Coursework on Generative Adversarial Network

Ran Cheng(01405413)
ran.cheng18@imperial.ac.uk

Yang Zhao(01561245)
yang.zhao18@imperial.ac.uk

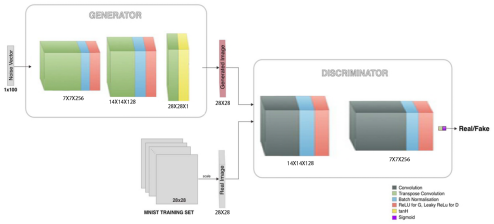

Figure 1. Architecture of DCGAN used as reference. The illustration is modified based on [3].

Table 1. Parameters as reference. See Architecture section for more detail.

Epoch	20
Batch Size	200
Keep Probability	0.6
Learning Rate	2e-4
G layers	3
D layers	3
Beta 1 (Adam Optimiser)	0.5
Restrictions on Loss Functions	$G \leq 2D, D \leq 2G$

1. DCGAN

The changes of DCGAN over conventional GAN are mentioned in [5]:

1. *Replace pooling layers with transpose (fractional-strided) convolutions in the generator and strided convolutions in the discriminator*, in order to extract the scale-invariant features efficiently and reduce the number of parameters to train.

2. *Use batch normalisation* to speed up learning, solve the problem of covariance shift, and avoid falling in the saturation region of the activation function to optimise the sensitivity of the network.

3. *Remove fully connected hidden layers* to maintain the information contained in the relative position of objects in images.

4. *Use ReLU in generators and LeakyReLU in discriminators* to avoid gradient vanishing and exploding.

1.1. Architecture

Figure 1 illustrates the architecture of DCGAN used as reference. The generator G produces samples x with desired features from noise z. There are three hyper-layers in the reference model, of which the first two consist of transposed convolution layer, dropout layer, batch normalisation layer and activation layer. The first layer takes a random vector of size 1×100 and performs transposed convolution by 256 kernels of size 7×7 with 1 stride and *valid* padding. Then, a dropout layer that randomly switches off some neurons [7] is used to avoid overfitting and reduce the training time and complexity. Next, it performs batch normalisation to avoid covariance shift and limit the input range of activation layers for sensitive error propagation. Finally, a leaky ReLU is used as the activation function. The second layer performs similarly, but the input is the result of the previous block. Also, 128 kernels of size 5×5 with 2 strides and *same* padding are used to produce a result of size $14 \times 14 \times 128$. Another same kernel is used in the convolution layer of the last layer to generate the output of size $28 \times 28 \times 1$, and an activation function of *Tanh* is used to synthesise the sample which is expected to carry some features.

The discriminator D takes an image of size 28×28 and output a value between 0 and 1 indicating the probability that the input is a real image. The transposed convolutions in the generator are replaced with strided convolutions, while Sigmoid function is used as the last activation function. The structure is the same as a reversed generator as implied by Figure 1. However, a bigger and deeper D can be used to optimise GANs.

1.2. Parameters, Architectures and Performance

Table 1 shows the parameters used in the reference model. The loss functions are based on the average value of 20 nearest iterations for each epoch. Notice that there exists some randomness in the GAN which can influence the result in a limited range.

1.2.1 Different Architectures

Figure 7 illustrates the output of D, loss functions, and synthesised samples for D and G with 2, 3, 4 layers. In the 2-layer model, the largest layer is removed. In the 4-layer model, an additional layer with 512 kernels of size 4×4 is added to D and G. As layers increase, the gap between $D's$ decision on real and fake images is narrowed from 0.4 to 0.3, which suggests the quality of synthesised samples is improved. For the 4-layer model, the trends of D and G loss are more stable with smaller relative entropies. It suggests a better strategy in the minimax game and enhanced network. Digits on the right also prove that increasing layers can reduce the noise. However, one more layer consumes about 15 seconds (10%) more per epoch.

1.2.2 Learning Rate

A small rate might lead to gradient vanishing because D may be too confident about the generated digits in the beginning, so the learning speed of G is slow. For a rate of 2e-5, the upper plots of Figure 8 denotes that the gap of $D's$ output on real and fake reaches 0.4 and D loss is significantly smaller than G loss (*i.e. D overpowers G*). As the learning rate increases, D is more likely to make mistakes, and the loss functions are balanced. However, the performance of network might be unstable if the learning rate is too large.

1.2.3 Batch Size

As illustrated by Figure 9, a large batch size uses more iterations in each epoch to push D real and fake to 0.5 and add balance to the network, but it might lead to overfitting. The result corresponding to a size of 300 (lower right) shows some "combined" samples made up of more than one digits. On the other hand, if the size is too small, the digits seem clearer but similar to each other (and possibly the training set). The synthesised samples with limited diversity are distinguishable from the real data, and the discriminator may regard them as "real" enough. This phenomenon is called mode collapse. It is a simple but effective strategy of G, but the cost is a restricted diversity [9]. Therefore, a balance between overfitting and mode collapse should be reached by carefully choosing the batch size.

1.2.4 Dropout Probability

The upper subplot of Figure 10 corresponds dropout layers with keep probability of 0.3. The digits are "thinner" than others because fewer nodes are engaged in the synthesis. In comparison, the lower subplot with probability 0.9 has a limited diversity. Since the parameters are shared, a suitable dropout probability can enhance the randomness and variety of the network.

1.2.5 Noise Distribution

Figure 11 shows the result of uniform noise. Gradient vanishing leads to meaningless outputs. Although D thinks good, the network is a failure. All digits look the same which is an extreme case of mode collapse.

1.2.6 Virtual Batch Normalisation (VBN)

For batch normalisation (BN), the mean and variance are obtained from the current minibatch, which inherently suffers the intra-batch correlation and leads to overfitting. VBN uses a minibatch as the reference and combines it with the current one before normalisation. It is expected to improve the generality of neural nets but the complexity is high, so it was used in the generator only.

Figure 12 proves the optimisation by VBN. The gap between $D's$ decision on real and fake images is slightly reduced, and the G loss is significantly improved. Also, the digits are more identifiable. Therefore, we regard VBN as an effective technique to improve GANs.

1.2.7 Balancing G and D

D tends to overpower G in most cases [3]. We tried a deeper D with an additional layer with 512 kernels of size 4×4 to optimise the network. As suggested by Figure 13, although the loss functions are similar, the digits by deeper D is with a better layout, but there are some scattered pixels. Using smaller kernels might help to solve the problem.

Training more D can be realised by adjusting restrictions of loss functions from balanced $G \leq 2D, D \leq 2G$ to biased $G \leq 1.5D, D \leq 3G$. Figure 14 shows that it might lead to a worse result because the gap between decisions of D on real and fake images is larger, which is unexpected. One possible reason is the randomness of the network. It can also be true that the method used to run more D may be invalid since the restrictions on the loss functions bring more extreme and imbalanced cases to the network. A fixed pattern like a run-time ratio of 2:1 may have a positive influence.

2. CGAN and Inception Score

GAN (DCGAN) can be extended to CGAN (CDCGAN) by feeding some extra condition which contains auxiliary information [4] to both generator and discriminator as an additional input. Condition y in this project is the label. In the training process, the number of digits should be balanced to avoid biasing or overfitting.

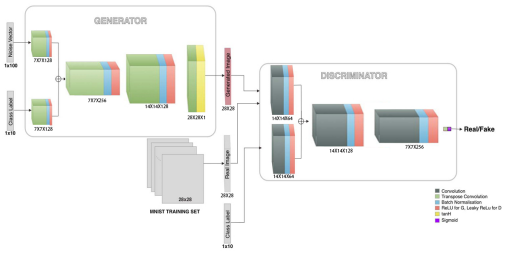

Figure 2. Architecture of CGAN used as reference. The illustration is modified based on [3].

Inception score (IS) measures the quality and diversity of the synthesised samples. The quality can be denoted by $p(y|x)$ where x is the image and y is the predicted label, and the diversity is measured by $p(y)$. GANs aim to reduce the conditional entropy of the predicted label by generating "real-enough" images. Moreover, if the generated samples are with good diversity, the entropy of the predicted label y will be large. Here we only consider the accuracy by comparing the prediction of a pre-trained classifier with the class label.

Compared with DCGAN, CGAN generally narrows the gap between $D's$ decision on real and fake samples and balances the D and G loss. Therefore, it is expected to generate samples better than those of DCGAN.

2.1. Architecture

Figure 2 shows a simplified structure of CGAN as reference. The one-hot code of label is concatenated with noise at the generator and concatenated with input image at the discriminator, then fed into the deconvolution network similar to DCGAN. The parameters in the reference model are the same as DCGAN to investigate the improvement by adding labels.

2.2. Parameters, Architectures and Performance

2.2.1 Different Architectures

Figure 15 illustrates the impact of layers. Adding layers benefits both sides in the minimax game and improves the quality and diversity of the synthesised samples. The result of the 2-layer model is somewhat similar for each class with minor inter-class mistakes, and the 4-layer model provides a great quality and diversity.

The success rate grows with the increase of layers. At the 20th epoch, the accuracy of models with 2 and 3 layers is around 0.8 while that of 4-layer model is more than 0.9. An interesting phenomenon is that the success rate is dominant by the weakest class (*e.g. number 8 or 9*), which is around 0.8 for the 4-layer and 0.7 for the models with 2 and 3 layers. It is recommended to train the network with biased data set with more 8 and 9 to reduce the complexity while maintaining the accuracy.

2.2.2 Learning Rate

Figure 16 examines the relationship between learning rate and output. At a rate of 2e-5, the model suffers from gradient vanishing and the output after 20 epochs are still ambiguous. The learning curve grows slowly, and more epochs are needed before saturation. In comparison, a rate of 2e-3 grants fast convergence (accuracy of 0.8 at the 4th epoch) but the ripple suggests instability. It can be seen in the lower plot that the D and G losses are growing instead of decreasing, and D overpowers G as training goes on. The rate should be determined to guarantee a stable and reproducible result with impact by randomness as small as possible.

2.2.3 Batch Size

As indicated by Figure 17, by increasing the batch size, D tends to make more mistakes. Also, the D and G loss are more likely to be coupled. For a size of 300 at epoch 20, the output of D on real and fake images is around 0.6 and 0.4, closed to Nash equilibrium (0.5 for both). In contrast to DCGAN where a size of 300 leads to overfitting, the label avoids inter-class impact effectively.

The advantage of large batch size is proved by the success rate. For a size of 300, the accuracy for digit 8 is only about 0.6, which is more than 0.8 for the other digits. In comparison, the error for a size of 200 is uniformly distributed around 0.7 for all classes. It proposes an optimisation using a large batch size and a biased training set with more digit 8.

2.2.4 Dropout Probability

Figure 18 implies an optimum keep rate of 0.9. Although using a lower keep rate can enhance the diversity, the pre-trained classifier might not recognise all variant forms. Therefore, a larger keep rate appears a prudent strategy if the evaluation is fully based on the output of the classifier. The lower right instance of digits also proves that the synthesised samples based on a large keep rate as 0.9 is dissimilar enough and there is no mode collapse issue.

2.3. Noise Distribution

Figure 19 based on noise with uniform distribution shows a typical case of mode collapse. It can be seen that the generated samples of the same class are almost identical, because the decision and feedback of D cannot influence G positively. Therefore, uniformly distributed noise is not an acceptable option.

2.3.1 One-Sided Label Smoothing

Label smoothing avoids extremely confident decisions by replacing 0 and 1 with smoothed probabilities as 0.1 and 0.9 then calculate the relative entropy. One-sided label smoothing only performs smoothing for true samples because that on the fake side would have no incentive to improve the generator [1] but to encourage it to synthesis samples similar to the training or existing data [8].

As Figure 20 suggests, smoothing with probability 0.9 is better than 0.85, and the performance of both cases is better than the unsmoothed one. The value of under-smoothed (*i.e. approaching 1*) and over-smoothed (*i.e. too small*) should be carefully chosen to balance the model and maximise the accuracy. It is a valid optimisation strategy because the fine-tuning only shifts the accuracy and loss curves vertically and the trends of loss functions are not influenced significantly.

2.3.2 Virtual Batch Normalisation (VBN)

It is proved by Figure 21 that VBN increases the success rate from 0.8 to 0.95 in epoch 20. Not only the accuracy on the worst digit class is raised from 0.6 to 0.9, but also the performance on all classes is significantly enhanced. Although the loss curves are similar, the digits synthesised with VBN are thicker and with fewer shakes, which benefits from the reference batch.

2.3.3 Balancing G and D

Figure 22 shows that by adding a layer to D, the gap between its output on real and fake samples is increased from 0.3 to around 0.4. Also, the D loss is significantly smaller than G loss, which means D overpowers G. It is somewhat strange because [3] mentioned that a larger and deeper D should mitigate the imbalance rather than enlarge it. However, the success rate grows from 0.8 to 0.95, providing more natural digits of class 2 and 4.

Not much changed if we run more D than G by adjusting restrictions, as illustrated in Figure 23. One possible reason is that improper limits may bring more imbalanced cases. Running D twice and G once in each iteration might be more useful to optimise the network.

3. Retraining the Handwritten Digit Classifier

3.1. Injection of Synthetic Images

The same architecture and hyper-parameter CGAN and classifier are applied in this part. When varying the portions, the number of the real training set is fixed to 1000, and only the number of synthetic images is changed. The classifier trained with 1000 real images already achieves an accuracy about 94.2%.

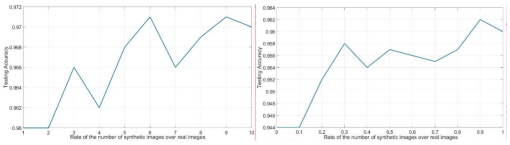

Figure 3. The evaluation accuracy on testing set with the ratio of synthetic images to real images greater than 1 (left) and less than 1 (right)

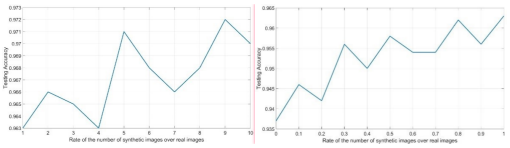

Figure 4. The evaluation accuracy on testing set with the ratio of synthetic images (without class 8) to real images greater than 1 (left) and less than 1 (right)

As shown in Figure 3, the accuracy grows with the increase of the portion of plenty synthetic images. "Plenty" means the number of synthetic images is greater than the number of real images. Furthermore, all evaluation accuracies with the addition of synthetic images are higher than the accuracy that trained with real samples only. Since only 100 per class real images are used, the diversity of the real set is low. Thus, the classifier trained with the purely real set has a limited generalisation ability. After the injection of plenty synthetic images in high quality (95.3% accuracy), the diversity of training set is improved, resulting in a higher generalisation ability.

If the number of synthetic images is less than the number of real images, the improvement is more obvious than injecting plenty synthetic images. The reason is that the enhanced diversity tends to saturate with plenty synthetic images. The accuracy of only using plenty real images as training set is 98.6%. It has the highest accuracy compared to injecting synthetic images, due to its high diversity and high quality. The effect of injecting synthetic images into a large number of real images can be ignored because a large real training set has already given the highest recognition rate.

3.2. Injection of Specific Classes of Synthetic Images

Results above suggests the class of number 8 have the lowest accuracy. Thus, a method of injecting synthetic images without class 8 is applied. This method causes an imbalanced training, however, Figure 4 shows that it slightly improves the evaluation accuracy when the portion of synthetic images is low.

After removing synthetic images of class 8, the average accuracy of the rest 9 classes is improved. However, the evaluation accuracy on purely synthetic images is reduced

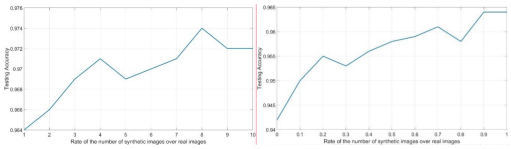

Figure 5. The evaluation accuracy according to confidence score on testing set with the ratio of synthetic images to real images greater than 1 (left) and less than 1 (right)

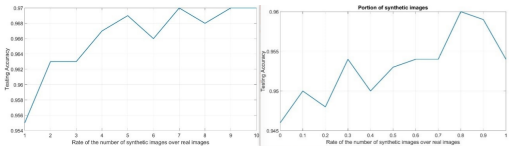

Figure 6. The evaluation accuracy with separate training on testing set with the ratio of synthetic images to real images greater than 1 (left) and less than 1 (right)

to 87.1%, because one class of images is missing. Therefore, this method only works when the portion of synthetic images is low. For the injection of plenty synthetic images, the method should be avoided.

3.3. Confidence Score

The output of classifier gives the probabilities of the label of the input image. The probabilities are proportional to *exp(output logits)*. The label with the highest probability is the prediction. To improve the evaluation accuracy, the synthetic images are input to a pre-trained classifier that trained with the real data. Next, the synthetic image is removed, if its probability of the predicted label is less than 99.9%. Finally, the rest synthetic images are added into the real images.

Figure 5 suggests that this training strategy achieves the highest accuracy, due to the high quality of synthetic images. After filtering by the pre-trained classifier, the synthetic images with lower quality are removed. The rest images have a similar quality to real images. Thus, the evaluation accuracy is improved.

3.4. Training in Turn

In this training strategy, the classifier is first trained using synthetic images, and then fine-tuned using the real images. The first step achieves high diversity and increases the generalisation ability of the classifier. The second step improves the accuracy of the classifier, due to the higher quality of real images.

Since the classifier training with purely synthetic images can achieve the accuracy of 95.3%, the parameters of the classifier are close to the ideal values after the first step. In the second step, the distance between the current parameters and the ideal parameter is reduced further, leading to a more stable performance. However, the strategy has a high requirement on the quality of synthetic images. If not high enough, the second step can not adjust the classifier to a high recognition accuracy.

3.5. Solved Problem

In the early stage of program development, the evaluation of testing set accuracy is performed more than once, and the results are found to be invalidated. The evaluation of the testing set only can be performed once. After each evaluation, all variables should be initialised, and the classifier should be not affected by the evaluation.

4. Ideas to Improve GAN

4.1. Generator: Feature Matching

The generated images might be too ”fake” at the early stage that leads to gradient vanishing and unstable training process of the generator. Feature matching avoids the problem by generating data that matches the statistics of features on an intermediate layer of the discriminator $f(\mathbf{x})$, rather than directly maximising its output. The discriminator is trained in the usual way. [6] mentioned that feature matching is effective for unstable GAN by empirical results although there is no guarantee to reach the practical distribution (optimum model).

4.2. Discriminator: Minibatch Discrimination

One possible solution to mode collapse is minibatch discrimination. It examines multiple examples in combination rather than in isolation, then calculates the similarity of the generated samples as feature map input to the next layer [6]. In other words, the discriminator not only classifies the real and generated data but also uses side information provided by other samples in the minibatch to encourage the generator to produce more diversity. Therefore, it is expected to create appealing samples in fewer epochs, and the training process can be more effective.

4.3. Activation: Maxout

GANs with on dropout layers are essentially subsets of a complicated model because their parameters are shared. Based on the idea of bagging, a maxout network containing m additional hidden layer each consists of k affine feature maps can be introduced as activation to optimise the performance of convolutional networks [2]. Each affine feature map performs a piecewise linear approximation to an arbitrary convex function, and a maxout is constructed by choosing the maximum to introduce nonlinearity. It can fit and replace activation functions and adapt the parameters through training. However, the number of parameters in the network increase as well.

References

[1] 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.

[2] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. *arXiv preprint arXiv:1302.4389*, 2013.

[3] T.-K. Kim. Lecture on generative adversarial networks, February 2019.

[4] M. Mirza and S. Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

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

[6] 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*, pages 2234–2242, 2016.

[7] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[8] 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*, pages 2818–2826, 2016.

[9] H. Thanh-Tung, T. Tran, and S. Venkatesh. On catastrophic forgetting and mode collapse in generative adversarial networks. *arXiv preprint arXiv:1807.04015*, 2018.

5. Appendices

5.1. DCGAN

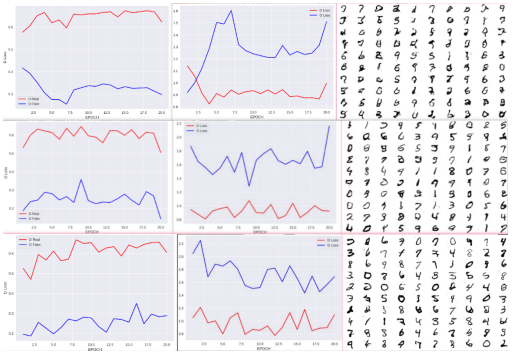

Figure 7. Top: 2-layer D and G. Middle: 3-layer D and G. Bottom: 4-layer D and G. [Left: output of D for real (red) and fake (blue) images. Middle: loss functions of D (red) and G (blue). Right: generated samples]

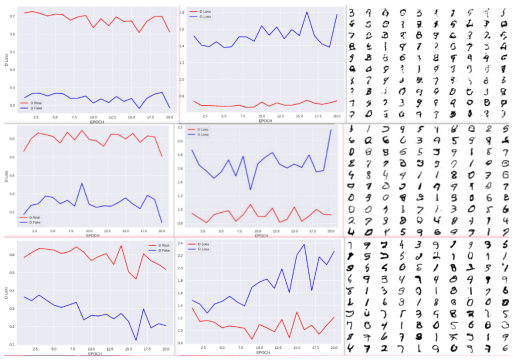

Figure 8. Top: learning rate 2e-5. Middle: learning rate 2e-4. Bottom: learning rate 2e-3. [Left: output of D for real (red) and fake (blue) images. Middle: loss functions of D (red) and G (blue). Right: generated samples]

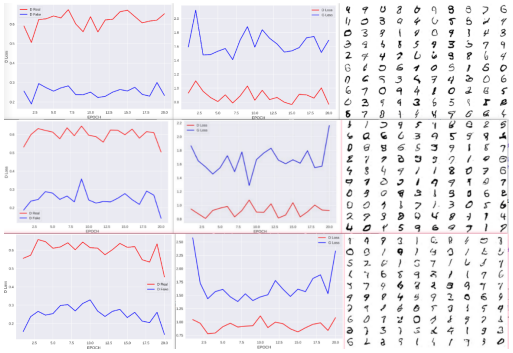

Figure 9. Top: batch size 100. Middle: batch size 200. Bottom: batch size 300. [Left: output of D for real (red) and fake (blue) images. Middle: loss functions of D (red) and G (blue). Right: generated samples]

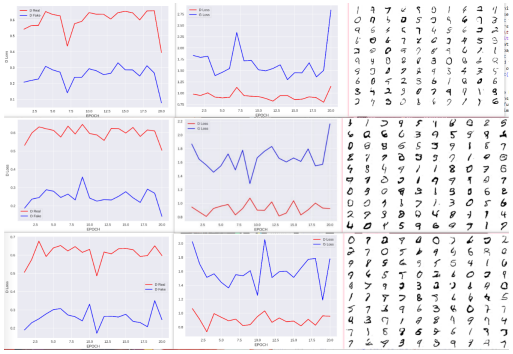

Figure 10. Top: keep probability 0.3. Middle: keep probability 0.6. Bottom: keep probability 0.9. [Left: output of D for real (red) and fake (blue) images. Middle: loss functions of D (red) and G (blue). Right: generated samples]

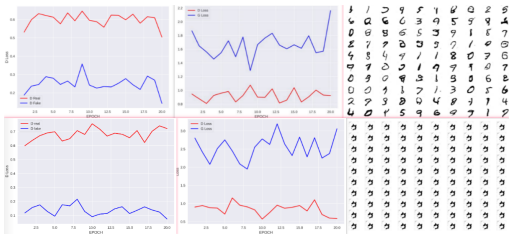

Figure 11. Top: Gaussian noise. Bottom: uniform noise. [Left: output of D for real (red) and fake (blue) images. Middle: loss functions of D (red) and G (blue). Right: generated samples]

Figure 12. Top: standard batch normalisation. Bottom: virtual batch normalisation. [Left: output of D for real (red) and fake (blue) images. Middle: loss functions of D (red) and G (blue). Right: generated samples]

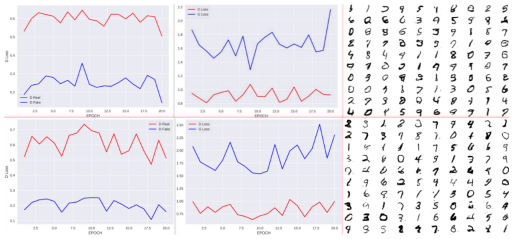

Figure 13. Top: 3-layer discriminator. Bottom: 4-layer discriminator. [Left: output of D for real (red) and fake (blue) images. Middle: loss functions of D (red) and G (blue). Right: generated samples]

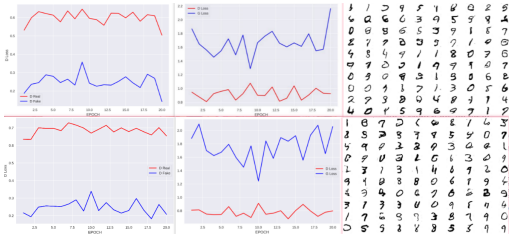

Figure 14. Top: train D and G for same times ($G \leq 2D, D \leq 2G$). Bottom: train more D than G ($G \leq 1.5D, D \leq 3G$). [Left: output of D for real (red) and fake (blue) images. Middle: loss functions of D (red) and G (blue). Right: generated samples]

5.2. CGAN

Figure 15. Top: 2-layer D and G. Middle: 3-layer D and G. Bottom: 4-layer D and G. [Left 1: Average and class accuracy. Left 2: output of D for real (red) and fake (blue) images. Right 1: generated samples. Right 2: loss functions of D (red) and G (blue)]

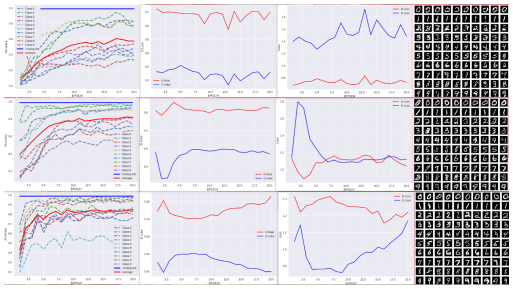

Figure 16. Top: learning rate 2e-5. Middle: learning rate 2e-4. Bottom: learning rate 2e-3. [Left 1: Average and class accuracy. Left 2: output of D for real (red) and fake (blue) images. Right 1: generated samples. Right 2: loss functions of D (red) and G (blue)]

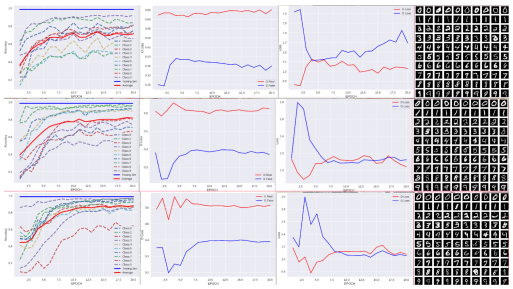

Figure 17. Top: batch size 100. Middle: batch size 200. Bottom: batch size 300. [Left 1: Average and class accuracy. Left 2: output of D for real (red) and fake (blue) images. Right 1: generated samples. Right 2: loss functions of D (red) and G (blue)]

Figure 18. Top: keep probability 0.3. Middle: keep probability 0.6. Bottom: keep probability 0.9. [Left 1: Average and class accuracy. Left 2: output of D for real (red) and fake (blue) images. Right 1: generated samples. Right 2: loss functions of D (red) and G (blue)]

Figure 19. Top: Gaussian noise. Bottom: uniform noise. [Left 1: Average and class accuracy. Left 2: output of D for real (red) and fake (blue) images. Right 1: generated samples. Right 2: loss functions of D (red) and G (blue)]

Figure 20. Top: unsmoothed. Middle: smoothed with probability 0.85. Bottom: smoothed with probability 0.9. [Left 1: Average and class accuracy. Left 2: output of D for real (red) and fake (blue) images. Right 1: generated samples. Right 2: loss functions of D (red) and G (blue)]

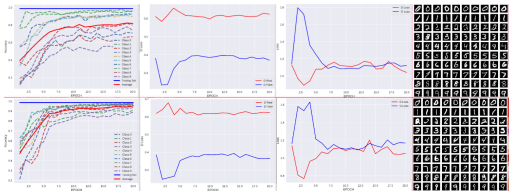

Figure 21. Top: standard batch normalisation. Bottom: virtual batch normalisation. [Left 1: Average and class accuracy. Left 2: output of D for real (red) and fake (blue) images. Right 1: generated samples. Right 2: loss functions of D (red) and G (blue)]

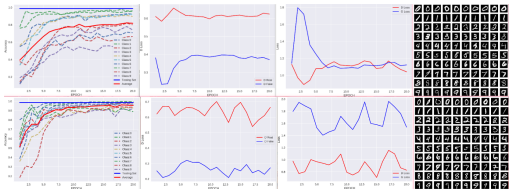

Figure 22. Top: 3-layer discriminator. Bottom: 4-layer discriminator. [Left 1: Average and class accuracy. Left 2: output of D for real (red) and fake (blue) images. Right 1: generated samples. Right 2: loss functions of D (red) and G (blue)]

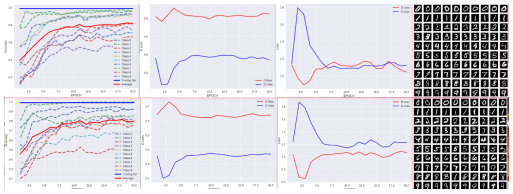

Figure 23. Top: train D and G for same times ($G \leq 2D, D \leq 2G$). Bottom: train more D than G ($G \leq 1.5D, D \leq 3G$). [Left 1: Average and class accuracy. Left 2: output of D for real (red) and fake (blue) images. Right 1: generated samples. Right 2: loss functions of D (red) and G (blue)]