Machine Learning for Computer Vision CW2

Tung-Duong Mai (20180745)
School of Computing
KAIST
Daejeon, South Korea
john_mai_2605@kaist.ac.kr

Tien Dat Nguyen (20190883)
School of Computing
KAIST
Daejeon, South Korea
kaistdat123@gmail.com

***Abstract*—Coursework 2 on Generative Adversarial Network**

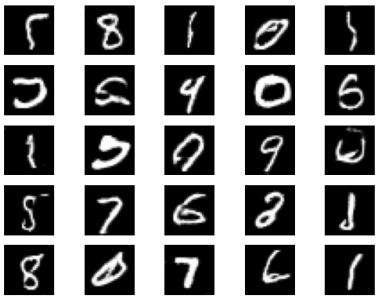
I. Deep Convolution GAN (DCGAN)

A. Definition of good GAN

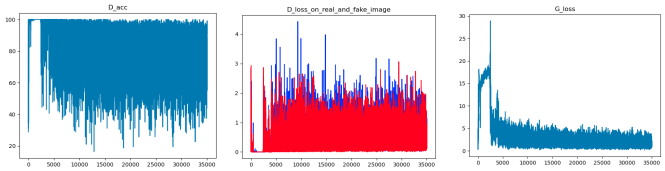
Generally, a good set of generated images should have high quality and diversity. Some rigorous evaluation schemes will be addressed in depth in Section. III.

B. Some common hindrances in training

1) Convergence failure and vanishing gradient: We first tried the architecture shown in Fig. 2a. Some other hyperparameters: latent dimension 25, batch size 64, Adam optimizer with learning rate 0.002 and beta_1 = 0.95. We conducted 35000 epochs of training, and some generated images are shown in Fig. 1a. Intermediate epochs' images are in Appendix.



(a) Some output images



(b) Accuracy (c) Discriminator loss (d) Generator loss

Fig. 1. DCGAN with convergence failure and vanishing gradient: (a) Some generated images, (b)-(d) Training history

The quality of generated image is improved continuously up to epoch 19000. From epoch 19000, the quality is not improved much (or even worse such as digits 6 and 9). At the end of the training (epoch 35000), some generated images look reasonable but there also exist many unrealistic ones. The model does not really work well.

Let's analyze the training history (Fig. 1b-1d). The training process is unstable. Generator loss, albeit stable after epoch 5000, still has a high variance. *Discriminator loss* on both generated (red) and real images (blue) has a very high variance and diverges. *Discriminator accuracy* varies wildly from 20% to 100%. We also observe many epochs when the discriminator reaches (or nearly reaches) 100% accuracy. In these epochs, the discriminator's overperformance might lead to near-zero gradient for the generator. Overall, our model fails to generate high-quality images due to unstable and diverging training.

2) Mode collapse: We do not face this problem

C. Resolve the hindrances

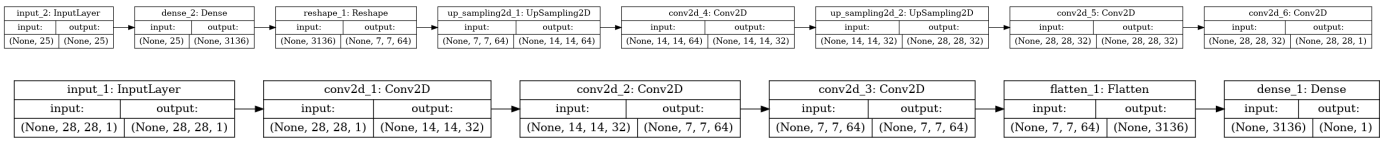
We revise the DCGAN architecture (Fig. 2b) and hyperparameters:

- With deeper architecture, the generator is easily overpowered by the discriminator [1]. We add more layers to the discriminator to make it deeper than the generator, and increase the number of channels in all layers to boost the model capacity.
- The batch size of 64 might be too big, making the discriminator overperformant. This hurts the performance during initial training. We reduce batch size to 32
- The learning rate 0.002 and beta1 0.9 seems to be too high, leading to unstable training. We decrease the them to 0.0002 and 0.5, respectively.
- The latent dim 25 is too small. We adjust this to 100.

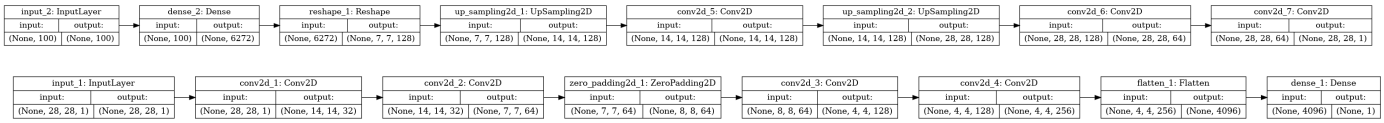
D. Successful model

The quality of generated images are improved continuously up to epoch 28500, with higher quality of images compared to previous model. The highest quality generated images is obtained at epoch 28500, shown in Fig. 3a (for other epochs, refer to the Appendix). All generated images at epoch 28500 looks very close to the MNIST images. We also observe a high diversity of generated images, with all digits from 0 to 9 appears at least once. Our model is successful, resolving the convergence failure, vanishing gradient and avoiding mode collapse.

Let's analyse the training history in Fig. 3b-3d. There are a stable trend in all generator loss and discriminator loss and accuracy. Although their variance increase over time, they do not show any erratic change, indicating a stable training process. Discriminator accuracy stays around 60% and never

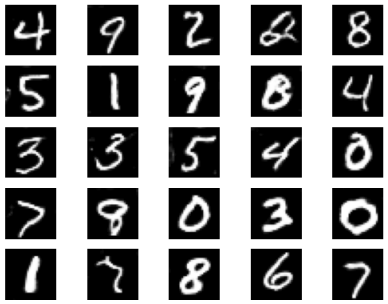


(a) Model with convergence failure and vanishing gradient (top: generator, bottom: discriminator)

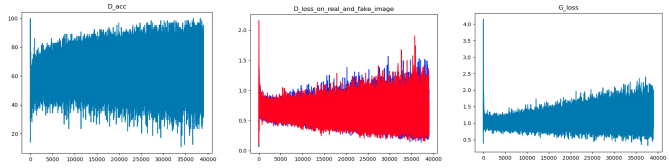


(b) Successful model (top: generator, bottom: discriminator)

Fig. 2. DCGAN architectures



(a) Some output images



(b) Accuracy (c) Discriminator loss (d) Generator loss

Fig. 3. Successful DCGAN: (a) Some generated images, (b)-(d) Training history

reaches 100%, eliminating the vanishing gradient. Note that the 60% accuracy is very close to the Nash equilibrium (50%), indicating a reasonable discriminator performance.

E. Further improvement

In GAN, if the discriminator is overconfident, it will only use a small set of features to discriminate real and fake images. The gradient might be too large, making the backpropagation too aggressive, and the generator will only try to produce the same set of features. We apply "one-sided label smoothing" by adjusting the label for real image to 0.9. If the prediction for any real images is higher than 0.9, it is penalized. We avoid the large gradient and subsequently improve the model.

After training the new model, we observe that the images and training history is similar with the successful model (Fig. 4). To rigorously check if there is an improvement, we compare their Inception score (detail in Section. III). The inception score for the first (diverge) model, second (successful) model and third (one-sided label applied) model are 8.260, 9.067, 9.037 respectively. The inception score ranges from 1 to 10 and higher inception score means better model performance. We conclude that applying the one-sided label smoothing technique does not bring any clear improvement to the performance of our model.

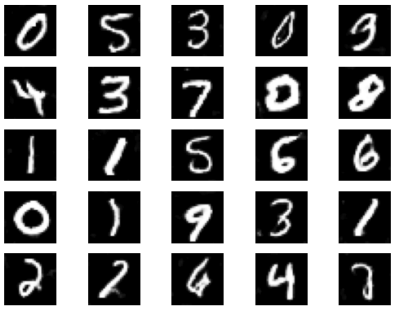


Fig. 4. Some generated images of DCGAN with smoothing

II. Conditional GAN (CGAN)

A. Some common hindrances in training

1) Mode collapse: We use the architecture shown in Fig. 6. Some other hyperparameters: latent dimension 1, Adam optimizer with learning rate 0.0002 and beta_1 = 0.5. Some generated images are shown in Fig. 5. We observes the mode collapse, where generated images for digits 0, 1, 4, 5 look almost the same. It seems that using a too small latent dimension has led to the mode collapse problem.

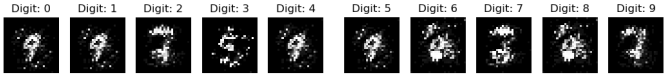


Fig. 5. Some generated images of CGAN with mode collapse (left to right: 0 to 9)

2) Convergence failure and vanishing gradient: We design a simple architecture with only two fully connected layers for the generator and two fully connected layers for the discriminator. There are only around 230k and 400k trainable parameters for generator and discriminator. Some hyperparameters: latent dimension 100, Adam optimizer Adam with learning rate 0.005 and beta_1 = 0.95.

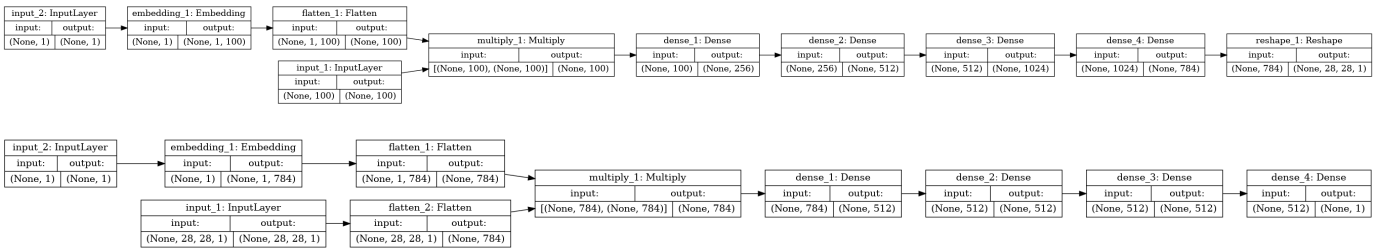
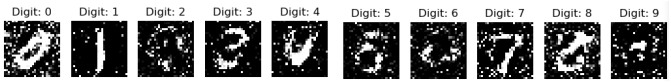
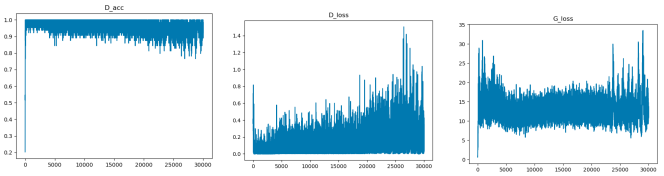


Fig. 6. CGAN architectures (top: generator, bottom: discriminator)



(a) Some output images (left to right: 0 to 9)



(b) Accuracy (c) Discriminator loss (d) Generator loss

Fig. 7. CGAN with convergence failure and vanishing gradient: (a) Some generated images, (b)-(d) Training history

From Fig. 7, the quality of generated images are extremely low even for later epochs. We attribute poor performance of the generator to the discriminator's overperformance (100% accuracy and 0 loss for many epochs). This leads to vanishing gradient for the generator. The loss of discriminator and generator are very unstable with large variance (generator loss ranges from 10 to 30, and discriminator loss ranges from 0 to 1.5). Our model fails to converge to the optimal point with an unstable training process, indicating convergence failure. We conjecture that the model is too simple and lacks training capacity, and the learning rate, beta_1 are so high that the backpropogation becomes too aggressive.

B. Resolve the hindrances

To solve the mode collapse, we increase latent dimension from 1 to 100. To overcome convergence failure and vanishing gradient, we use an adequately complex and capable model for both discriminator and generator. The architecture is shown in Fig. 6. We also decrease the learning rate and beta_1 to 0.0002 and 0.5, respectively, to regulate the back-propagation.

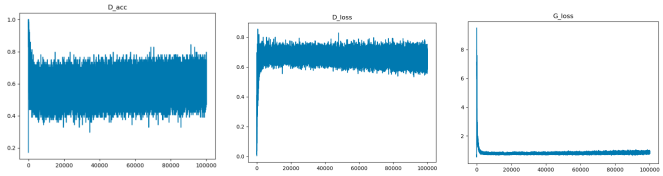
C. Successful model

The quality of generated images are remarkably improved, and at the end of training, generated images of all digits look very realistic. (Fig. 8a). We also observe a high diversity of generated images, thus our successful model does not get mode collapse.

The losses of discriminator and generator become more stable and have reasonable ranges without erratic change (Fig. 8b-8d). Discriminator accuracy stays around 60% and never reaches 100%, eliminating the vanishing gradient. Note that the 60% accuracy is very close to the Nash equilibrium (50%), indicating a reasonable discriminator performance. Thus, our successful model does not suffer convergence failure.

We also report the success rate and Inception score of this model (detail in Section. III) as 97.6% and 9.620. These high scores confirm the generated images are realistic and diverse.

(a) Some output images (left to read: 0 to 9)



(b) Accuracy (c) Discriminator loss (d) Generator loss

Fig. 8. Successful model: (a) Some generated images, (b)-(d) Training history

D. Further improvement

We apply one-sided label smoothing (explained in Section. I). We set the positive target value for discriminator to 0.9 instead of 1 and then train the model again. Some generated images are shown in Fig. 9. The quality are good and nearly identical to the previous model. Training history is also similar with the old model. To be rigorous, we report the success rate and inception score (detail in Section. III) as 97.9% and 9.717 (compared to 97.6% and 9.620 before smoothing). We conclude that applying one-sided label smoothing technique slightly improves the performance of our model.

III. Evaluation Metrics

We train a neural network classifier that include convolutional layers and a softmax layer, using the MNIST training set. The model architecture is shown in the Appendix. From now on, we will refer to this classifier as *pretrained classifier*.

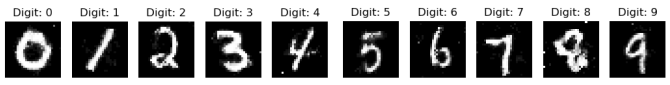


Fig. 9. Some images generated by CGAN with smoothing

A. Success rate

The generator of from previously trained (successful) CGAN models is used to generate 1000 for each digit from 0 to 9 (10000 images in total). Use the pre-trained classifier to predict the labels for them, and then compare these predicted labels with the labels that we fetch into CGAN to generate these images. Then the success rate is computed as the ratio between the number of correct prediction and the number of generated images. To have a reasonable baseline, we also perform classification on MNIST real testing set and observe the classifier accuracy.

The success rate of our trained CGAN is 97.6% and the accuracy of the pretrained classifier on MNIST real testing set is 98.38%. The accuracy of our classifier on the MNIST real testing set is nearly 100%, meaning that it performs very well on the classification task and is reliable. Expectedly, the success rate on the generated images is smaller than the accuracy on real testing images. Importantly, the success rate on generated images are also very high, showing that our trained CGAN is able to generate very high-quality and realistic images.

We also perform the evaluation for different models in Section. II and report the result in Table. I

B. Inception score [2]

$$\text{IS}(\mathbb{P}_g) = e^{\mathbb{E}_{x\sim\mathbb{P}_g}[KL(p_M(y|x)\|p_M(y))]} \quad (1)$$

M is the pretrained network. The KL divergence is higher if $p_M(y|x)$ close to a point mass and has $p_M(y)$ close to uniform, (i.e. M is very confident about the images and the images distribute equally to all categories). This suggests the image has high quality and diversity.

Inception score ranges from 1.0 to number of classes supported by the classifier (10 in our setting). Inception score was designed to measure both the image quality and image diversity. A high inception score means that the CGAN performs very well, able to generate high-quality images of good diversity. A low inception score means that the CGAN generates low-quality images and/or can only generate images of low diversity (mode collapse problem).

The generator of the trained CGAN model is used to generate 10000 images, with 1000 images for each digit from 0 to 9. Then, we use the pretrained classifier to get the predicted class probability vectors for each generated image. The 10000 collected vectors is divided into 10 groups. Inception score each calculated for each group by formula. The average of these 10 inception scores is reported in Table. I.

The successful CGAN model achieves a very good inception score (9.620, almost reaching the optimal 10). This indicates that our CGAN is able to generate high-quality images with good diversity. The model with mode collapse problem has quite low inception score (4.264), since it can only generate images of low diversity, thus penalized by the inception formula. Finally, the model with problem of convergence failure also has a low inception score (4.403), due to the low-quality generated images, thus penalized by the inception formula.

TABLE I
EVALUATION RESULTS OF CGAN

Model	Success	Mode collapse	Convergence failure
Success rate ↑	**0.976**	0.212	0.296
Inception score ↑	**9.620**	4.264	4.403
FID ↓	**0.001**	0.338	0.342
MMD ↓	**0.014**	0.203	0.240
Mode score ↑	**0.998**	0.440	0.546

↑: the higher the better; ↓: the lower the better

C. Fréchet Inception Distance (FID) [3]

We approximate the embedding of real and fake images with Gaussian distributions of the same mean and variance. FID score calculate the Fréchet distance between these two distributions. A small FID score indicate the proximate distributions of real and fake images, i.e. generated images are realistic.

$$\text{FID}(\mathbb{P}_r, \mathbb{P}_g) = \|\mu_r - \mu_g\| + \text{Tr}(C_r + C_g - 2(C_r C_g)^{1/2}) \quad (2)$$

We report the FID scores for several CGANs in Table. I. While the successful model has almost 0 FID, other model has quite high distance (around 0.3). This suggests FID can catch both mode collapse and divergence problem.

D. Others

We also use some other measurements: MMD score, mode score [3]. MMD measures the dissimilarity between two distribution; while mode score is enhanced version of Inception score extended with similarity between two distribution. These scores are also reported in Table. I, following the same trends with other metrics, with the best score on successful model.

IV. RE-TRAINING THE HANDWRITTEN DIGIT CLASSIFIER

A. When is GAN-generated examples helpful?

We use the successful CGAN's generator to generate 60000 fake images of 10 digits (6000 each). Then, we use different portions of real and fake images to retrain the classifier and the result is reported in Table II.

TABLE II
CLASSIFICATION ACCURACY WITH DIFFERENT DATA SIZE

Real \ Fake	**0**	**6k**	**12k**	**30k**	**54k**	**60k**
0	NA	0.9000	**0.9053**	0.9047	0.8960	0.8971
600	0.8704	0.9103	**0.9265**	0.9241	0.9258	0.9188
6k	0.9588	0.9595	0.9600	**0.9637**	0.9591	0.9543
12k	0.9691	0.9685	0.9702	**0.9710**	0.9700	0.9683
30k	**0.9812**	0.9807	0.9797	0.9809	NP	NP
54k	**0.9857**	0.9844	0.9857	0.9842	NP	NP
60k	**0.9855**	0.9838	0.9849	0.9845	NP	NP

*NA: Not applicable
*NP: Not performed, due to the lack of computational resources and the trend is already clear.

For very small size of real data ($\leq$ 600), addition of small portion of generated fake images significantly improves the performance. Because the generator captures the information

of 60k images, the generated images add useful information to the model. The optimal fake data size is around 12k. Further addition decreases the performance, suspectedly due to the overfitting to the fake data (fake data, despite very similar with real data, still has some small distinct characteristic that the model should not learn).

For the medium size of real data (6k - 12k), the performance only increases marginally with the addition of fake images. Because we use bigger real data, we now have more information and the fake images are less useful. Expectedly, more fake data is needed to boost the performance. The optimal fake data size is 30k, and further addition leads to overfitting.

For the big size of real data ($\geq$ 30k), additional fake images does not improve the performance and leads to slight overfitting. The fake images adds no useful information to the model, since we already have abundance of real data.

Expectedly, the model performs better with increasing size of real data. The performance saturates at around size 54000.

We conclude that, generated images is practically helpful with small size of real data (around 600).

B. Optimal ratio of real and fake images

Observing that the generated data is helpful with data size around 600, we have a closer look into this range and perform experiment on different ratio for the total training size of 120, 300, 600 and 1200. The accuracy is report in Table.III.

TABLE III
ACCURACY FOR DIFFERENT REAL AND FAKE IMAGES RATIO

Real : Fake ratio	**Total training size**			
	120	**300**	**600**	**1200**
1.0 : 0.0	0.6947	0.8180	0.8680	0.9087
0.9 : 0.1	0.7077	0.8334	**0.8777**	**0.9116**
0.8 : 0.2	0.7507	0.8368	0.8720	0.8999
0.5 : 0.5	0.7813	0.8432	0.8715	0.8944
0.2 : 0.8	0.7818	**0.8619**	0.8715	0.8899
0.1 : 0.9	0.7907	0.8500	0.8751	0.8811
0.0 : 1.0	**0.7967**	0.8395	0.8727	0.8804

For all cases, using some fake images improve the accuracy. For extremely small training size of 120 and 300, the accuracy is improved by 5-10%. For bigger training size, the accuracy is marginally improved (less than 1%). We attribute the improvement to the informativeness of fake images, which capture information from 60k images when training with CGAN. With increasing training size, this information appear more and more in the real data, and become less useful.

The optimum fake images ratio drops with increasing training size. The generated images have some noises that the model should avoid to learn. As we train with bigger size, the informativeness is less helpful and outweighed by the noise effect. Further addition leads to overfitting to noises of fake images.

C. Training strategies

The strategy we use so far is referred as mixing (pretrain with real and retrain with both real and fake images). Some different training strategy is performed. We try to pretrain the model with fake images and then finetune with real images. We also try to pretrain the model with real images, then finetune with the most confident (elite) fake images. The results are reported in Table. IV. Mixing outperform across all training size. This suggests that there is some correlation or interaction between real and fake images which is helpful in training. Separating the two sets of image adversely affect the performance of the training.

TABLE IV
ACCURACY OF DIFFERENT TRAINING STRATEGIES

Training size	120	300	600	1200
Optimal real : fake images ratio	0.0 : 1.0	0.2 : 0.8	0.9 : 0.1	0.9 : 0.1
Mixing	**0.7967**	**0.8619**	**0.8777**	**0.9116**
Pretrain: fake Retrain: real	-	0.7992	0.8652	0.9044
Pretrain: real Retrain: (elite) fake	-	0.8420	0.8419	0.8940
Baseline All real images	*0.6947*	*0.8180*	*0.8680*	*0.9087*
Benchmark Augmentation	*0.7410*	*0.8635*	*0.9243*	*0.9308*

D. Baseline and benchmark

Table. IV also reports the baseline and benchmark for a better grasp of our model's performance. Our baseline is simply a model train with all real data. The benchmark is model trained with the conventional augmentation techniques (rotation, zoom, brightness adjustment, translation).

Mixing outperforms the baseline in all cases, proving the informativeness of fake images (which is generated by CGAN on 60k real images). Mixing performs better than the benchmark in small size, but worse with bigger size. This suggests that GAN-generated images could be used as the augmentation techniques, especially for small training size.

V. FURTHER EXPERIMENTS

A. GAN vs PCA

Purpose of PCA is finding a subspace that maximizes the projected data variance. PCA supports incremental learning. Rather than generate new images, PCA represents (concisely) current data. Training PCA is straightforward with a well-defined procedure.

The purpose of GAN is to estimate the distribution of training data. GAN does not support incremental learning. A well-trained GAN can generate new data usable for other tasks (classification, ...). Training GAN is hard with a wide range of architectures and hyper-parameters that requires trial-and-error. Training GAN can fall into many of problems such as mode collapse, convergence failure, vanishing gradient, ...

B. Embedding and confidence

We extract the classifier's penultimate layer's activation and perform t-SNE to get 2-D projection shown in Fig. 11a. We can see that the fake images is very close to and usually surround the same-class real images cluster. This makes the generated

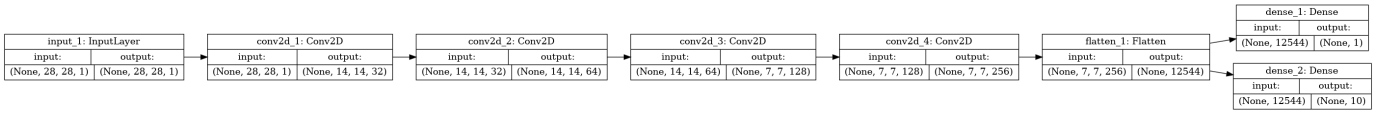
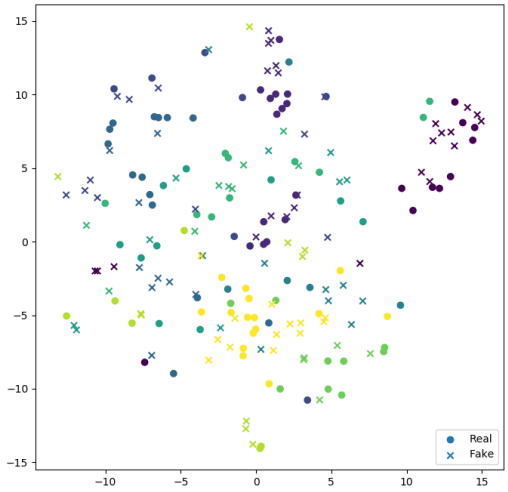
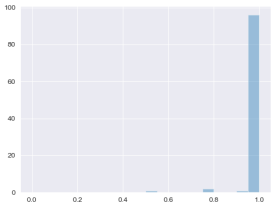


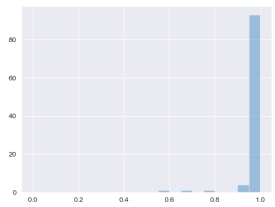
Fig. 10. Architecture of ACGAN discriminator



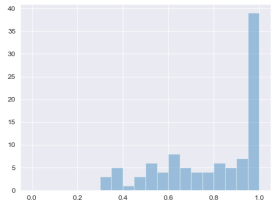
(a) Penultimate layer's embedding



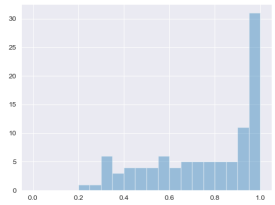
(b) Confidence on real images



(c) Confidence on fake images



(d) Confidence on real images



(e) Confidence on fake images

Fig. 11. Embedding and confidence of classifier on 100 real and fake images: (a) Embedding (b)-(c) Confidence on a good classifier, (d)-(e) Confidence on a mediocre classifier

image realistic (close), yet still fake (the fake images often located at the margin of each cluster).

The confidence score is shown in Fig. 11b-11c. Due to high performance, the model is confident about both real and fake images, and the distribution for fake images spreads out more. The result in a mediocre classifier (Fig. 11d-11e) could illustrate this better.

C. Classification loss - ACGAN [4]

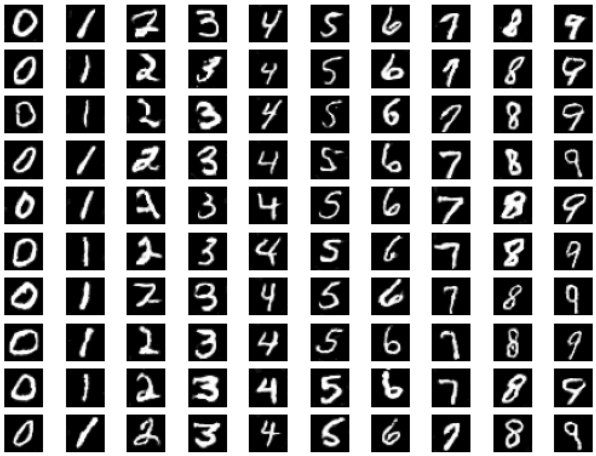


Fig. 12. Some generated images by ACGAN (left to right: 0 to 9)

We extend our CGAN with classification loss to make AC-GAN model. Rather than receive the label as input, ACGAN discriminator predict the class label. The new discriminator architecture is shown in Fig. 10. The discriminator receives image input and output two layers, one layer for fake/real prediction and one layer for class prediction. We compile discriminator with two different losses for these two layers (fake/real binary loss and multi-label classification loss).

Some the images generated by ACGAN is shown in Fig. 12. These images look more realistic than images generated by the CGAN model (CGAN images are shown in Fig. 17 of Appendix). To be rigorous, We also report the success rate and inception score of our ACGAN as 99.57% and 9.912, respectively. Compare to the statistic of CGAN (97.6% and 9.620), we conclude that our AC-GAN model is considerably more performant than CGAN. We attribute this improvement to the auxiliary classifier, which help the discriminator predict both the "realness" and class label of images. The classification loss would help the generated images maintain the class label, and in turn, improving the performance.

ACKNOWLEDGMENT

This coursework was done based on knowledge from module *'Special Topic in Computer Science: Machine Learning for Computer Vision'* at KAIST. Some codes are adapted from machinelearningmastery.com.

REFERENCES

[1] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Networks," 2014, accessed: 12-19-2020. [Online]. Available: https://arxiv.org/pdf/1406.2661.pdf

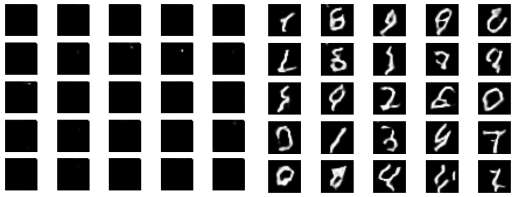
[2] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training GANs," 2016. [Online]. Available: https://arxiv.org/pdf/1606.03498.pdf

[3] Q. Xu, G. Huang, Y. Yuan, C. Guo, F. Wu, and K. Q. Weinberger, "An empirical study on evaluation metrics of generative adversarial networks," 2018, accessed: 12-19-2020. [Online]. Available: https://arxiv.org/pdf/1806.07755.pdf

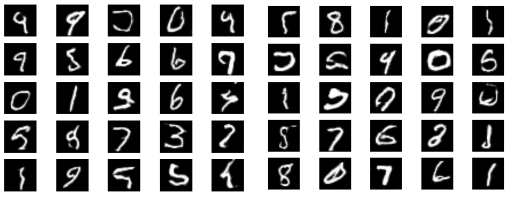
[4] A. Odena, C. Olah, and J. Shlens, "Conditional Image Synthesis with Auxiliary Classifier GANs," 2016, accessed: 12-19-2020. [Online]. Available: https://arxiv.org/pdf/1610.09585.pdf

APPENDIX

A. Images in different epochs

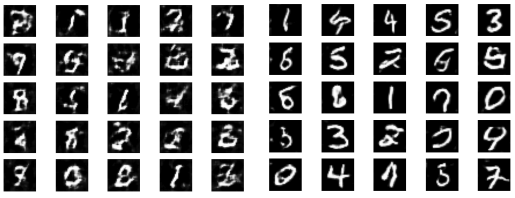


(a) Epochs 1000 (b) Epochs 6000

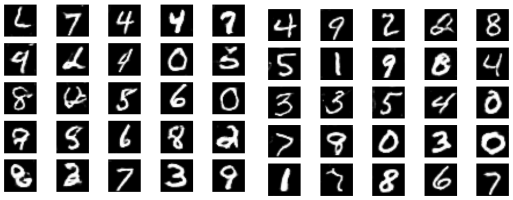


(c) Epochs 19000 (d) Epochs 35000

Fig. 13. DGAN (convergence failure)

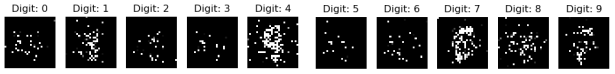


(a) Epochs 1000 (b) Epochs 8000



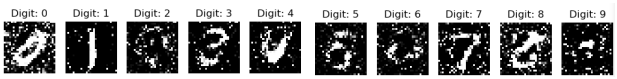
(c) Epochs 20500 (d) Epochs 28500

Fig. 14. DGAN (success)



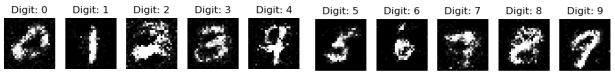
(a) Epochs 3000

(b) Epochs 13200

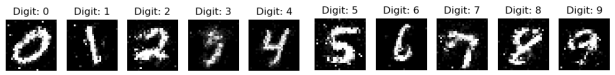


(c) Epochs 25600

Fig. 15. CGAN (failure to converge)

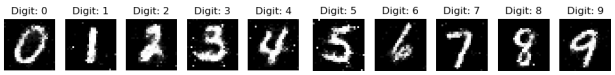


(a) Epochs 3000



(b) Epochs 13200

(c) Epochs 25600



(d) Epochs 53200

Fig. 16. CGAN (success)

B. CGAN generated images

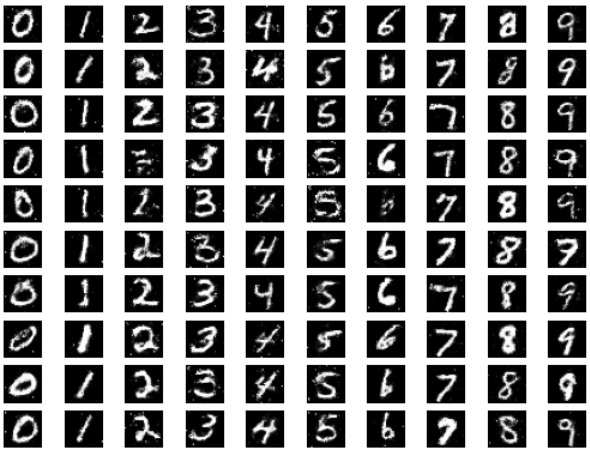
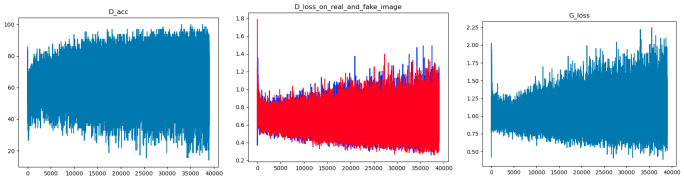


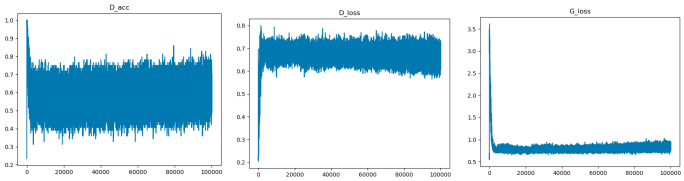
Fig. 17. Some generated images by CGAN (left to right: 0 to 9)

C. Label smoothing training history



(a) Accuracy (b) Discriminator loss (c) Generator loss

Fig. 18. DCGAN



(a) Accuracy (b) Discriminator loss (c) Generator loss

Fig. 19. CGAN

D. Results of t-SNE on 1000 images

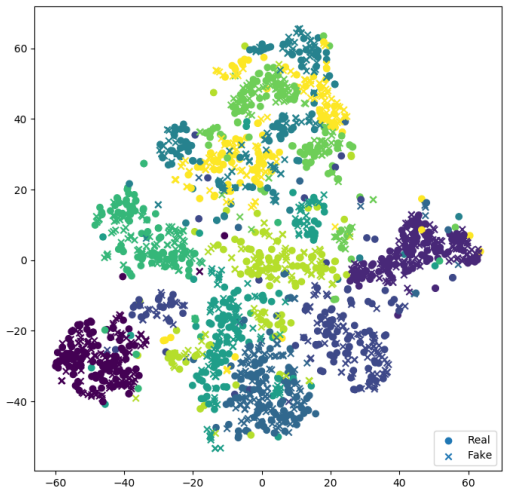


Fig. 20. Embedding of classifier penultimate layer on 1000 real and fake images