Due Date: April 12, 2020

Problem 1

1. **(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 $p(\boldsymbol{z}) = \mathcal{N}(\mathbf{0}, \boldsymbol{I}_L)$ 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:

   (a) Train a model to achieve an average per-instance ELBO of ≥ -102 on the validation set.

   Answer:

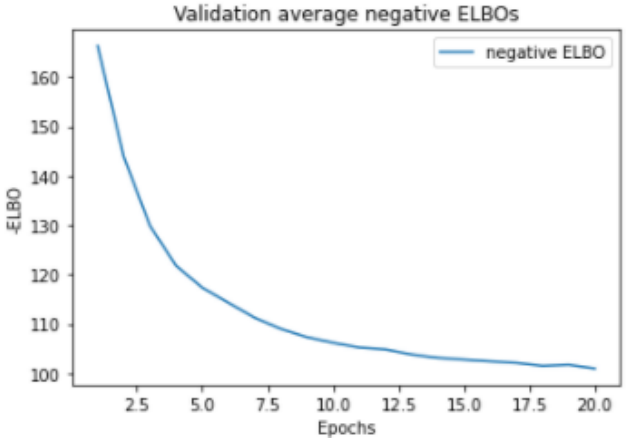


Figure 1: Validation average negative ELBOs

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

```
-elbo:  166.3352813720703
-elbo:  143.85409545898438
-elbo:  129.87220764160156
-elbo:  121.76470947265625
-elbo:  117.33319854736328
-elbo:  114.26793670654297
-elbo:  111.24293518066406
-elbo:  108.97356414794922
-elbo:  107.30304718017578
-elbo:  106.16422271728516
-elbo:  105.19488525390625
-elbo:  104.81182861328125
-elbo:  103.78592681884766
-elbo:  103.09915161132812
-elbo:  102.75752258300781
-elbo:  102.38983154296875
-elbo:  102.09286499023438
-elbo:  101.48159790039062
-elbo:  101.69996643066406
-elbo:  100.93998718261719
```

Figure 2: Validation average negative ELBOs

2. **(`report`, 15 pts)** 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.

   Answer:

   log p(x): -95.59159851074219

Problem 2

1. **(`report`, 15 pts)** Execute the jupyter notebook to train a dirac generator. Substitute the random seed with your matricule number and use the predetermined configurations denoted by the prefix '`dirac-`'. In the report, provide with the final plot of the parameter trajectories for each case and interpret the results.

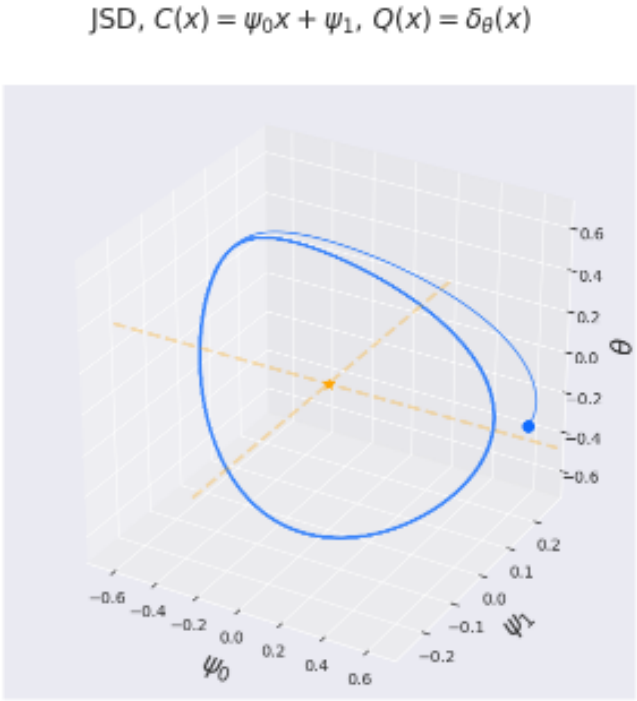


Figure 3: Final plot of Configuration 1: JSD with no gradient penalty and generator alpha ema=None

$\longrightarrow$ With JSD and no gradient penalty, the algorithm diverge.

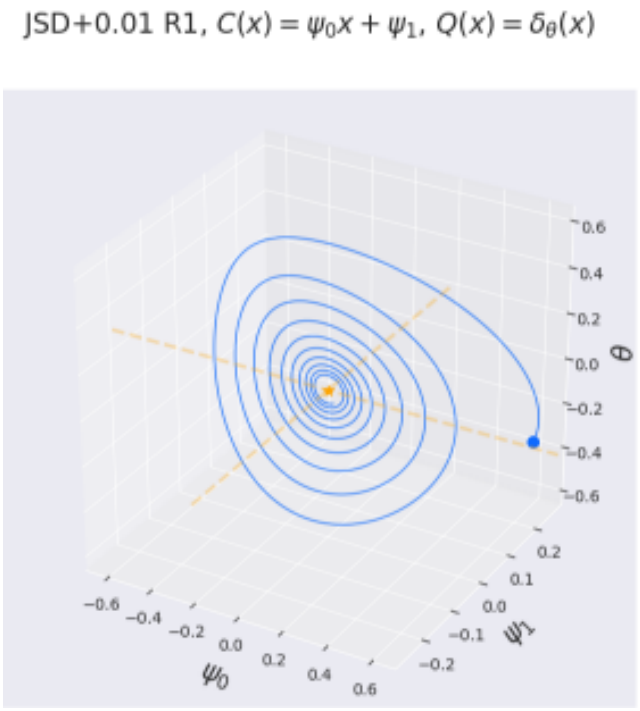


Figure 4: Final plot of Configuration 2: JSD with R1 gradient penalty, 0.01 regularization for critic and generator alpha ema=None

$\longrightarrow$ JSD with R1 gradient penalty and 0.01 regularization get closer to the target but very slow.

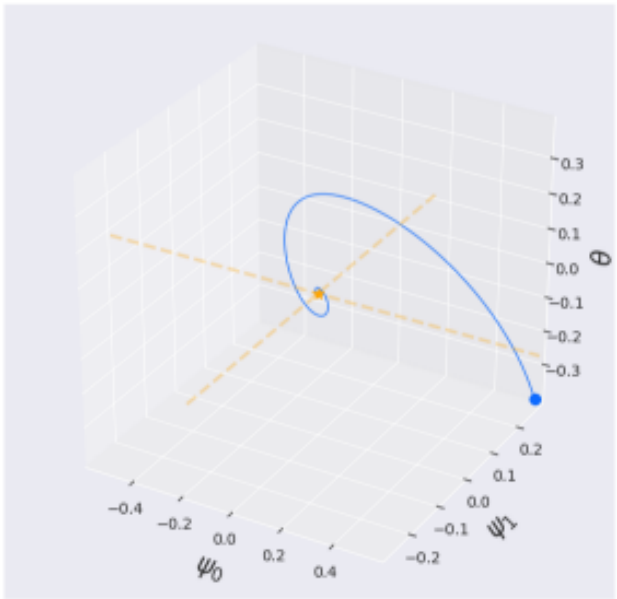


Figure 5: Final plot of Configuration 3: JSD with R1 gradient penalty, 0.1 regularization for critic and generator alpha ema=None

⟶ JSD with R1 gradient penalty and 0.1 regularization converge quickly because of 0.1 regularization.

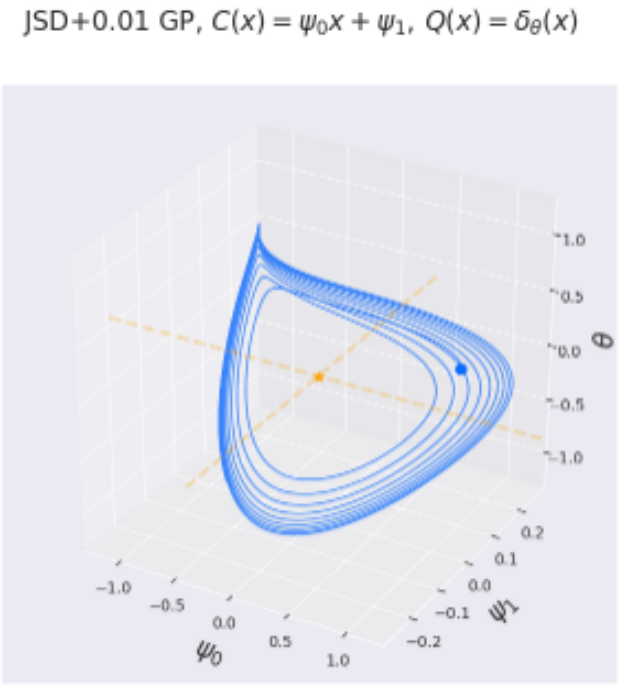


Figure 6: Final plot of Configuration 4: JSD with GP gradient penalty, 0.01 regularization for critic and generator alpha ema=None

⟶ JSD with GP gradient penalty, 0.01 regularization diverge and far more and more.

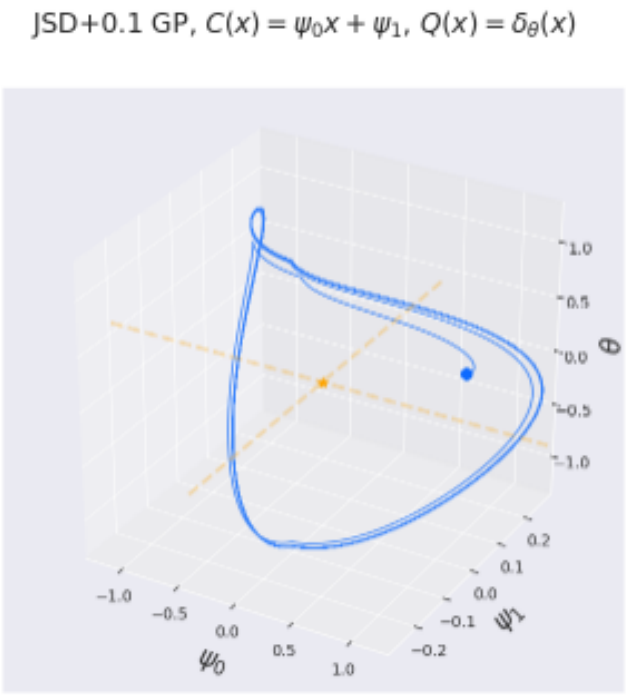


Figure 7: Final plot of Configuration 5: JSD with GP gradient penalty, 0.1 regularization for critic and generator alpha ema=None

$\longrightarrow$ JSD with GP gradient penalty, 0.1 regularization diverge.

W1, $C(x) = \psi_0 x + \psi_1$, $Q(x) = \delta_\theta(x)$

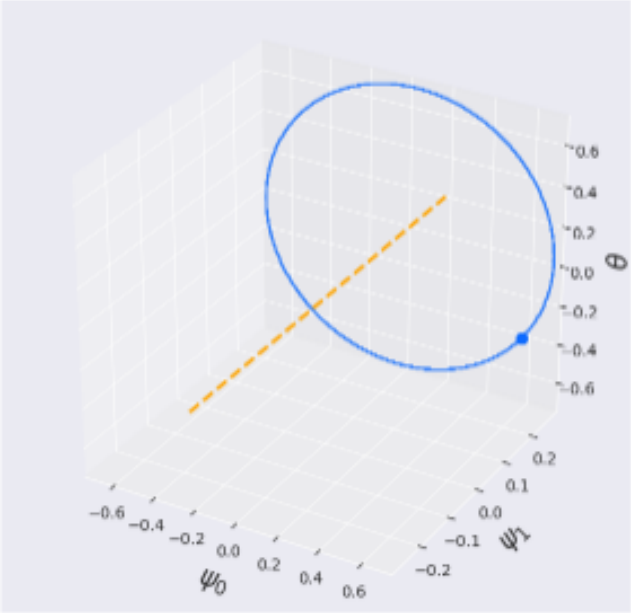


Figure 8: Final plot of Configuration 6: W1 with no gradient penalty and generator alpha ema=None

$\longrightarrow$ W1 with no gradient penalty fails to converge.

W1+0.01 R1, $C(x) = \psi_0 x + \psi_1$, $Q(x) = \delta_\theta(x)$

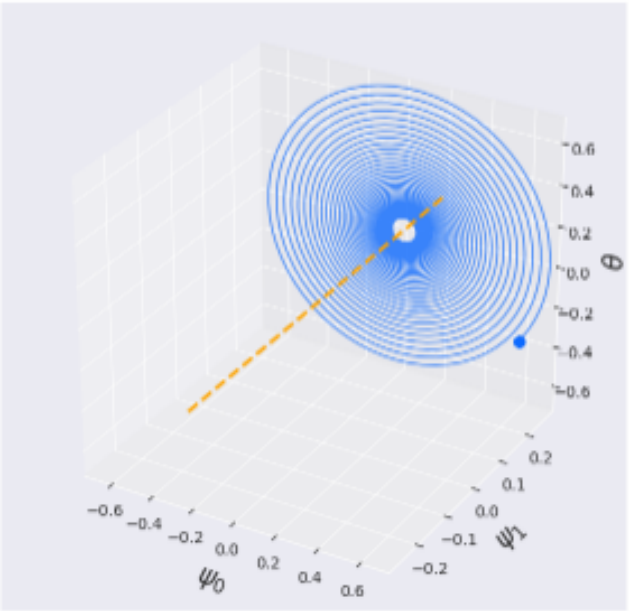


Figure 9: Final plot of Configuration 7: W1 with R1 gradient penalty, 0.01 regularization for critic and generator alpha ema=None

⟶ W1 with R1 gradient penalty, 0.01 regularization tends to converge slowly.

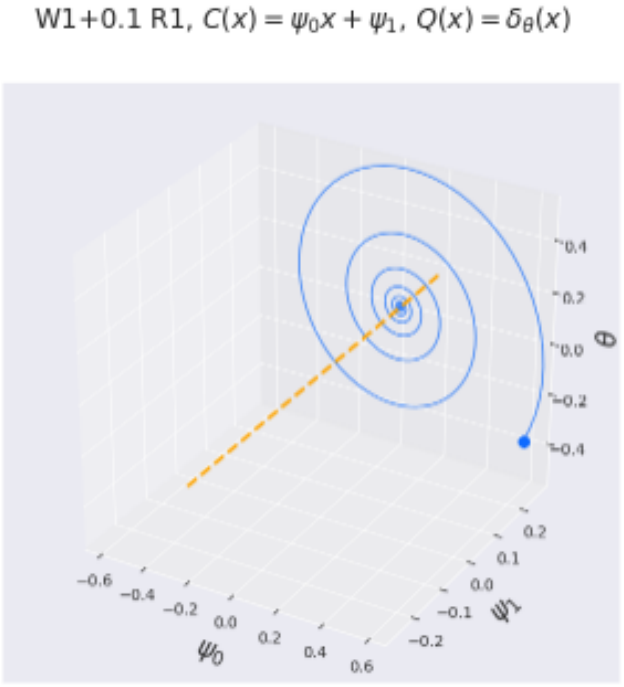


Figure 10: Final plot of Configuration 8: W1 with R1 gradient penalty, 0.1 regularization for critic and generator alpha ema=None

⟶ W1 with R1 gradient penalty, 0.1 regularization converge.

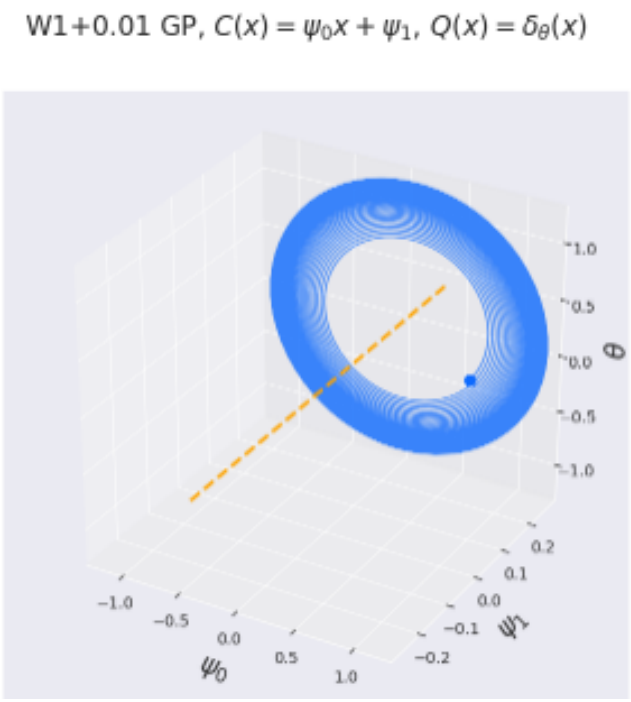


Figure 11: Final plot of Configuration 9: W1 with GP gradient penalty, 0.01 regularization for critic and generator alpha ema=None

$\longrightarrow$ W1 with GP gradient penalty, 0.01 regularization diverge.

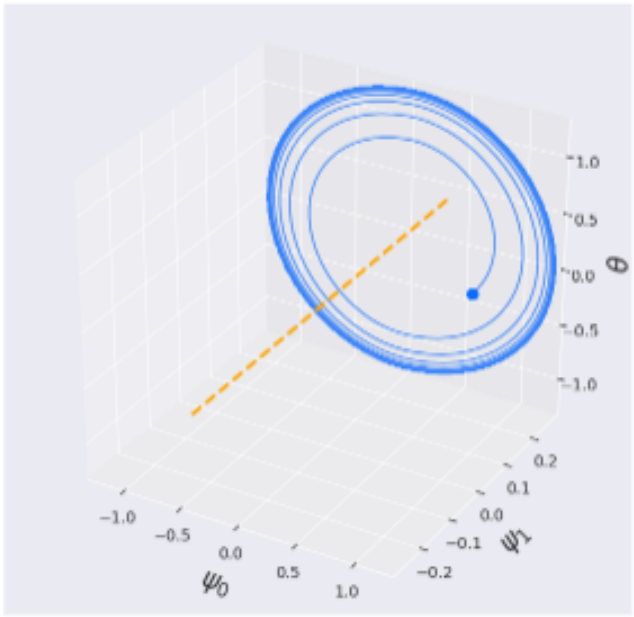


Figure 12: Final plot of Configuration 10: W1 with GP gradient penalty, 0.1 regularization for critic and generator alpha ema=None

⟶ W1 with GP gradient penalty, 0.1 regularization diverge.

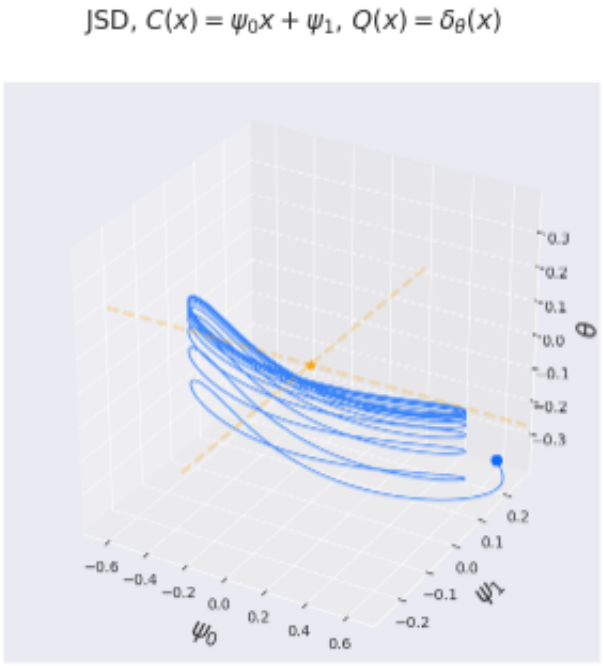


Figure 13: Final plot of Configuration 11: JSD with no gradient penalty

⟶ JSD with no gradient penalty but with generator alpha ema tends to diverge (get closer but diverge).

JSD+0.01 R1, $C(x) = \psi_0 x + \psi_1$, $Q(x) = \delta_\theta(x)$

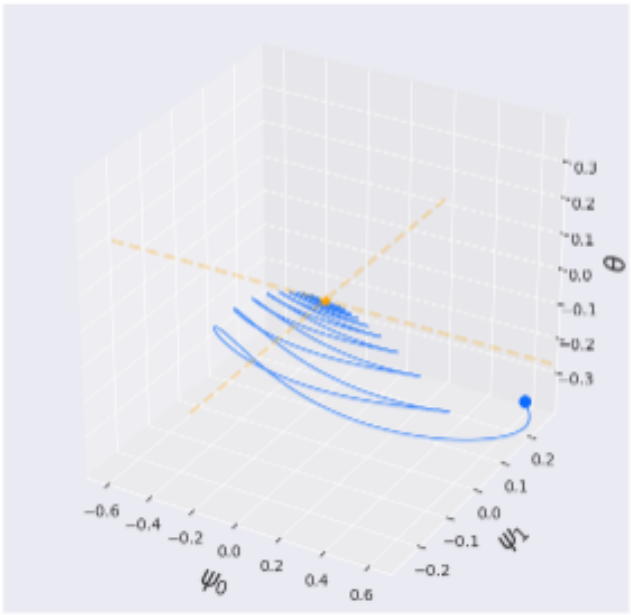


Figure 14: Final plot of Configuration 12: JSD with R1 gradient penalty and 0.01 regularization

⟶ JSD with R1 gradient penalty and 0.01 regularization converge slowly.

JSD+0.1 R1, $C(x) = \psi_0 x + \psi_1$, $Q(x) = \delta_\theta(x)$

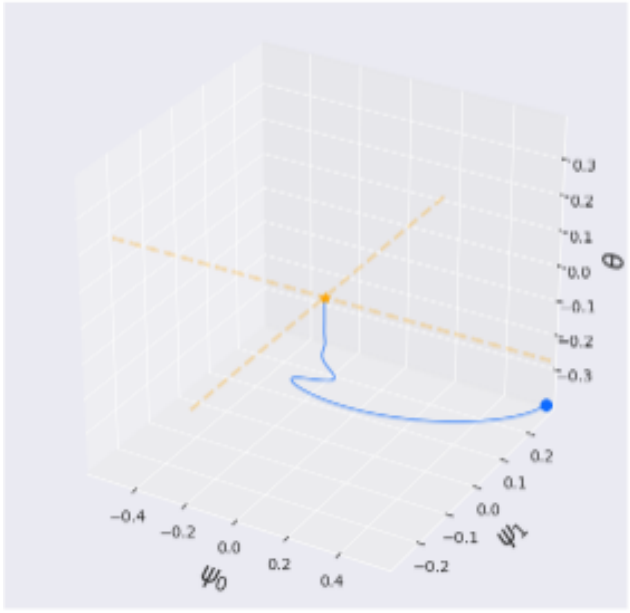


Figure 15: Final plot of Configuration 13: JSD with R1 gradient penalty and 0.1 regularization

⟶ JSD with R1 gradient penalty and 0.1 regularization converge well because the regularization term gets bigger than the above experiment.

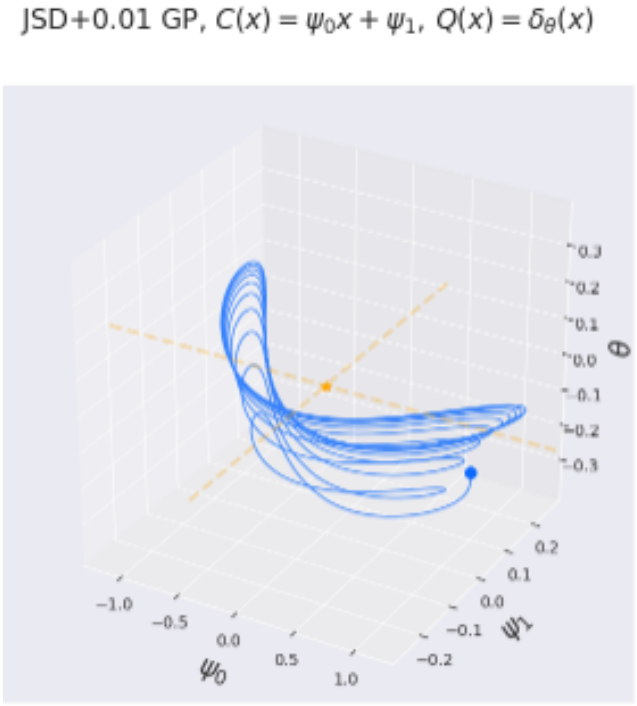


Figure 16: Final plot of Configuration 14: JSD with GP gradient penalty and 0.01 regularization

⟶ JSD with GP gradient penalty and 0.01 regularization diverge.

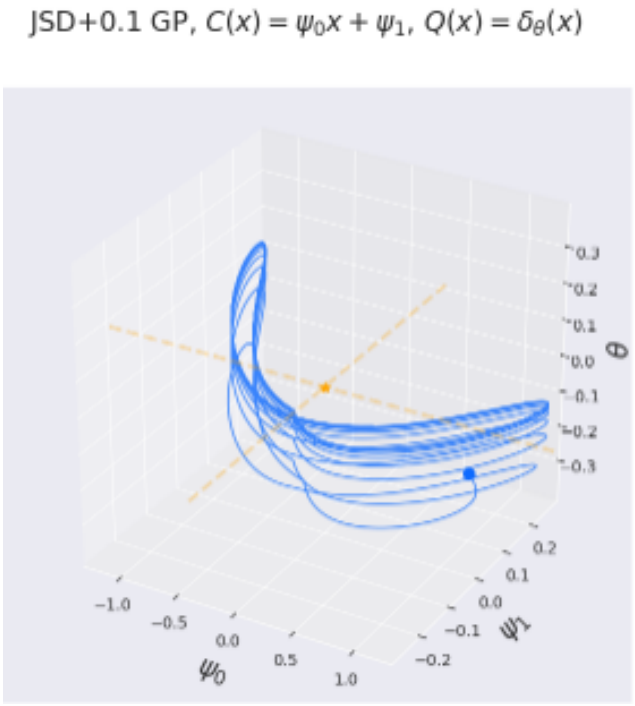


Figure 17: Final plot of Configuration 15: JSD with GP gradient penalty and 0.1 regularization

$\longrightarrow$ JSD with GP gradient penalty and 0.1 regularization diverge.

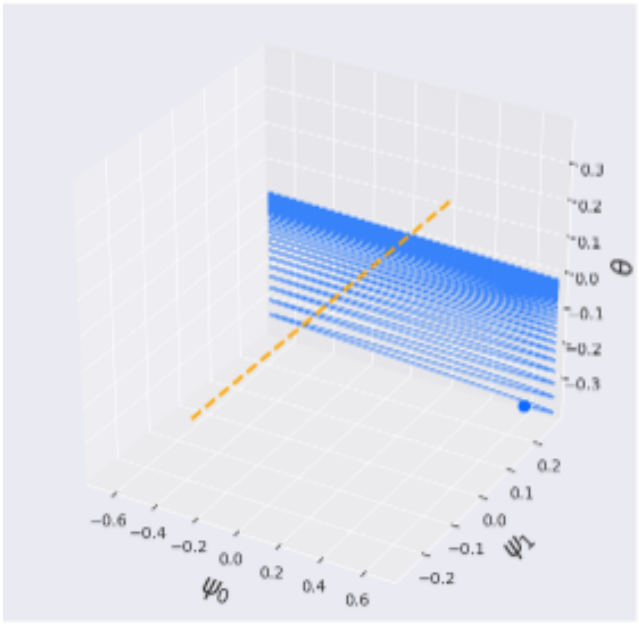


Figure 18: Final plot of Configuration 16: W1 with no gradient penalty

$\longrightarrow$ W1 with no gradient penalty fails to converge.

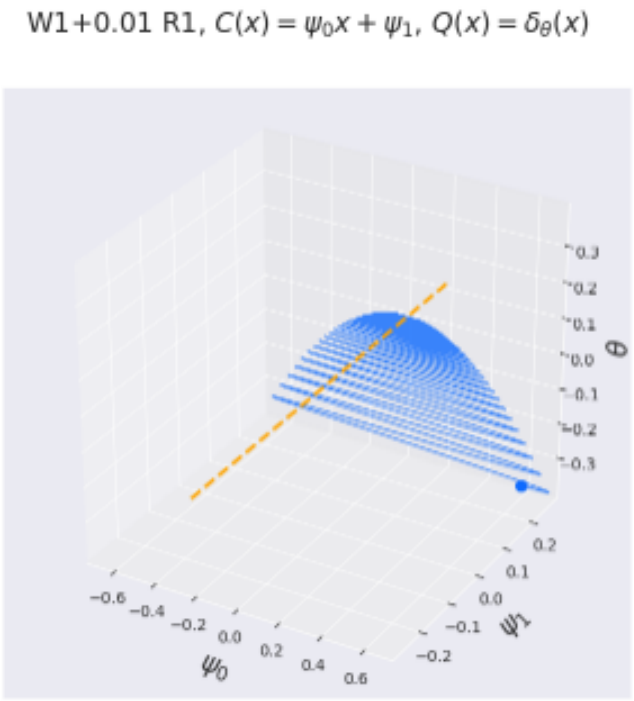


Figure 19: Final plot of Configuration 17: W1 with R1 gradient penalty and 0.01 regularization

$\longrightarrow$ W1 with R1 gradient penalty and 0.01 regularization converge slowly.

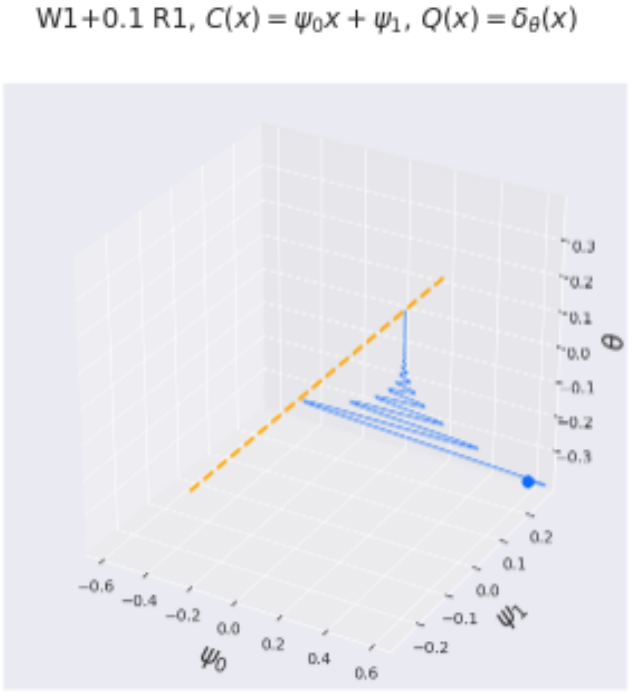


Figure 20: Final plot of Configuration 18: W1 with R1 gradient penalty and 0.1 regularization

$\longrightarrow$ W1 with R1 gradient penalty and 0.1 regularization converge well , quickly.

W1+0.01 GP, $C(x) = \psi_0 x + \psi_1$, $Q(x) = \delta_\theta(x)$

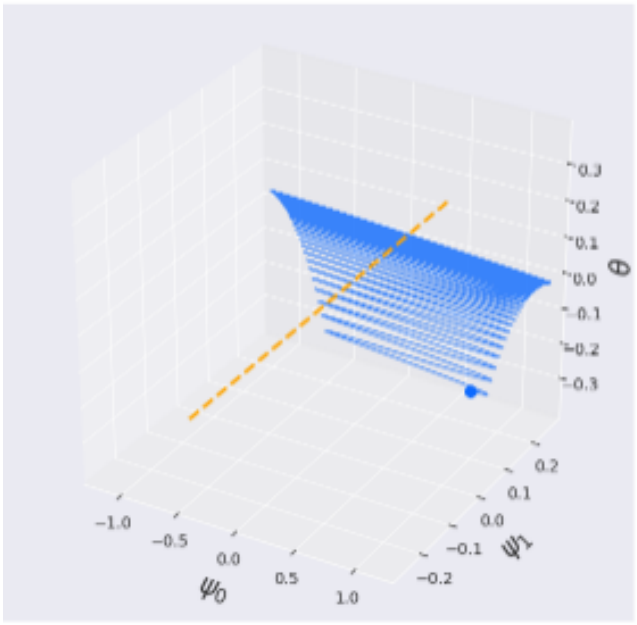


Figure 21: Final plot of Configuration 19: W1 with GP gradient penalty and 0.01 regularization

$\longrightarrow$ W1 with GP gradient penalty and 0.01 regularization diverge.

W1+0.1 GP, $C(x) = \psi_0 x + \psi_1$, $Q(x) = \delta_\theta(x)$

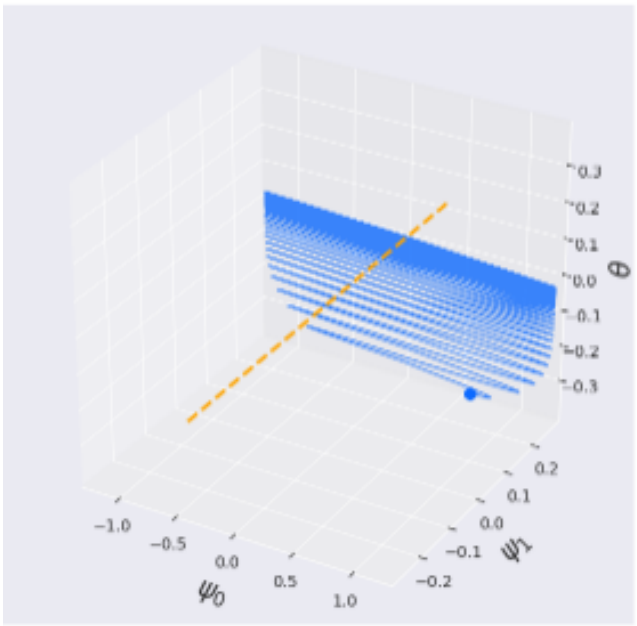


Figure 22: Final plot of Configuration 20: W1 with GP gradient penalty and 0.1 regularization

$\longrightarrow$ W1 with GP gradient penalty and 0.1 regularization fails to converge.

7. **(`report`, 25 pts)** We will use CIFAR10 dataset with predetermined generator and critic architectures, as well as the hyperparameter configurations denoted by the prefix '`cifar10-`'. In the report, you are requested to provide with final samples, Inception Score and Fréchet Inception Distance (FID) and compare between the 5 configurations. Notice that in this section, we have included spectral normalization as a way to constrain critic's Lipschitz constant. Also, provide with a plot over training steps for the approximated metric (JSD or W1) and FID for each run. Contrast the two quantities into the same plot. What do you observe? Which metric is more indicative of training's progress?[2]

Final samples for each configuration:

- **Configuration 1: cifar10 jsd 1**

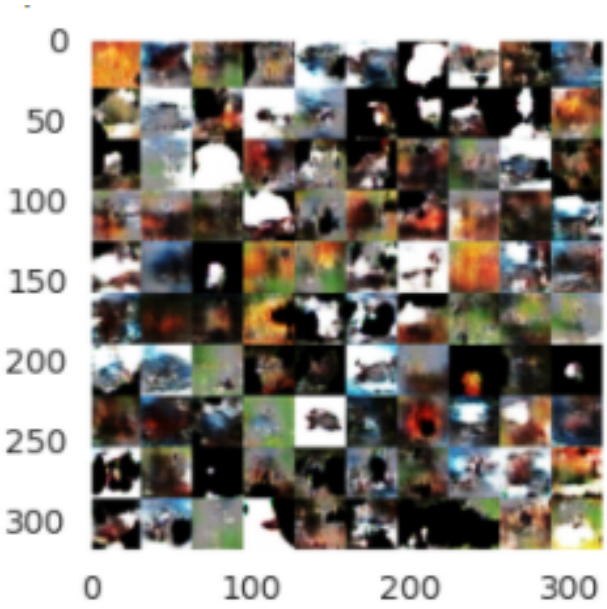


Figure 23: The final sample of the experiment cifar10 jsd 1

Final Inception Score = 3.483 and Final Fréchet Inception Distance (FID) = 124.472

- **Configuration 2: cifar10 jsd 2**

[2]Extra: Why? Find the answer at: Towards Principled Methods for Training Generative Adversarial Networks

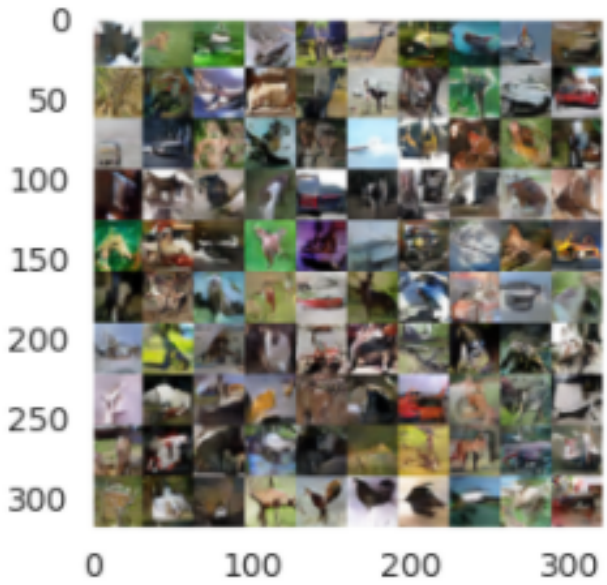


Figure 24: The final sample of the experiment cifar10 jsd 2

Final Inception Score = 4.873 and Final Fréchet Inception Distance (FID) = 40.891

- **Configuration 3: cifar10 jsd 3**

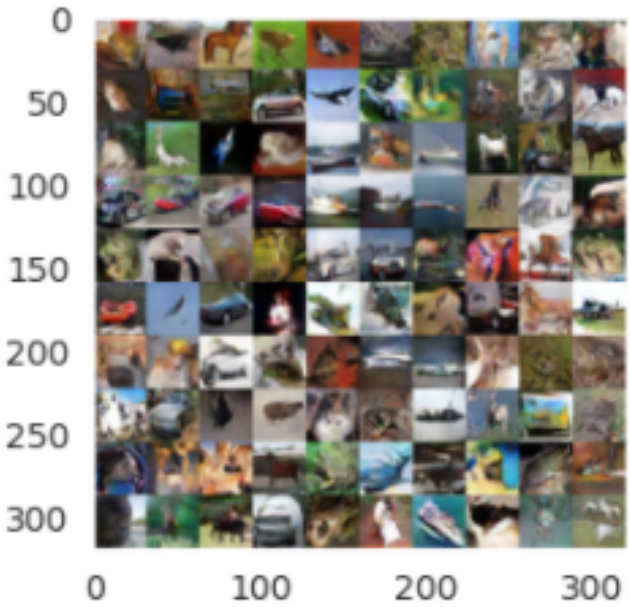


Figure 25: The final sample of the experiment cifar10 jsd 3

Final Inception Score = 5.424 and Final Fréchet Inception Distance (FID) = 33.065

- **Configuration 4: cifar10 w1 2**

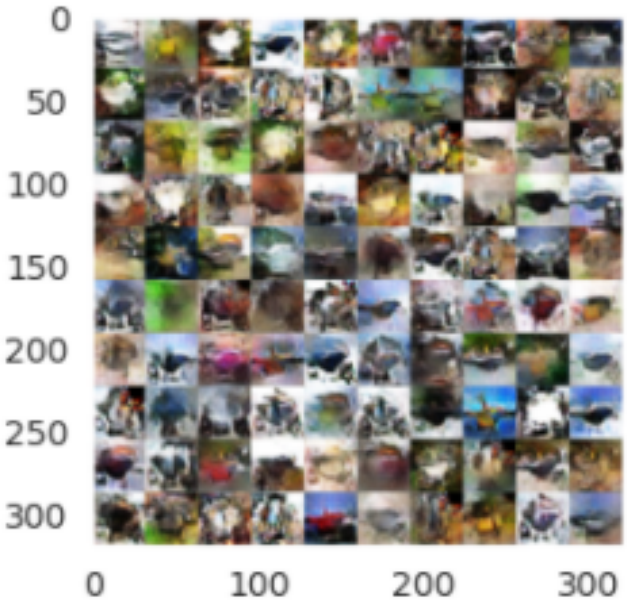


Figure 26: The final sample of the experiment cifar10 w1 2

Final Inception Score = 3.642 and Final Fréchet Inception Distance (FID) = 84.269

- **Configuration 5:cifar10 w1 3**

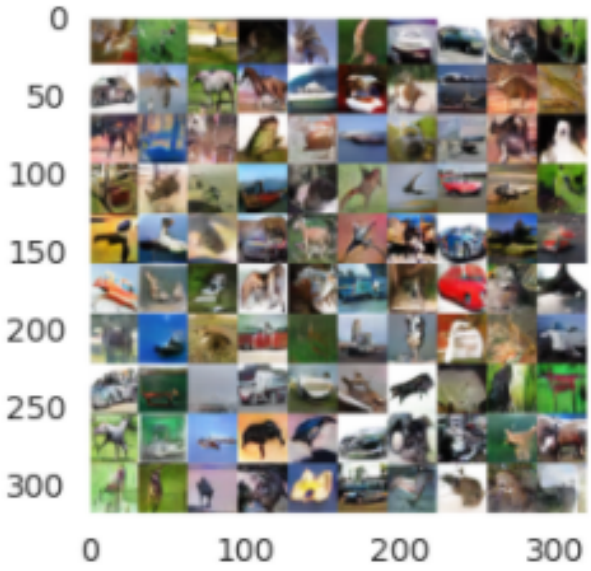


Figure 27: The final sample of the experiment cifar10 w1 3

Final Inception Score = 5.812 and Final Fréchet Inception Distance (FID) = 24.911

Comparision:
Comparing W1 and JSD experiments, we can notice that with W1 loss we get better results, better

generation of images. For JSD loss, we get blurry images especially for the first configuration where we did not add a regularization, high FID value and small IS. However, when we add gradient penalty, we get some better results for the configurations 2 and 3. The best results are using W1 loss with R1 gradient penalty.

Plot over training steps for the approximated metric (JSD or W1) and FID for each run:

- **Configuration 2: cifar10 jsd 2**

Figure 28: Plot JSD over training steps

Figure 29: Plot FID over training steps

Figure 30: Plot over training steps for the approximated metric JSD and FID

- **Configuration 3: cifar10 jsd 3**

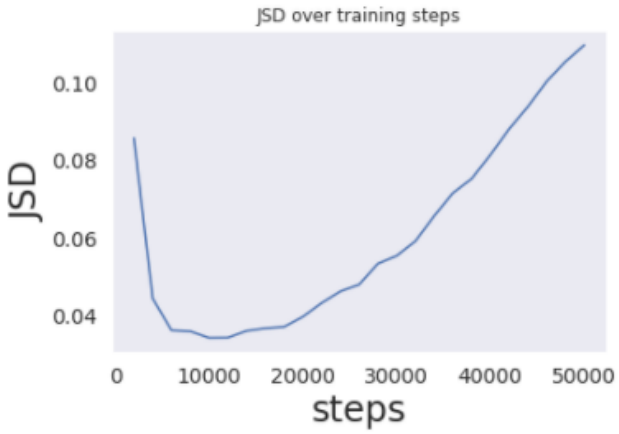


Figure 31: Plot JSD over training steps

Figure 32: Plot FID over training steps

Figure 33: Plot over training steps for the approximated metric JSD and FID

- **Configuration 4:cifar10 w1 2**

Figure 34: Plot W1 over training steps

Figure 35: Plot FID over training steps

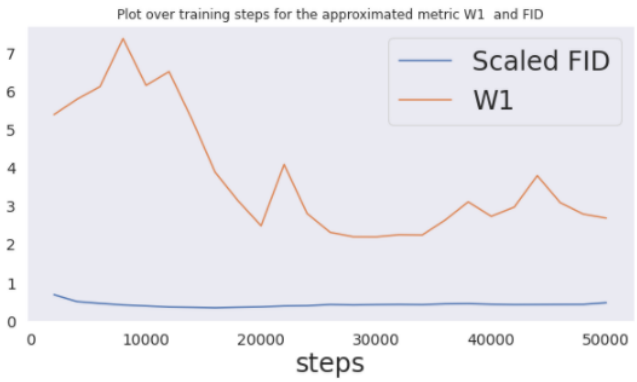


Figure 36: Plot over training steps for the approximated metric W1 and FID

- **Configuration 5:cifar10 w1 3**

Figure 37: Plot W1 over training steps

Figure 38: Plot FID over training steps

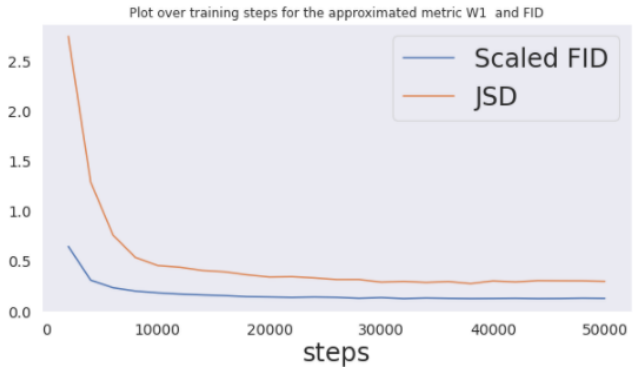


Figure 39: Plot over training steps for the approximated metric W1 and FID

Observation:
For the first three configurations, we notice that the JSD loss increases with training steps, i.e, more the training steps, jsd increases and FID decreases.

For the two last experiments, we can see that W1 loss decreases with training steps and even becomes stable for the last configurations.

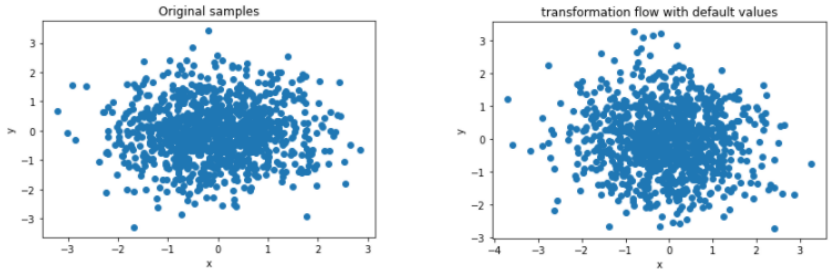
Which metric is more indicative of training's progress?
W1 is more indicative of training's progress

Problem 3

1. **(report, 2 pts)** Test your function by applying it to samples from a $N(\mathbf{0}, \boldsymbol{I})$ for different value of $\boldsymbol{u}$, $\boldsymbol{w}$, b. Describe what the transformation correspond to and add plots to show the effect of different transformations.

   Answer:
   For this experiment, we test the function by applying it to 1000 samples from a $N(\mathbf{0}, \boldsymbol{I})$ for different value of $\boldsymbol{u}$, $\boldsymbol{w}$, b.

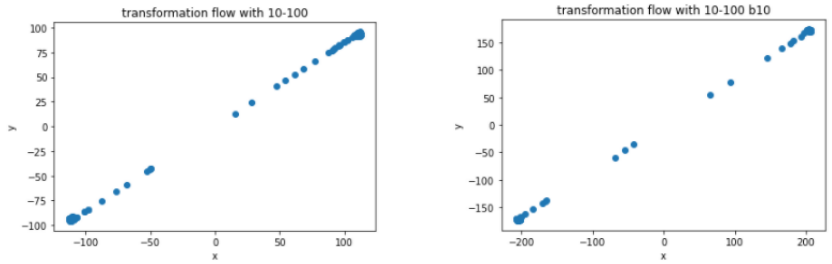
   In the figure below, we present the original samples from $N(\mathbf{0}, \boldsymbol{I})$ and the transformed samples using transformation planar with the default values of $\boldsymbol{u}$, $\boldsymbol{w}$, b.



(a) Original (b) Planar flow default values

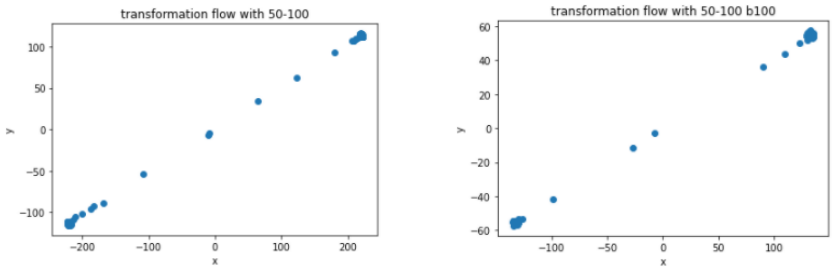
Figure 40: Original vs Transformation flow default values

   There is no major difference between the two subplots, therefore we change the default values of $\boldsymbol{u}$, $\boldsymbol{w}$, b with bigger ones. The results are shown in the figures below:



(a) (b)

Figure 41: For the $\boldsymbol{u}$, $\boldsymbol{w}$ values from a torch.normal(10, 100, (1, 2)) but (a) with the same default value of b and (b) $b = 10$



(a) (b)

Figure 42: For the $\boldsymbol{u}$, $\boldsymbol{w}$ values from a torch.normal(50, 100, (1, 2)) but (a) with the same default value of b and (b) $b = 100$

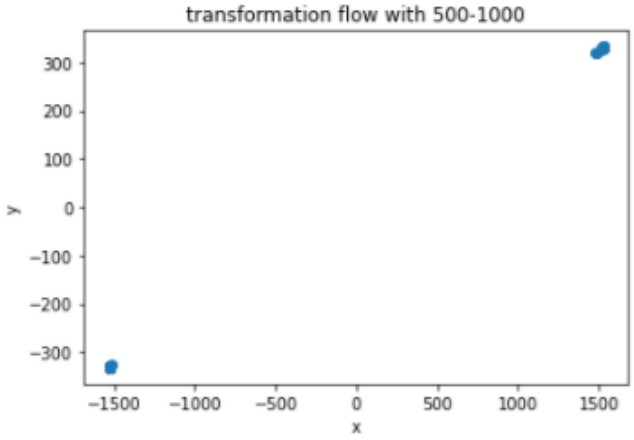


Figure 43: For the $\boldsymbol{u}$, $\boldsymbol{w}$ values from a torch.normal(500, 1000, (1, 2)) with the same default value of b

For each configuration, we get a different representation of the samples and a different scale on the axis of the plot. The higher the values of $\boldsymbol{u}$, $\boldsymbol{w}$ and b, the clearer the representation is. It seems that the transformation aims to make groups of samples. The bigger the parameters values are , we get two groups.

2. **(report, 4 pts)** Calculate the log determinant of the Jacobian of a planar transformation using the matrix determinant lemma. What is the complexity (using the big $\mathcal{O}$ notation) of evaluating it?

Answer:
We have:

$$f(\boldsymbol{z}) = \boldsymbol{z} + \boldsymbol{u}h(\boldsymbol{w}^\top \boldsymbol{z} + b)$$

The jacobian can be expressed as follow:

$$\frac{\partial f(\boldsymbol{z})}{\partial \boldsymbol{z}} = 1 + \boldsymbol{u}\frac{\partial h(\boldsymbol{w}^\top \boldsymbol{z} + b)}{\partial \boldsymbol{z}}$$

$$\frac{\partial h(\boldsymbol{w}^\top \boldsymbol{z} + b)}{\partial \boldsymbol{z}} = h'(\boldsymbol{w}^\top \boldsymbol{z} + b)\boldsymbol{w}^\top$$

$$\frac{\partial f(\boldsymbol{z})}{\partial \boldsymbol{z}} = 1 + \boldsymbol{u}h'(\boldsymbol{w}^\top \boldsymbol{z} + b)\boldsymbol{w}^\top$$

Using the matrix determinant lemma:

$$\det(\boldsymbol{A} + \boldsymbol{V}\boldsymbol{W}^T) = \det(I + \boldsymbol{W}^T\boldsymbol{A}^{-1}\boldsymbol{V})\det(\boldsymbol{A})$$

$$|\det \frac{\partial f(\boldsymbol{z})}{\partial \boldsymbol{z}}| = |\det(I + \boldsymbol{u}h'(\boldsymbol{w}^\top \boldsymbol{z} + b)\boldsymbol{w}^\top)|$$

However, the lemma above is a general case. Here we have $\boldsymbol{A} = I$ [special case proof]. So:

$$|\det \frac{\partial f(\boldsymbol{z})}{\partial \boldsymbol{z}}| = |\det I + h'(\boldsymbol{w}^\top \boldsymbol{z} + b)\boldsymbol{u}\boldsymbol{w}^\top)| = |1 + h'(\boldsymbol{w}^\top \boldsymbol{z} + b)\boldsymbol{w}\boldsymbol{u}^\top|$$

Therefore, the log determinant of the Jacobian of a planar transformation:

$$\log|\det(\frac{\partial f(\boldsymbol{z})}{\partial \boldsymbol{z}}| = \log|1 + h'(\boldsymbol{w}^\top \boldsymbol{z} + b)\boldsymbol{w}\boldsymbol{u}^\top|$$

The complexity of evaluating is $\mathcal{O}(d)$ (by using the matrix determinant lemma, maintain a triangular Jacobian, the determinant becomes easy to compute)

3. **(report, 2 pts)** Let $U_0 \sim N(\mathbf{0}, \boldsymbol{I})$ be the base distribution. Express the induced log density $\ln q_k(\boldsymbol{u}_k)$ obtained by applying a flow constituted of k planar transformation, where $\boldsymbol{u}_k = f_k \circ f_{k-1} \circ \cdots \circ f_1(\boldsymbol{u}_0)$. What is the complexity of evaluating the log determinant of the normalizing flow?

   Answer:

   For single flow, where $\boldsymbol{u} = f(\boldsymbol{u}_0)$ we have:

$$q(\boldsymbol{u}) = q(\boldsymbol{u}_0)|\det \frac{\partial f^{-1}}{\partial \boldsymbol{u}}|$$

   Applying the inverse function theorem $J_{F^{-1}}(q) = [J_F(p)]^{-1}$:

$$q(\boldsymbol{u}) = q(\boldsymbol{u}_0)|\det \frac{\partial f}{\partial \boldsymbol{u}_0}|^{-1}$$

   and because $\ln(X^{-1}) = -\ln(X)$:

$$\ln q(\boldsymbol{u}) = \ln q(\boldsymbol{u}_0) - \ln|\det \frac{\partial f}{\partial \boldsymbol{u}_0}|$$

   For the case where we have $\boldsymbol{u}_k = f_k \circ f_{k-1} \circ \cdots \circ f_1(\boldsymbol{u}_0)$: applying the chain rule (inverse function theorem), we get:

$$\ln q_k(\boldsymbol{u}_k) = \ln q_0(\boldsymbol{u}_0) - \sum_{i=1}^{k} \ln|\det \frac{\partial f_i}{\partial \boldsymbol{u}_{i-1}}|$$

   The density $\ln q_k(\boldsymbol{u}_k)$ obtained by successively transforming a random variable $\boldsymbol{u}_0$ with distribution q_0 through a chain of K transformations. [source]

   Recall results from the previous question:

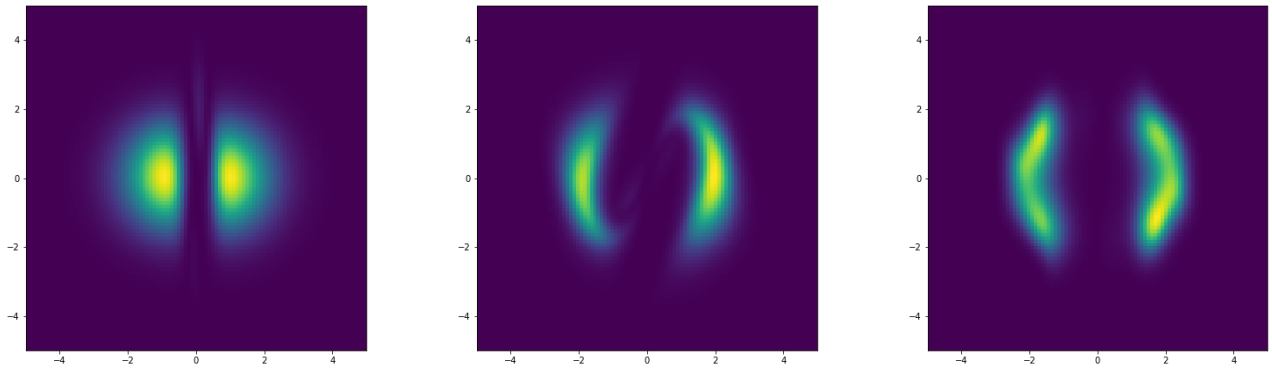
$$\ln q_k(\boldsymbol{u}_k) = \ln q_0(\boldsymbol{u}_0) - \sum_{i=1}^{k} \ln|1 + h'(\boldsymbol{w}^\top \boldsymbol{u}_{i-1} + b)\boldsymbol{w}\boldsymbol{u}_i^\top|$$

   the complexity of evaluating is $\mathcal{O}(kd)$

4. **(`report`, 4 pts)** Now that the flow is completed, train the model on the dataset '`data_arcs`' and '`data_sine`' with $k = \{2, 8, 32\}$ for 20000 iterations using the function '`launch_experiments_from_sa` from '`train.py`'. For each setting of k and dataset, plot the learned density (these plots are automatically generated by the code) and report the negative log likelihood. Briefly comment the effect of k.

   Answer:

   The figure below present the plot the learned density on the dataset '`data_arcs`'for each setting of k and the table present the negative log likelihood obtained for each configurations.



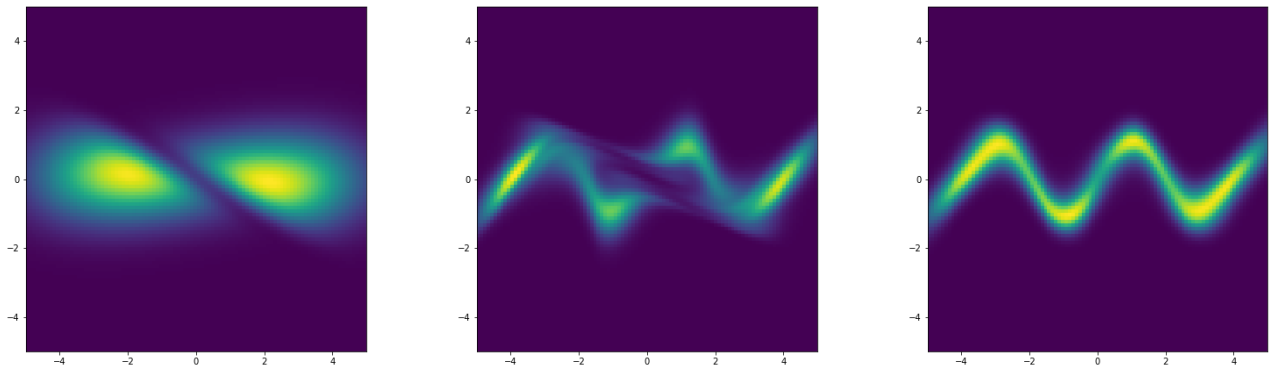
(a) k=2 (b) k=8 (c) k=32

Figure 44: Learned density data arcs with different k values

k	Loss
2	3.4643
8	2.7259
32	2.7233

Table 1: Loss for each k value data arcs

   The figure below present the plot the learned density on the dataset '`data_sine`'for each setting of k and the table present the negative log likelihood obtained for each configurations.



(a) k=2 (b) k=8 (c) k=32

Figure 45: Learned density data sine with different k values

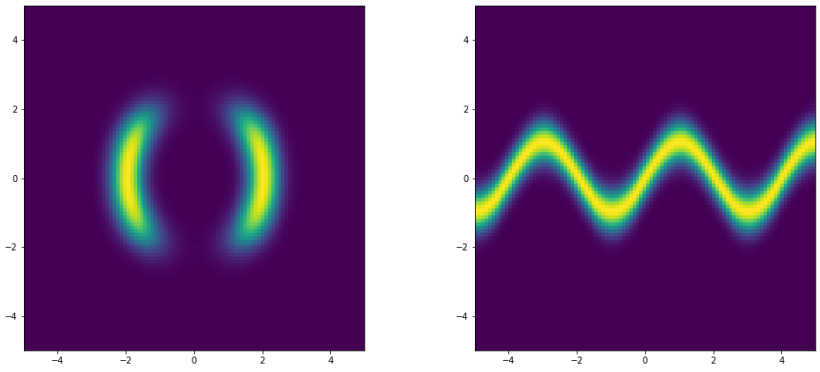
k	Loss
2	3.5468
8	3.0403
32	2.9112

Table 2: Loss for each k value data sine

The effect of k: As we increase the k value, We see an improvement in the approximation quality of the data, also, the loss decrease.

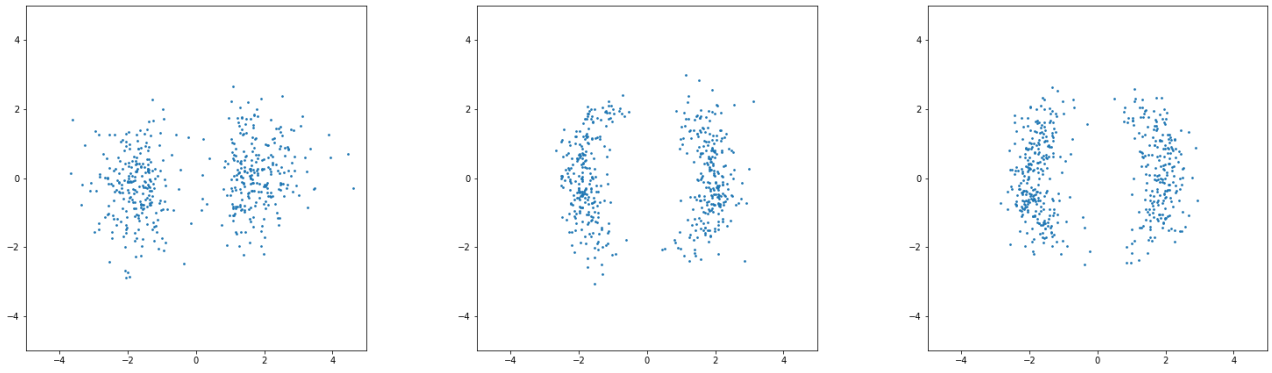
5. **(`report`, 2 pts)** Train the model on the density '`density_arcs`' and '`density_sine`' with $k = \{2, 8, 32\}$ for 20000 iterations using the function '`launch_experiments_from_density`' from '`train.py`'. For each setting of k, report the plots of the samples generated by the normalizing flow (these plots are automatically generated by the code).

   Answer:



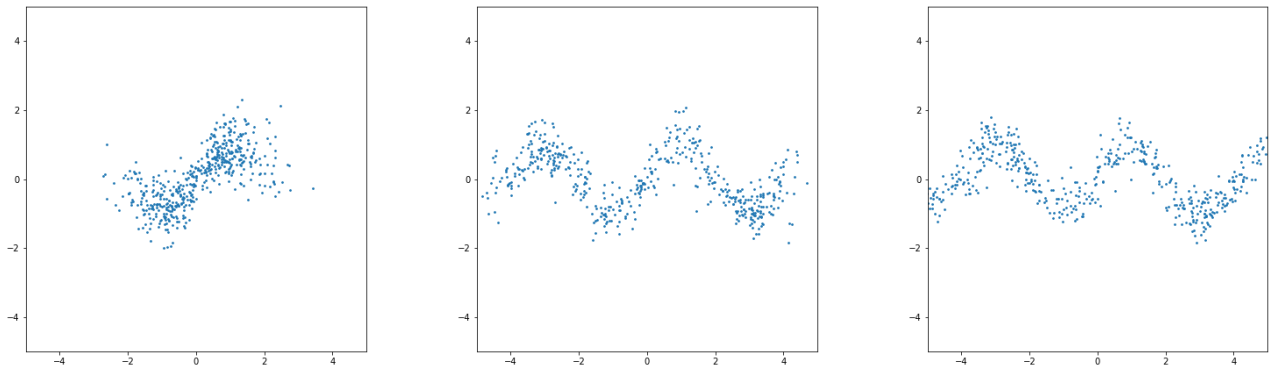
(a) density arcs (b) density sine

Figure 46: The densities



(a) k=2 (b) k=8 (c) k=32

Figure 47: samples arcs with different k values



(a) k=2 (b) k=8 (c) k=32

Figure 48: samples sine with different k values

6. **(`report`, 4 pts)** Normalizing flows can also be used as a posterior of variational autoencoders. Write the loss of a VAE using a planar flow. What is the advantage of using a normalizing flow as the posterior of a VAE?

   Answer:

   - Loss of a VAE using a planar flow:
     To obtain the loss, we replace the posterior in the standard VAE loss with planar flow, we get

$$\iint \tilde{p}(x)q(u)\log\frac{\tilde{p}(x)q(u)}{q(x|F_x(u))q(F_x(u))\left|\det\left[\frac{\partial F_x(u)}{\partial u}\right]\right|}dudx$$

     from [source]
   - The advantage of using a normalizing flow as the posterior of a VAE:
     VAE with normalizing flows addresses the problem of variational inference, that of simplifying the posterior distribution too much. With normalizing flows are able to transform an easy paramaterizable base distribution, by passing it through a series of transformations, in a more complex approximation for the posterior distribution.