Report to Deep Learning Applications

Lecheng WANG, Guanyu CHEN
M2A, Sorbonne University

Contents

Below are the corresponding colab links:

- **2-a: Transfer Learning**
- **2-b: Visualizing Neural Networks**
- **2-c: Domain Adaptation**
- **2-de: GANS**

2-a: Transfer Learning

★Question 1

A	B	C	D	E	F	G
	Function	Input	Output	um paramete	porportion	
1	Conv2d	3	64	1,792	14,714,688	sum(convolution)
2	Conv2d	64	64	36,928	123,642,856	sum(linear)
3	MaxPool2d	-	-		0.119009609	conv/linear
4	Conv2d	64	128	73,856	138,357,544	total
5	Conv2d	128	128	147,584		
6	MaxPool2d	-	-			
7	Conv2d	128	256	295,168		
8	Conv2d	256	256	590,080		
9	Conv2d	256	256	590,080		
10	MaxPool2d	-	-			
11	Conv2d	256	512	1,180,160		
12	Conv2d	512	512	2,359,808		
13	Conv2d	512	512	2,359,808		
14	MaxPool2d	-	-			
15	Conv2d	512	512	2,359,808		
16	Conv2d	512	512	2,359,808		
17	Conv2d	512	512	2,359,808		
18	MaxPool2d	-	-			
19	Linear	25,088	4,096	102,764,544		
20	Linear	4,096	4,096	16,781,312		
21	Linear	4,096	1,000	4,097,000		

Figure 1: A table to count the parameters in each layer.

This table shows nearly 90% of parameters are produced by fully connected layers.

★Question 2

Output size of last layer of VGG16 is (1,1000), each element in the vector represents the probability (or score) of the input image belonging to one of the 1000 classes.

★Question 3

Role of the ImageNet normalization:

1. **Stabilize the training procedure:** Stabilize the training procedure. Normalization reduces the chance of vanishing or exploding gradients, thus, leads to a faster and more stable convergence.
2. **Improve generalization ability:** Improve generalization ability. Normalization helps to center the data and scale it, reducing the effects of, for example, varying light conditions. This allows the model to focus on more important features like shapes and textures thus, leads to a better performance.

Setting the model to evaluation mode ensures that it remains unchanged, preventing accidental modifications. Additionally, since gradient calculations are not required, computation becomes faster. In evaluation mode, networks with dropout layers will also behave differently compared to training mode.

Question 4

We can see the different output channels have focus on different features. We use (i,j) to refer the picture in i th row and j th column. Picture (4,2) and (4,4) extract features from large, distinctly varied color blocks. Such feature extraction helps the network differentiate between the background and the main subject of the image. Picture (4,3) can be regarded as the extraction of textures in the finer details. Picture (2,4) high lights the brighter areas within the image.

It is difficult for the human brain to fully understand the meaning of each channel, but there is no doubt that each of these channels extracts some type of feature of the image, and these features are useful for later operations.

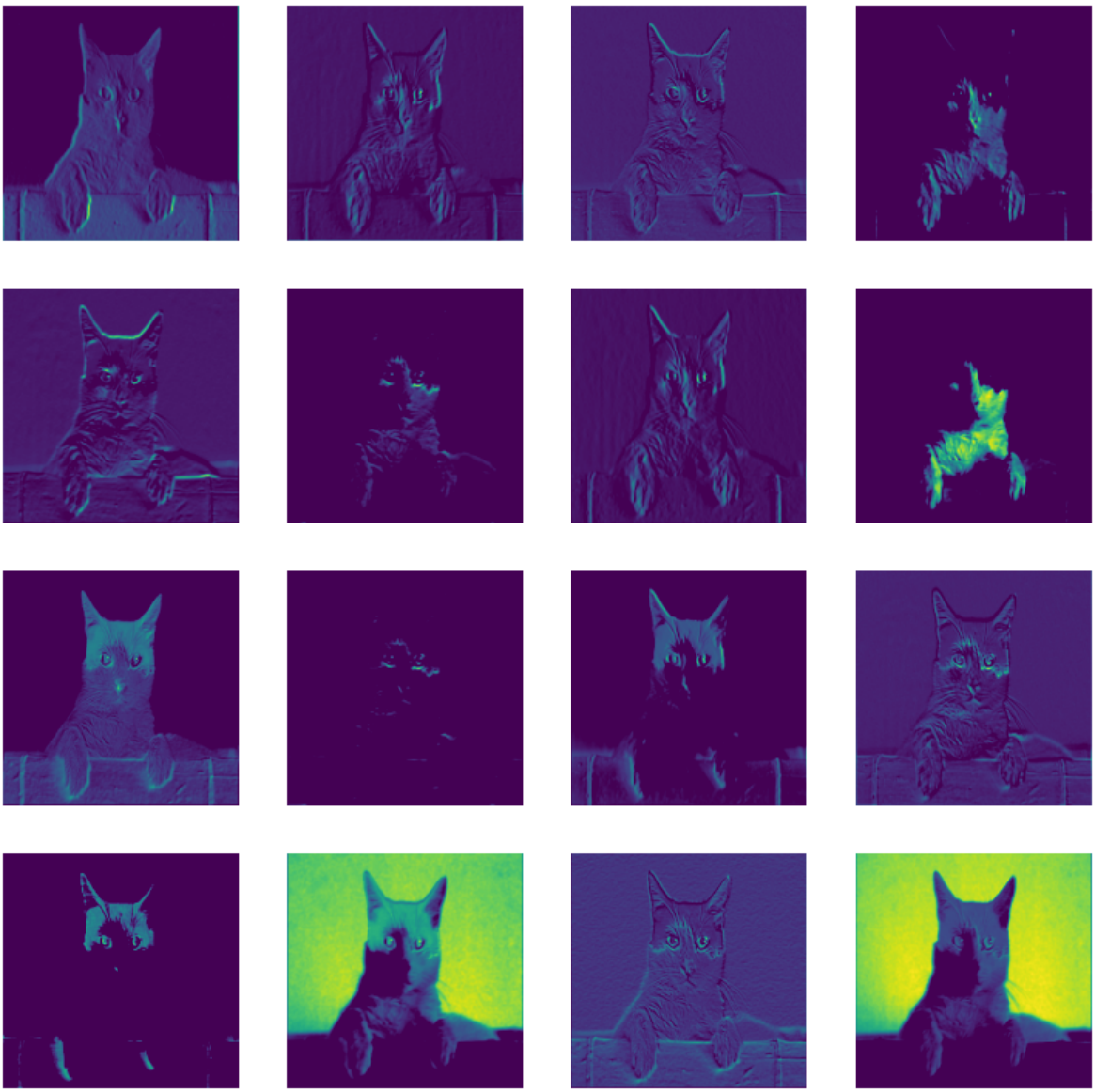


Figure 2: Visualize the output of first convolution.

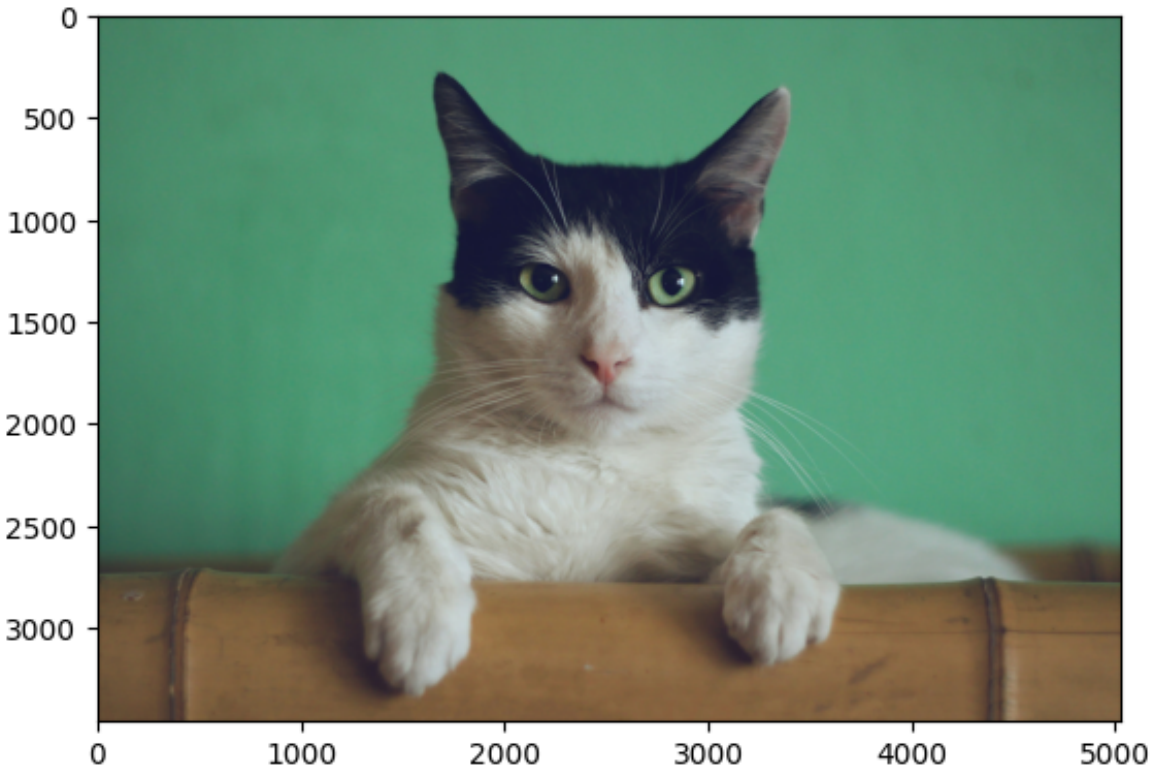


Figure 3: Origin input image of a cat.

★Question 5

Because VGG16 has too many parameters and 15 Scene is a relatively small dataset. If we directly train VGG16 on 15 Scene, it can cause significant overfitting.

★Question 6

Pre-training on ImageNet provides a strong baseline model that has already learned to recognize a wide range of features, such as edges, textures, shapes, and other complex patterns. This ability serves as a "universal" foundation for feature extraction, reducing training time and helping to prevent overfitting (as noted in question 5). This makes it a more effective approach for building a model that performs well on the 15 Scene dataset.

Question 7

First, the training dataset (ImageNet) may differ from the test dataset (15 Scene), so the features learned and the normalization parameters may not fully align. Second, since the task has changed, there might be a more suitable network architecture for extracting features specifically tailored for this classification task.

Question 8

In the early layers, the model primarily extracts basic graphical features like edges, textures, and light and shade. As more convolutional layers are added, the extracted features grow more complex, capturing shapes and partial structures of objects. In the final layers, the model can interpret complete information about the entire image, such as the object's class.

Question 9

Since the image is black and white, there is only one channel. We can make three copies of this channel after normalizing it and use it as the input to the RGB three channels

Question 10

A fully connected layer can certainly replace the SVM classifier. However, this approach has some drawbacks. For instance, it brings more parameters to be learned, and if the dataset is small, the large number of parameters can lead to overfitting.

Question 11

In the Jupyter notebook, we explored the impact of the following choices on the model's performance.

1. **Change the layer at which the features are extracted.** In the first experiment, we chose to copy to the $relu6$ layer, resulting in an accuracy of 0.908, which improved the model's performance. In the second experiment, we copied to the $relu5$ layer, achieving an accuracy of 0.894, which was comparable to the original performance at the $relu7$ layer. Based on these two experiments, we conclude that for the VGG16 model, copying to the $relu6$ layer yields the best performance.

2. **Try other available pre-trained networks.** In this experiment, we used ResNet50 as the pre-trained model and copied to the last $ReLU$ layer, achieving an accuracy of 0.917. The performance improved significantly. ResNet50's deeper network architecture enables the extraction of richer features, which enhances the performance of the support vector machine classifier.

3. **Tune the parameter C.** In this experiment, we attempted to adjust the parameter C in both the ResNet50 and VGG16 models, but it did not lead to significant improvement. The results were similar to those obtained when $C = 1$.

4. **Replace the last layer of VGG16 with a new fully-connected layer.** In this experiment, we replaced the last layer of the VGG16 model with a fully connected layer and trained it for 40 epochs. When tested with a cat photo, the model made an incorrect prediction. We speculate that using a fully connected layer for classification may not lead to performance improvement. While extensive training might eventually achieve decent performance, the computational cost would far exceed that of using a support vector machine.

5. **Look into methods for dimensionality reduction.** In the final experiment, we performed dimensionality reduction before classification, reducing the dimensions to 1200 and comparing it with the experiment without dimensionality reduction. The SVM training time after dimensionality reduction was 1.77 seconds with an accuracy of 0.910, while the training time without dimensionality reduction was 1.08 seconds with an accuracy of 0.908. Although the training time increased after dimensionality reduction, the accuracy improved, indicating that dimensionality reduction can enhance the model's performance.

2-b: Visualizing Neural Networks

★Question 1

From the following images, we can see that the primary object in the scene significantly influences the output of a trained image recognition neural network. In the second to fifth images, it is evident that the brightest part of the heat map corresponds to the location and contours of the animals. Similarly, in the first image depicting a haystack, the brightest regions correspond to the areas containing hay.

This demonstrates that the neural network effectively filters out background noise during computation, successfully identifying the main subject and producing accurate classifications.

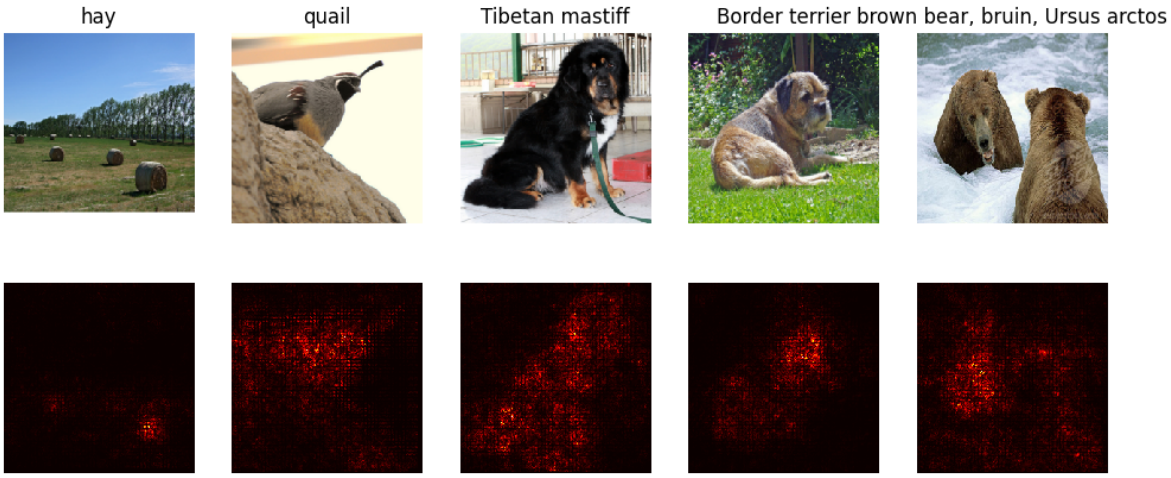


Figure 4: Saliency.

Question 2

1. **Gradient disappears:** In deep networks with highly non-linear activation functions, we may find the loss of gradient. This saturation may ignore some points that are also critical to the result.
2. **Class dependence:** Saliency maps only backward the gradient of a specific class. There may be other information that helps the network determine that the picture doesn't belong to another class, which is also important for the classification task, and is ignored.
3. **Lack of explanation:** Saliency maps only indicate which pixels contribute most to the model's prediction but do not provide the reason. It is purely mathematical and lacks explanation.

Question 3

It can be used for different purpose.

1. **Adversarial attack detection:** In the following part, we are already using it. This method can identify areas which is most sensitive to changes, enabling us only slightly modify the image to fool the classifier.
2. **Model debugging:**If the generated feature images are barely visible in relation to the object itself, this can indicate that our model for classification may not be effective.
3. **Class visualization:** As we will perform in Section 3, it can help us create the feature of a certain class.

Question 4

In Figure 5, we present the saliency map generated using the pre-trained VGG16 model. It can be observed that the saliency map from VGG16 has sharper contours and captures finer image details, although it performs poorly on larger images. This indicates that the VGG16 model has a stronger capability to handle finer details.

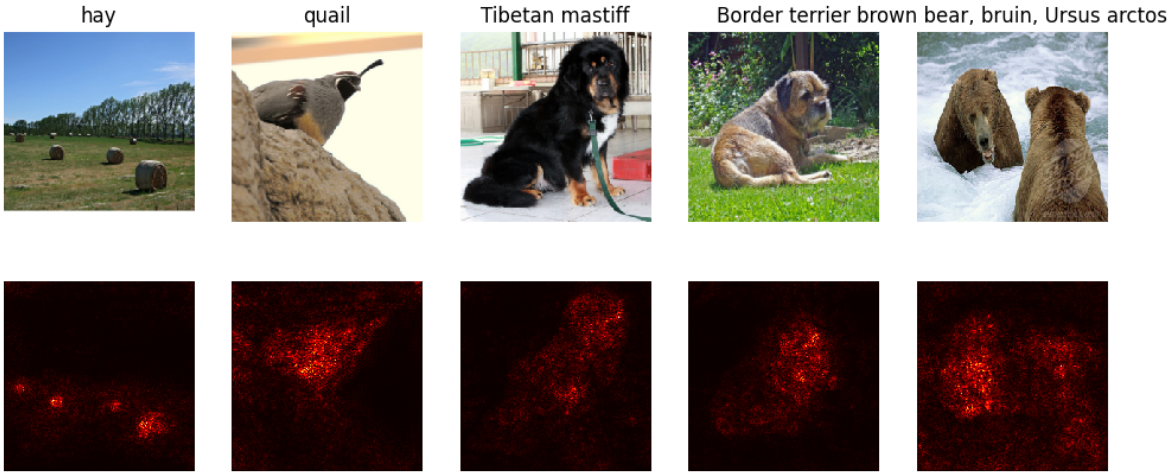
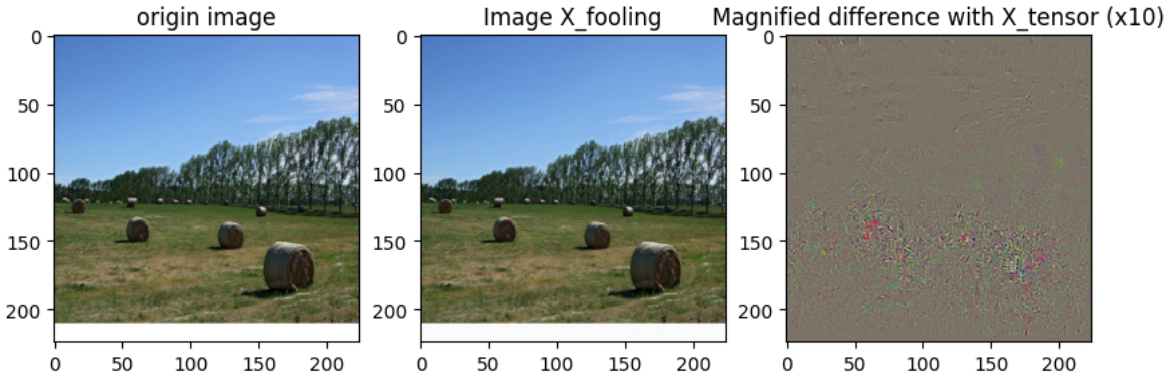


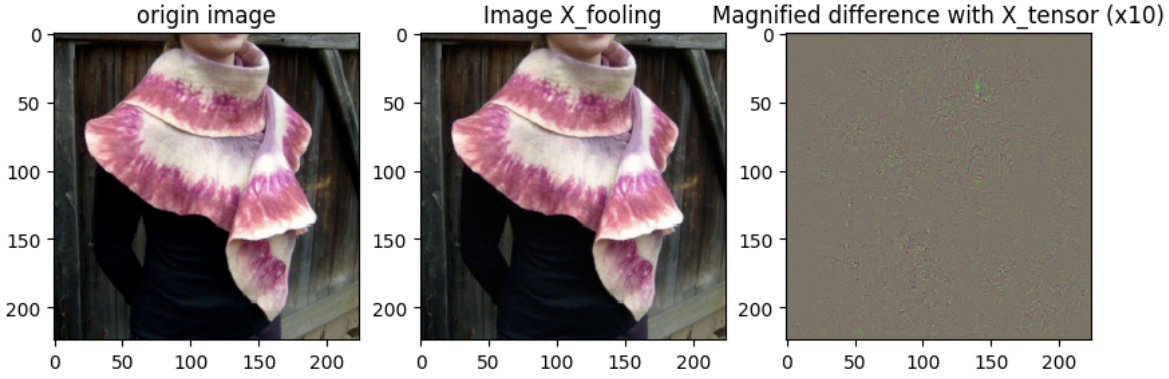
Figure 5: Saliency (VGG16).

★Question 5

The results are remarkable. Under conditions where the changes are nearly imperceptible to the naked eye, we successfully deceived the neural network into making an incorrect classification. This indicates that alterations to key pixels have a significant impact on the classification outcome. Furthermore, the original image label and the altered target label have a weak correlation, rather than being a simple confusion between similar categories, such as Labrador and Border Collie (both are dogs). This suggests that this approach is generalizable to such networks and has a low dependence on specific categories.



(a) From hay to stingray



(b) From stole to stingray

Figure 6: Comparison of the original picture and the fooling image.

Question 6

1. **Copyright Protection:** This method can be employed as a means to protect copyright. There are artists who object to their creations being utilized as training material for AI systems, particularly without their consent. By employing this technique, they can modify their works prior to publication, thereby complicating the process for certain network to categorize them effectively and remain almost imperceptible to the human eye.

2. **Robustness Assurance and further study:**In certain domains, errors in recognition can lead to significant safety issues, such as in the field of autonomous driving. Therefore, it is imperative to investigate how to prevent situations where a small amount of noise causes the overall failure of image recognition. Such research also aids in a deeper understanding of the principles behind how neural networks recognize images.

3. **Class visualization:** As we will perform in Section 3, it can help us create the feature of a certain class.

Question 7

1. **Dependence on Full Model Access:** Dependence on Full Model Access: The method requires the computation of gradients, which necessitates access to the model's parameters. In practical scenarios, such as attacking black-box models, obtaining such access is often not feasible.

2. **Limited Generalization Ability:** The dependency on specific models means that adversarial examples generated may not perform effectively across other models.

3. **Computational Cost:** The requirement to compute gradients can lead to time consuming computation, for example, high-dimensional inputs or complex models.

Alternative ways:

- Fast Gradient Sign Method (FGSM), which is computationally efficient.
- Carlini & Wagner (C&W) Attack, which generates more imperceptible and effective examples.
- For black-box model, we can try to fine similar known model to generate fool examples. If we know noting about the model, we can try *Query-based Attacks* , e.g.:Zeroth-Order Optimization (ZOO),NES Attack...

★Question 8

The following figure 7 (figure 8) visualizes class tarantula (snail) from a irrelevant white noise (a picture of hay). In figure 7, without the disturb of original image, we can see more clearly that the network identifies this category by capturing features similar to spider legs or bodies. This visualization method indicates that what the network learns are not specific objects, but rather patterns of features, such as edges, textures, and shapes.

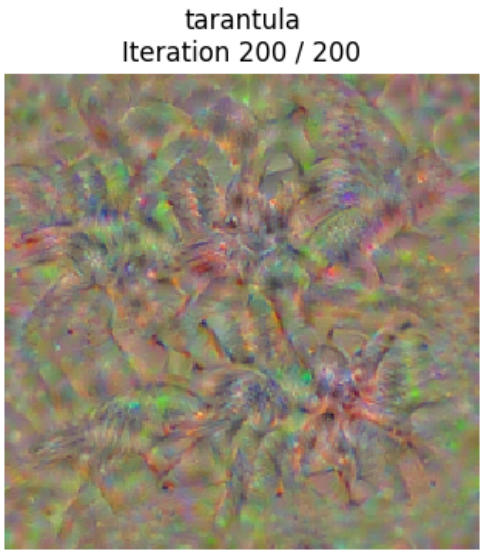


Figure 7: Noise to tarantula.

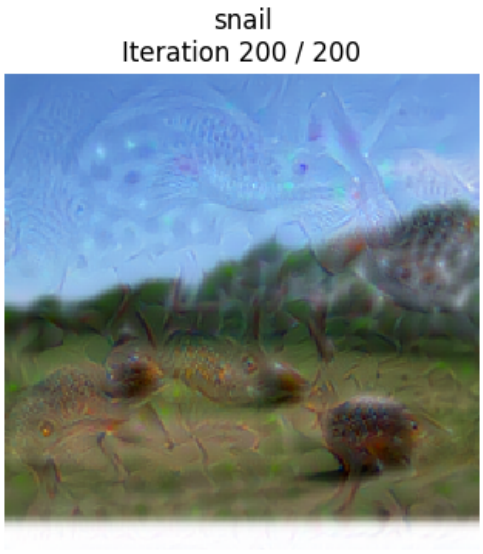


Figure 8: Hay to snail.

Question 9

In the Jupyter notebook, we modified the number of iterations, learning rate, and L_2 regularization weight based on the original parameters. The following images show the results of the runs.

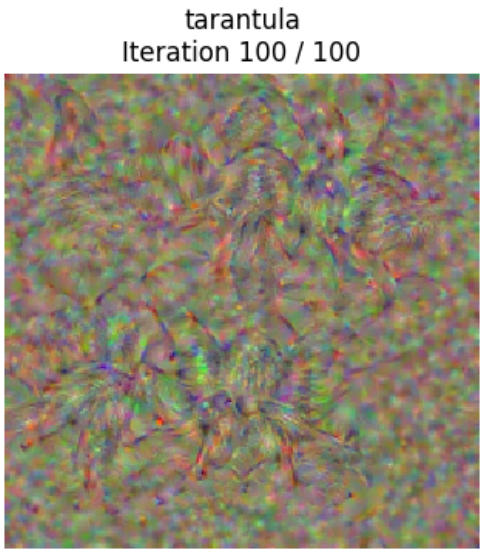


Figure 9: Noise to tarantula (num_iter = 100).

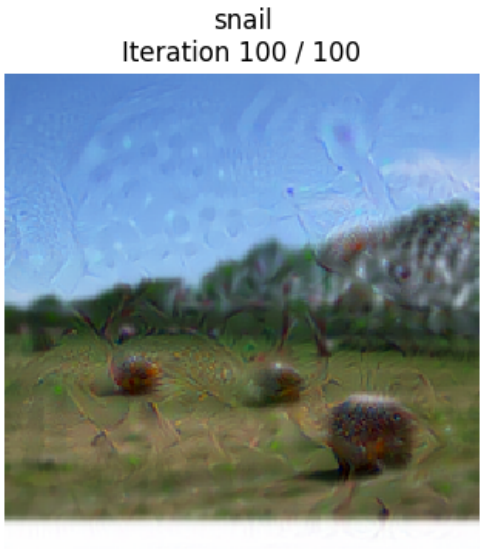


Figure 10: Hay to snail (num_iter = 100).

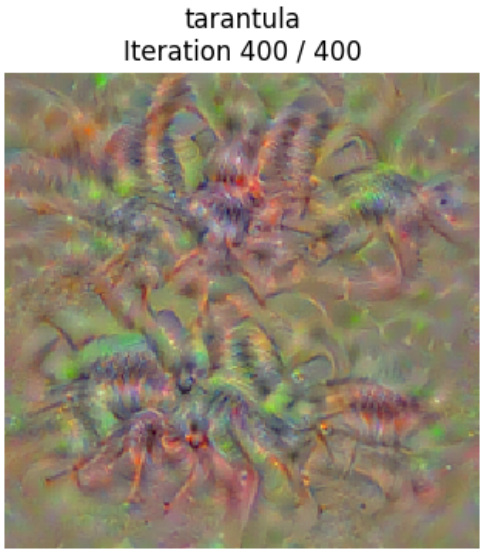


Figure 11: Noise to tarantula (num_iter = 400).

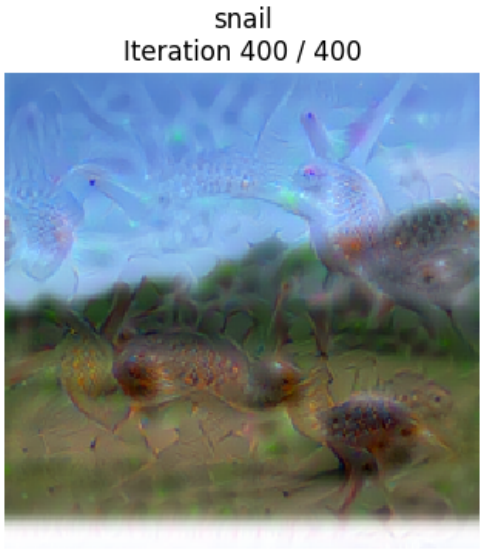


Figure 12: Hay to snail (num_iter = 400).

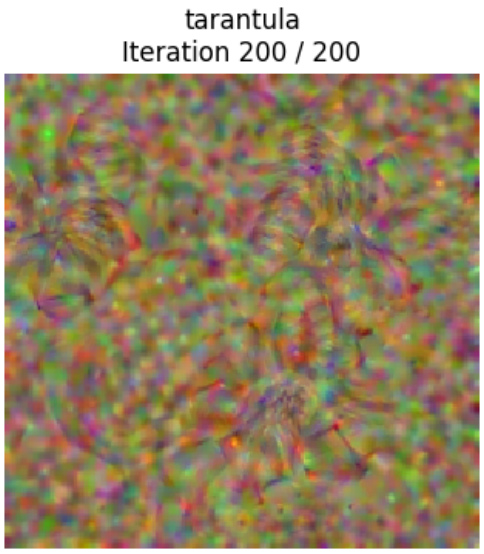


Figure 13: Noise to tarantula (lr = 1).

Figure 14: Hay to snail (lr = 1).

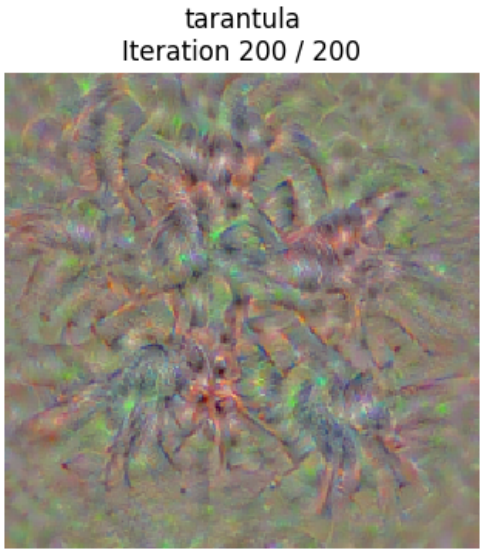


Figure 15: Noise to tarantula (lr = 10).

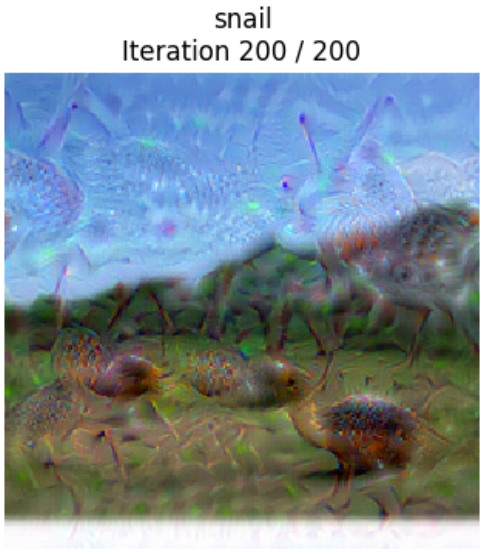


Figure 16: Hay to snail (lr = 10).

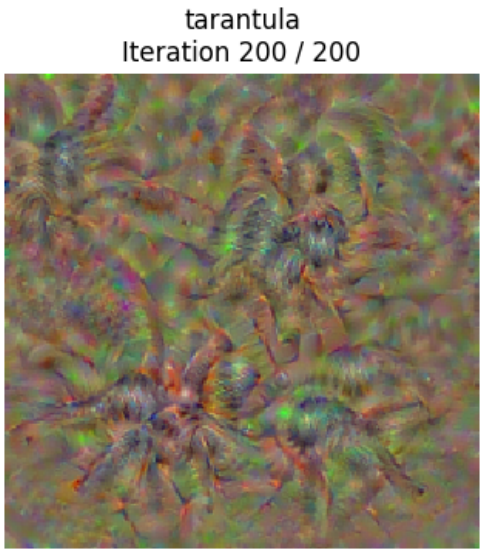


Figure 17: Noise to tarantula (L_2 reg = 0.1).

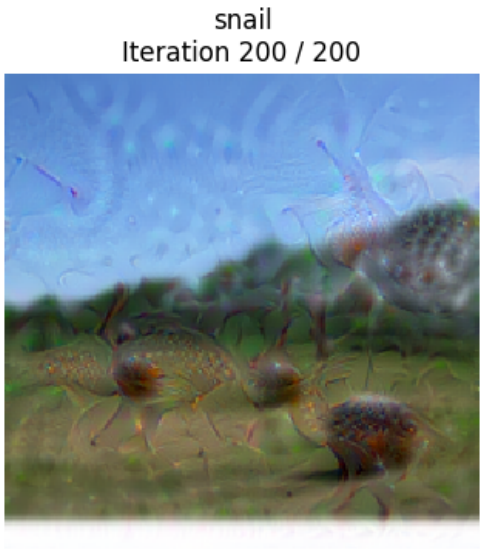


Figure 18: Hay to snail (L_2 reg = 0.1).

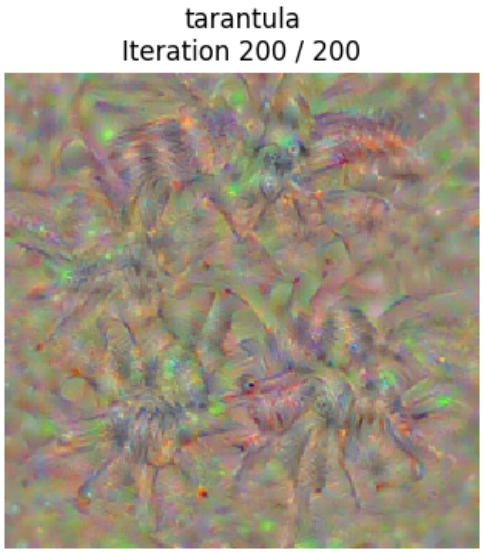


Figure 19: Noise to tarantula (L_2 reg = 0.01).

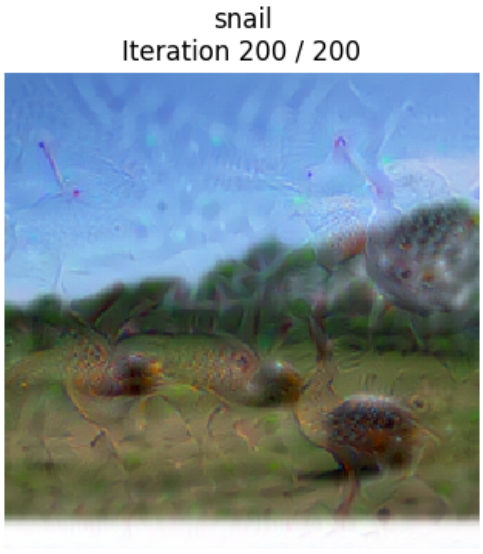


Figure 20: Hay to snail (L_2 reg = 0.01).

Question 10

The figure 8 visualizes class snail from a picture of hay. In Figure 8, we can see that the network added snail body features to the image of hay, causing it to be recognized as a snail. This visualization method demonstrates that by adding the features of one image to another, we can effectively deceive the neural network.

Question 11

The following images are the results generated using the VGG16 model. Compared to the original model, VGG16 exhibits significantly different characteristics. VGG16 demonstrates a stronger ability to capture features such as edges and textures, resulting in more pronounced local features.

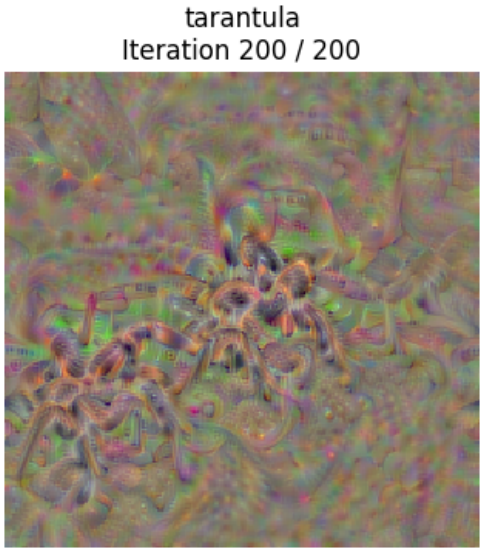


Figure 21: Noise to tarantula (VGG16).

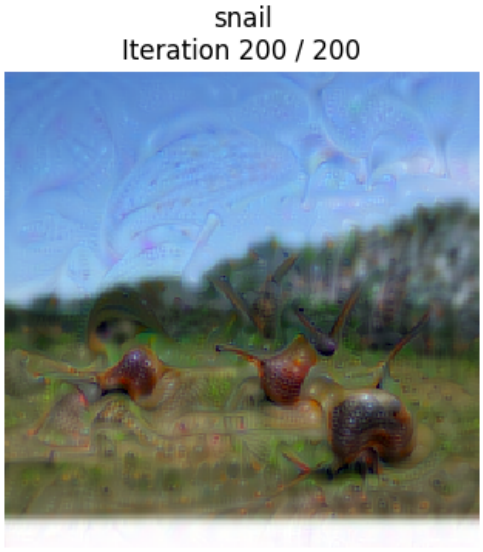


Figure 22: Hay to snail (VGG16).

2-c: Domain Adaptation

Question 1

If we remove the GRL, the feature extractor will try to tell the domain of the input instead of make it impossible to discriminate both domains. This will make CNN learn more domain feature rather than the feature that is more crucial to tell which number is presented. The classifier may perform better on original domain but worse on the target domain.

Question 2

During the training, the CNN adversely reduce the ability of domain classifier. This might sacrifice some features highly relevant to the source domain, thereby reduce the classification ability on the source data. After broaden the domain, the task become more complicated. With same number of parameters, it is nature that the performance reduced. This is a trade-off in optimization procedure.

Question 3

We call the negative number λ. The greater the absolute value of lambda is, the convolution network will learn domain-invariant features harder, make the domain classifier harder to learn origin domain and target domain. However, like we stated in previous question, this is a trade-off, focus more on learning domain-invariant features can cause the performance on original domain decrease. Meanwhile, a larger λ can be considered as a larger learning rate in optimizing procedure, which can cause oscillation of the network during the training. That is why we need to balance λ.

Question 4

Pseudo-labeling is a semi-supervised machine learning technique. This technique can use unlabeled data as training source. This technique can be simplified as:

1. Using labeled model to learn a basic model.
2. Using basic model to label unlabeled data (for the stability, only take high-confidence predictions, e.g., softmax output should be higher than a threshold).
3. Using original data plus high-confidence data as source to train the model.
4. Using new model to relabels all unlabeled data (including those we have labeled with previous model), take high-confidence predictions.
5. Repeat 3 to 4 with some adjustment of parameters (learning rate, threshold ...)

This method is quite useful when the original domain is similar to the target (unlabeled) domain, but if the domains are not similar, the original model trained on labeled domain can give many wrong labels, thus jeopardize the training procedure.

2-de: Generative Adversarial Networks

Question 1

The equation (6) produces loss for the generator while the equation (7) produces the loss for discriminator. If we only use one of them, the on of them will remain in a very trivial stage, which means the generator can easily fool a trivial discriminator or the discriminator can easily tell which picture are generated by a trivial generator. In the first case, the generator won't improve even the generated images have low quality, because most of the fake images pass the test, thus there is no "direction" for parameters to update. The second situation is similar, only use one of the loses will lead to a poor ability for both networks.

Question 2

Ideally, the generator G should transform the distribution $P(z)$ to P_{data} which is the distribution of data.

Question 3

The "true" equation should be: $\min_G E_{z\sim P(z)}\left[\log\left(1 - D(G(z))\right)\right]$.

However, in practice, from very beginning, the discriminator D can easily distinguish generated samples from real samples, $D(G(z))$ will be very close to 0. In this case, $1 - D(G(z)) \approx 1$, and the generator's loss approaches 0, leading to the vanishing gradient problem.

Question 4

1. **Progress of the generations:** From the very beginning, the images generated are basically noisy without visible information. After 400 iterations, we can roughly see some pattern similar to MINIST dataset,for some picture, We can somewhat understand the content of the image. After 2000 iterations, most of the images generated are easily to tell which number they represent, but still have trouble in fooling human eyes. Only a few of them can pass for the real ones, looking almost indistinguishable from the images in the original dataset.

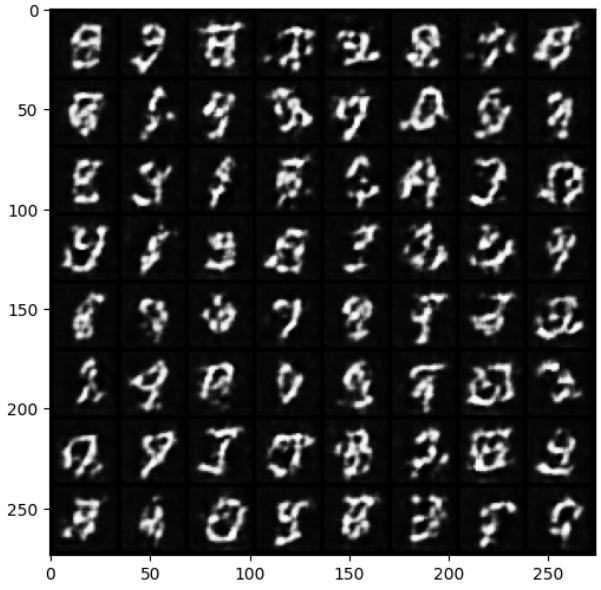


Figure 23: Default in notebook,400 iterations.

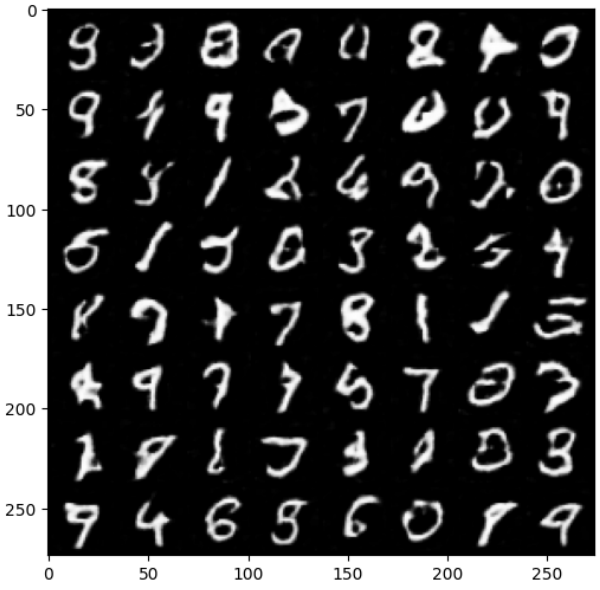


Figure 24: Default in notebook,2000 iterations.

2. **Loss:** We notice that both the generator loss and the discriminator loss are decreasing, but the fluctuations are relatively large, the generator has more significant fluctuations than the discriminator. The generator is relatively more difficult to train, resulting in larger fluctuations . Occasionally, the discriminator's loss shows relatively higher values, which may be due to the generator producing samples that are harder to distinguish or samples representing entirely new modes, making them more difficult to discriminate. The training process does not stabilize completely towards the end. We believe improvements can be made by adjusting the parameters, such as reducing the learning rate or decreasing the discriminator's update frequency, to help the generator learn more effectively (as the generator is usually harder to train, in the picture, the loss of generator present an increase tendency, which might indicate that the discriminator is too strong).

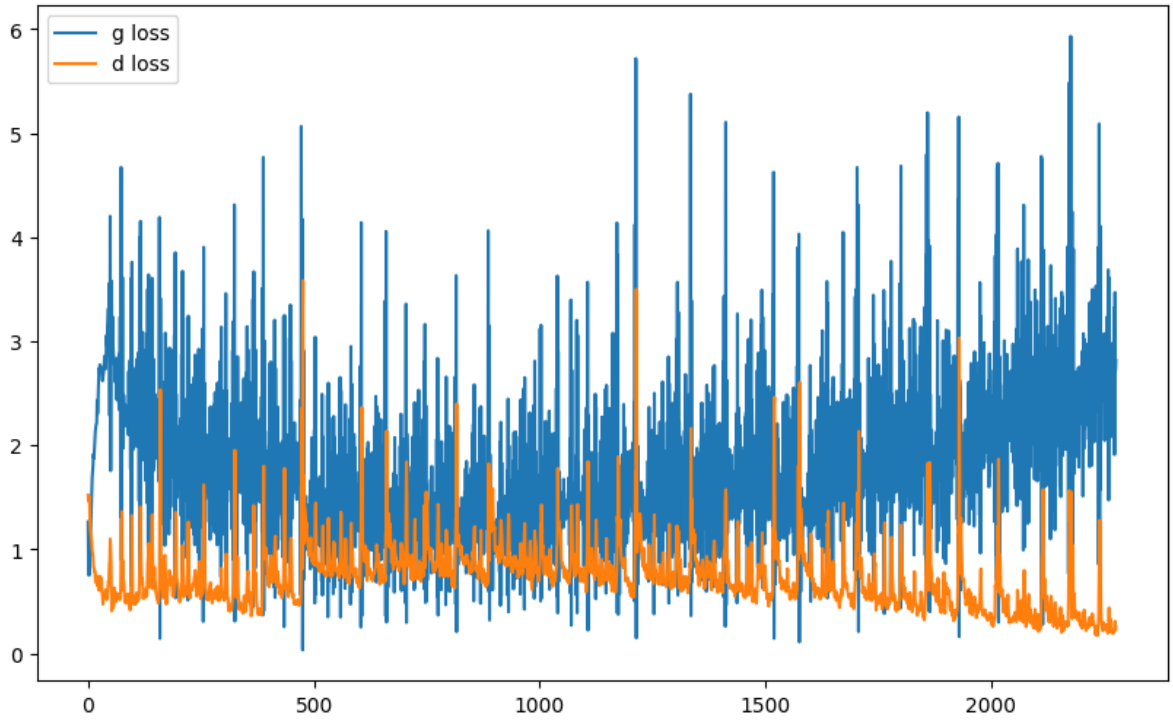


Figure 25: Default in notebook, loss value.

3. **Stability:** We observe the stability of the training process by the loss functions. The losses do not show signs of obvious divergence or significant anomalies (such as a large generator loss with a discriminator loss of zero, which would indicate that the generator is producing entirely poor samples). However, compared to the training of other models, the stability of these loss functions is relatively poor. We suspect that this is caused by the fluctuations inherent in the adversarial nature of GANs training.

Question 5

Due to the limited training resources in colab, without specific statement, we train each type of GANs with epoch=5.

1. **Change ndf and ngf:** We try ndf=128 (ngf hold) and ngf=128 (ndf hold). We found that more filters (within a reasonable range) tend to make the model perform better. In the first case, the ndf is too strong (because the hard part is training gernerator) and generator loss remain high. In the second case, the generator can fool the discriminator better and has a lower loss. We believe that if we replace the discriminator with a better one (pre-trained and with more ndf), we can let the generator has a better performance

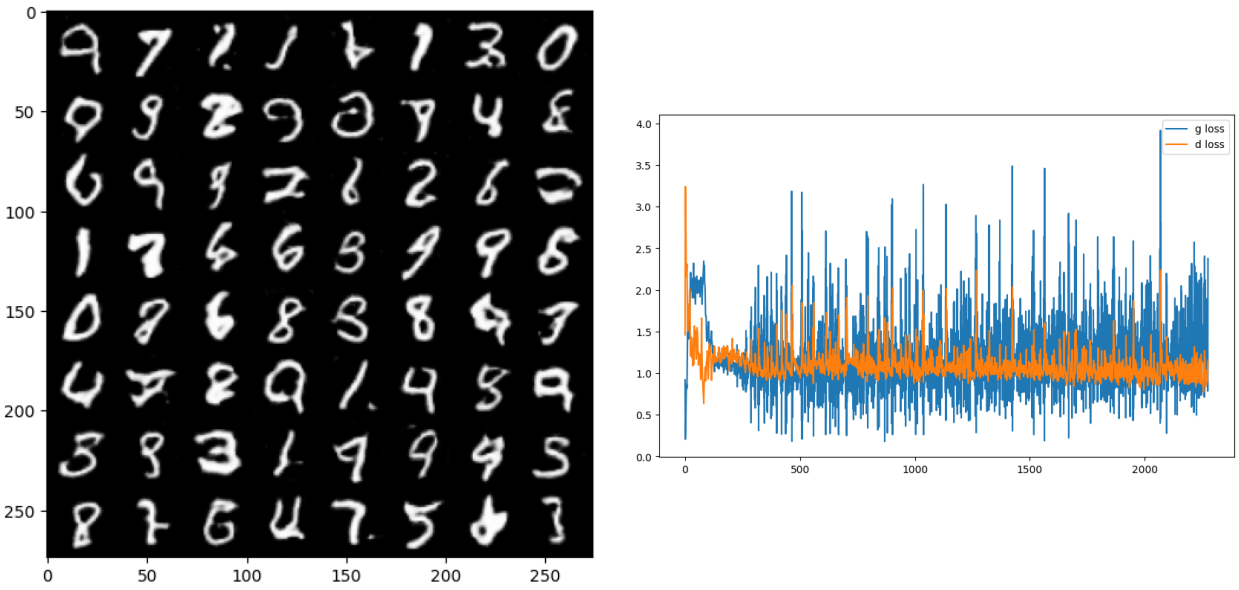


Figure 26: ngf=128,5 epochs: (Left) Generated images; (Right) Training loss.

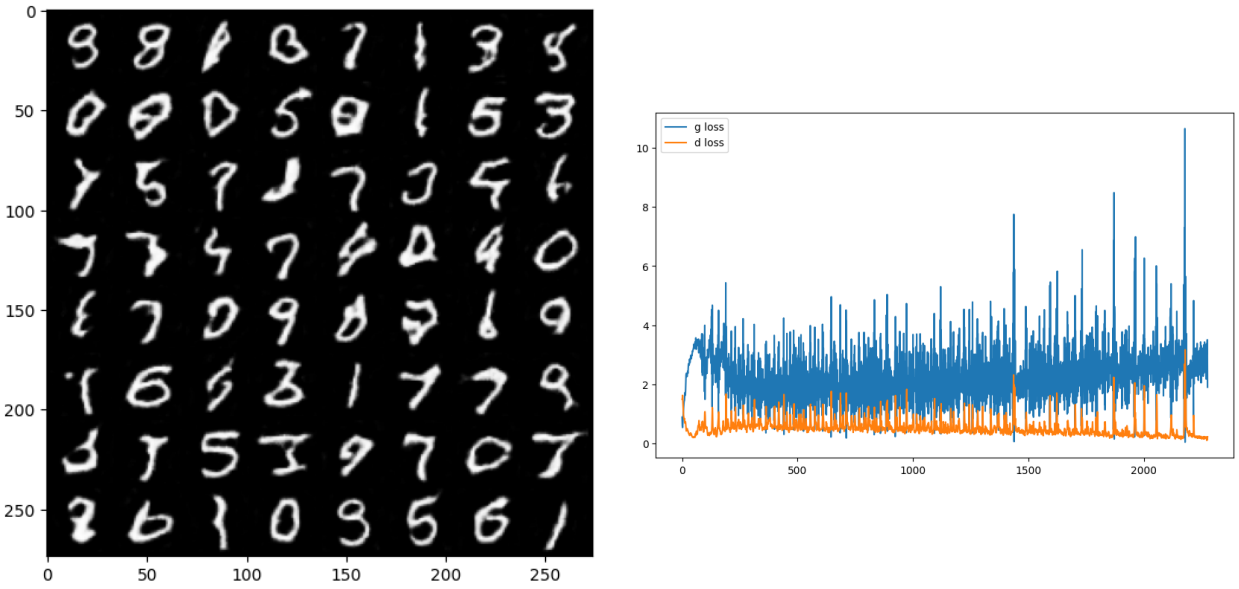


Figure 27: PyTorch's default initialization: (Left) Generated images; (Right) Training loss.

2. **Replace the custom weight initialization with PyTorch's default initialization:** The performance is slightly more fluctuant than the original model, but the difference is not significant.

3. **Replace with true loss (figure 28):** We use the loss presented in question 3, the images generated always seem to have some "white dots" which indicate the there is something wrong with the generator, and thoes dots won't disappear after more training. There is also a significant fluctuation appears in the end, both of them show the necessity of using an improved loss.

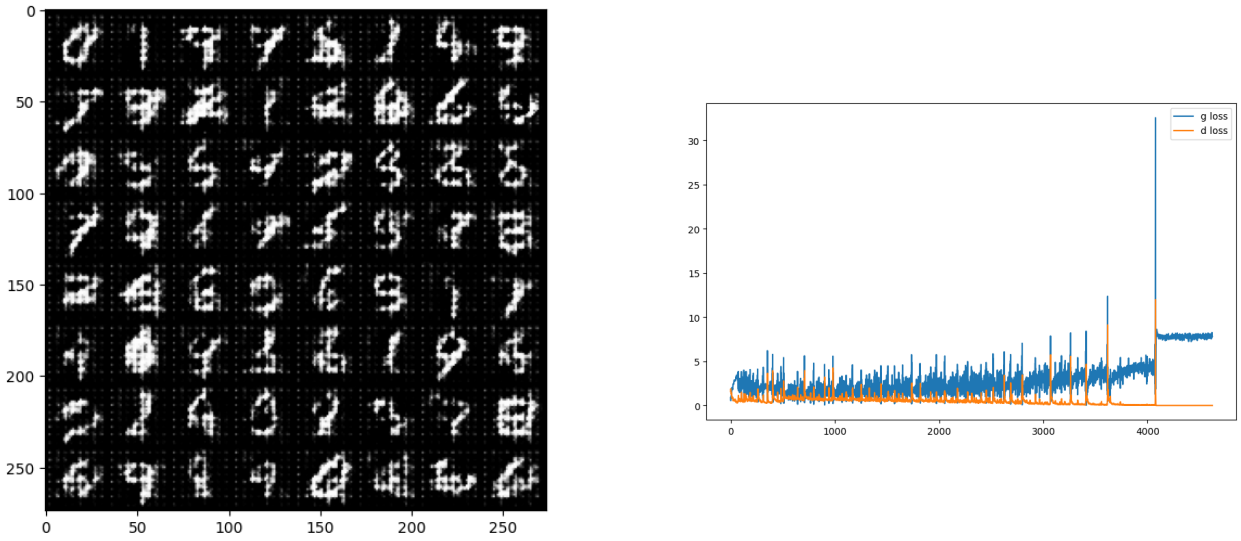


Figure 28: True loss: (Left) Generated images; (Right) Training loss.

4. **Change learning rate for both models (figure 29,30):** We multiply the learning rate by 10. The model quickly reached a relatively stable state (indicating that neither the generator nor the discriminator losses showed a significant increasing or decreasing trend). However, instability emerged during the subsequent training, as the generator's loss function began to increase.

   When we divide learning rate by 10 (figure 30), we have an interesting result: the generated model has worse results (not because the epoch isn't enough). It is suspected that the generator updates too slowly, making the discriminator significantly stronger than the generator. As a result, the images generated are too far from deceiving the discriminator, leaving the generator with insufficient information to update its gradients. Consequently, the generator remains in a suboptimal state.

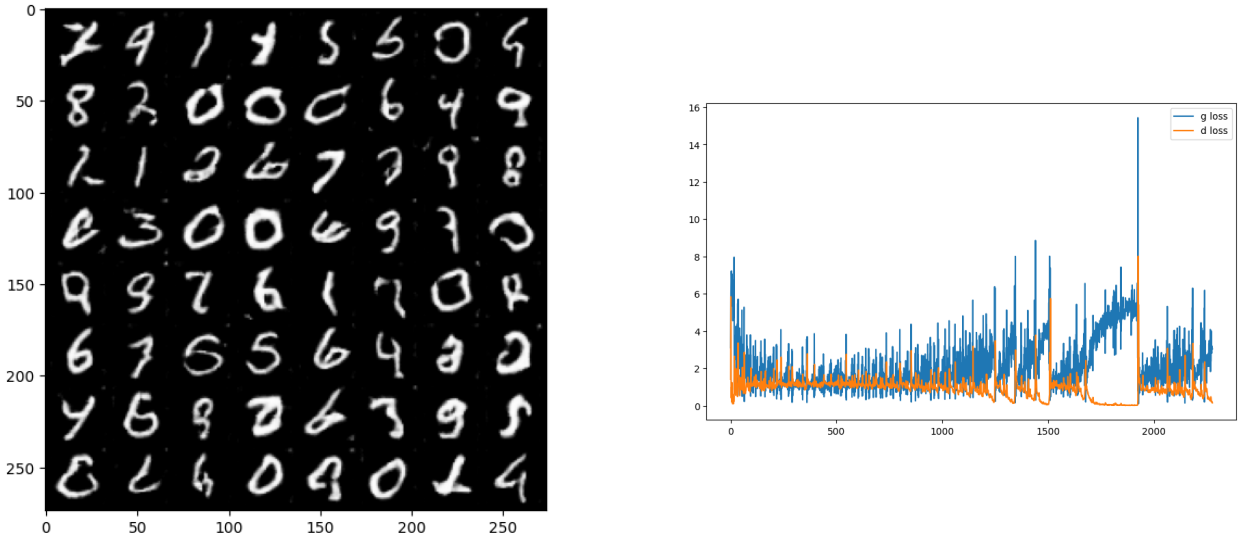


Figure 29: Learning rate $\times 10$: (Left) Generated images; (Right) Training loss.

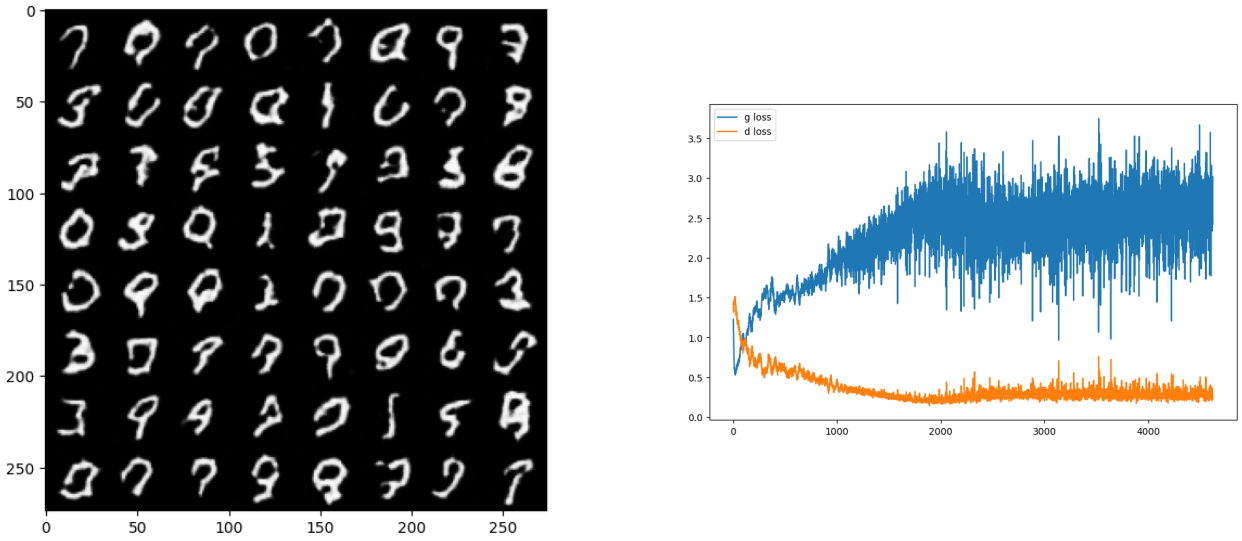


Figure 30: Learning rate ÷10: (Left) Generated images; (Right) Training loss.

5. **The original parameter in notebook with 30 epoch:** The plot of loss show a periodicity pattern especially after the abscissa value is greater than 6000. After the x-axis value exceeds 6000, the generator occasionally finds a technique with limited applicability to fool the discriminator. However, this technique is not highly generalizable. Once deceived, the discriminator can relatively easily address the failure through training. This adversarial process results in sudden increases and decreases in the discriminator's loss, but after each dramatic fluctuation, the loss quickly returns to normal. This indicates that the training has reached a bottleneck, 30 epochs is a wast of computing resources, we can also observe that the generated images don't have great improvement than the results after 5 epoch.

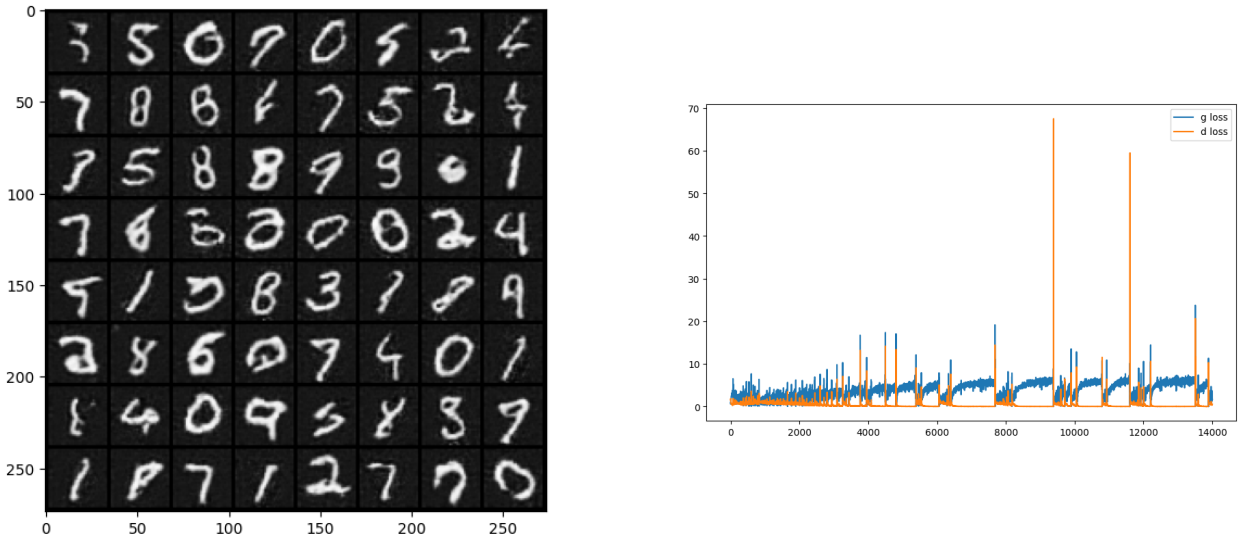


Figure 31: Default in notebook, 30 epochs: (Left) Generated images; (Right) Training loss.

6. **Change n_z(figure 32):** With different n_z, loss images are similar. When changing the value of n_z, apart from noticing differences in training time (smaller n_z leads to faster training), other differences are relatively less perceptible. Subjectively, I feel that a higher-dimensional n_z produces images with more complex textures and wrinkles (although this may not necessarily good for the MNIST dataset). Generally, the dimension of

n_z should not be too low, as it may lose part of the information related to the distribution, leading to mode collapse where only a few similar patterns appear. On the other hand, n_z should not be too high, as it increases computational cost while also making the model less stable.

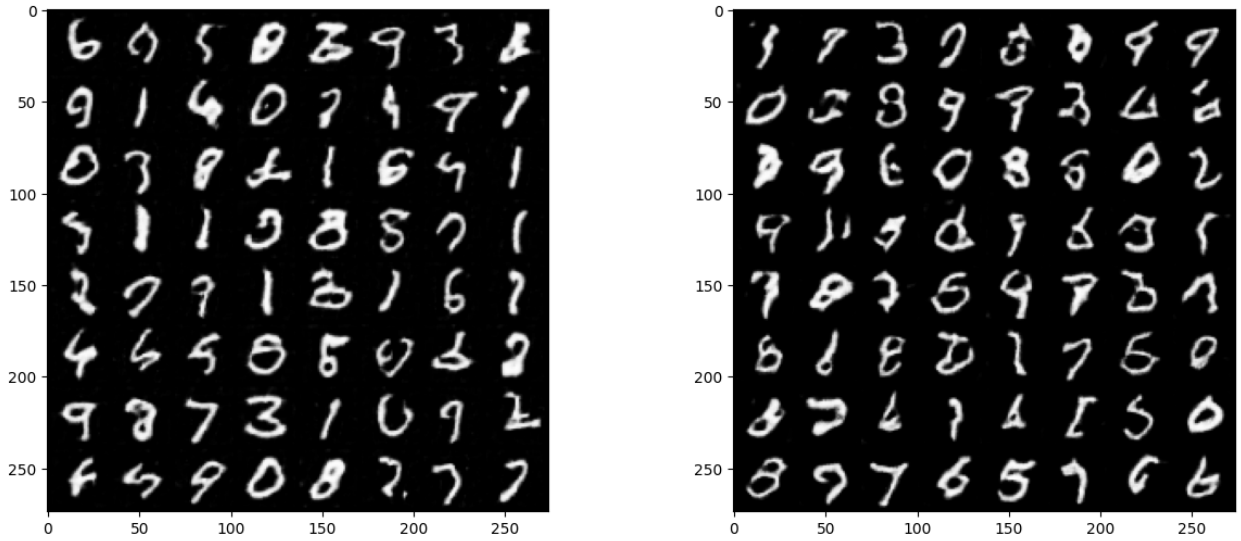


Figure 32: Different n_z, : (Left)$n_z = 10$; (Right) $n_z = 1000$.

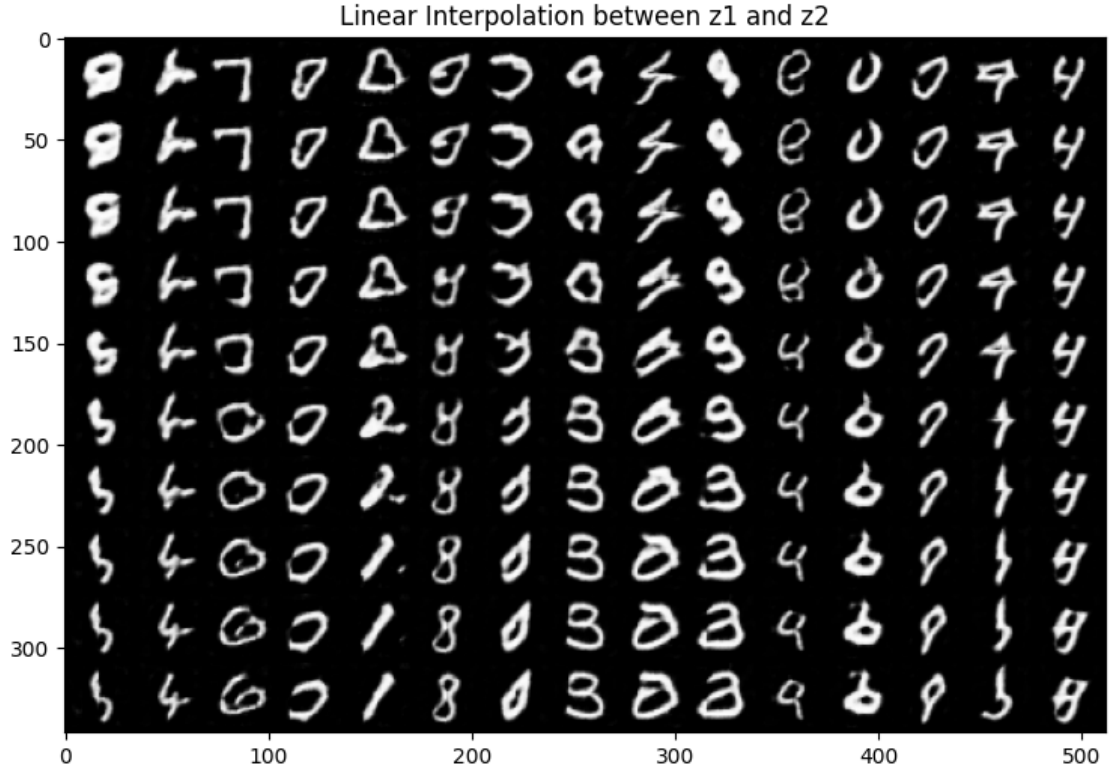


Figure 33: Images vary with input vector z.

7. **Take 2 noise vectors z_1 and z_2 and generate the images(figure 33):** For each column, we sample 2 fixedz_1, z_2, for each line 1 to line 10, we gradually increase α from 0 to 1 and count $z_{input} = \alpha z_1 + (1 - \alpha) z_2$. The input vector gradually transitions from z_1 to z_2. During this process, we can observe that the output images change in a "gradual" manner, transforming from the first to the last image in each column. This variation demonstrates that the output can be controlled by manipulating the input vector, and such changes tend to be continuous rather than abrupt. This showcases the fundamental principle of training a Conditional Generator and Discriminator.

8. **Generate 64*64 pictures (figure 34:** The generated results are worse than 32*32 case. During the training process, the generator took a long time to generate some images slightly useful (before x-axis value over 1200, the generator loss has a increase tendency).The poor performance may due to:

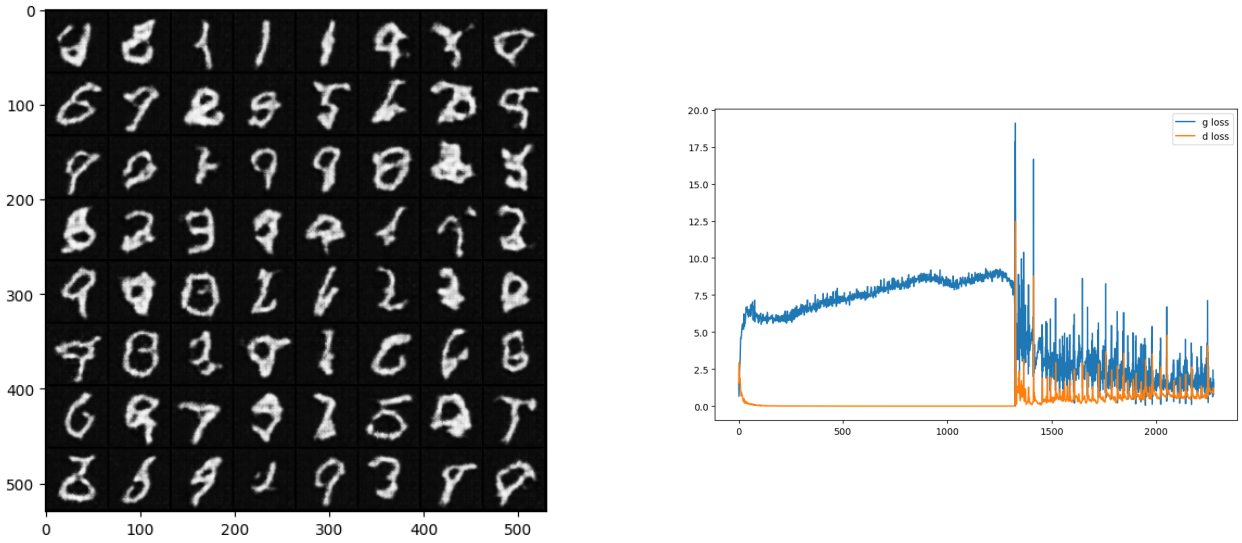


Figure 34: 64*64 Generated image and training loss.

- More parameters to learn but the training epoch isn't enough.
- The original MNIST images have limited details, and directly scaling them up to 64×64 amplifies the sparsity of pixels, which may result in generated images appearing blurry or unnatural.

9. **Try on CIFAR-10:** CIFAR is more complicated than MINIST, the generator trained 5 epochs by default has poor performance.These images barely allow one to perceive that there is an object in a certain position of the image but provide almost no additional information about the object. However, after 15 epochs of training, the edges of objects and their background become recognizable. Some images reveal waterbirds swimming on the water or birds flying in the sky, while others resemble frog-like animals. Nevertheless, the generator we obtained has limitations, as there is no significant improvement in image quality between 10 epochs and 15 epochs of training, nor is there a clear trend of change in the loss function. This suggests that adjustments to the parameters and network structure may be necessary to achieve better results.

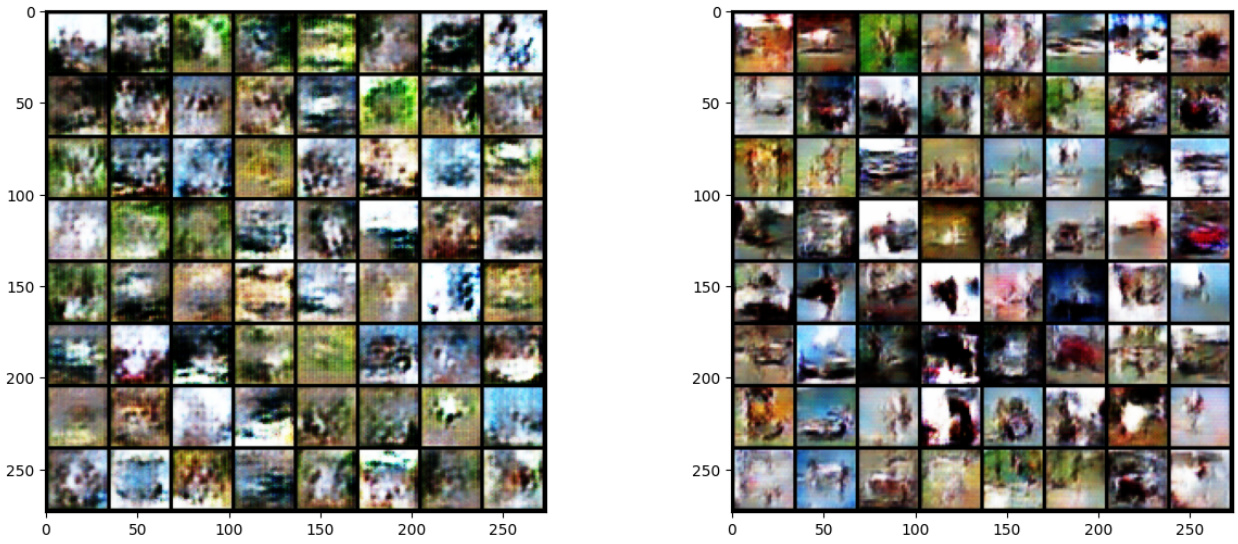


Figure 35: Generated images of CIFAR-10, (left) epoch=5, (right) epoch=10.

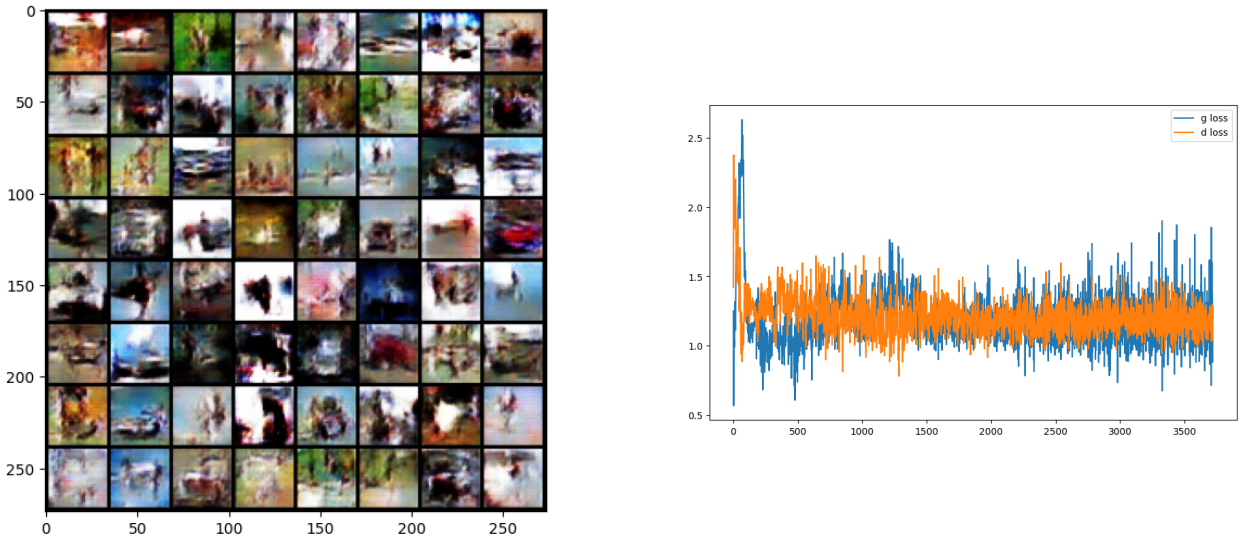


Figure 36: Generated images of CIFAR-10 (epoch=15) and training loss.

Question 6

Conditional DCGAN generates samples that meet specific conditions by combining the condition y with the input z. In experiments, this approach typically produces more diverse images that align with the condition y. In our experiment, it is way better than previous results.

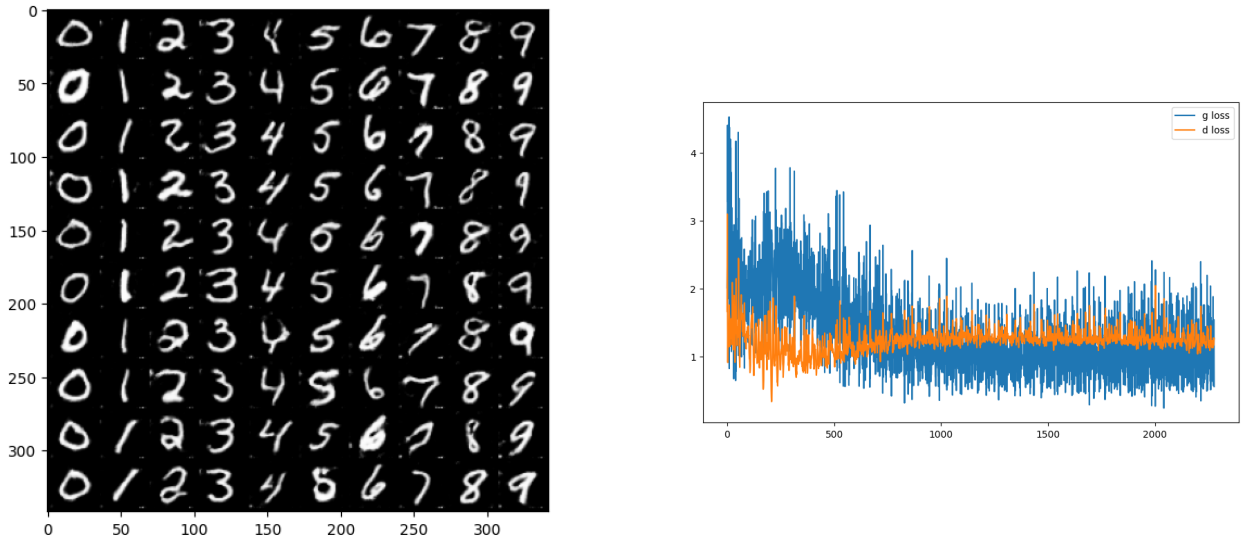


Figure 37: Generated images by DCGAN after 5 epochs,(right) training loss.

Question 7

We can't remove the vector y from the input of the discriminator. If the input of the discriminator is changed from $cD(x, y)$ to $cD(x)$, the discriminator has no information about supplementary condition y and will be unable to verify whether the generated sample x' matches the condition y,thereby violating the original intent of Conditional GAN design, which is generate images related to supplementary condition.

Question 8

Way more successful. We assume that this is because the condition y provides additional information, it can better guide the generator to produce target samples. In the original experiments, the generator had to learn the features of all different digits simultaneously, resulting in generated images that often combined features of various digits and appeared blurry overall. However, after providing labels, the similarity among samples within the same label category significantly increased, making it easier for the generator to learn.

Question 9

The noise vector here can represent certain latent features of the samples, such as texture or style For example, when the label is the same digit "4," some vectors might represent a "4" with a triangular-shaped enclosed space in the middle, while others might represent a "4" with longer endpoints.

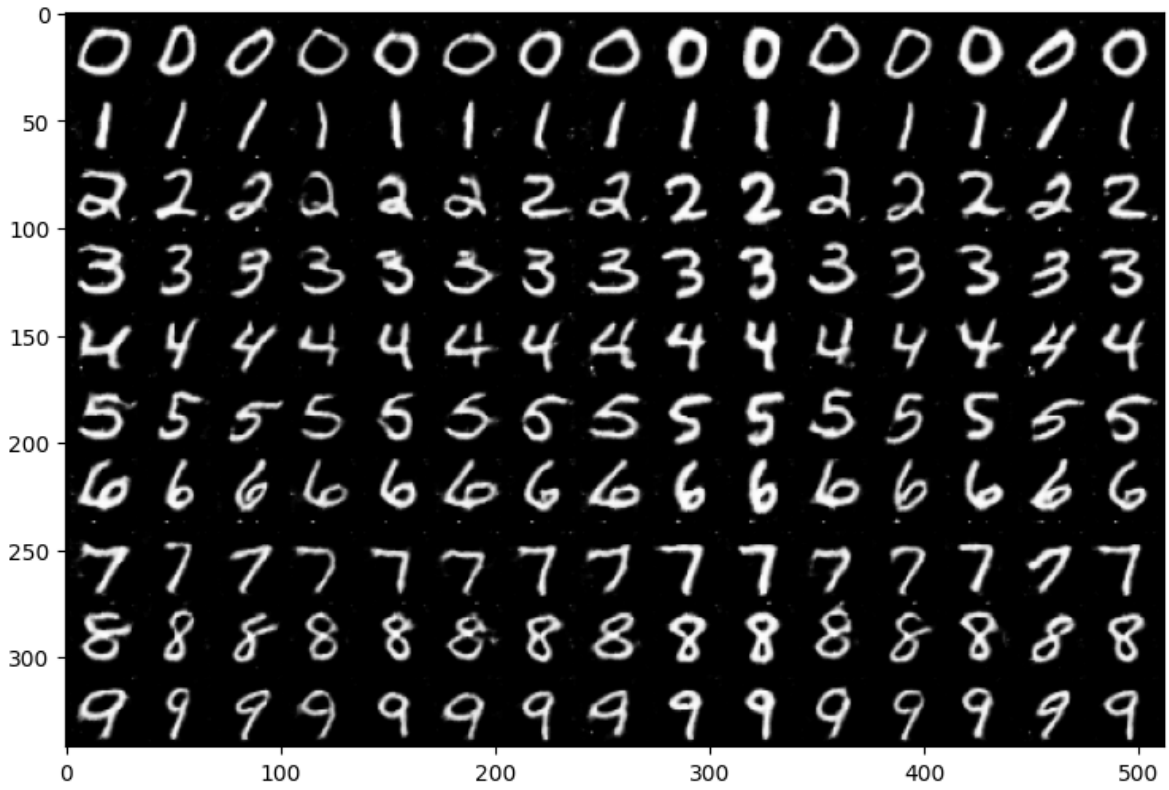


Figure 38: Generate number, each column corresponds to a unique noise vector z.