Hena Ghonia (20213256)

Problem 1

6. (report, 15 pts) Consider a latent variable model $p_\theta(\boldsymbol{x}) = \int p_\theta(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})dz$. The prior is define as $p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_L)$ and $\boldsymbol{z} \in \mathbb{R}^L$. Train a VAE with a latent variable of 100-dimensions ($L = 100$). Use the provided network architecture and hyperparameters described in '`vae.ipynb`'[1]. Use ADAM with a learning rate of 3×10^{-4}, and train for 20 epochs. Evaluate the model on the validation set using the **ELBO**. Marks will neither be deducted nor awarded if you do not use the given architecture. Note that for this question you have to:Train a model to achieve an average per-instance ELBO of ≥ -102 on the validation set, and report the ELBO of your model. The ELBO on validation is written as:

$$\frac{1}{|\mathcal{D}_{\text{valid}}|} \sum_{\boldsymbol{x}_i \in \mathcal{D}_{\text{valid}}} \mathcal{L}_{\text{ELBO}}(\boldsymbol{x}_i) \geq -102$$

Feel free to modify the above hyperparameters (except the latent variable size) to ensure it works.

Solution: We used same hyperparameters mentioned in question. Figure 1 indicates ELBO value for 20 epochs when VAE trained with loss - KL divergence calculated with Monte carlo sampling. At 20th epoch ELBO value is -90.523872 which is greater than -102 on validation set.
In Figure 2, it seen that ELBO values calculated using Monte-carlo method are almost similar to ELBO values calculated using analytical method.

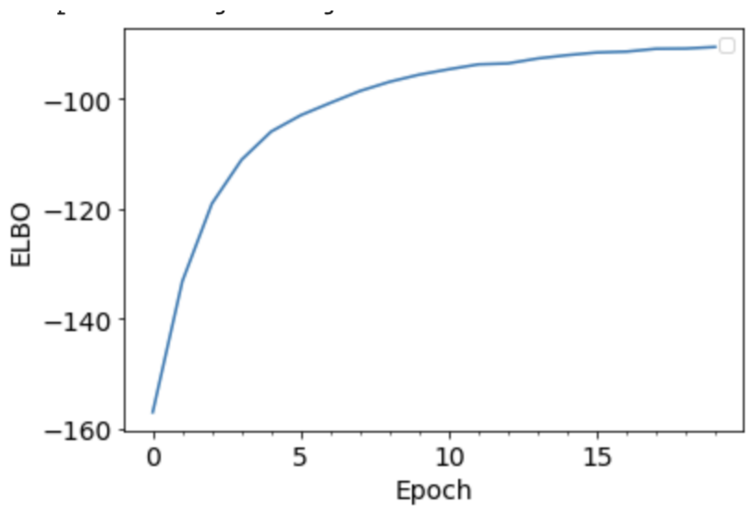


Figure 1: ELBO value for 20 epoch on validation set

[1]This file is executable in Google Colab. You can also convert `vae.ipynb` to `vae.py` using the Colab.

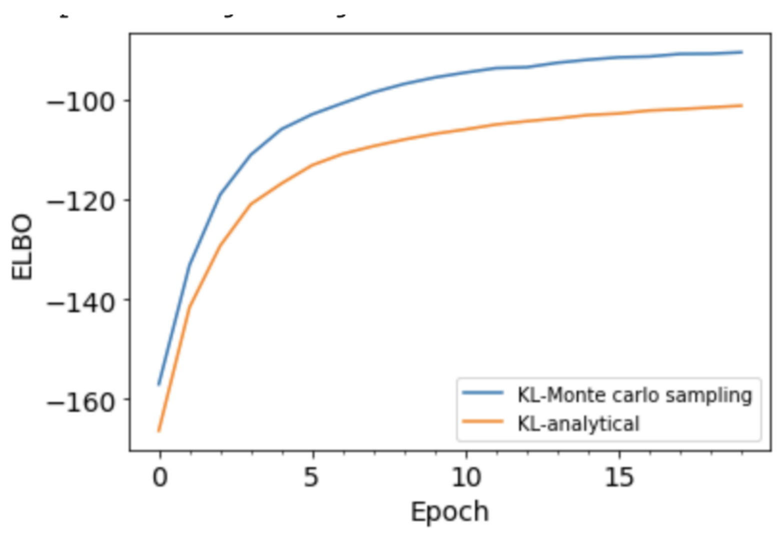


Figure 2: ELBO value by calculating loss analytically and with monte carlo samppling

7. Evaluate *log-likelihood* of the trained VAE models by using importance sampling, which was covered during the lecture. Use the codes described in '`vae.ipynb`'. The formula is reproduced here with additional details:

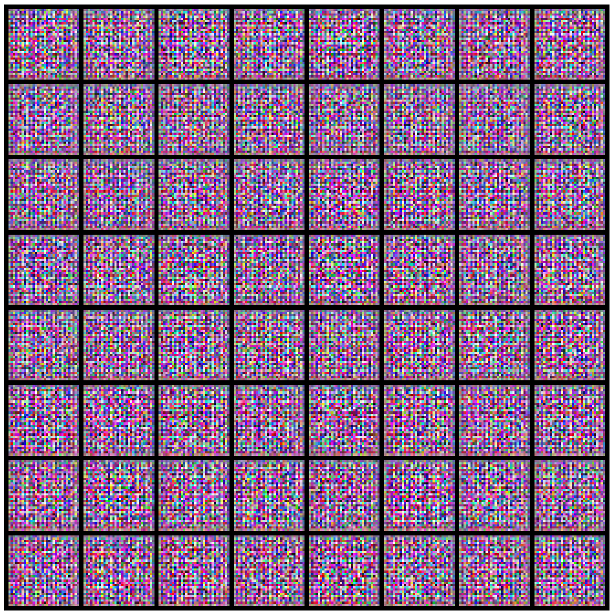
$$\log p(\boldsymbol{x} = \boldsymbol{x}_i) \approx \log \frac{1}{K} \sum_{k=1}^{K} \frac{p_\theta(\boldsymbol{x} = \boldsymbol{x}_i | \boldsymbol{z}_i^{(k)})\; p(\boldsymbol{z} = \boldsymbol{z}_i^{(k)})}{q_\phi(\boldsymbol{z} = \boldsymbol{z}_i^{(k)} | \boldsymbol{x}_i)}; \quad \text{for all } k\text{: } \boldsymbol{z}_i^{(k)} \sim q_\phi(\boldsymbol{z}|\boldsymbol{x}_i)$$

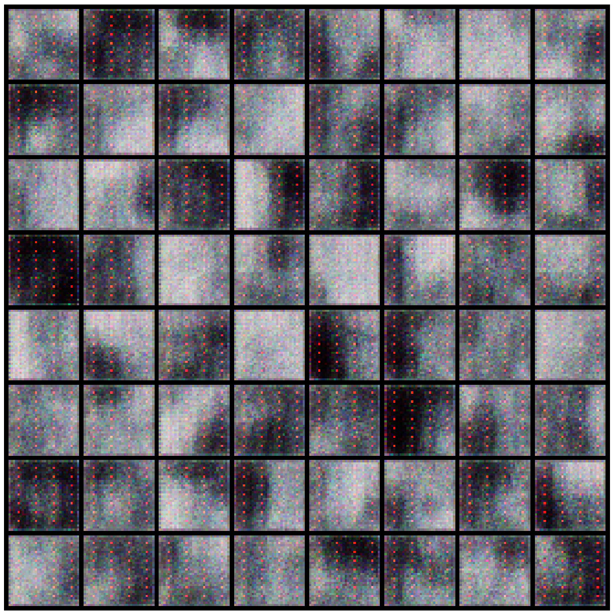
and $\boldsymbol{x}_i \in \mathcal{D}$.

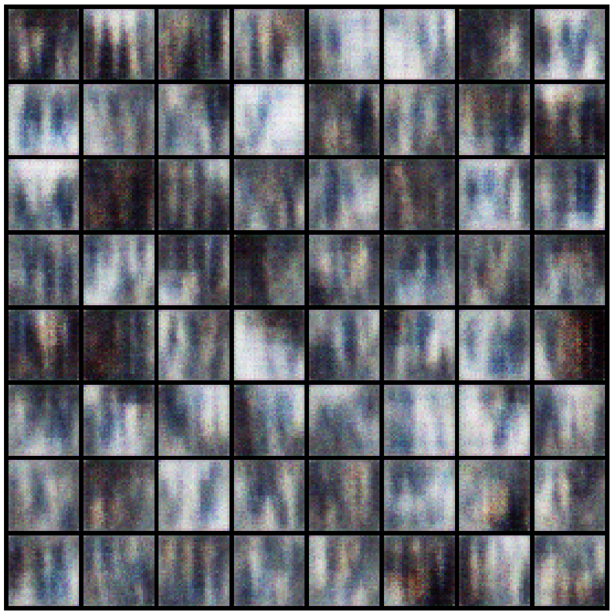
(a) Report your evaluations of the trained model on the test set using the log-likelihood estimate ($\frac{1}{N}\sum_{i=1}^{N} \log p(\boldsymbol{x}_i)$), where N is the size of the test dataset. Use $K = 200$ as the number of importance samples, D as the dimension of the input ($D = 784$ in the case of MNIST), and $L = 100$ as the dimension of the latent variable.
Solution: We get $\log p(x)$: -95.8 when model trained with loss as KL divergence calculated using Monte carlo sampling whereas $\log p(x) = -95.4$ is obtained when model trained with loss which is calculated analytically.

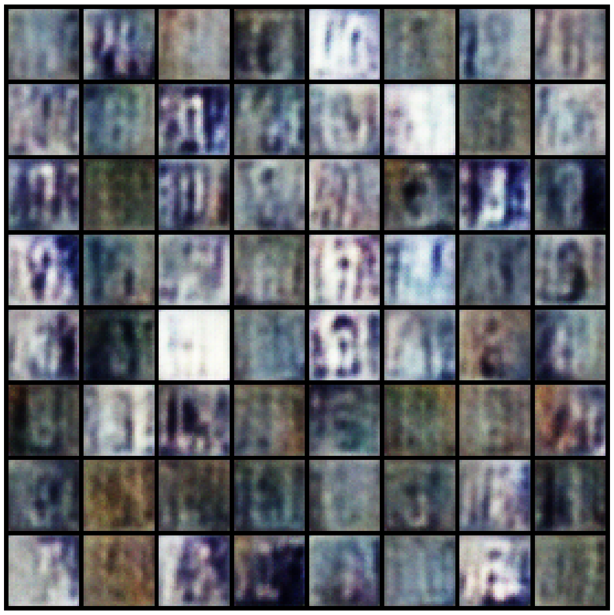
Problem 2

Question- 2 Provide visual samples. Comment the quality of the samples from each model (e.g. blurriness, diversity).
Solution: As you can see in below figure (a), at 0th iteration generator outputs noise, and slowly it learns and generates better images which are less blurry. One can see progression in below figure (b), (c), (d), (e), (f), (g) and (h). At 30,000 iteration and at 49,900 iteration it is less blurry and more diverse representation is seen.

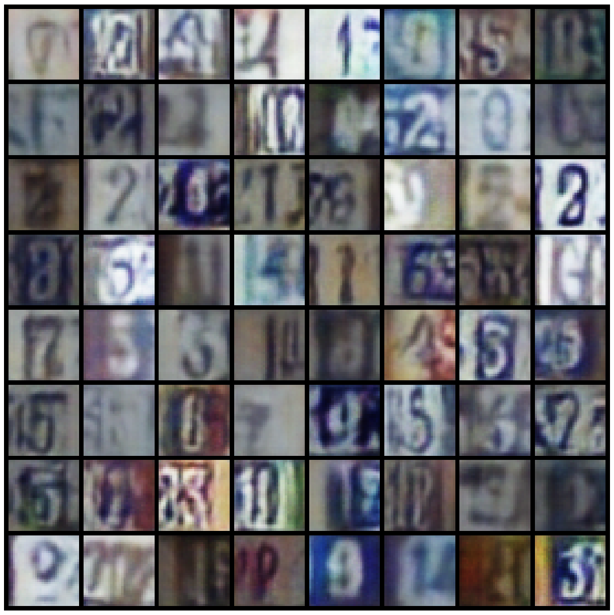

(a) At 0th iteration

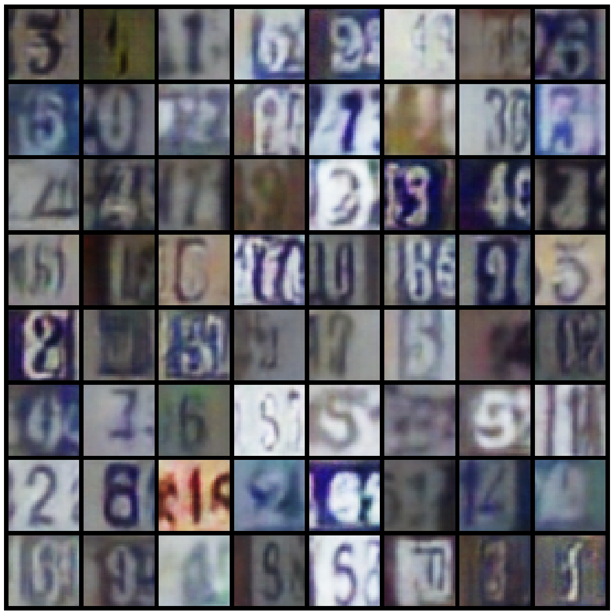

(b) At 100th iteration


(c) At 200th iteration


(d) At 1000th iteration

(e) At 2000th iteration

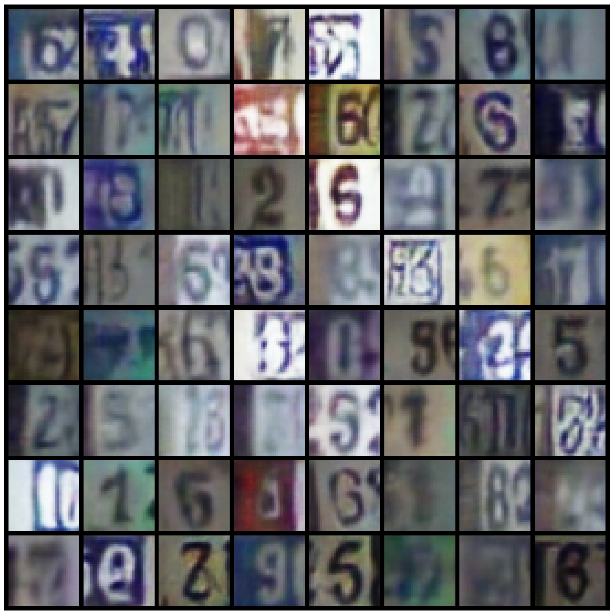

(f) At 16000th iteration

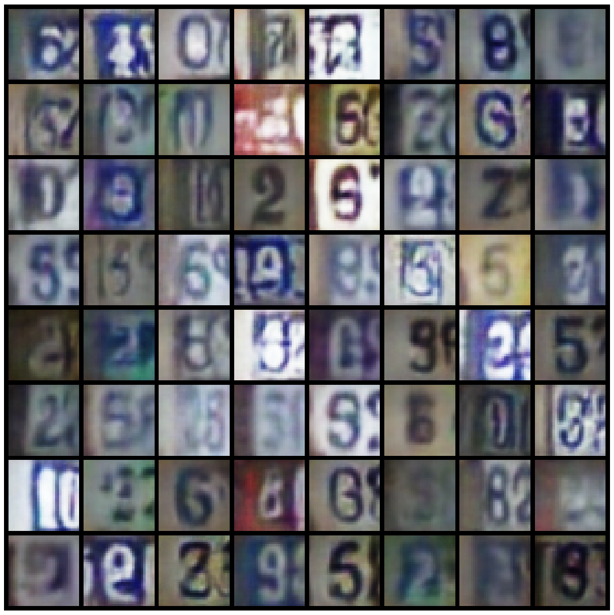

(g) At 30000th iteration

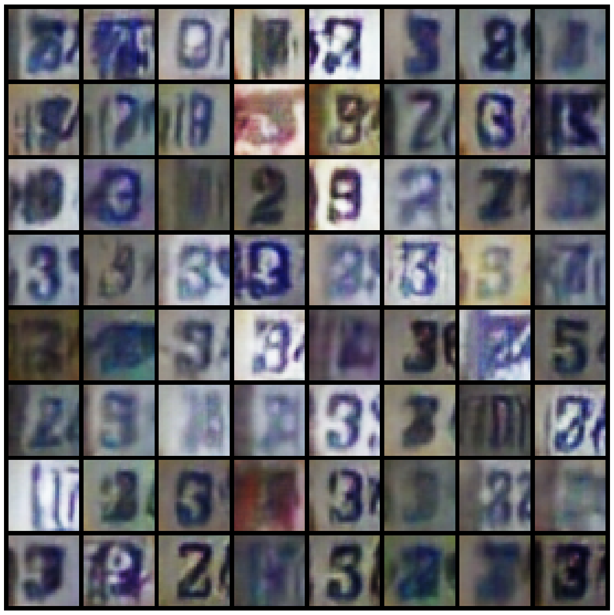
(h) At 49900th iteration

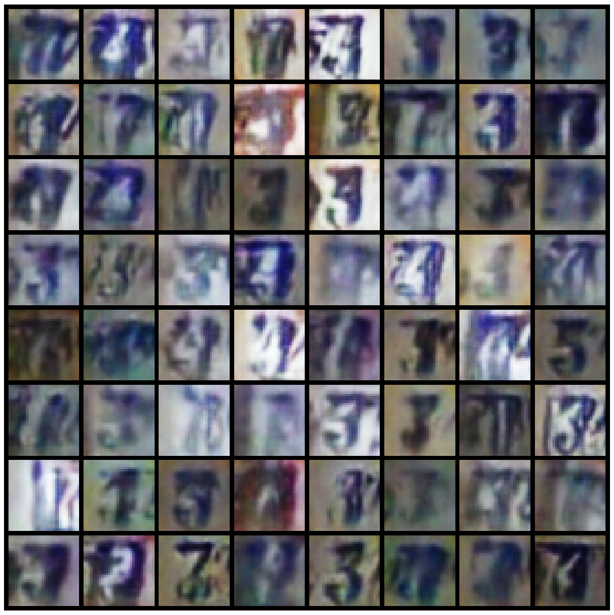
Question- 3 We want to see if the model has learned a disentangled representation in the latent space.
Sample a random z from your prior distribution. Make small perturbations to your sample z for *each dimension* (e.g. for a dimension i, $z'_i = z_i + \epsilon$). ϵ has to be large enough to see some visual difference. For each dimension, observe if the changes result in visual variations (that means variations in $g(z)$). You do not have to show all dimensions, just a couple that result in interesting changes.

Solution: $\epsilon = 5$ was added at each dimension. In below figure, (i) and (j) which are result of perturbation at dimension 0 and 2 is seen. One can see the difference in image (j) and (k) where 6 changes to 8 and in figure (l) it changes to 3 at location (3,5) in batch of images displayed and can observe that model has learnt disentangled representation in latent space.


(i) Perturbation at dimension i = 0


(j) Perturbation at dimension i = 2


(k) Perturbation at dimension i = 6


(l) Perturbation at dimension i = 16

Question- 4 Compare between interpolating in the data space and in the latent space. Pick two random points z_0 and z_1 in the latent space sampled from the prior.

For $\alpha = 0, 0.1, 0.2 \ldots 1$ compute $z'_\alpha = \alpha z_0 + (1-\alpha) z_1$ and plot the resulting samples $x'_\alpha = g(z'_\alpha)$.

Solution: In below figure, from $\alpha = 0.4$ to $\alpha = 0.7$ images are more blurry. You can observe in below figure (q), (r), (s), (t).

(m) For $\alpha = 0$.

(n) For $\alpha = 0.1$

(o) $\alpha = 0.2$

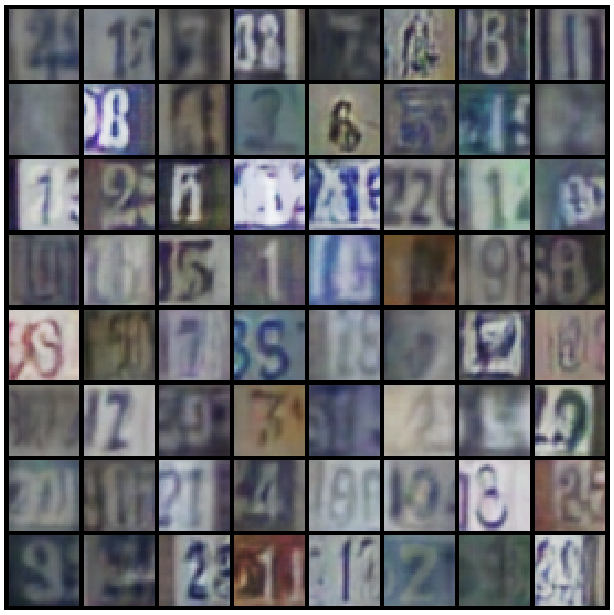
(p) $\alpha = 0.3$

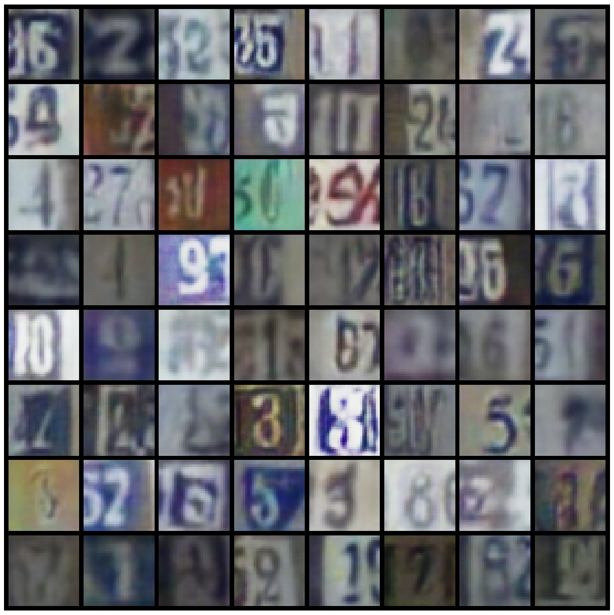
(q) $\alpha = 0.4$

(r) $\alpha = 0.5$

(s) $\alpha = 0.6$

(t) $\alpha = 0.7$


(u) $\alpha = 0.8$


(v) $\alpha = 0.9$

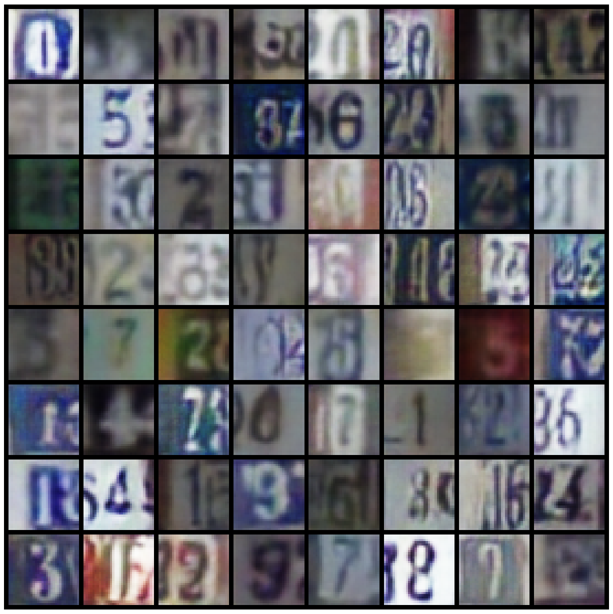


Figure 3: $\alpha = 1$

(a)

(b) Using the data samples $x_0 = g(z_0)$ and $x_1 = g(z_1)$ and for $\alpha = 0,\ 0.1,\ 0.2 \ldots 1$ plot the samples $\hat{x}_\alpha = \alpha x_0 + (1-\alpha) x_1$.
Solution: In below figure, from $\alpha = 0.4$ to $\alpha = 0.7$ images are more blurry. You can observe in below figure (e), (f), (g), (h).

(a) For $\alpha = 0$.

(b) For $\alpha = 0.1$

(c) $\alpha = 0.2$

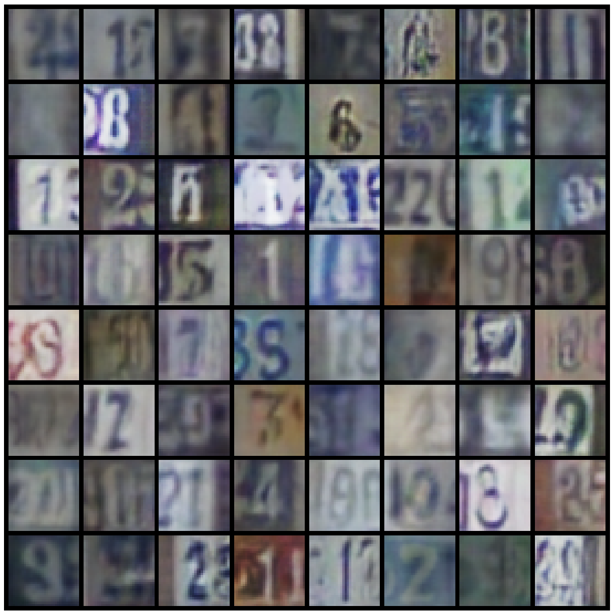
(d) $\alpha = 0.3$

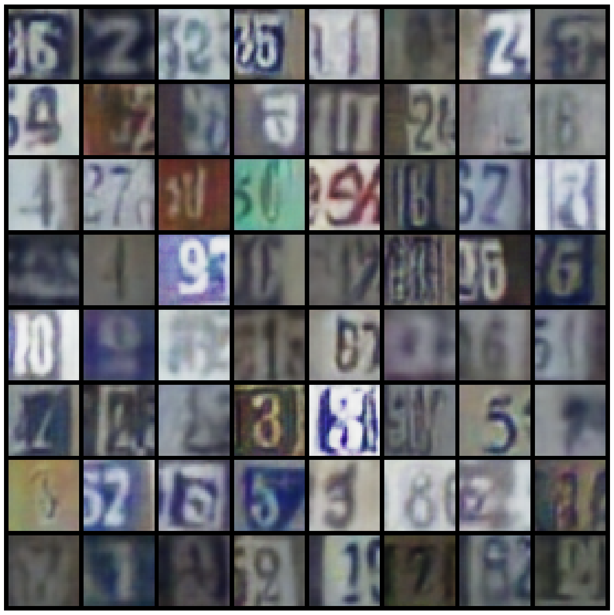
(e) $\alpha = 0.4$

(f) $\alpha = 0.5$

(g) $\alpha = 0.6$

(h) $\alpha = 0.7$


(i) $\alpha = 0.8$


(j) $\alpha = 0.9$

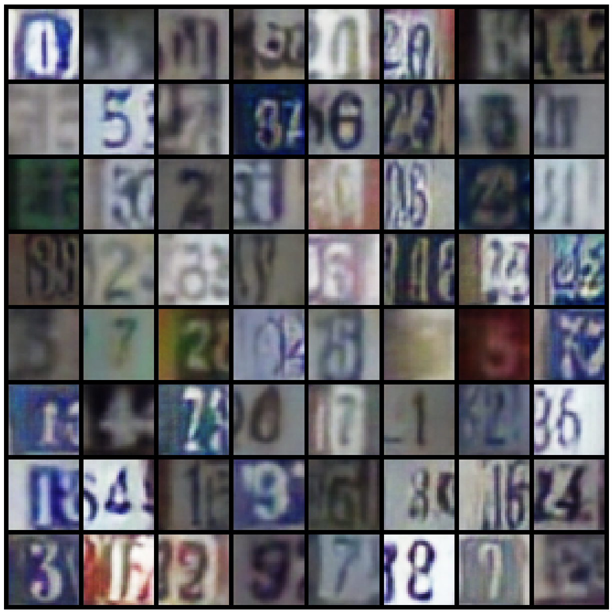
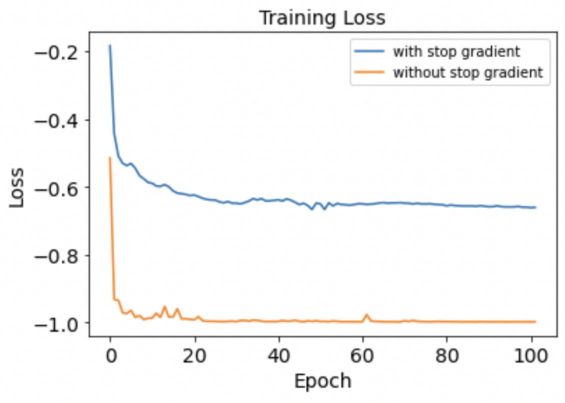


Figure 4: $\alpha = 1$

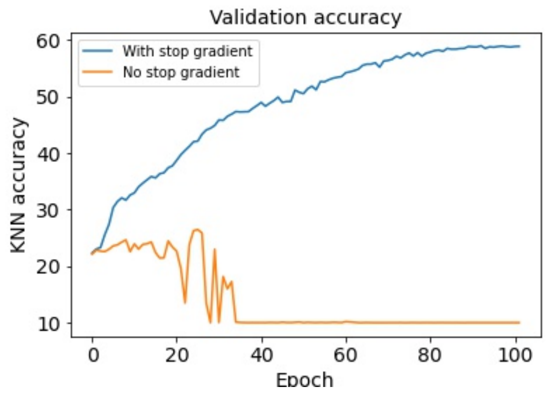
Explain the difference with the two schemes to interpolate between images.
Solution: Images generated in part (b) are more blurred than part (a). It is observed that in both part (a) and part (b) during $\alpha = 0.4$ to $\alpha = 0.7$ images are more blurry.

Problem 3

1. **(report, 10 pts)** Train the model for 100 epochs with and without gradient stopping. Plot training loss and Knn accuracy against training epochs.
   Solution: As shown in below Figure 5 (a) Training loss for without gradient (orange curve)is comparatively less than with stop gradient(blue curve) but it does not decrease much after 10th epoch for without stop gradient.
   For Knn accuracy, in Figure 5 (b) validation KNN accuracy decreases for no stop-gradient(orange curve) whereas accuracy increases for with stop-gradient(blue curve), also without stop gradient after 40th epoch accuracy remains at 10% which means its not learning and representational collapse might be occuring.
   Hence including stop gradient results in stable learning. Also, from theory assignment 3, we learnt that including stop gradient and predictor network helps to avoid representational collapse.



(a) Training loss



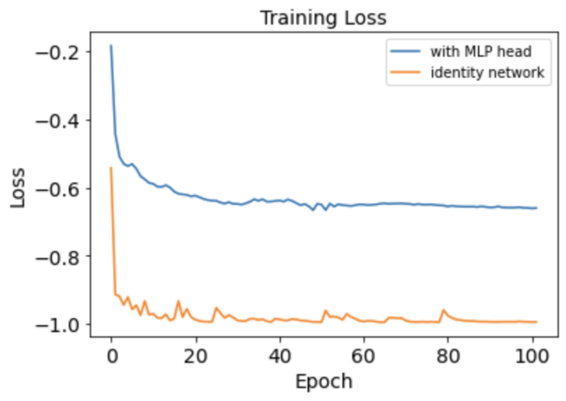
(b) validation KNN accuracy

Figure 5: Plot of training loss and validation knn accuracy with and without gradient stopping

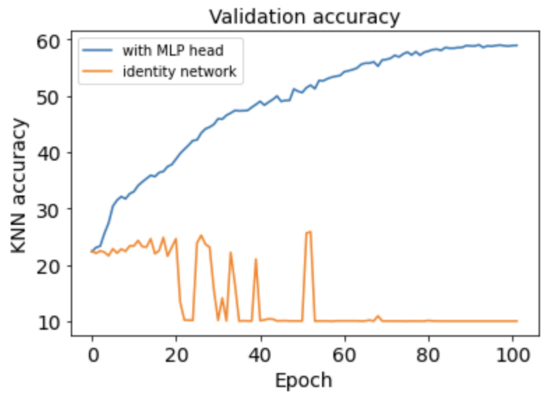
2. **(report, 10 pts)** Investigate the effect of the predictor network (MLP) by experimenting the below setting(s). Plot training loss and KNN accuracy against training epochs.

   (a) Remove the predictor by replacing it with an identity mapping.
   Solution: As shown in below Figure **??** (a) Training Loss decreases smoothly for with MLP head(i.e with predictor network) indicated by blue line whereas for without predictor network i.e by replacing predictor network with identity network (orange line) loss value fluctuates as during training. However training loss is comparatively less in removing the predictor with identity network.
   For Knn accuracy, in Figure **??** (b) validation KNN accuracy decreases for replacing predictor network with identity mapping(orange curve) whereas accuracy increases for including MLP - predictor network(blue curve), also by replacing predictor network with identity mapping(or not including predictor network), after 60th epoch accuracy remains at 10% which means its not learning and representational collapse might be occuring which is like similar scenario when we exclude stop gradient.

So including predictor network in SimSiam model results in stable learning. Hence excluding either stop gradient or predictor network lead to representational collapse.



(a) Training loss



(b) validation KNN accuracy

Figure 6: Plot of training loss and validation knn accuracy with MLP (with predictor network)and replacing predictor with identity network