Exercise 1 - Generative Modelling

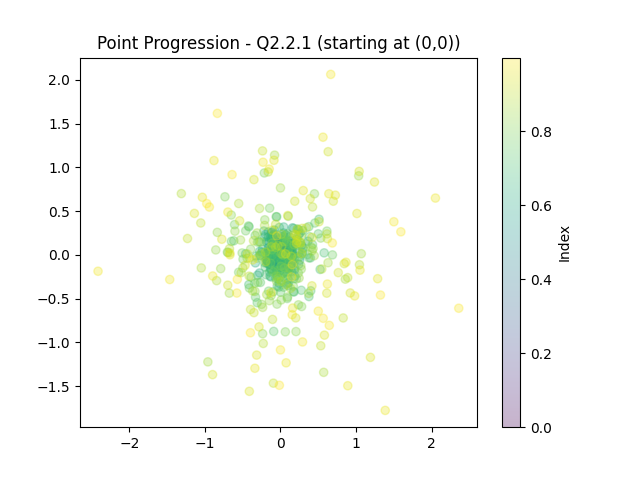
67912 - Advanced Course in Machine Learning

Yuval kenneth – 208435834

2.2.2 – unconditional:

1.

I chose (0,0) as the starting point:



2.

A picture containing text, screenshot, plot, diagram

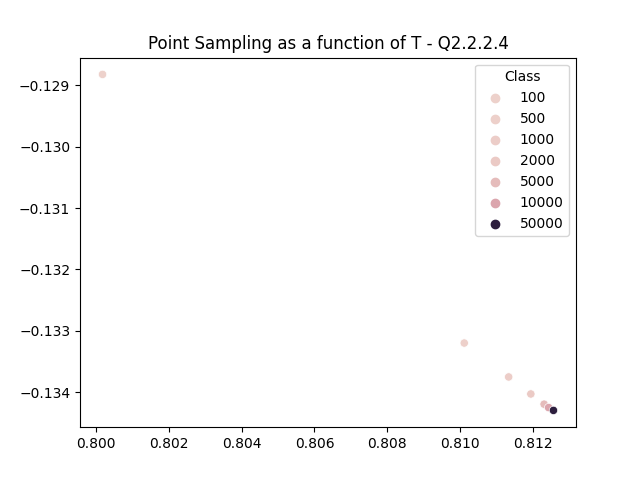
Description automatically generated

3.

A picture containing pattern, symmetry

Description automatically generated

4.



In this scatter plot, we can see that sampling (with the same seed) using increasing sampling steps, converges as T increases.

5.

A picture containing text, pattern

Description automatically generated

In the presented scatter plots, I wanted to see how sampling works using the same noise scheduler, noisier ones, and a less noisy one. F(1) = 1 for all used schedulers. It is visible that using the same noise schedulers used in training, spreads the samples more uniformly, matching the training data. Also, it is visible that using a less noisy scheduler, the data is more dense in the center of the square, as opposed to the noisier schedulers, that “pushed” the samples to the edges of the square. In summary, in order to sample exactly like the training data, I would use the same noise scheduler, but if I wanted to control how my samples spread, I will choose a scheduler based on those findings.

6.

Using DDIM sampling resulted in the same output every time as expected. I implemented another sampler, with the only change is in eq.8 in the exercise. In each sampling step, I added a small amount of noise (sampled from normal distribution and multiplied by a small factor (<1).

I was thinking that the added noise will still result in the same “Area” (populated areas from the training set), but will add a little variance to the outputs from the same noise input.

A picture containing screenshot, diagram, plot, text

Description automatically generated

2.2.2 – conditional:

1.

A picture containing screenshot, text, colorfulness

Description automatically generated

I divided the square into 5 classes, such that each class width is 0.4 on the x-axis.

#added loss over batches for conditioned model

A picture containing text, screenshot, plot, diagram

Description automatically generated

2.

In order to insert the conditioning, I implemented a similar denoiser (NN), only this time I added the class of the input as another input, and added an embedding layer to the network. All other functions (such as sampling and estimation) were updated to match both denoisers.

3.

A picture containing text, screenshot, line, diagram

Description automatically generated

It is visible that the points reach their designated class. (class 0 and 4 reach the line between their neighboring classes)

4.

A picture containing screenshot, text

Description automatically generated

5.

Overall, it seems that the spatial distribution is very similar to the input distribution. However, it seems that classes 0 and 4 are more difficult than the others. A possible explanation is that from normal distribution, it is less likely to sample an initial point in those areas, so maybe the denoiser is less likely to succeed in reversing the noise to those classes.

6.



From the first two points we can see that the same point, once assigned to the right class and the other to a different one, the probability of the former is much higher. The least likely point is far away from the square as expected. It seems as if points in the square that are assigned to their class have similar probabilities, as can be expected given that the data was uniformly sampled from the square and divided equally (space-wise) between the classes.

3.2.4 – GPT-2:

1.

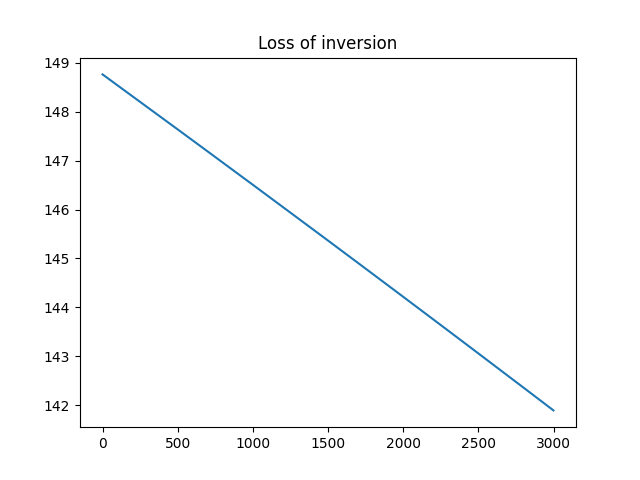
A picture containing plot, line, diagram, screenshot

Description automatically generated

2.

The inversion process did not output a correct input vector. In my opinion there are number of possible explanations. First, it is possible that the model did not learn patterns in the training set, whether it’s because the training was not long enough, or the model itself isn’t strong enough. Another explanation could be that there is no similar sentence in the text (the sentence itself and most of the words are not part of the training set). Lastly, I don’t think that using an inversion process like that can guarantee success even when trying to output a single word. It depends on the training data and the learning process and there is no guarantee that a specific word will get a positive probability to be the next word no matter what the context it, so trying to output a specific sentence should be possible if the pattern of that sentence is dominant in the training data, or if that sentence is very common.

Inversion loss:



3.

A picture containing text, screenshot, colorfulness

Description automatically generated

4.

A picture containing text, screenshot, colorfulness

Description automatically generated

5.

