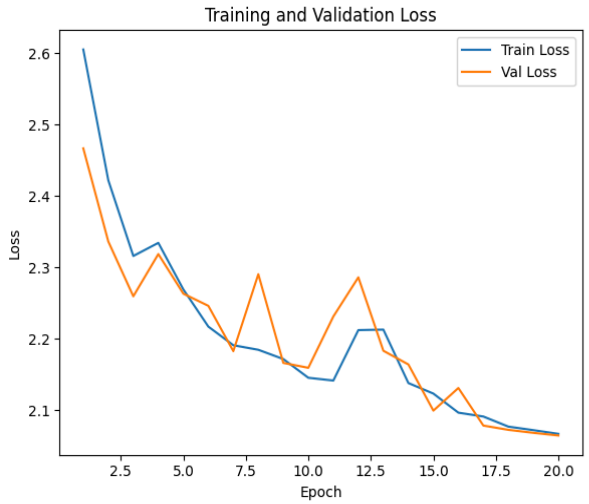
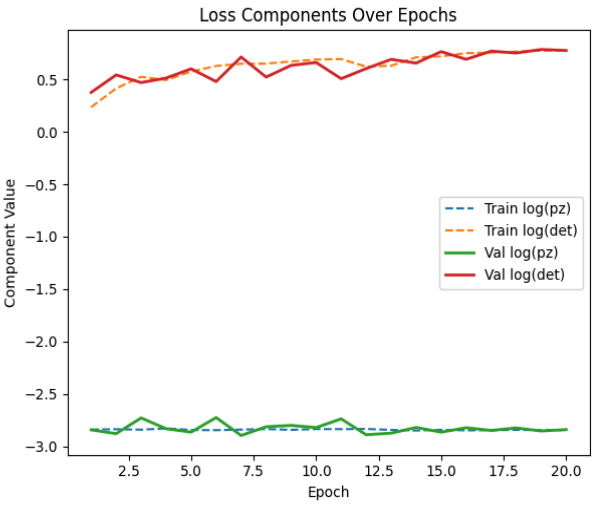
Flow Models

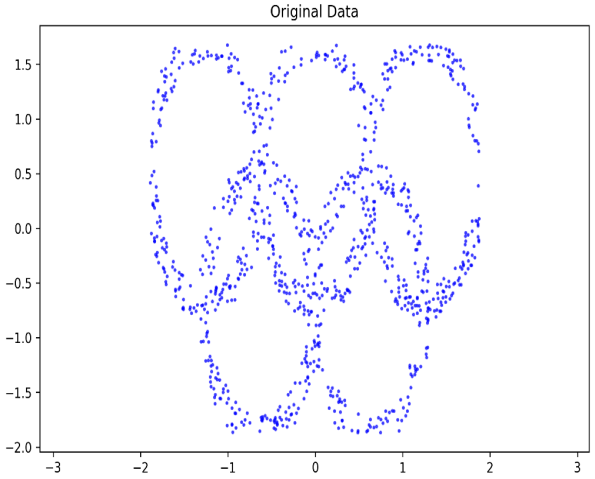
Normalizing Flows

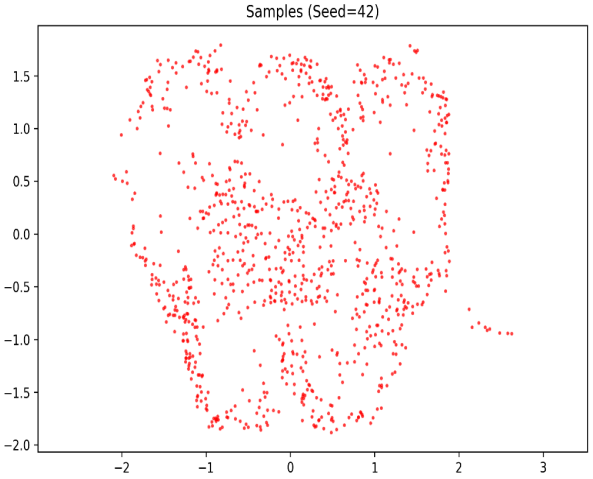
Loss

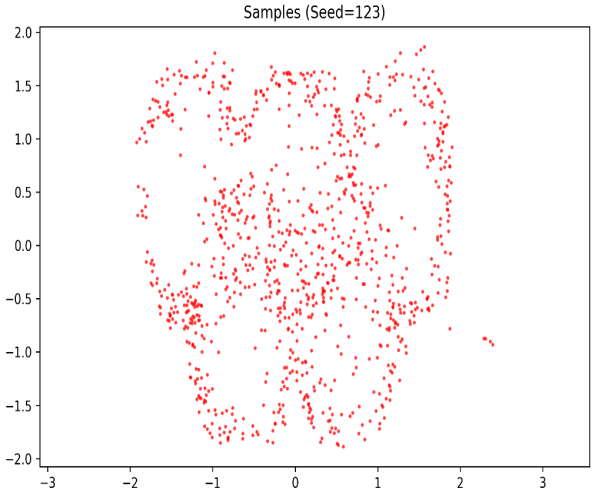


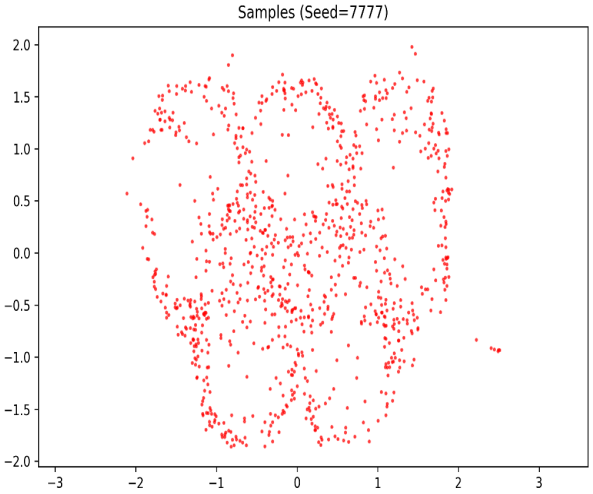


Sampling

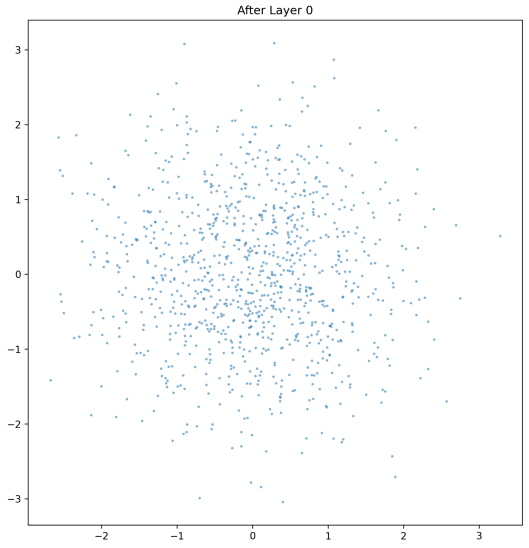


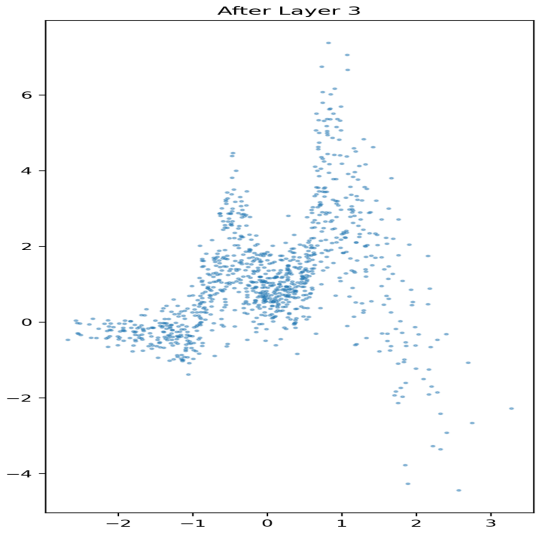


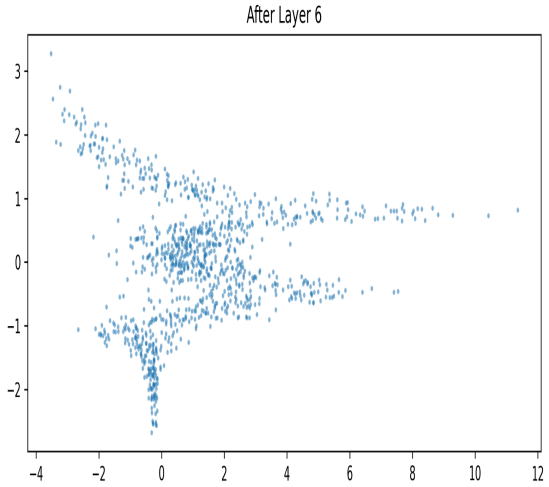


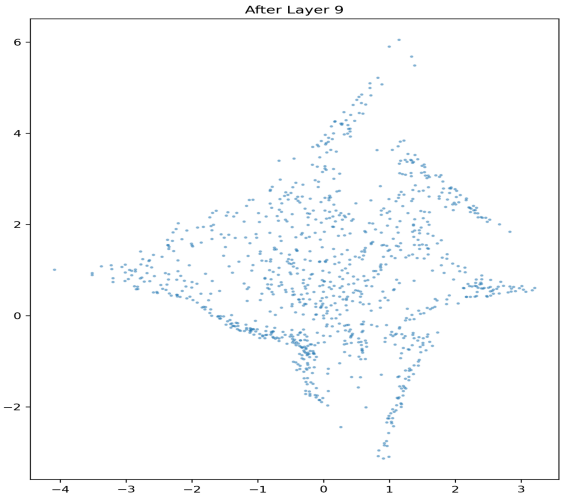


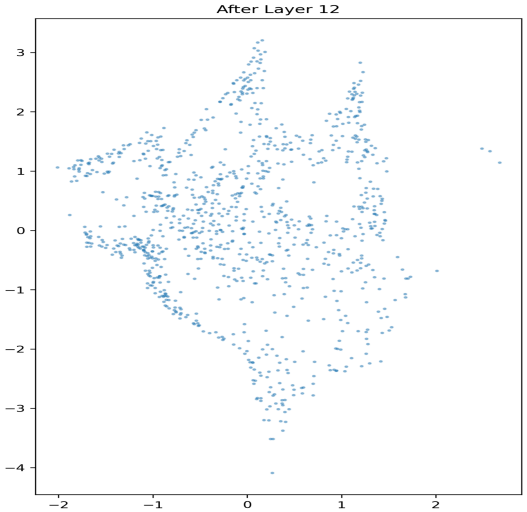
Sampling over time

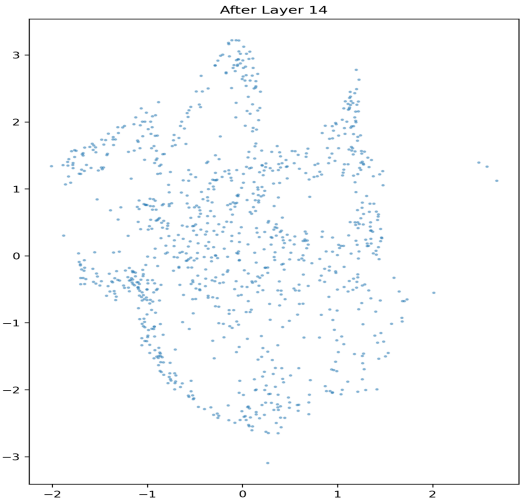




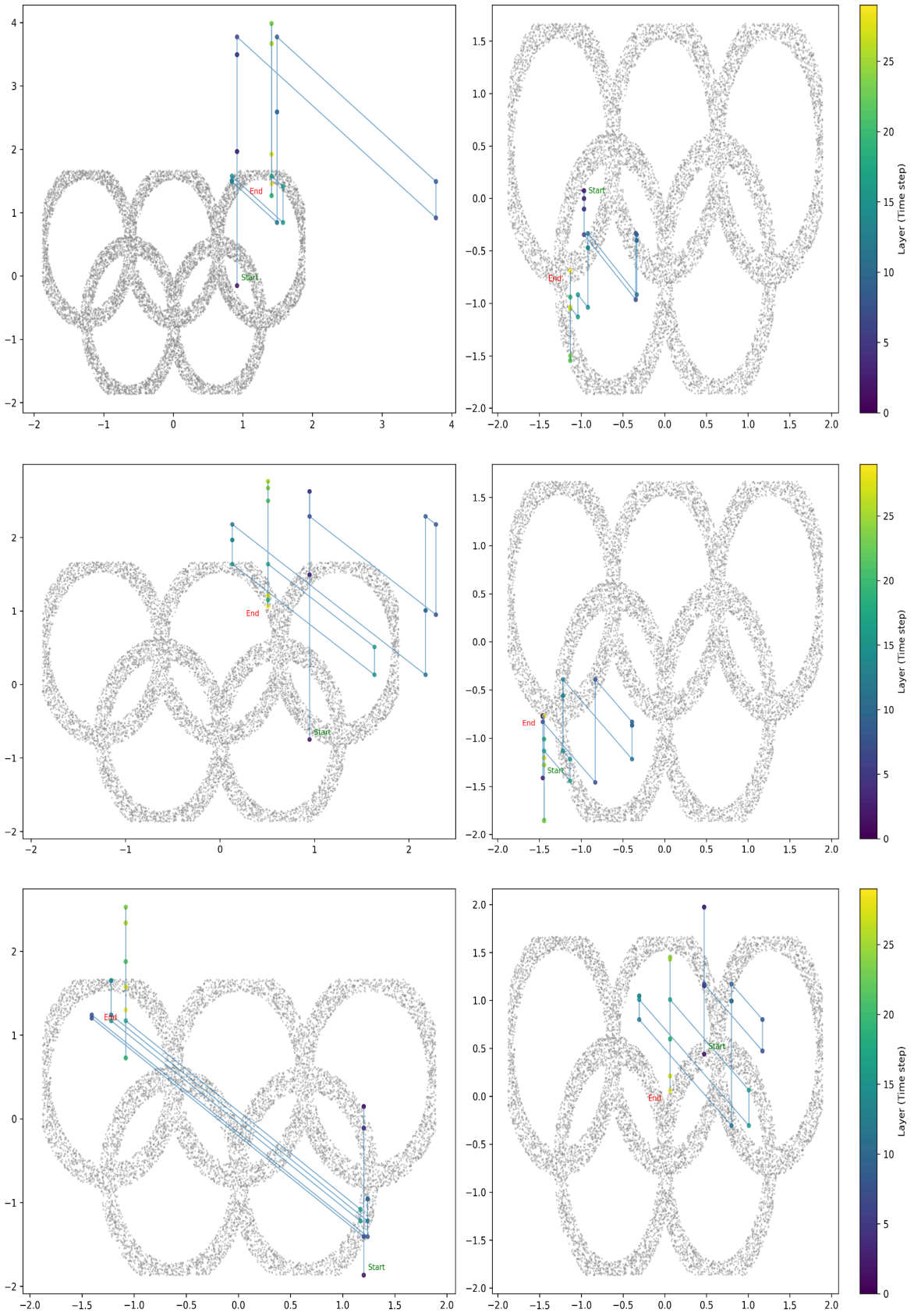


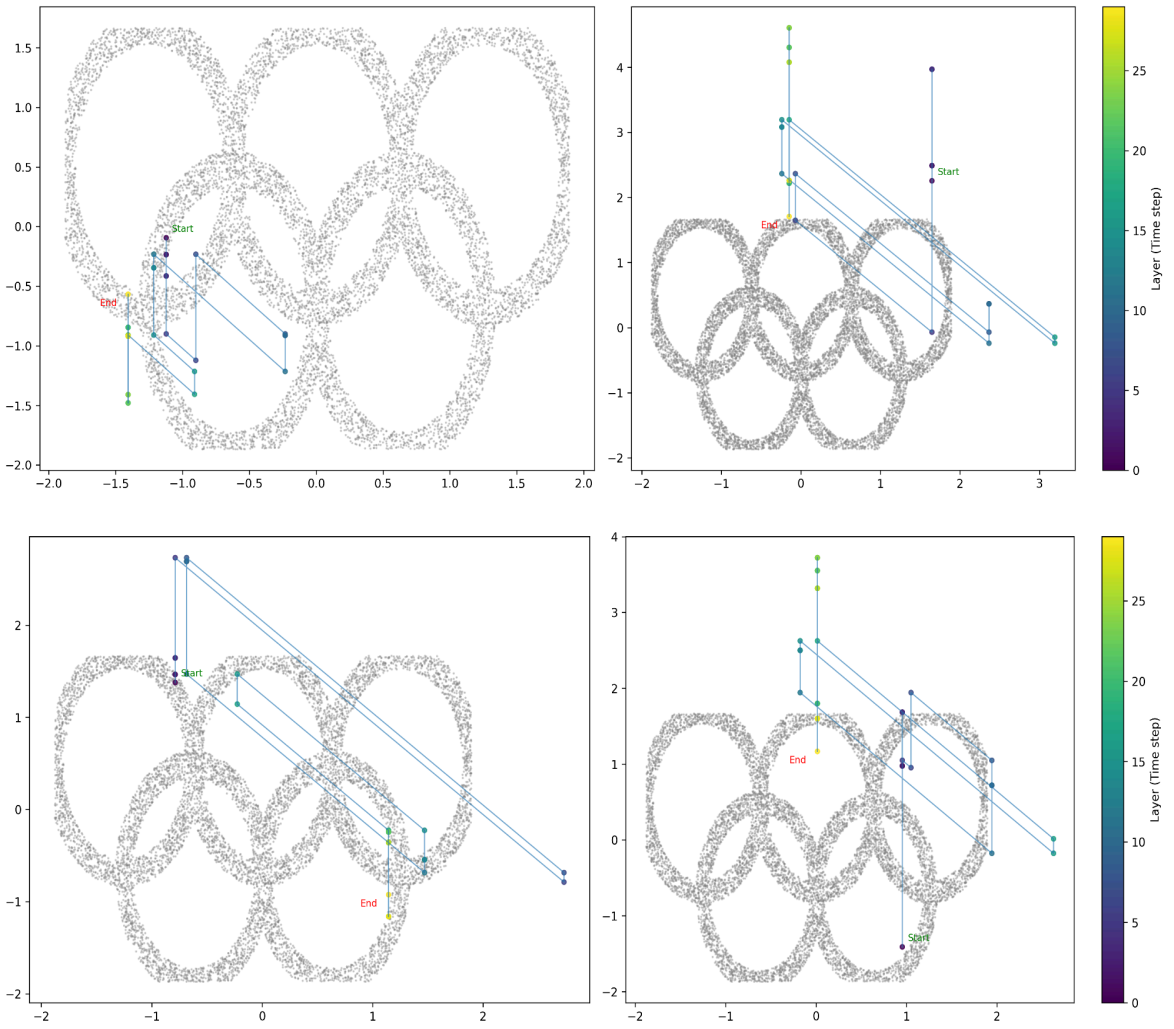






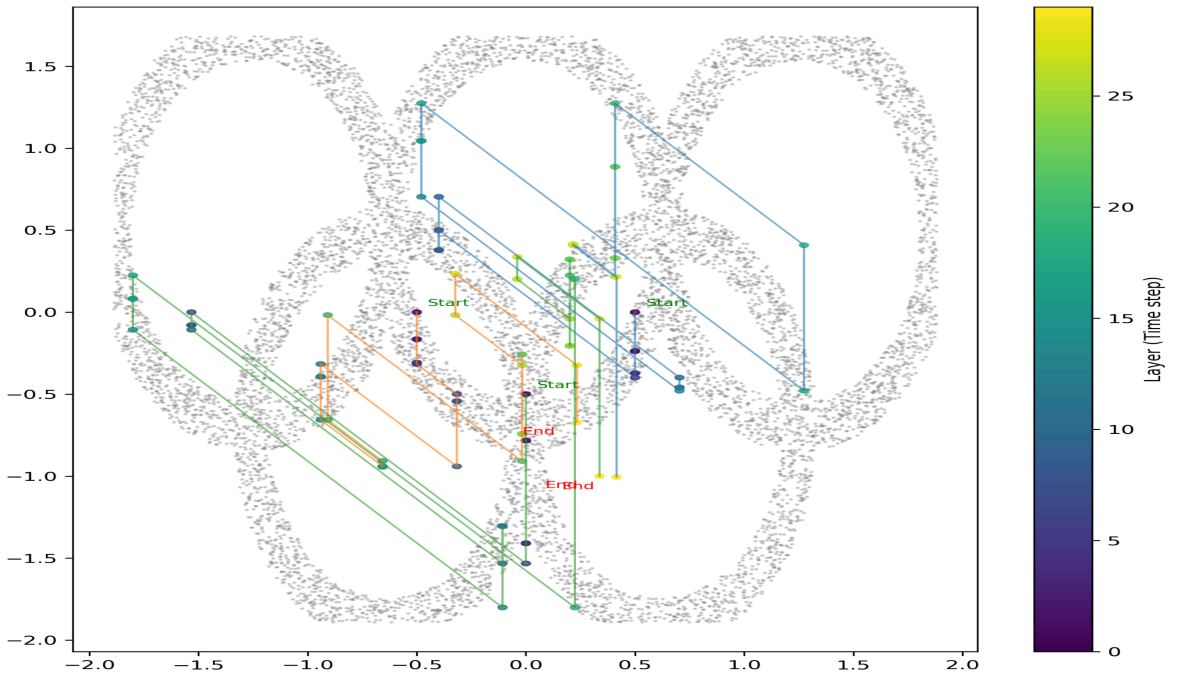
Sampling trajectories



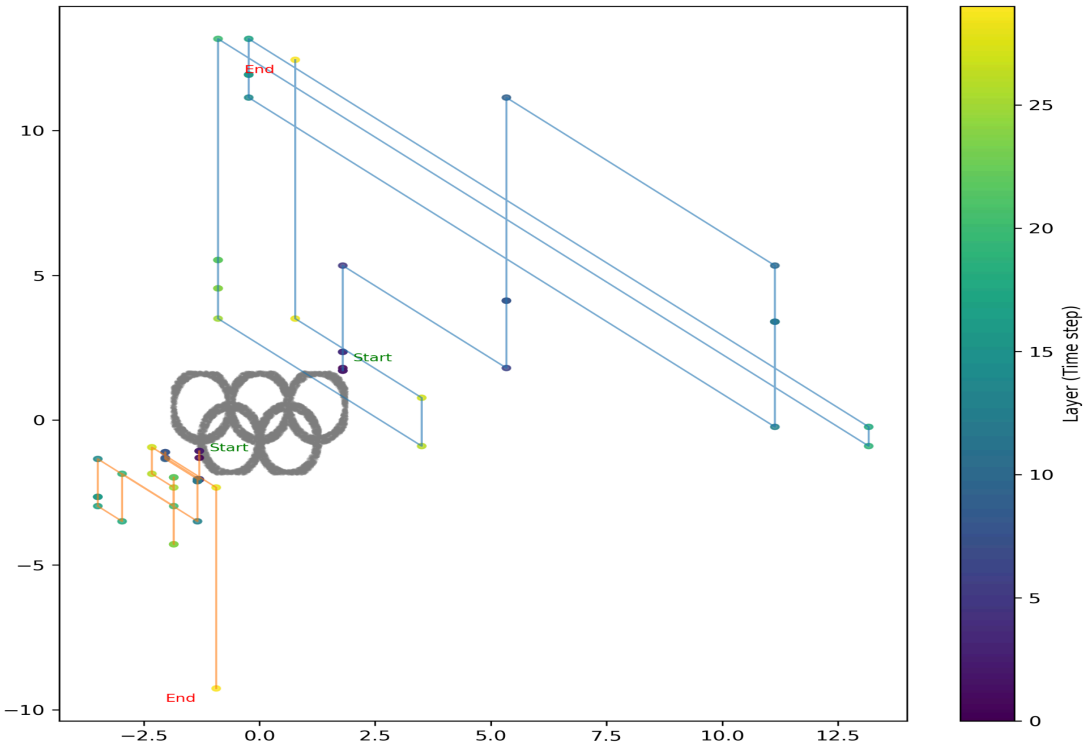


Probability estimation

Inside points



Outside points

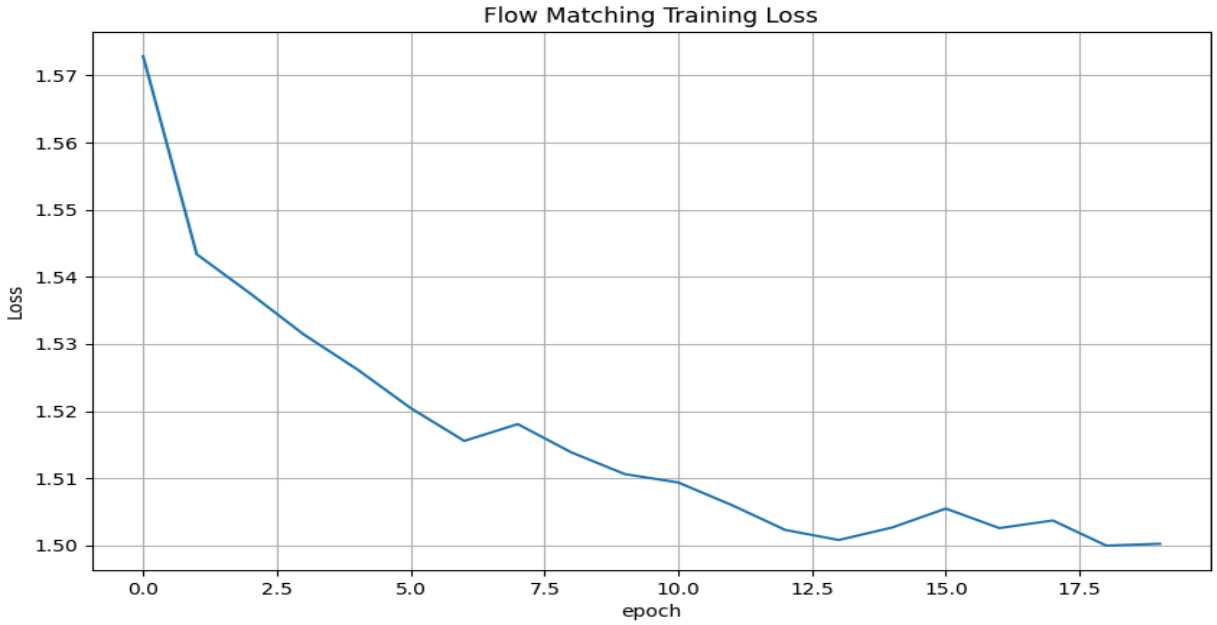


--- Summary of Log Probabilities log(p_X(x)) ---
Inside 1 ([0.5, 0.0]): -1.7133
Inside 2 ([-0.5, 0.0]): -2.0947
Inside 3 ([0.0, -0.5]): -2.0304
Outside 1 ([1.75, 1.75]): -12.8457
Outside 2 ([-1.5, -1.5]): -61.6288

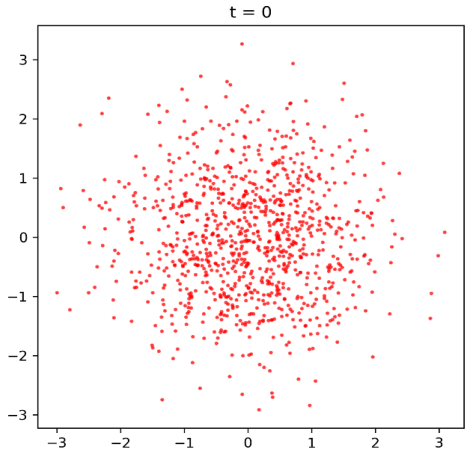
Average log(p_X(x)) for 'Inside' points: -1.9461
Average log(p_X(x)) for 'Outside' points: -37.2372

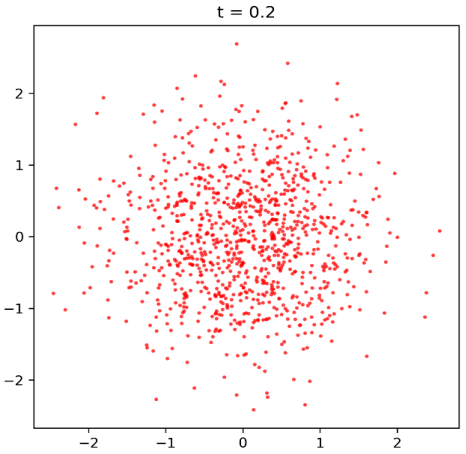
As seen by the provided log probabilities points outside are less likely,
This can be explained by the fact that Normalizing flows are designed to transform a complex data distribution (like the Olympic rings) into a simpler base distribution (e.g., a standard Gaussian). The core idea is that points belonging to the original data distribution should map to high-density regions in the latent space of the base distribution and as a result points 'Inside' get higher probability then points 'Outside'.

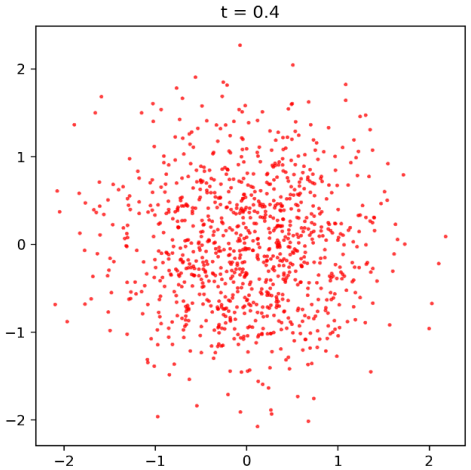
Flow Matching
Unconditional
Loss

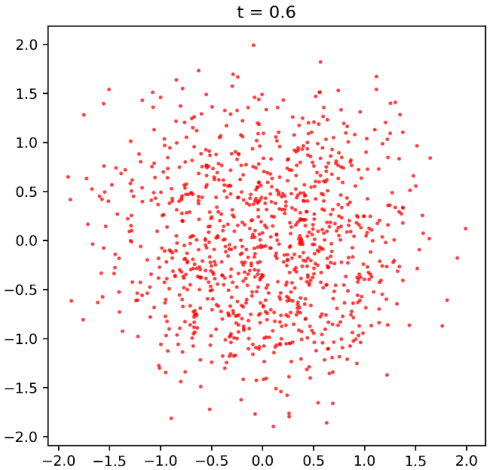


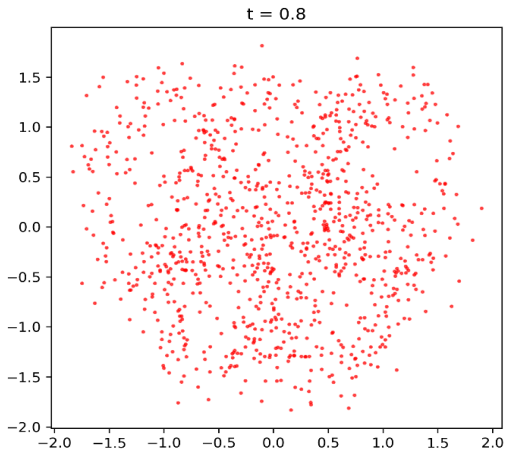
Flow Progression

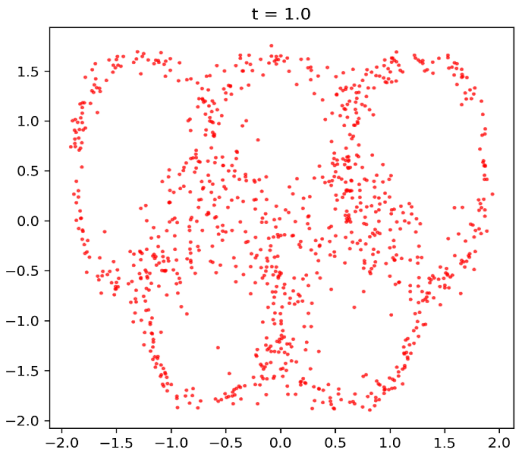




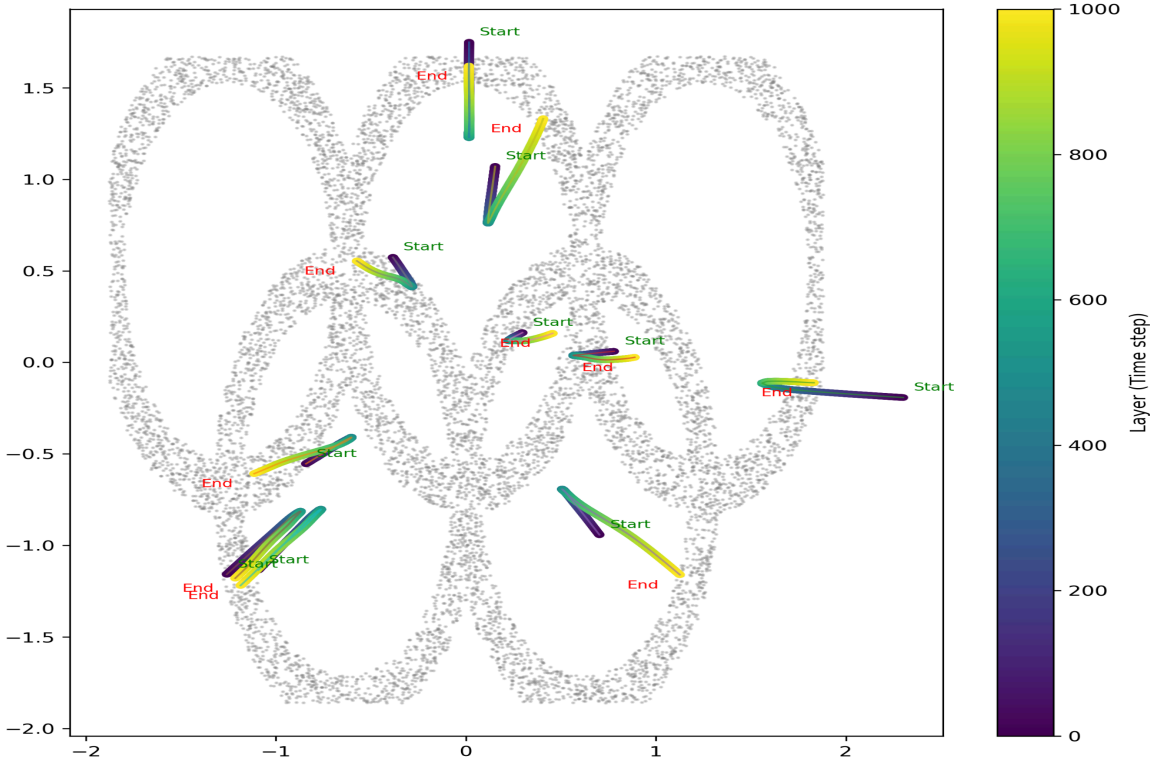








Point Trajectory



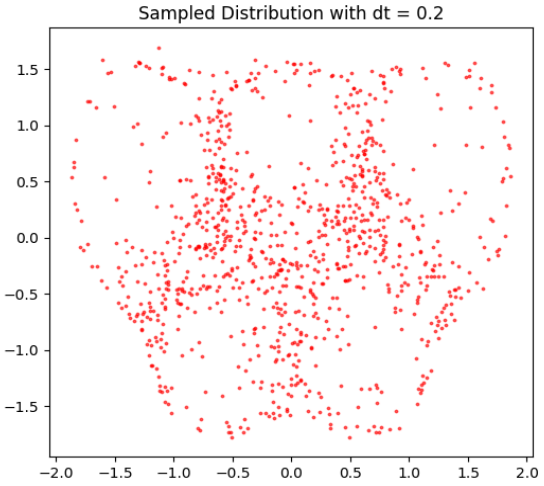
Looking at the plots we can see fundamental differences between Normalizing Flows and Flow Matching.

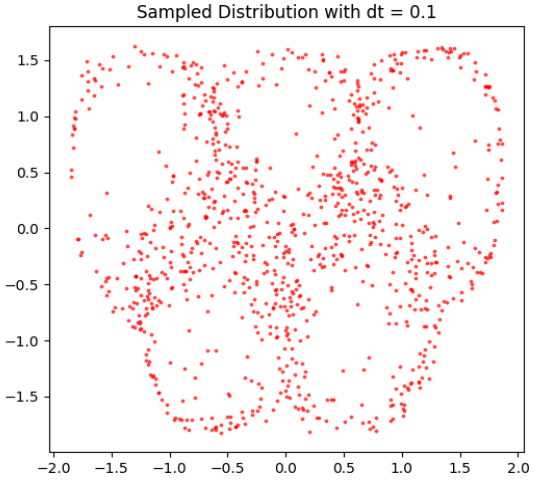
Flow Matching demonstrates a smoother and more consistent transformation from the initial Gaussian distribution to the target Olympic rings. This is because it learns a continuous vector field, and the transformation occurs via small, iterative dt steps, making the point trajectories resemble "flowing water".

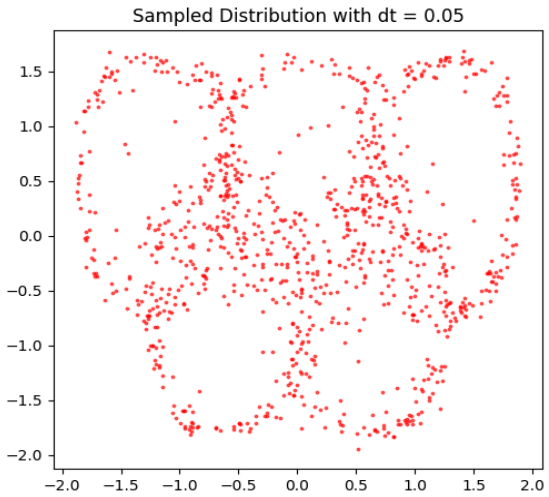
In contrast, Normalizing Flows transform data through discrete, invertible layers.

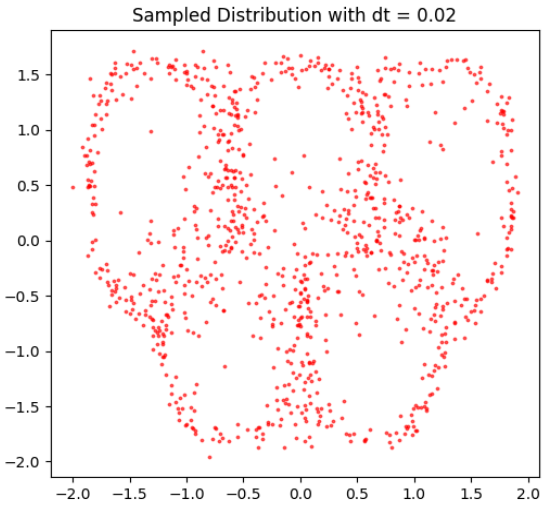
The progression can appear faster and less smooth, as each layer applies a distinct transformation, often moving one coordinate at a time. So, the distribution can seem to jump around or quickly shift from one place to another between layers. And when you look at a point's journey, it's more like a zigzag or a series of distinct hops rather than a smooth curve.

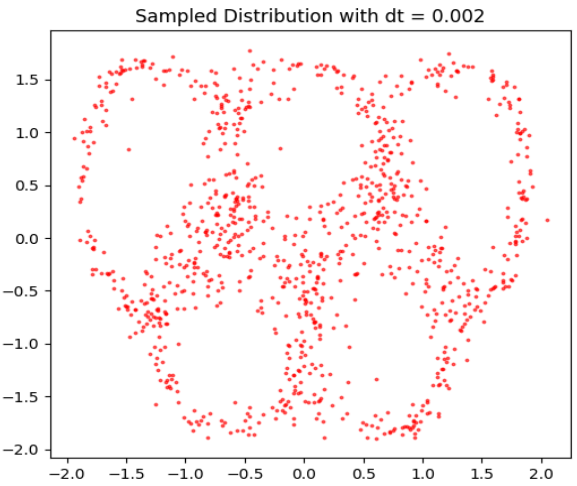
Time Quantization





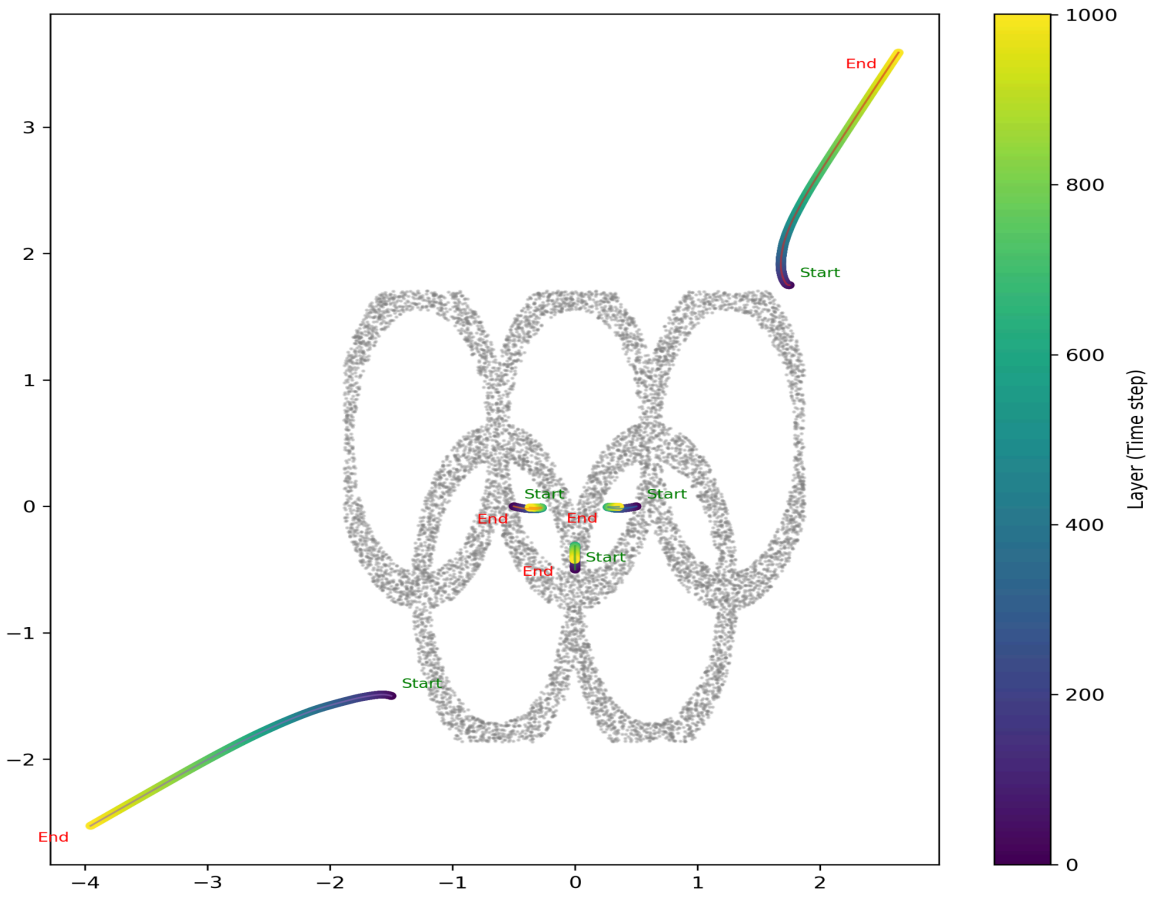






The choice of Δt significantly impacts the sampled distribution in Flow Matching. When we use smaller Δt values, like 0.002, we get very accurate samples that look a lot like the Olympic rings. This is because they enable precise calculations of the continuous vector field. On the other hand, if we opt for larger Δt values, say 0.2, the results are much less reliable. The errors start to add up, causing the points to stray from their actual paths.

Reversing the Flow



The reverse sampling process for Flow Matching shows points flowing smoothly back towards the Gaussian prior. Compared to Normalizing Flows, Flow Matching trajectories are more continuous, as they result from numerically integrating a learned vector field backward in time. Normalizing Flows, conversely, reverse through discrete layer transformations, making more segmented paths.

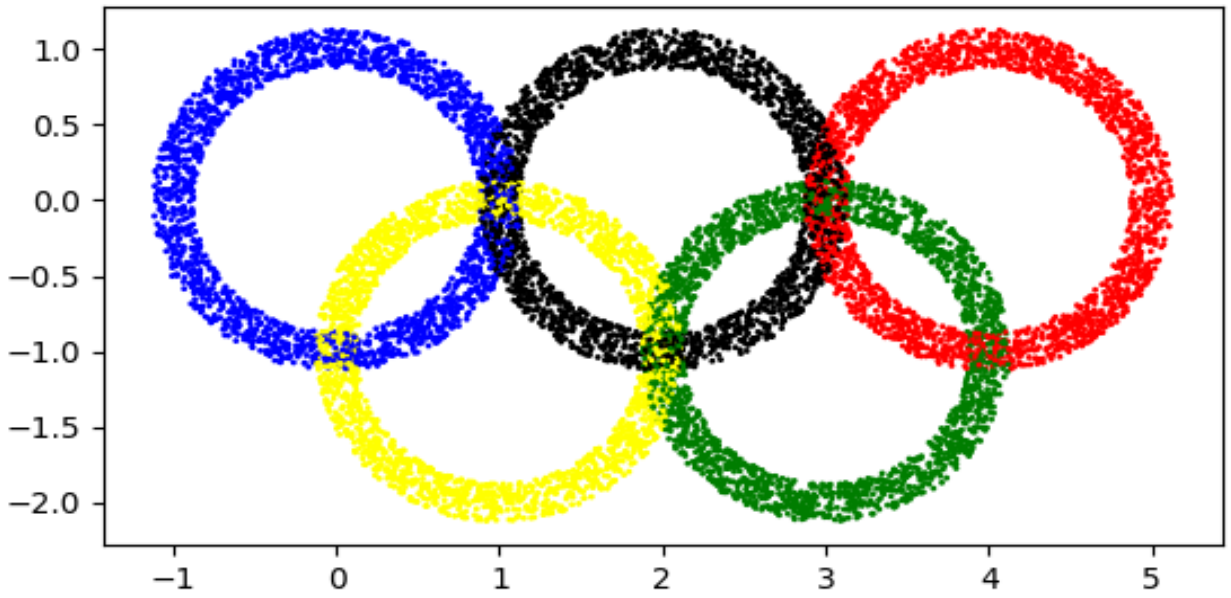
The outputs are not the same, the exact latent space coordinates (outputs) for a given input will most of the time be diffrent between Flow Matching and Normalizing Flows. This is because Normalizing Flows use explicit, compositionally invertible functions, while Flow Matching approximates a continuous flow via vector field integration. Their distinct internal mappings lead to different numerical results in the base distribution.

For Flow Matching, you would get approximately the same points back. The forward and reverse processes are designed to be inverses of each other, but the accumulation of small errors during the discrete dt steps prevents perfect reconstruction, leading to minor deviations from the original points.

On the other hand in Normalizing Flows you would theoretically get the exact same points back. Normalizing Flows are built from explicitly invertible functions; each layer has a defined inverse. Applying the inverse transformation and then the forward transformation should perfectly reconstruct the original input, barring only floating-point precision issues.

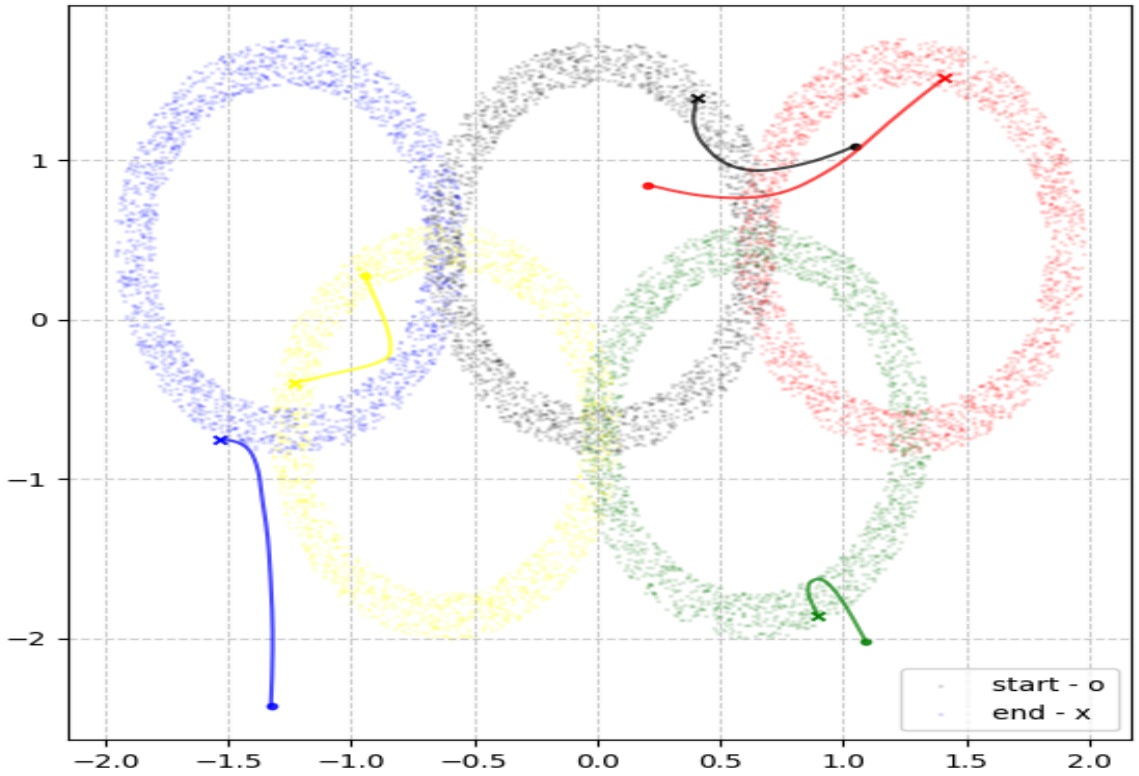
Conditional

Plotting the Input

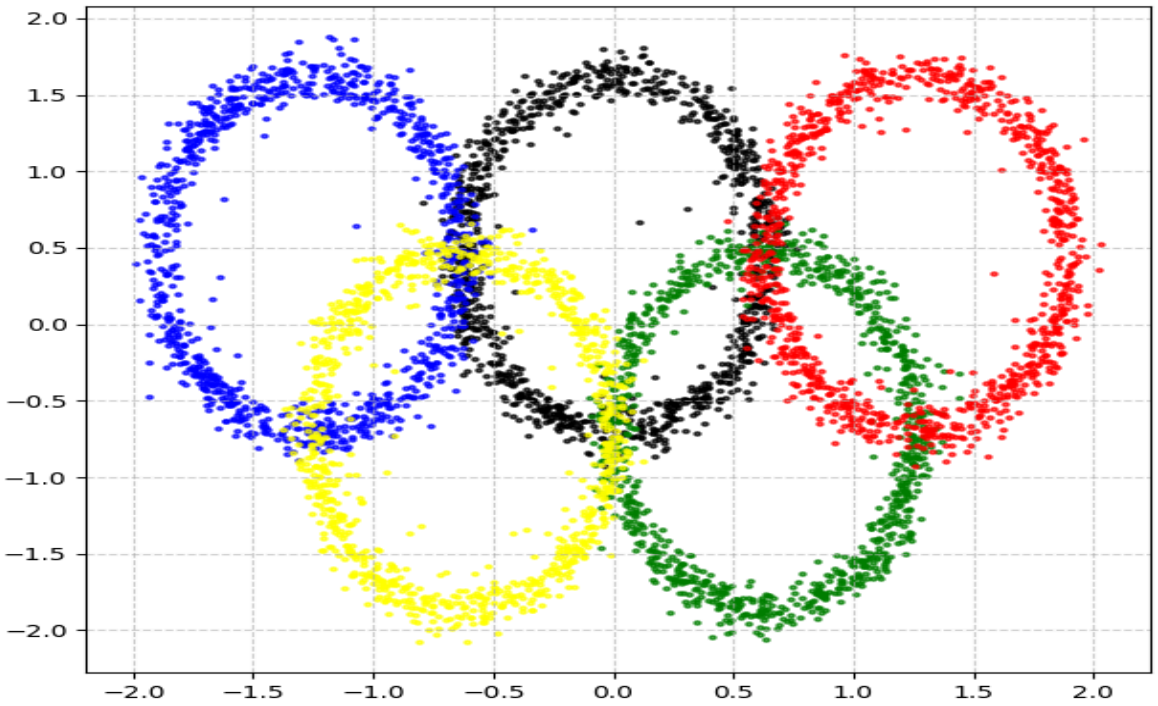


the core change was made to the neural network V itself, which approximates the vector field vt in the Flow Matching loss function. While the mathematical form of the loss (Equation 13) doesn't explicitly show the class variable, the implementation implicitly modifies the vt term. By feeding the class embedding into the network V(y,t,class_labels), we essentially make the learned vector field vt conditional on the class, i.e., vt(y,t,class). This means the network now learns a distinct velocity field for each class.

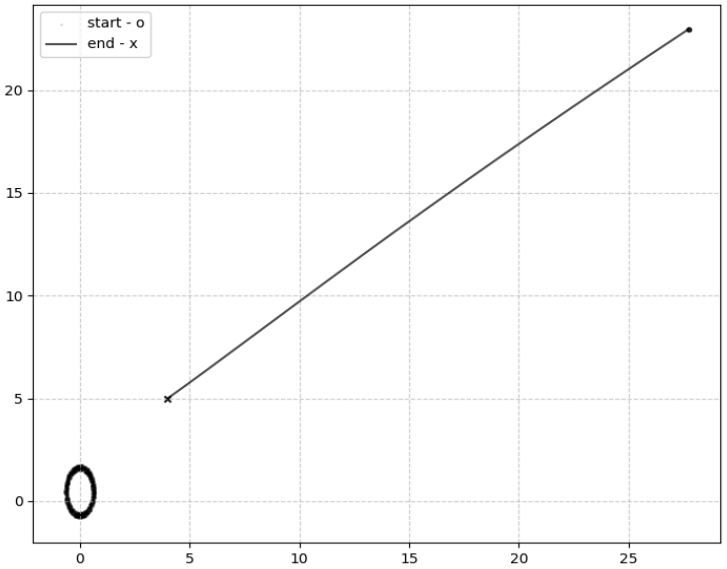
A Point from each Class



Sampling



Output specific point



I put the point (4,5) in the Conditional model in reverse with class 2 (moved opposite to the vector field we trained) and then i just take the point i got at t = 1(in my case the point is (28,24)) and put it in the model with the same class to get the (4,5).(in conclusion because the point is far away then the wanted distribution we will need a point that is far away from (0,0) both are unlikely and because we use linear flow we want to take a point such that the the point we want to get is in a liner line between the point we toke and the wanted distribution).