of the generator (0.0004 and 0.0001 for the discriminator and generator, respectively). Furthermore, the ratio of generator to discriminator training was set to 1:1, as otherwise led to significant performance issues in 1. This improved image quality significantly and made training more stable, as can be seen in Figures 9a and 9b.

The next change made to the architectures was the application of Label Smoothing, which was shown to be a good regularisation in [15] since it avoids extreme logits. GANs depend on a small number of features to detect real images, the generator may produce only these futures to exploit the discriminator. This leads to a greedy, over-confident optimisation, producing no long-term benefits. Then, [14] proposed to use only one-side label smoothing; apply label smoothing only to real samples changing its label from 1 to 0.9. The reasoning is because applying label smoothing on fake samples leads to fake mode on data distribution. This led to even better image quality as seen in Figures 10a and 10b, and once again, the CGAN-MBL performance achieved the superior performance in less epochs.

It is well known that making the training of the discriminator more difficult is beneficial for the overall stability [5].



(a) CGAN (b) CGAN-MBL

Figure 3: Images produced after applying Noise Injection

One of the most known methods to increase the complexity of the discriminator training is adding noise to both the real and synthetic data; in the mathematical world this should work because it helps giving some stability to the data distributions of the two competing networks [1]. For this reason, noise injection was tested by adding noise to 10% the discriminator output. This resulted in the images of best quality being generated (seen in Figures 3a and 3b).

3. Question 3

This last question consists in performing quantitative comparisons by measuring the inception scores of the different hyper-parameter and architectures of Question 2. The Inception Score (IS) is an objective metric used to evaluate the quality of synthetic images produced by a generative adversarial network. The score seeks to capture two characteristics in the collection of images:

- **Saliency**: do the images look like a specific object?
- **Diversity**: are there different class objects generated?

If both properties are present, the score will be high; the higher score, the better. The IS has a lowest value of 1.0 and a highest value equal to the number of classes supported by the classification model [3].

The classification neural network was created in `PyTorch`. It contained two convolutional layers, with kernels of size 5 and dropout of 20%, followed by 3 fully connected layers, the first two with a ReLU activation and the last one, the output, returned a log softmax activation. To train the network, an ADAM optimiser with learning rate of 0.001 was used. The loss criterion used was a negative log likelihood loss function, which combined with the log softmax output gave an equivalent cross entropy loss for the 10 classification classes.

The classifier was then trained with the MNIST training set of 60,000 images for 4 epochs giving a test accuracy of 98.86% in 54 seconds.

The IS uses this pre-trained classifier and calculates a statistic of the network's output.

$$IS(G) = exp(\mathbb{E}_{\mathbf{x}} D_{KL}(p(y|\mathbf{x}) \parallel p(y))) \qquad (2)$$

Saliency is expressed as $p(y|x)$ - a distribution of classes for any individual image should have low entropy, i.e, the Inception Network should be confident there is a single object in the image. Diversity is expressed as $p(y)$ - the distribution of classes across the images should have high entropy, i.e., the generative algorithm should output a high diversity of images with the absence of dominating classes. If both of these components are met, then a large KL-divergence D_{KL} between the distributions $p(y|x)$ and $p(y)$ is expected, resulting in a large IS.

To calculate the Inception Score, the dataset is split into chunks of size $\frac{N}{n_{splits}}$, where $N = 60,000$. Then, the estimator is applied to these chunks repeatedly, obtaining the mean and standard deviation of the Inception Score. [14] used $n_{splits} = 10$, and therefore, is the default split used for these experiments.

The Inception Score is bounded between 1 and the maximum number of classes supported by the classifier [3]. Therefore, $1 \leq IS(G) \leq 10$. The IS calculated on the real MNIST dataset achieved a result of 9.495 ± 0.037. This result agrees with the saliency and diversity of the real MNIST dataset, and consequently, approximates to the theoretical maximum score of 10.

The following table summarises the different architectures and hyper-parameter updates carried in Question 2 with their corresponding mean inception scores.

Updates	Figures	Model	IS
Vanilla	8a	CGAN	1.879
implementation	8b	CGAN-MBL	4.611
Optimal DCGAN	9a	CGAN	3.562
hyper-parameters	9b	CGAN-MBL	5.420
Label	10a	CGAN	4.211
Smoothing	10b	CGAN-MBL	7.309
Noise	3a	CGAN	5.013
Injection	3b	CGAN-MBL	7.859

Table 1: Inception Score (IS) on CGAN and CGAN-MBL models.

[14] introduced the Inception Score because, according to their experiments, it correlated well with human judgement of image quality. Since there is a correlation between the human judgement of image quality and the IS, lower score will correspond to lower quality images and higher score will correspond to better quality images.

The lowest score obtained was 1.879 with the CGAN vanilla implementation. Figure 8a does not meet neither the saliency trait since the images are of very low quality, nor the diversity trait since mode collapse can be seen. Thus, a very low divergence is expected and consequently a very low IS close to the minimum.

According to the scores obtained, only the CGAN with noise injection produced better quality images than the vanilla CGAN-MBL implementation. This is a very interesting point since, according to the Inception Network, image (3a) has better quality digits than image (8b). A small experiment was run across Imperial College students in which they were asked to compare which image, 3a or 8b, had better quality numbers. A sample of 70 students were asked; 51 people agreed the vanilla CGAN-MBL (Figure 8b) had better image quality numbers versus 19 people who voted for the noise injection CGAN (Figure 3a). These results contradict with the correlation mentioned by Salimans *et al.* in [14] since only 27% agree with the IS results for this specific comparison.

The highest IS was 7.859, obtained by the CGAN-MBL with noise injection. Figure 3b met the saliency trait since the numbers are of the best quality compared to other CGAN images. It has also met the diversity trait since the digits are different from each other, opposite to what happened with mode collapse. Overall, this has led to a high KL-divergence and thus a large IS.

The Inception Score is a metric that has its limitations. It has first been seen with the results obtained from the student's poll in which 73% of the population opposed the scores from the IS. Therefore, it can be said the IS can yield misleading results when the score difference between two different images is less than 0.5.

The second limitation is due to the variable n_{splits}. An experiment was run on the CGAN-MBL noise injection model which measured the IS values with respect to the n_{splits}. It was found that the size of the splits directly affect the scores obtained, even though the digit's image quality is still the same. The graph beneath shows that increasing the split number lowers the inception mean score; all the values can be found in Table 4.

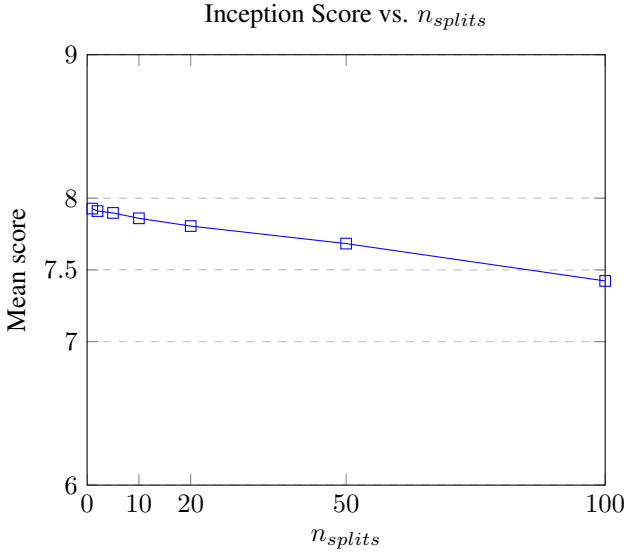


Figure 4: IS of CGAN-MBL noise injection model with varying n_{splits}

References

[1] Y. S. Abu-Mostafa, M. Magdon-Ismail, and H.-T. Lin. *Learning From Data*. AMLBook, 2012.

[2] M. Arjovsky, S. Chintala, and L. Bottou. Wassertein gan, December 2017.

[3] S. Barratt and R. Sharma. A note on the inception score, June 2018.

[4] J. Gauthier. Conditional generative adversarial nets for convolutional face generation, March 2015.

[5] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville. Improved training of wasserstein gans, December 2017.

[6] M. Heusel, H. Ramsauer, T. Unterthiner, and B. Nessler. Gans trained by a two time-scale update rule converge to a local nash equilibrium, January 2018.

[7] Keras. Optimizers. `https://keras.io/optimizers/`. Accessed: 2019-02-22.

[8] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, December 2014.

[9] Y. LeCun, C. Cortes, and C. J. Burges. The mnist database of handwritten digits. `http://yann.lecun.com/exdb/mnist/`. Accessed: 2019-02-22.

[10] J. Li, H. He, and L. Li. Cgan-mbl for reliability assessment with imbalanced transmission gear data, September 2019.

[11] P. Mahalanobis. On tests and measures of group divergence. 1. theoretical formulae. *Jour And Proc Asiatic Soc Bengal*, 26(4):541–588, 1930( ), 1933.

[12] M. Mirza and S. Osindero. Conditional generative adversarial nets, November 2014.

[13] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2016.

[14] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. In Advances in Neural Information Processing Systems, 2016.

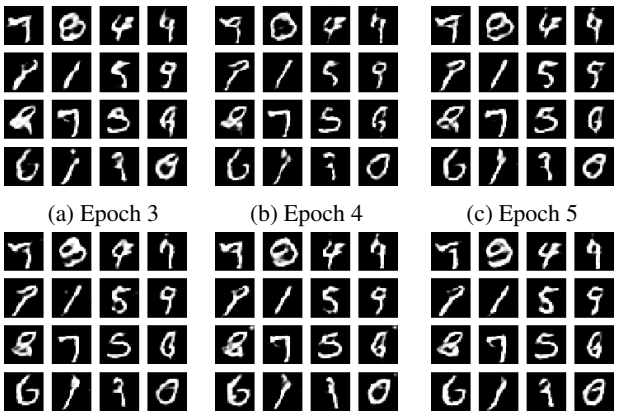
[15] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016.

A. Question 1

A.1. Hyper-parameter tuning for DCGAN and WGAN

Updates	Figures	Model	Disc. lr	Gen. lr	Ratio	Epochs	Training time (s)
Vanilla implementation	1	DCGAN	0.001	0.001	1:1	20	3,600
	5	WGAN-GP	0.001	0.001	1:1	8	10,700
Decreased learning rates	6a	DCGAN	0.0001	0.0001	1:1	25	4,600
	6b	WGAN-GP	0.0001	0.0001	1:1	10	12,800
Training ratio	7a	DCGAN	0.0001	0.0001	5:1	23	4,100
	7b	WGAN-GP	0.0001	0.0001	5:1	6	8,400
	7c	DCGAN	0.0001	0.0001	1:5	22	3,950
	7d	WGAN-GP	0.0001	0.0001	1:5	8	10,500
Increased discriminator lr	2a	DCGAN	0.0004	0.0001	1:1	19	3,700
	2b	WGAN-GP	0.0004	0.0001	1:5	10	11,000

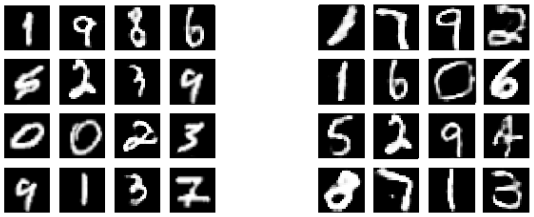
Table 2: Hyper-parameter tuning on DCGAN and WGAN-GP models. Ratio corresponds to the number of generator updates per discriminator update.



(a) Epoch 3 (b) Epoch 4 (c) Epoch 5

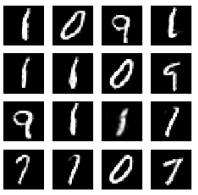
(d) Epoch 6 (e) Epoch 7 (f) Epoch 8

Figure 5: Images generated by the Vanilla WGAN trained on different epochs.

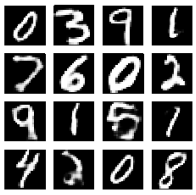


(a) DCGAN (b) WGAN

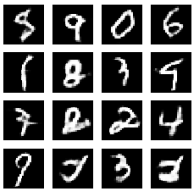
Figure 6: Images produced by GANs with learning rates for both competing networks set to 0.0001



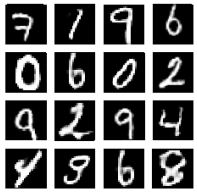
(a) DCGAN - 5:1 ratio



(b) WGAN - 5:1 ratio



(c) DCGAN - 1:5 ratio



(d) WGAN - 1:5 ratio

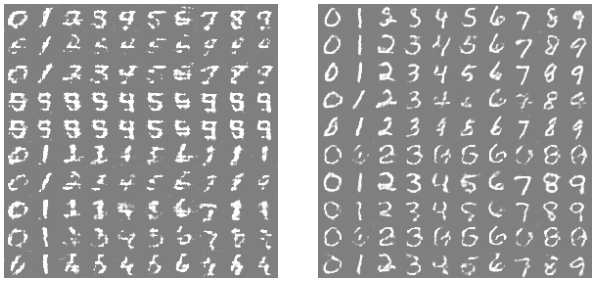
Figure 7: Images generated by the Vanilla WGAN trained on different epochs.

B. Question 2

B.1. CGAN

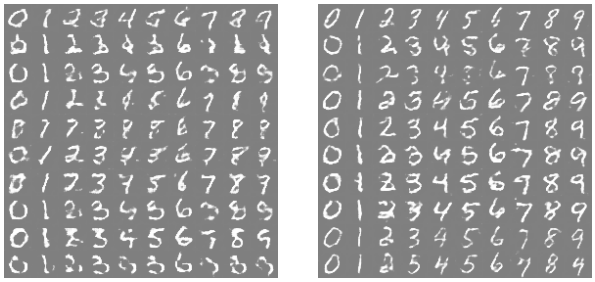
Updates	Figures	Model	Disc. lr	Gen. lr	Ratio	Epochs	Training time (s)
Vanilla implementation	8a	CGAN	0.001	0.001	1:1	15	3,000
	8b	CGAN-MBL	0.001	0.001	1:1	10	3,200
Optimal DCGAN hyper-parameters	9a	CGAN	0.0004	0.0001	1:1	19	3,700
	9b	CGAN-MBL	0.0004	0.0001	1:1	13	3,900
Label Smoothing	10a	CGAN	0.0004	0.0001	1:1	17	3,500
	10b	CGAN-MBL	0.0004	0.0001	1:1	14	4,100
Noise Injection	3a	CGAN	0.0004	0.0001	1:1	17	3,600
	3b	CGAN-MBL	0.0004	0.0001	1:1	13	3,800

Table 3: Hyper-parameter tuning on CGAN model. Ratio corresponds to the number of generator updates per discriminator update.



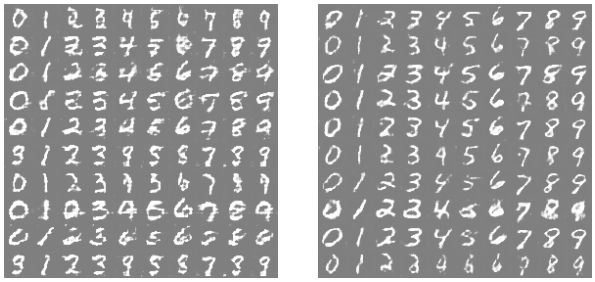
(a) CGAN (b) CGAN-MBL

Figure 8: Images produced with vanilla CGAN and CGAN-MBL architectures.



(a) CGAN (b) CGAN-MBL

Figure 9: Images produced with Optimal DCGAN hyper-parameter values found in Question 1.



(a) CGAN (b) CGAN-MBL

Figure 10: Images produced after applying Label Smoothing

C. Question 3

C.1. Inception Score

n_{splits}	1	2	5	10	20	50	100	200
Mean	7.9274	7.9101	7.8972	7.8596	7.8057	7.6833	7.4231	7.0440
Standard deviation	0	0.00241	0.1009	0.1796	0.2321	0.3176	0.3951	0.4881

Table 4: Inception Score for different n_{splits} of the CGAN-MBL model with noise injection