EX5 - Advanced Practical Course In Machine Learning
Generative Latent Optimization

Daniel Afrimi
203865837

January 06, 2021

Contents

1 Goal

The goal is to learn disentangled representations for the domain (in our case 0-10) in the dataset and a content representation for each digit image. In **Figures 1** we can see the architecture of the model.

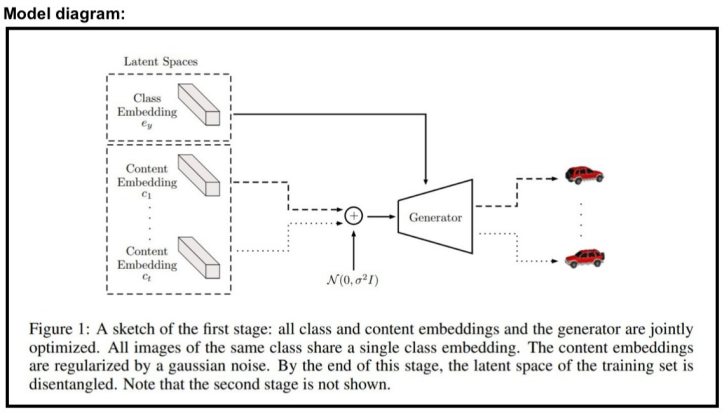


Figure 1: Generative Latent Optimization

2 Implementation Details

I trained the model on the MNSIT Dataset (256 samples).

Table 1: Training GLO

Parameter	Default value
Epochs	50
Batch size	32
$\sigma(fornoise)$	0.3
Learning rate Generator	0.001
Learning rate Class Embedding	0.001
Learning rate Content Embedding	0.001
Loss Function	L1 + L2

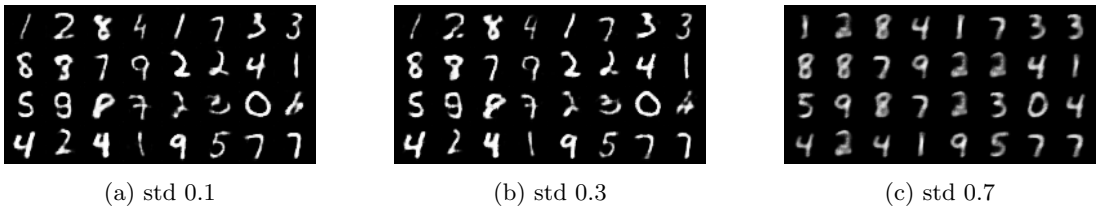
3 Content Noise

Using noise with a unlearned variance (and mean 0). This prevents information leakage through the content code, which can learn all but we want to give meaning to the class embedding. In the general case disentanglement will not happen without it as the content code can learn all the image attributes (both known and unknown) if unregularized.

In **Figures 2** we can see the reconstruct images with different variance factor (for the noise added to the content vectors). It can be seen that when the noise is small, the content code has more meaning and therefore the generating of the images is similar to the original images. But, as the noise increases, the model relies less on the content code and there is more meaning to the class code - so it can be seen that for digits that look different, the generated digits brings the same digit.

3.1 Content regularization

In order to disentanglement, we want to keep the norm of the content code small. During the loss calculation we added a regulation variable, which keeps the norm of this vector small. In practice this did not help to prevent information from leaking from the content code to the department code.
According to Yedid, in practice, because of the special properties of latent optimization, sometimes things work without it on very easy datasets - like MNIST.



(a) std 0.1 (b) std 0.3 (c) std 0.7

(d) Ground Truth (e) Ground Truth (f) Ground Truth

Figure 2: Different Noises with $std \in [0.1, 0.3, 0.7]$

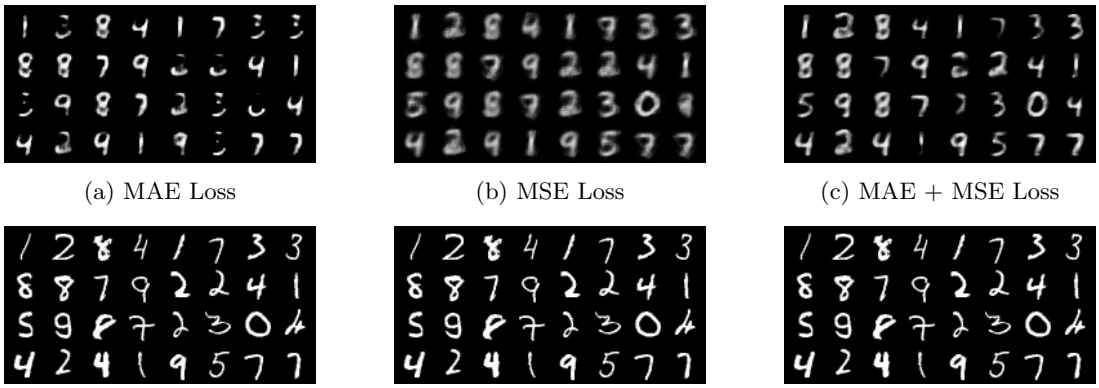
4 Comparing Different Loss Functions

L1/L2 losses work great when mapping is many-to-one, They do not work when mapping is one-to-many. As Yedid explained in the lecture, if we want to fill in a missing part of the image by different loss functions, L2 will give us an average result between the pixels and therefore the image will look blurry.

So in this section I compared the model on different loss functions. In the cited paper (Demystifying Inter-Class Disentanglement), the optimization originally implemented with a perceptual loss, which I included in my code (but I didn't ran it cause its too heavy).

In **Figures 3** we can see the reconstruct images with the different loss function. we can see that when using both L1 and L2 losses we can obtain the best result (in reconstruction). L2 is too blurry, and L1 missing some details.

In **Figures 4** we can see the training loss for each loss function.



(a) MAE Loss (b) MSE Loss (c) MAE + MSE Loss

(d) Ground Truth (e) Ground Truth (f) Ground Truth

Figure 3: Comparison between MAE, MSE, Combined MSE+MAE Losses

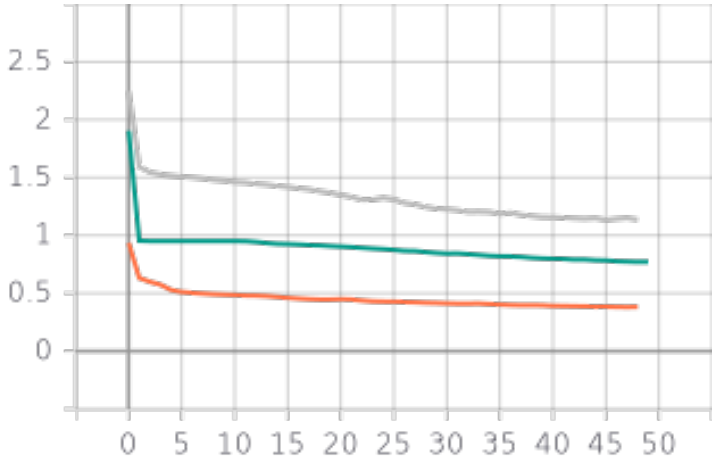


Figure 4: Training Loss with different losses - L2 (orange), L1 (green), combined (gray)

5 Test

When the model finished to train, we random a batch of 32 vectors (instead of the embedding of the class content). In **Figures 5** we can see the result of the model, which the images it generated looks like a digits. I assume that each vector in the input was similar (in approximation) to a vector corresponding to the class of a particular digit and therefore there are images that are similar to these and other digits.

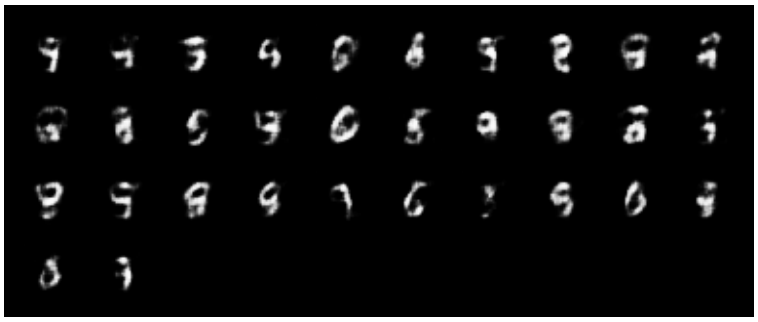


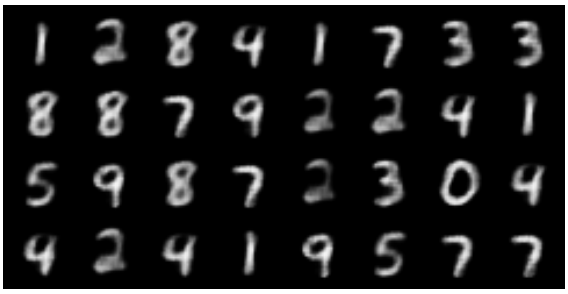
Figure 5: Result of the model on random Input

6 Theoretical Questions

6.1 What will happen if you will set the noise too high?

Note that when we pass input to the model, it gets a embedding of the class code and the content code (contacting). That is, the model relies on the embedding of both and knows from this codes how to generate the new digit. If the noise is high, we'll rely more on the class code. That is, for a specific image of a number, we will pay less attention to what the digit looks like but to the knowledge that it came from class "X".

Notice that as the noise is higher - the variance between the digits from the same class is smaller. In **Figures 6** we can see the reconstructs digits without any meaning to the content of the real digits given to us in that batch. It does not matter what the digit itself looks like but what class it came from, i.e. for digits from the same class we get the same image since the code for the content is noisy and the model gives it less attention. For example we can see that we have different types of the digit '8' (on the right), but the generated '8' digits is exactly the same one.



(a) Digits reconstruction with $\sigma = 1$

(b) Ground Truth

Figure 6: Digits reconstruction with high variance

6.2 What will happen if you will set the noise too low?

If we use low noise, we will not restrict the content code, and it will be able to hold all the information (known and unknown) of the image. That is, if we use low noise we will get that the generating of the images

during the training will be very similar to the digit itself we got and the recovery will be quite optimal. But, there will be no meaning to class code and therefore there will be less regularization and the model will probably go into over-fit.
Adding the noise to the content representation ensures that the class information does not get into the content code, and the class code will have meaning. I guess if we will try to sample new digit (after training with low noise) the model will not succeed to generate sample that looks like some digit, this is because we the digit-embedding vector will have no meaning.
In order to quantify this effect we can and check if the content code contains more information about the class, we can use adversarial discriminator, which get as input content code, and need to classify this code to a class. If successful, we will assume that information about the class appears in the content code, and there is no complete separation (quantify the disentanglement of the content codes from the class).