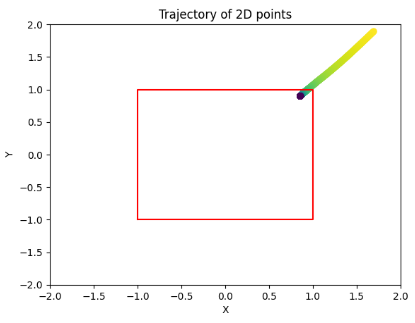
Ex1 Advanced Course in Machine Learning

Amitai Ovadia 312244254

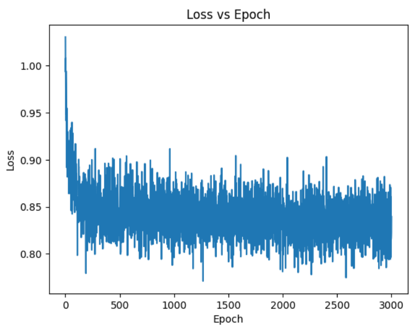
Part 1

Unconditional Diffusion Model:

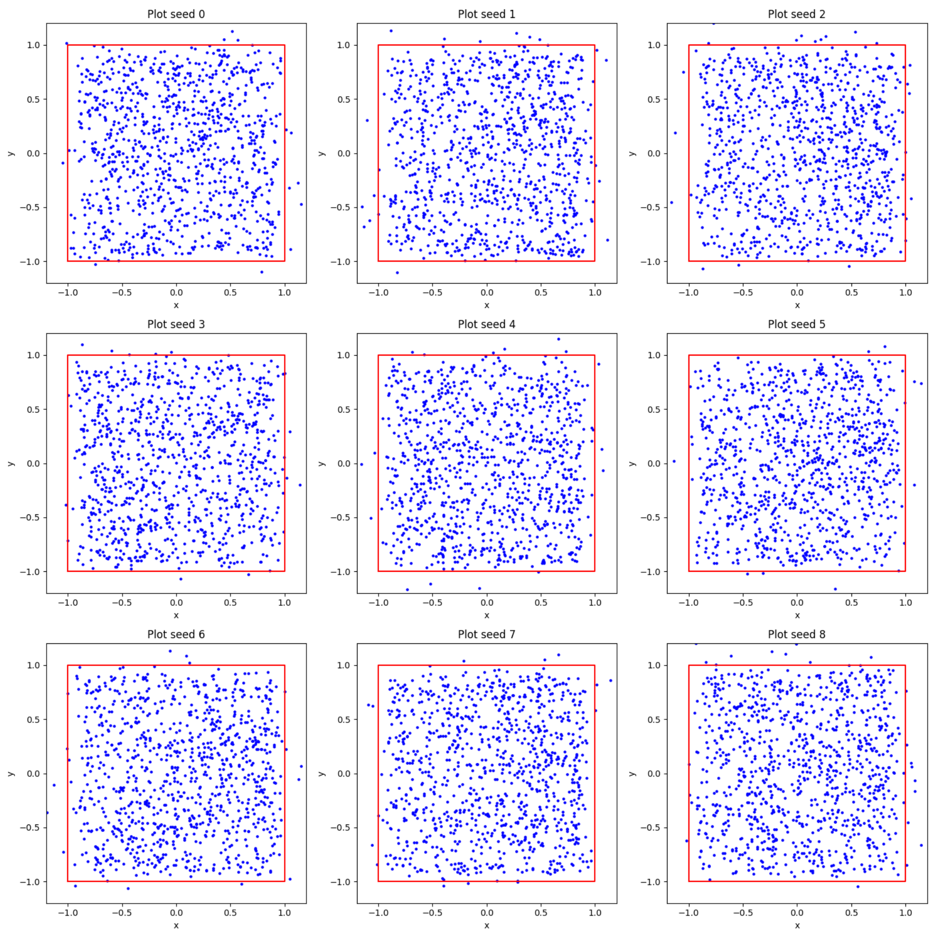
1. The point started at (1.7, 1.9) and ended inside the square. Done using the unconditional denoiser model. The pace of the diffusion slowed as the point neared the distribution area.



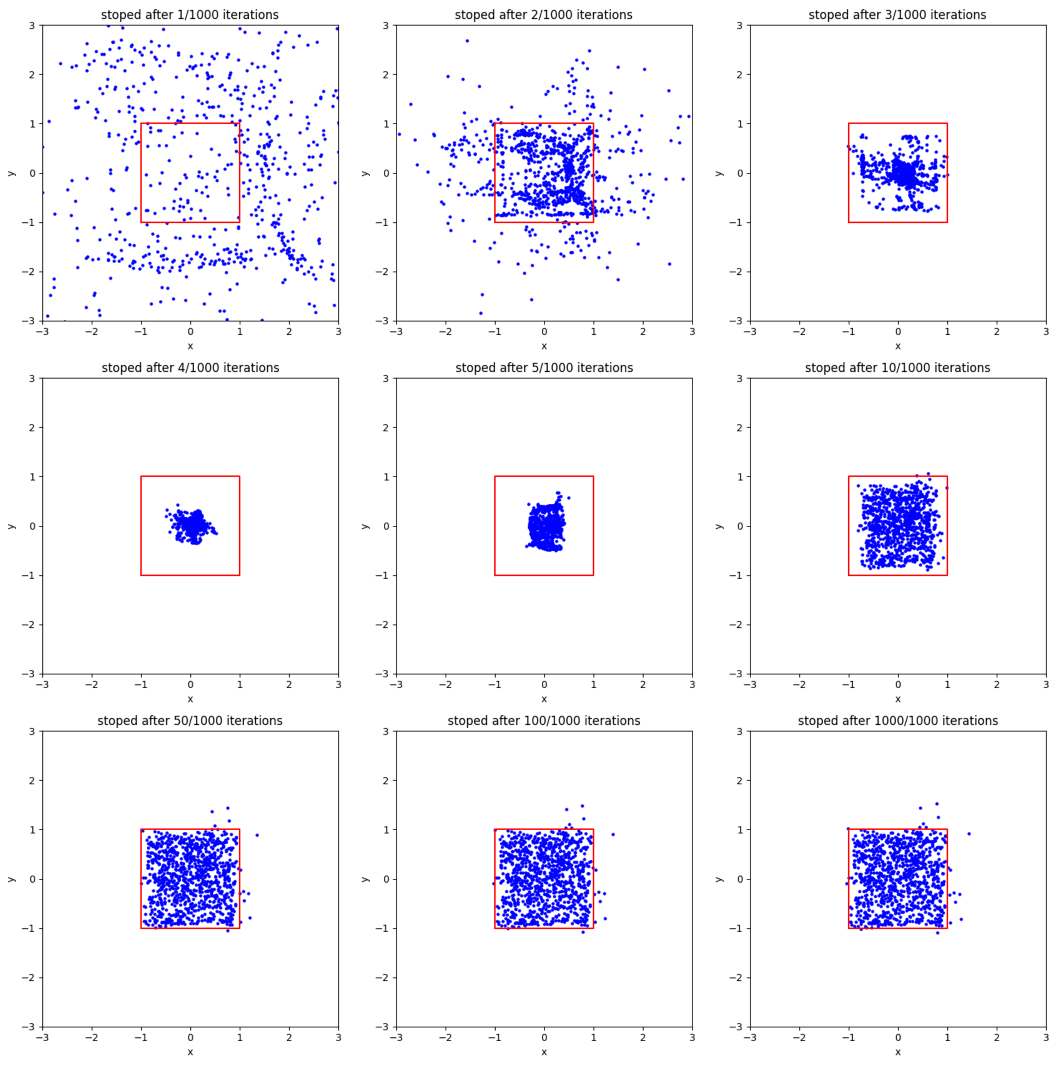
2. Loss of training as a function of epochs. At first the loss drops, as it is not so difficult to estimate noise of points outside the train set distribution, but the loss gets noisy as the points are already inside the train set distribution.



3. 3X3 figure of sampling of different seeds (0 -> 8))
started at random N(0,1) and then applied DDIM sampling. The points sampled are different of course due to the random initialization using different seeds.

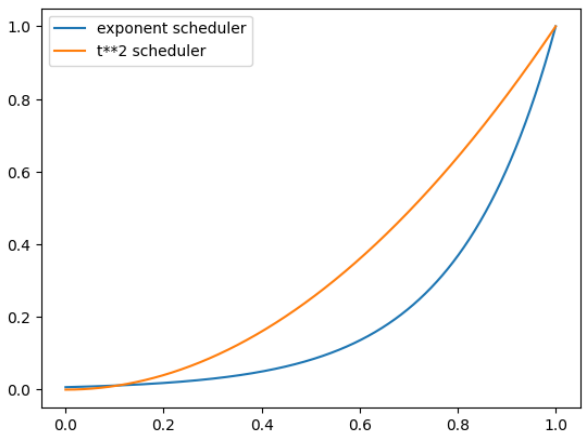


4. Here I tried 9 different t's:

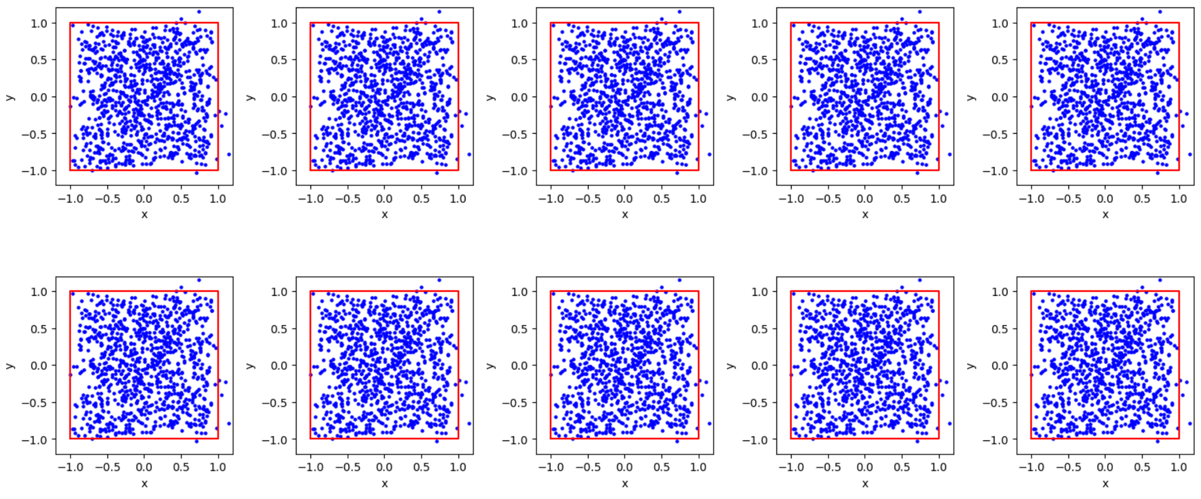


We can see that when T is very small (1,2), then the denoiser couldn't really estimate the noise clearly and not all the samples fell within the -1,1 square of the training set data. Then for T = (3,4,5) the points were pushed to the mean of the training set (more coarse noise estimation), and for larger and larger T's the points are much closer to the original dataset distribution. We can deduce that when dt is smaller and we have more steps, there is a much more thorough exploration of the data distribution and a better result.

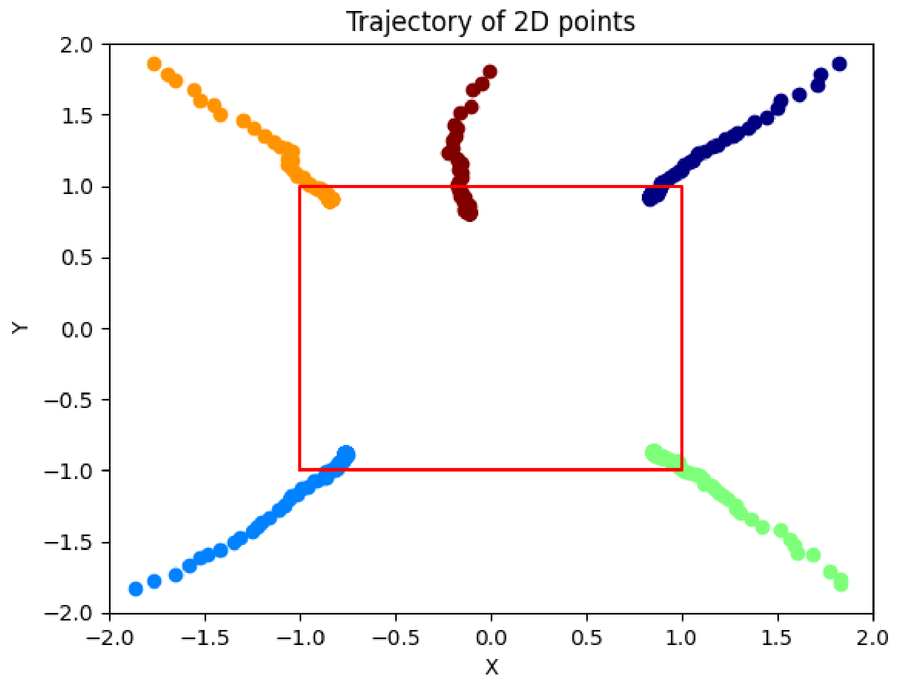
5. Here is the original sampler vs the t^2 sampler. The rules of thumb you find for an effective sampler are that we want to do the diffusion process in a coarse to fine manner: start in large steps and decrease them as you get closer to the training set distribution, so the resulting sampling distribution will look as close as possible to the dataset distribution.



6. As can be seen, the sampling is deterministic using the DDIM method.

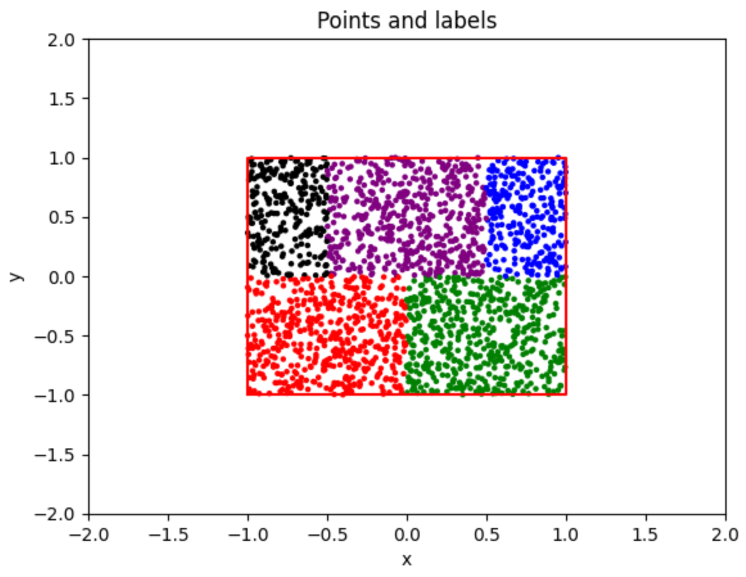


I implemented DDPM sampling, according to the equations presented in class, using lambda = 0.01 and T=100.
Here are trajectories of 5 points on the same figure.
The trajectories are indeed noisy, and so nondeterministic.

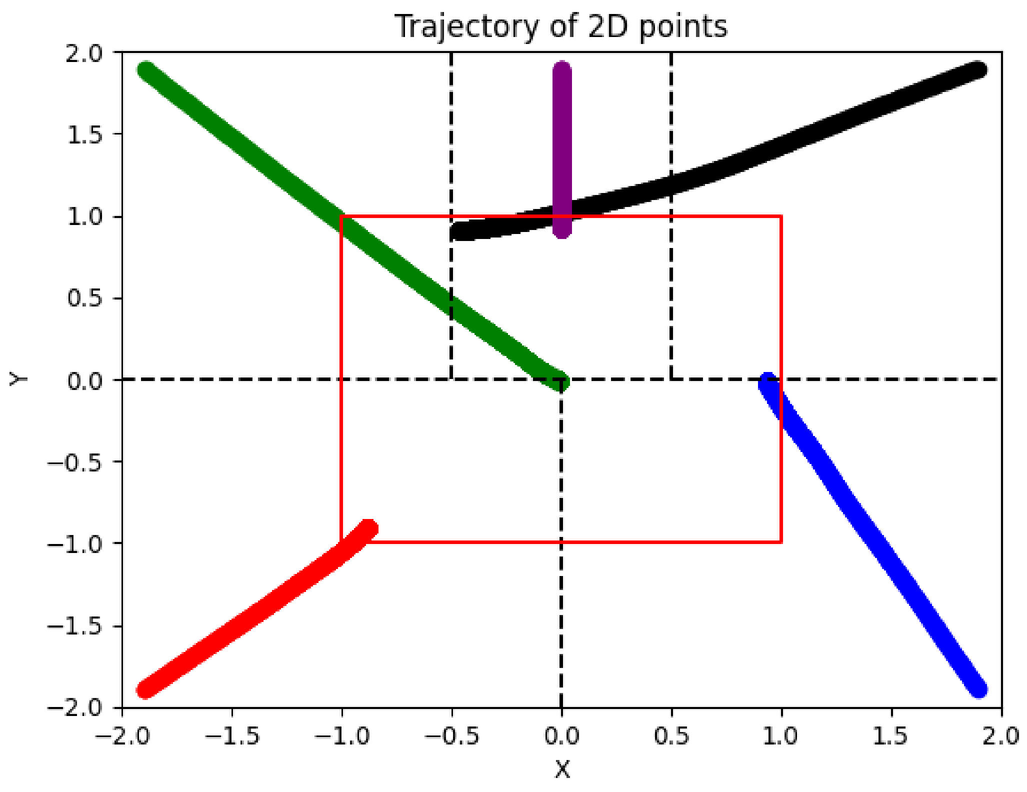


Conditional Diffusion Model:

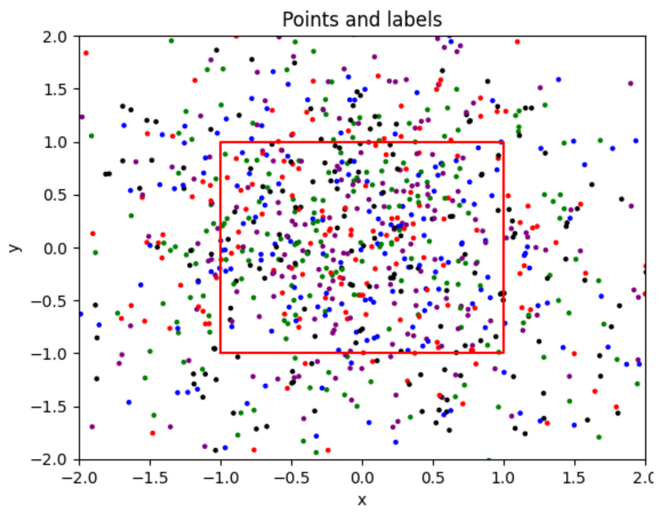
1. The training dataset plotted according to the different labels.

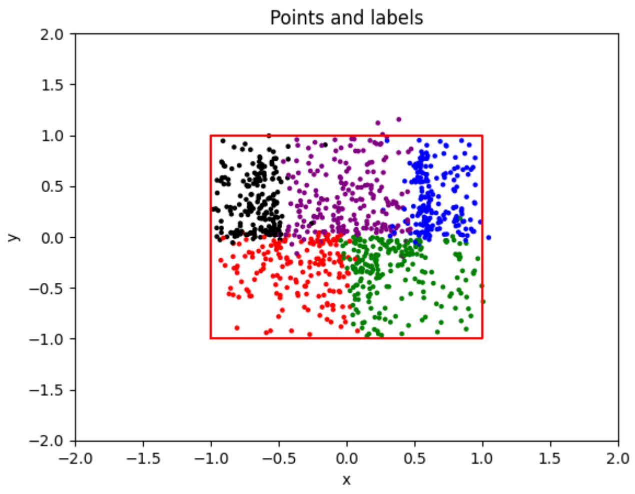


2. In order to insert the conditioning I used a different neural network model for the denoiser, which now has the (x,y,t) as well as a 5 dimensional embedding of the class label. so the input to the network is now an 8 dimensional vector (instead of 3). Apart from that, no equations had to be changed.
3. A plot of the points trajectories: we can see that they reach (or almost reach) their class boundary as can be seen in the figure in question 1.
   Interestingly, they all reached the boundary of their class, and stopped close to the boundary, and didn't go to a random location inside the class boundary like you would expect from the dataset distribution.



4. Sampled points coloring them by their classes: before and after the sampling process:

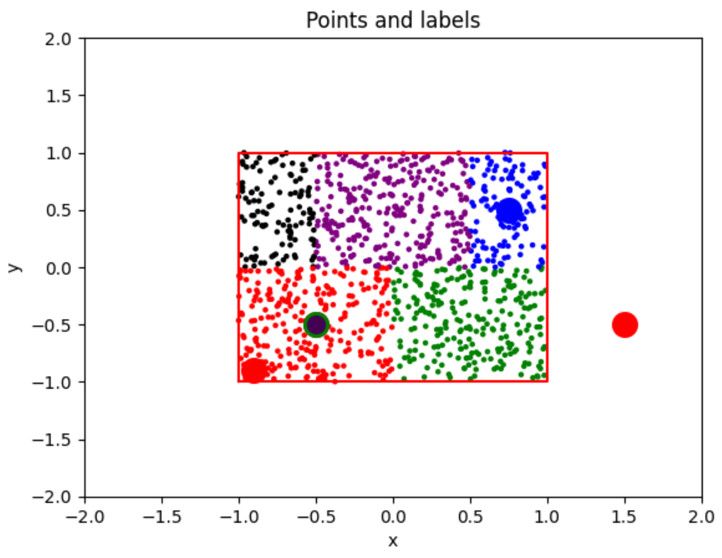




5. The results are pretty good: the classes are well separated from each other. but their distribution inside their respective areas does not resemble the original data distribution. For some reason most of the points are closer to the boundaries (exaclty like question 3 results), which are not the high probability areas (closer to the middle of the class area).
   Another finding is that the points are more concentrated on the boundaries with other classes, and less so in the outer boundaries (the +-1 x,y axis)

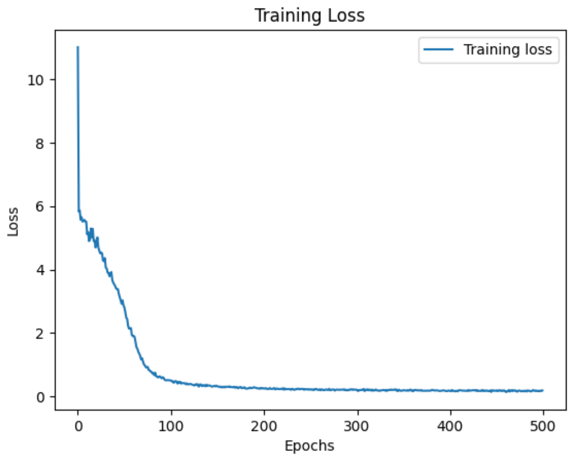
6. The chosen points:
   P1 (-0.5,-0.5) label 'red' : -0.2×10^{6} (in distribution mean)
   P2 (-0.5,-0.5) label 'green' : -2.8×10^{6} (out of distribution)
   P3 (1.5, -0.5) label 'red' : -7.7×10^{6} (don't match distribution)
   P4 (0.75,-0.5) label 'blue' : -0.5×10^{6} (in distribution mean)
   P5 (-0.9,-0.9) label 'red' : -0.56×10^{6} (far from mean of distribution)

   We can see that the closest the points is to the mean of the distribution then the higher the ELBO value of it.
   (P1 vs P5 and P3)
   also we can see that points that are in the wrong class also get low ELBO (P2)

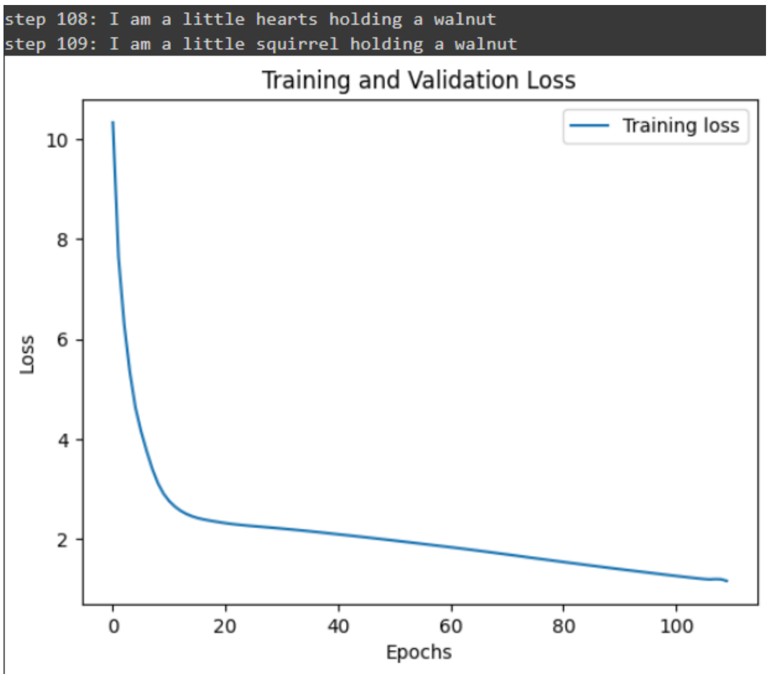


<u>Part 2</u>

1. loss term of the GPT-2 model over time: I trained the GPT-2 model, with 123M parameters.



2. Inversion: I performed gradient descend on the embedding vector of size [1,9,768]: The ‘9’ is the 9 tokens of the sentence, and the ‘768’ is the embedding dimension. I used a trained GPT-2 model, and created a new class InvGPT that encapsulates the trained GPT model:
   It takes a trained model as input and uses a modified version of the forward method such that the input vector replaces ‘tok_emd’ (the token embedding).
   The loss is calculated as a cross entropy between the output of the model, and the target output: both matrices of size [1, 9, 50257]
   a matrix of 9 ‘one-hot’ vectors, each entry represents a word in the vocabulary.



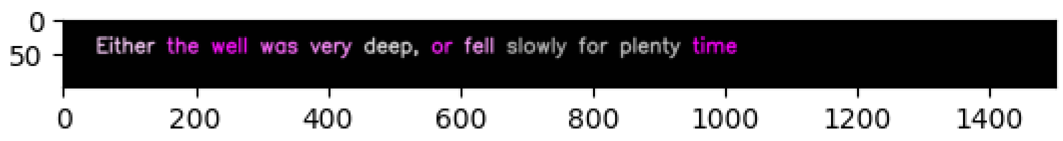
```
step 108: I am a little hearts holding a walnut
step 109: I am a little squirrel holding a walnut
```

3. I entered the prompt `prompt = "Either the well was very deep, or"`

   And the comptorion was:

```
Either the well was very deep, or fell slowly for
 plenty time she down look everybody, she obliged everybody
 about the of as went of
onder was to any, isn any!' Alice not to. Oh had as as came proper of
 his--
```

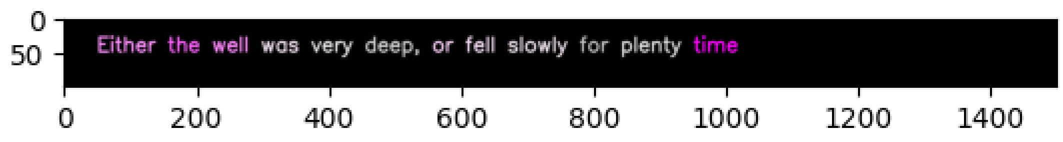
   Here I show the results of the last attention layer.



```
0.04301264509558678
0.006908044684678316
0.0015561794862151146
0.013636487536132336
0.021088873967528343
0.12866245210170746
0.003659454407170415
0.02459331415593624
0.28580471873283386
0.2359730750322342
0.2351047545671463
0.0
```

   Here purple means low attention score, and white means high score:
   BY analyzing the attention scores we can say that the
   word “**time**” was chosen mostly due to the last 2 words, but is mostly
   connected to the word ‘slowly’ and also influenced by the word ‘deep’.
   ‘Slowly’ got the highest attention score which is pretty cool actually.
   Also, the attention scores sum up to 1.

4. These are the results on the same sentence using the first attention layer:



```
0.021429454907774925
0.01202754583209753
0.01930335722863674
0.06661903113126755
0.10587217658758163
0.20291155576705933
0.04512729123234749
0.05753448233008385
0.07424075901508331
0.2550179660320282
0.13991637527942657
0.0
```

   The results are different from the last layer, here the higher scores were
   achieved by the words “**for**” and then “**deep**”.
   It appears that each attention layer “looks” at different contexts, semantic
   or grammatical ones.

5. I entered the prompt `prompt = "Alice had learnt several things"`

   And got the completion:

```
Alice had learnt several things this in ons the room and this not
without seen when noticed
```

   The probability scores were:

```
[0.96844816 0.99767333 0.9583966 0.48838913 0.998796 0.99667275
0.9999672 0.9159985 0.7606309 0.59195423 0.7214095 0.99251103
0.859273 0.9818033 0.40744302 0.53757244 0.9935469 0.97086614
0.9388271 0.8968002 ]
```

   Log probability: $log(P(x)) = log(\prod_i P(x_i)) = \sum_i log(P(x_i))$

   So the log probability of the sentence was `tensor(-3.9145)`