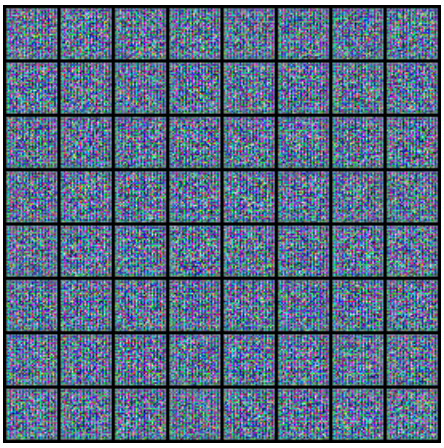
Problem 1

6.
The ELBO value obtained by $kl - gaussian - gaussian - mc$ with all the parameters set to default is -101.2971, which meets the criteria of being greater than -102. For this result, we can see that the ELBO value is close to the analytical method which is -101.2571. As a result of the prior $p(\boldsymbol{z}) = \mathcal{N}(\mathbf{0}, \boldsymbol{I}_L)$ in this question, both techniques produce similar results. However, Monte Carlo can be used for distributions that cannot be calculated analytically.

7.
The log-likelihood estimate ($\frac{1}{N}\sum_{i=1}^{N} \log p(\boldsymbol{x}_i)$) is -95.3737, where we use importance sampling to get the approximate $\log p(\boldsymbol{x} = \boldsymbol{x}_i)$. We can see that the ELBO of the trained model is the lower bound of the log-likelihood estimate ($\frac{1}{N}\sum_{i=1}^{N} \log p(\boldsymbol{x}_i)$).

Problem 2

2.


model0


model70

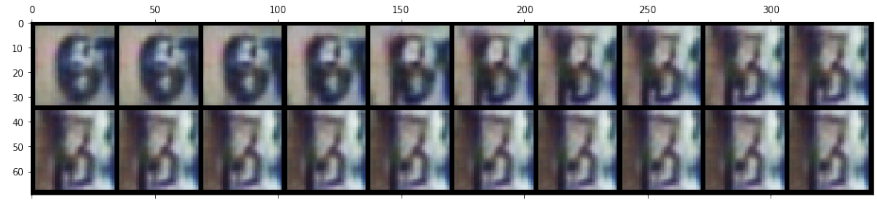

model200


model499

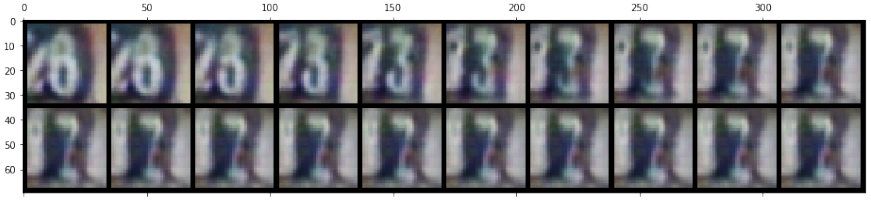
We can see from the four models above that model 0 generated nothing but noise because it has yet to learn anything. After 7000 training iterations, we can observe that the sample created is much better, but still too indistinct to detect the numbers. Continuing to train the model to 20000 epochs, we can see that the majority of the digits are proper despite some blurriness. However, in terms of diversity, the numerals are primarily 2, 6, and 8, which have a close form. The final model, with 50000 training iterations, produced far more diversified and crisper images.

Problem 2 continued on next page...

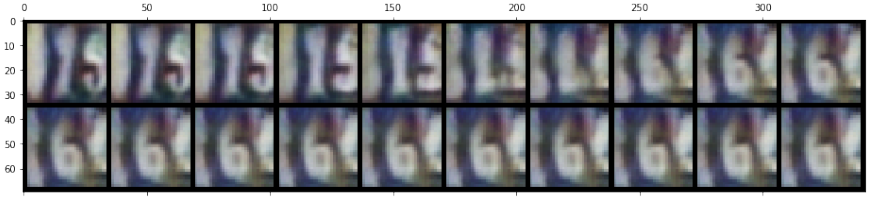
3.



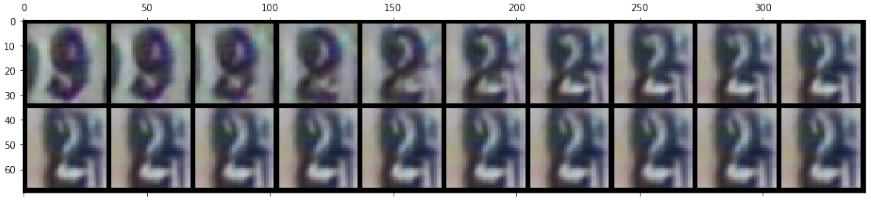
latent representation z3



latent representation z30



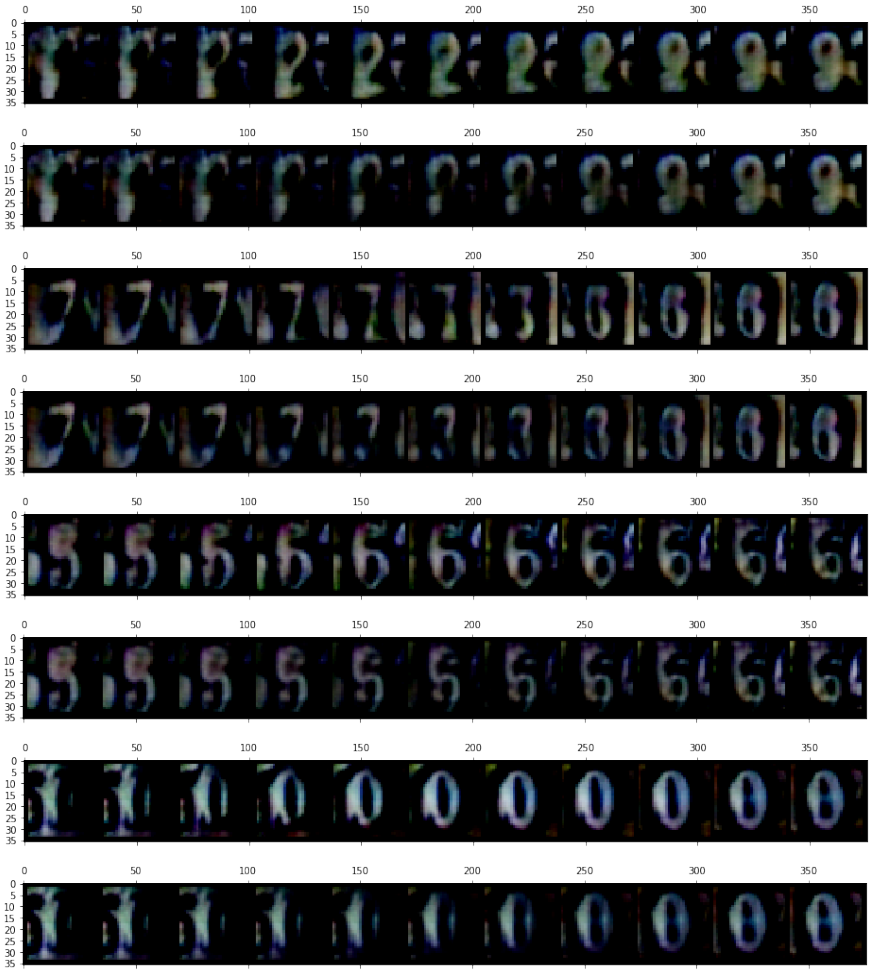
latent representation z35



latent representation z50

The four images above depict visual variations on latent representation z = 3, 30, 35, and 50 with epsilon values ranging from 1 to 20. We can observe that the third latent dimension changes from 6 to 3; the 30th latent dimension changes from 6 to 7/2; the 35th latent dimension changes from 15 to 6 or converges to 6 more clearly; and the 50th latent dimension changes from 9 to 2.

4.



interpolate example 1-a

interpolate example 1-b

interpolate example 2-a

interpolate example 2-b

interpolate example 3-a

interpolate example 3-b

interpolate example 4-a

interpolate example 4-b

Each pair of the four photos uses the same z_1 and z_0 as starting and finishing points. It demonstrates the distinction between the two interpolation schemes. The interpolation differs because (a) it is a latent representation interpolation or (b) it is a pixel space interpolation of x_0, x_1 pictures with matching pixels.

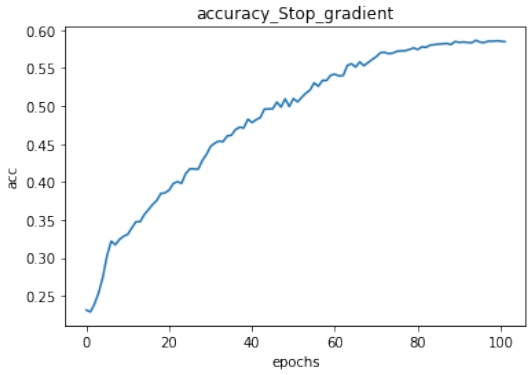
In the (a)latent space interpolation, we can observe from the example plots (interpolate example-1-a,2-a,3-a,4-a) that the number progressively changed from image created by $g(z_1)$ to image generated by $g(z_0)$ with a minor zoning effect. The interpolated picture appears to be a composite of the two original photos, indicating that it is still there in the original data manifold.

In (b) pixel space interpolation, we can observe from the example plot (interpolate example-1-b,2-b,3-b,4-b) that the interpolated picture progressively morphs and eventually changes to the image created by $g(z_0)$. We could barely tell the interpolated image from the original photograph. This indicates that the pixel space interpolation does not remain in the data manifold.

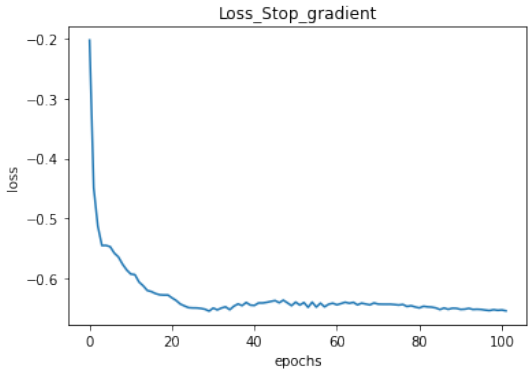
The distinction between the two methods for interpolating between pictures demonstrates the significance of latent representation.

Problem 3

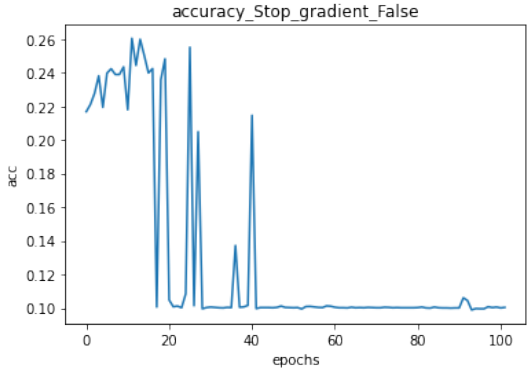
3.4 The plots are as following:



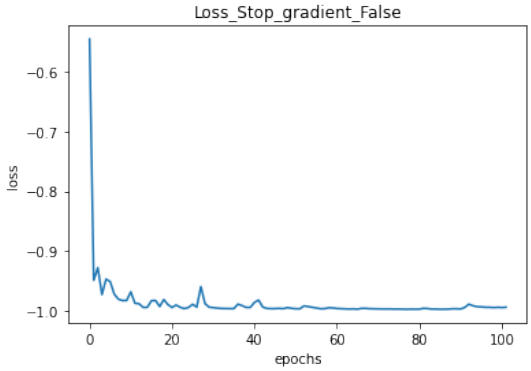
Accuracy(stop-gradient=T)



Loss(stop-gradient=T)



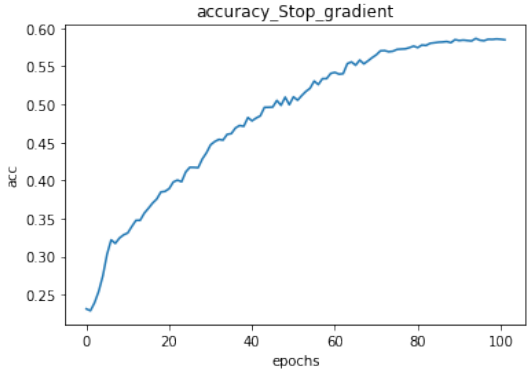
Accuracy(stop-gradient=F)



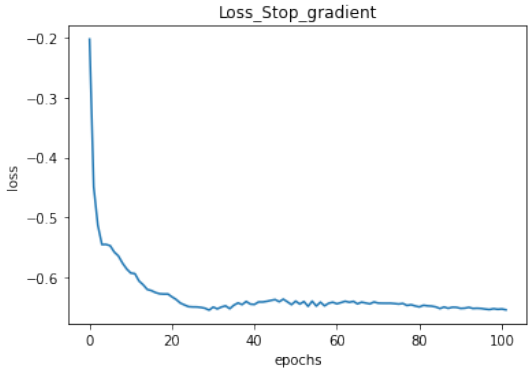
Loss(stop-gradient=F)

From the above plots, we can see that SimSiam achieves a accuracy of 0.60. But without stop-gradient, the optimizer quickly finds a degenerated solution and reaches the minimum possible loss of 1. The model has collapsed. The accuracy drops from 0.6 to 0.1 without the gradient stopping.

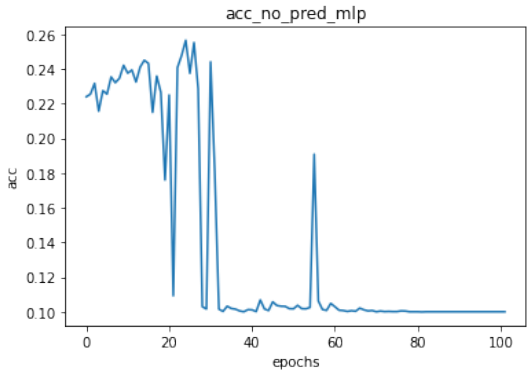
3.5 The plots are as following:



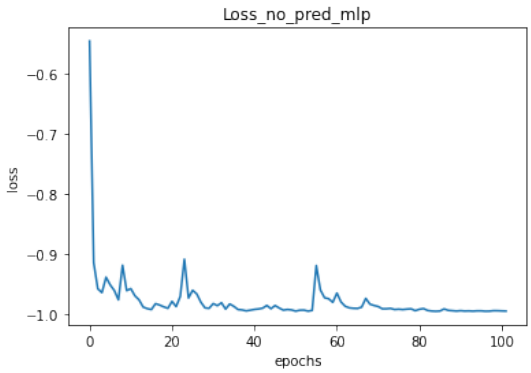
Accuracy(stop-gradient=T,MLP=None)



Loss(stop-gradient=T, MLP=None)



Accuracy(stop-gradient=T,MLP=no-pred-mlp)



Loss(stop-gradient=T, MLP=no-pred-mlp)

From the above plots, we can see that without the MLP preditor, the model also does not work when h is the identity mapping. The accuracy drops from 0.6 to 0.1 without the MLP preditor. Also, the loss does not decrease smoothly till convergence. The reason for this is that the symmetric loss is used as an objective function$(0.5 * D(p_1, stop - grad(z_2)) + 0.5 * D(p_2, stop - grad(z_1))$. The loss is now equal to $0.5 * D(z_1, stop - grad(z_2)) + 0.5 * D(z_2, stop - grad(z_1))$.Its gradient is in the same direction as that of $D(z1, z2)$, but the magnitude is scaled by 1/2.In this case, using stop-gradient is equivalent to removing stop-gradient and scaling the loss by 1/2. Hence, collapse is observed.