RL-Course 2024/25: Final Project Report

Florentin Doll

February 26, 2025

1 Experiences

So i will just start at the beginning. I started out with super simple networks that just predicted one value, or a matrix of values for z. I already had a GRU for my hidden state h since this one needs to be a recurrent memory. Since i didnt have distributions i couldnt use the normal dreamer losses and just used MSE or similar losses, i played around with that a litte but never got a good result.
For most of these steps i was using the weak opponent from the environment as an actor, since i wanted to focus on the world model first.

Since i was permanently underfitting i tried adding more layers, added initialization weights, normalization and everything.

The next big step that i took was to pretrain an autoencoder on a lot of data, that i could then freeze and use in my world model.
This autoencoder would sorta work, i still had issues with speeds being off my like 1.5 sometimes, so i got something like 28.5 instead of 30 but it was not too bad. I am not sure in hindsight, if that would have worked if everything else was properly set up (which it wasn't). I was already using the kl losses back then, on non distributions, so that didn't do anything i think.

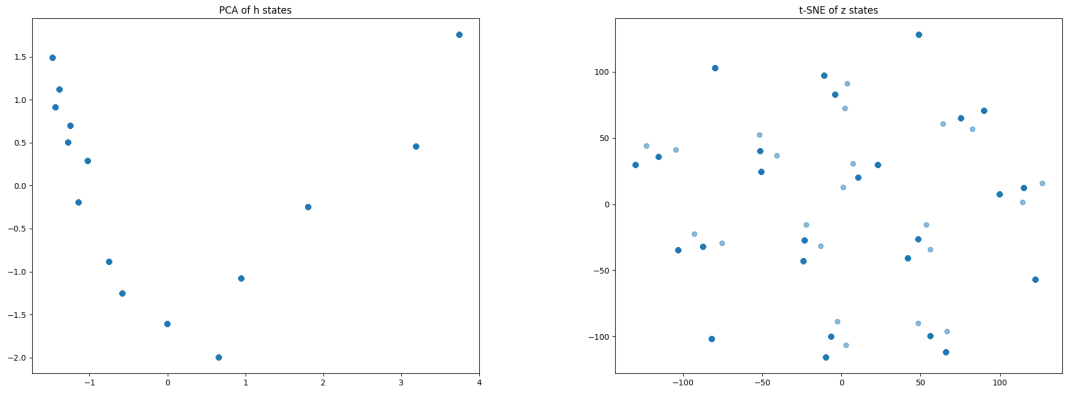
Since i couldn't get this to work i swapped to more dreamer like implementations. I rewrote most of my code to make every network put out distributions and i went from an uncontrolled z to a binary z. I didnt use one hot vectors yet, instead i used bernulli distributions to pull samples from probabilities my networks gave me.
With that i could finally use exactly the same losses as dreamer, but i was still heavily underfitting.

After adding stuff like kl weighting and testing with different values I now swapped to categoricaly one hot vectors for z. Before i read some papers on one hot vectors improving training and latent space quality, which is why i decided for that move. At this point i also added nice logging with tensorboards for each loss.

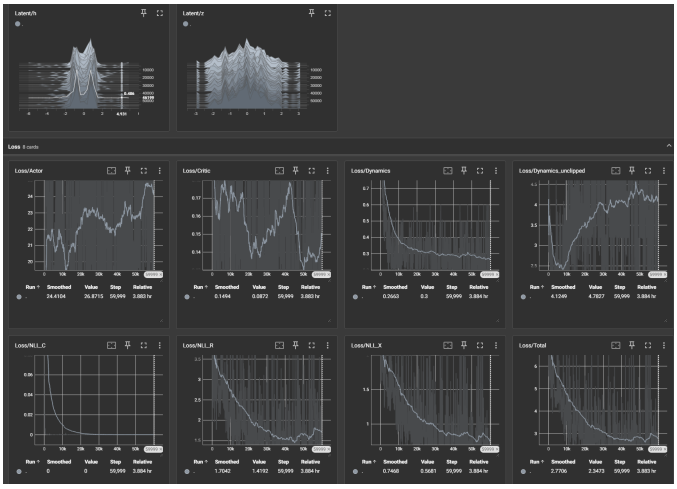
To explain my questionable decision making here, i was obsessed with making my architecture more complex to avoid the underfitting, because i could not get rid of it, no matter what i did. So for some reason i got the idea to add a Transformer for interpreting the latent space z instead of a simple embedding layer. I am not quite sure why i thought this was a smart move, but i added a SlotAttentionTransformer to my world model, which made it like 3 times slower and, shocker, did not help training.

Appart from that i also found a lot of issues in my code, that arose by me changing the architecture

multiple times.
I also added my own actor now, that i would now train together with my world model, because i really thought it would work after me fixing that many things. I also added plotting to my models at this point, so that i would get plots of my latent z and my hidden h after training.



Here you can see one of the underfitted z and h spaces (the clusters come from the gumble softmax i think but im not sure.)
Now my last step was where i finally came to think about the way i trained over multiple dreams each step, which would then lead me to comparing the predictions of observations that were made under the assumption of different actions than the ones that were actually taken. After removing that and swapping to the pendulum instead of the hockey i got some pretty nice looking hidden states, but i still can't predict observations accurately and i don't know why.



This is my best version of the pendulum i think, The actor is really bad, but that is due to the implementation error that i talked. What I mean by the best is the reconstruction loss X that is the lowest I have ever produced. (Top left are visualizations of h and z)

So as a final thought: I really dont know what hinders my world model from learning good representations, i feel like i fixed literally everything, i literally basicly did nothing but working on this for the last 7 days. I really want this to work and i will keep working on it until it does. So if you accidentally find anything that might cause this, please let me know, i am really running out of ideas here (Maybe fixing the agents gradients did the trick, cant confirm right now).