AIML 431 ASSIGNMENT FOUR REPORT

RuoHao Sun

VUW

1. Introduction (5 points). 1-2 paragraph to briefly introduce what has been done in the implementation and what you have achieved.

In this implementation, we have developed a Deep Convolutional Generative Adversarial Network (DCGAN) and incorporated features of Conditional GAN, enabling the generation of specific images under designated conditions. This means the produced images are not purely random but are generated based on provided labels or descriptions. By constructing and fine-tuning both the generator and discriminator neural networks, this model aims to produce images that are indistinguishable from real samples, showcasing the prowess of GANs in image synthesis. The introduction of the Conditional GAN adds a layer of flexibility and broadens the application spectrum. Through our iterative training process, the generator has progressively enhanced its capability to craft realistic images, successfully highlighting the potential of DCGANs and Conditional GANs in applications ranging from image generation to advanced uses such as data augmentation and anomaly detection.

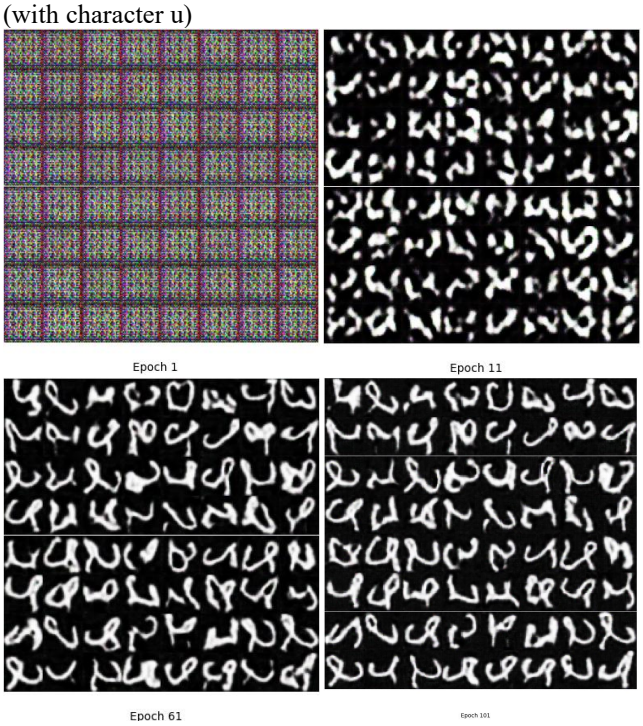
2. Implementation (5 points). You could discuss the issues you met in your implementation, and justifications on some steps or approaches you use to solve your problems. It doesn't need to be very long

The first issue I met in implementation is how to encode labels and concatenate with noise into the network to make sure the network can learn the label information. So I convert label 0 or 1 into one-hot encoding and then extend the label's length to match that of the noise, the generator deeply intertwines the label information within the generation process. This allows for the creation of images that are distinctly tailored based on the input label.

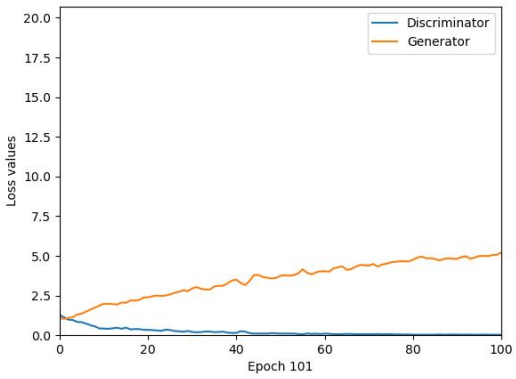
The second issue I met is my conditional DCGAN cannot learn map different noise into different image whereas my normal DCGAN for single class of input image is working well. In the other words, my generator cannot generate different images in different noises with same weights after adding labels together into the network. I personally think it might because the generator is not complicated enough on processing input data, I try to add the complexity by using different way of combination of noise and labels but I didn't figure it out at the end.

Experiment discussion (10 points). Please show your final generation results of your DCGAN model. At least 50 images are required. Then provide your insights about the training process and the relationship between the training schemes and the result quality. Some example questions (but not limited to these questions): if you choose different character's folders to train the network, which one can achieve a satisfactory generation earlier? How do the losses for generator and discriminator change during training? Does the generated image quality always get improved during training? Does data augmentation lead to a better model? Does a network with different layers and convolutional kernels provide different result quality?

This is the normal single class DCGAN output:
(with character u)



As we can see from above result, the generated images are getting better and better with the increasing number of epoch.



From my results of change of loss for discriminator and generator, I found it does not show what I expected since the generator loss is increasing and discriminator is getting to a very low value. But I think loss divergence is not fixed in the training of GAN, my results show that my Discriminator is too strong and my generator loss can not get decreased with such a strong discriminator, anyway, the image results do get improved in training.

Does data augmentation lead to a better model?

In this project, data augmentation does lead to a better model but it is not always the case, since GAN training has instabilities and introducing data augmentation might further complicate the training process. However, in certain situations, especially when training data is limited, data augmentation might help in increasing the diversity of the generator and the robustness of the discriminator.
The actual effect can vary based on the chosen augmentation techniques, the dataset being used, and the specifics of the training configuration. For instance, for some datasets, rotations or crops might be meaningful augmentations, while for others, these transformations might not be as relevant.

In summary, data augmentation can be beneficial for DCGAN in certain contexts, but it isn't always advantageous and might require careful tuning to yield positive outcomes.

Does a network with different layers and convolutional kernels provide different result quality?

The architecture of a DCGAN, including the number of layers and the size of convolutional kernels, can significantly influence the quality of the generated results. Different layers and kernel sizes determine the receptive field, the level of detail, and the complexity of patterns the network can capture.

For example, a deeper network might capture more intricate and hierarchical features in the data, which could potentially lead to higher-quality generation. However, it also poses the risk of overfitting if the training data is limited. On the other hand, larger convolutional kernels can grasp more extensive spatial patterns, but they also introduce more parameters, potentially increasing the training time and risk of overfitting.

Adjusting the network's depth and convolutional kernel sizes necessitates a balance between model complexity, computational cost, and the nature of the data. Therefore, while varying these parameters can certainly lead to differences in result quality, optimal configurations often require empirical testing and are dataset-dependent.

Does the generated image quality always get improved during training?

No, the quality of generated images does not always improve consistently during training. As we can see from the results, the image quality has a significant improvement at the beginning but there is no huge different from epoch 61 to epoch 100. While the training process generally aims to enhance the generator's ability to produce realistic images, there can be instances of mode collapse, overfitting, or oscillations where the quality might degrade. Regular monitoring of the generated samples and validation metrics, along with appropriate regularization and architectural choices, is crucial to ensure consistent improvement.

if you choose different character's folders to train the network, which one can achieve a satisfactory generation earlier?

If one character's folder contains images that are relatively similar and less complex, the network might achieve satisfactory generation earlier compared to a folder with diverse and complex images. Conversely, a character with more intricate features may require more training to generate satisfactory results, as the network has to capture more nuances.

Why does DCGAN use a tanh activation function in the output layer of the generator?

In neural networks, the choice of activation function is crucial for both the performance and stability of the model. For Generative Adversarial Networks (GAN) and its variant DCGAN, the aim of the generator is to produce data that's as close as possible to the real data distribution. In many image processing tasks, pixel values are typically normalized to the range [-1,1]. Using tanh as the output activation ensures that the output of the generator also lies within this range.

At the same time, one advantage of the tanh function over the sigmoid function is that its output is centered around 0.

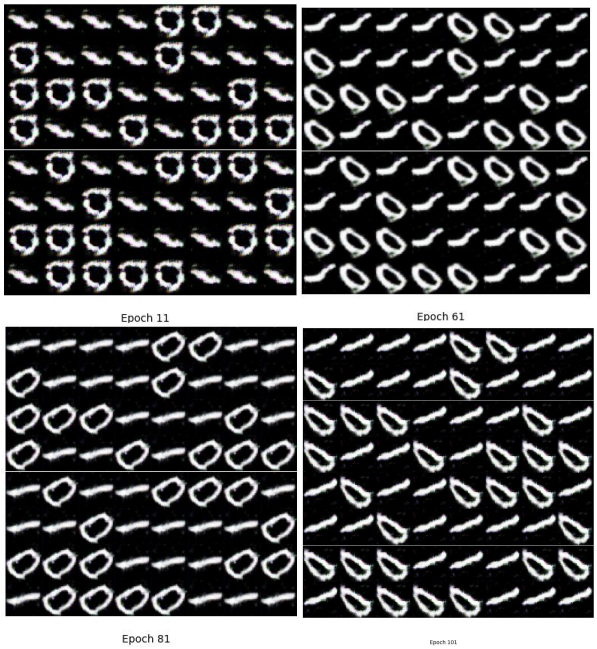
This means it has a balanced range of negative and positive output values, which can help the model to learn and adapt better during training. When generating image data, this balance can help maintain the brightness and contrast of the images. This is why in DCGAN, the output layer of the generator opts for tanh over other activation functions.

How to choose the appropriate dimension of the latent space for DCGAN?

The dimension of the latent space is a critical hyperparameter in GANs, determining the size of the input to the generator and the latent feature space. Choosing the appropriate dimension requires balancing the following considerations:

Representational Power: A larger latent space dimension can capture more variations and complexities in the data, offering more diversity in the generated samples.
Training Stability: An overly large dimension might lead to unstable training. A high-dimensional latent space might lead to mode collapse where the generator produces a very limited set of samples, neglecting other possible outputs.
Computational and Memory Overhead: Increasing the dimensionality adds to the computational complexity and memory required.
Risk of Overfitting: An overly large latent space might lead to overfitting, especially on limited datasets.

Results from conditional DCGAN:
(with characters o and 一)



As we can see from above results, there is a bug that model cannot map different noises into different images whereas labels working well. Besides, the training process is not stable, the final results is actually not as well as epoch 81, which is an common issue for GAN.

This is the loss divergence of conditional DCGAN