Task 1:

What is reinforcement learning:

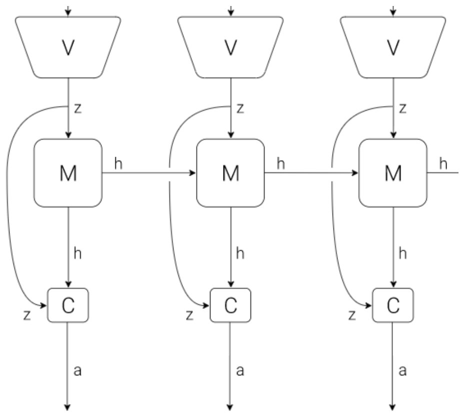
Reinforcement learning is a training process to make a sequence of decisions to achieve maximum awards. The environment it faces is highly uncertain and complex and the learning uses trials and loss to come up with a decision-making process. Unlike supervised learning There is no label suggesting an action is right or wrong, it only uses reward system with many trials to find ways to maximize rewards.

Model based reinforcement learning:

In model-based reinforcement learning, the system uses various machine learning models such as random forest, gradient boost, or more complicated one like deep neural network. This is different from model-free reinforcement learning where it does not use models, instead, it only uses non-ML algorithms for policy selection. Model based reinforcement learning lacks prior knowledge which indicates that this method needs to explore more policies than other reinforcement learning methods. In this World Models paper, the main point is that I do not need to actually interact with the environment, instead, I need to create a model which represents the environment and learn using this model.

General structure of World Models:

The agent model has three components: Vision model, Memory RNN model and a small controller. The vision model is used to take an input image and transform it into a low-dimensional latent vector. This model is achieved by using VAE (Variational Autoencoder). The Memory RNN model is the part to use previous information to predict future actions. The controller is a decision-making part to decide which action to take based on previous vision and memory RNN model results.



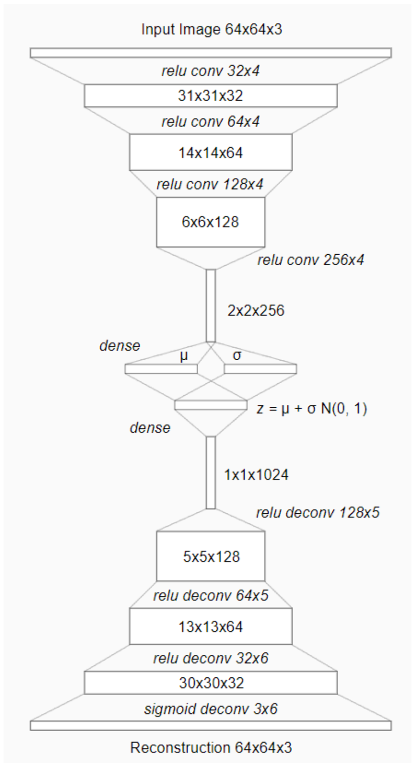
Steps for Car Racing experiment:

1. Generate random samples

Firstly, they go to the environment and collect 10000 random rollouts (640 in my reproducing experiment). In the meantime, at each time step, random actions (policies) are collected as well.

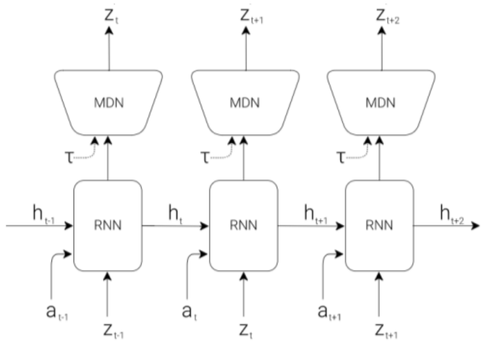
2. Train VAE

In this particular experiment, the input image has the size of 64*64*3 which is high-dimensional. This could make the learning not efficient. Therefore, such observations need to be reduced. Variational Encoding is one such method. It is basically a data compression and decompression process. In this experiment, for encoding process, the input image is fed into a convolutional network where it is finally converted into a much more compact, dense latent space which is low dimensional. When decoding, there is another convolutional network to reconstruct the latent space back to 64*64*3 image. There will be some loss of information as the reconstructed image can be very blurry.



As you can see form the structure image on the left, The input image is of size 64*64*3. Then, it is passed through 4 convolutional layers which is eventually a latent vector z of size 32. With such low dimension, the agent can learn much more efficiently. This latent vector consists mean and log variation of the distribution of the reconstructed image. With the mean and log variation, and epsilon, a blurry image can be reconstructed again through multiple convolutional layers. All layers use RELU activation function except the final stage of the decoding part since a sigmoid function is necessary for [0, 1] output.

3. Train MDN-RNN



This part is where the "model-base" works. This model is used as a predictive model which gets the latent vector of previous time step and predict the Gaussian distribution of the latent vector in the next time step, and beyond. The neural network used in this stage is LSTM. LSTM is useful in this scenario because it can learn using the past information store in memory.

Under the Car Racing context, The RNN model will take the action at time t, the hidden state at time t, and the latent vector at time t to predict the latent vector at time t+1. The temperature parameter is added to adjust the uncertainty of the generated model of the environment.

4. Train controller

The controller is the actually policy learner. In this experiment, a controller is divided from the large neural network model. The large neural network is used to solely train to predict how latent vectors change over time with previous knowledge. The controller, on the other hand, is much smaller and is solely responsible for the reward information. This makes the learning of controller faster since there are only 867 parameters in the controller model. The model is as follow:

$$a_t = W_c \, [z_t \; h_t] \; + \; b_c$$

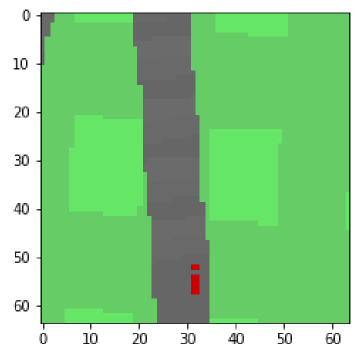
In this equation, z_t is the current latent vector, h_t is the current state of RNN which models the environment over time. W_c is the weight matrix and b_c is a bias parameter that mapps the latent vector and hidden state to the action. As you can see this is a pretty simple function and there are only very few parameters to learn a policy.

One interesting thing that is mentioned in the paper is that since the model created can model the future, it can basically create a fake racing scene. Since we have the latent distribution, we can sample a latent at t+1 given the latent at t. Then, we can train Controller in this dream environment. The agent can learn inside the dream and the policy learned can be used in actual environment.

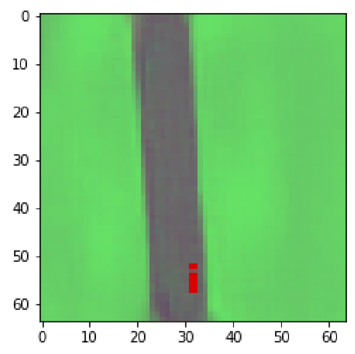
Task 2:

The reconstructed image using VAE is quite blurry.

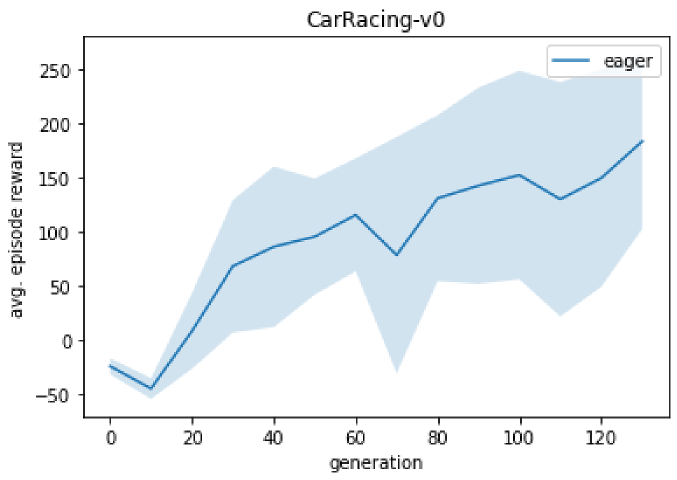
input:



VAE Reconstructed:



The result:



The speed of learning a policy is still a bit slow. This could be because the time steps I choose is high.

Speed of learning could be faster if I adjust the time step and some other parameters

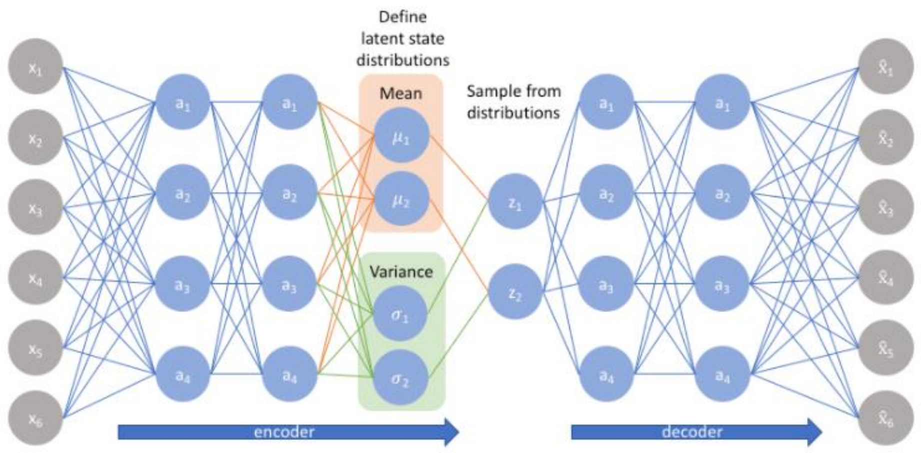
Task 3 (Document an approach of VAE/GAN):

In the paper, the authors suggests that current similarity metric used in generative models like VAE is not very suitable for image data as you can see that the reconstructed image is very blurry. It is mainly because the loss function is not very good. Therefore, they introduce a combination of VAE and GAN, which through experiment, can outperform VAE alone. To fully understand VAE/GAN. We need to first explain VAE, GAN separately and combine them together.

VAE Explained:

Variational Encoding has two parts: Encoder and Decoder. Encoder is a process to decrease the dimensionality of the input image into a latent space which describes the probability distribution. Decoder is a process that take in the latent space and reconstruct the image with the probability distribution.

The structure of a VAE is simple. It is basically a well-structured neural network which takes high dimensional image as the input and using encoder to condense it into latent representation of the particular image. Then decoder, which is also a neural network, uses that latent space to reconstruct it.



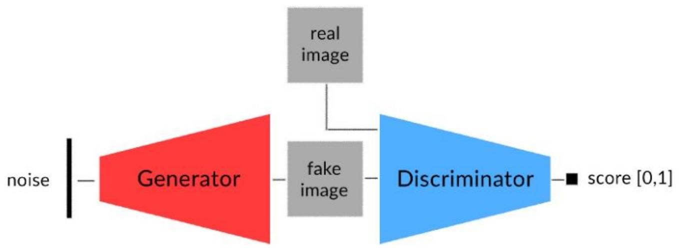
As you can see from the image above, The latent state distribution are the result from the encoder which is then used to make samples. Then those samples are used to reconstruct the image. In the sampling process, since we need to use backpropagation for final loss, we cannot use this for a random sampling process, therefore, VAE uses a “reparameterization trick” where we take a randomly sampled epsilon which is then shifted and scaled.

Typical loss function is:

$$\mathcal{L}(x, \hat{x}) + \sum_j KL\left(q_j(z|x) \,||\, p(z)\right)$$

GAN explained:

Generative Adversarial Networks is a type of unsupervised learning and are made of a set of generative models. Like VAE, Generative models are able to generate new content. It is made of two neural networks: one generative network and one discriminative network. The generator takes the noise and trys to create samples as an image. The discriminator is another neural network that examines the one generated from the generator and evaluate them. During the training, the generator will become better at creating the image and the discriminator will become better at evaluating the genreated samples.



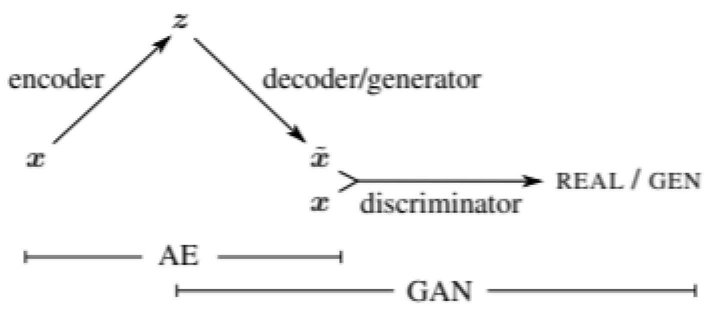
The loss function for GAN is:

$$E_x[log(D(x))] + E_z[log(1 - D(G(z)))]$$

D(x) is the discriminator's estimate of the probability that x is real, E_x is the expected value, G(z) is the generator's output if z is the input noise, D_z is the expected value over the random inputs.

Combine VAE and GAN:

When combining VAE with GAN, it is essentially using the generator in the GAN as the decoder in the VAE.



In VAE part, there is loss function:

$$\mathcal{L}_{\text{llike}}^{\text{pixel}} = -\mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}[\log p(\boldsymbol{x}|\boldsymbol{z})]$$
$$\mathcal{L}_{\text{prior}} = D_{\text{KL}}(q(\boldsymbol{z}|\boldsymbol{x})\|p(\boldsymbol{z})) ,$$

In GAN, the loss function is:

$$\mathcal{L}_{\text{GAN}} = \log(\text{Dis}(\boldsymbol{x})) + \log(1 - \text{Dis}(\text{Gen}(\boldsymbol{z})))$$

Since the element-wise error is not good enough for image learning, the loss function has to be changed. Therefore, a new loss function is used:

$$\mathcal{L}_{\text{llike}}^{\text{Dis}_l} = -\mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}\left[\log p(\text{Dis}_l(\boldsymbol{x})|\boldsymbol{z})\right]$$

The combined loss function is:

$$\mathcal{L} = \mathcal{L}_{\text{prior}} + \mathcal{L}_{\text{llike}}^{\text{Dis}_l} + \mathcal{L}_{\text{GAN}}$$

Since we are not updating all the parameters on the combined loss function, we can train VAE and GAN models simultaneously. The algorithm updates the parameters by using:

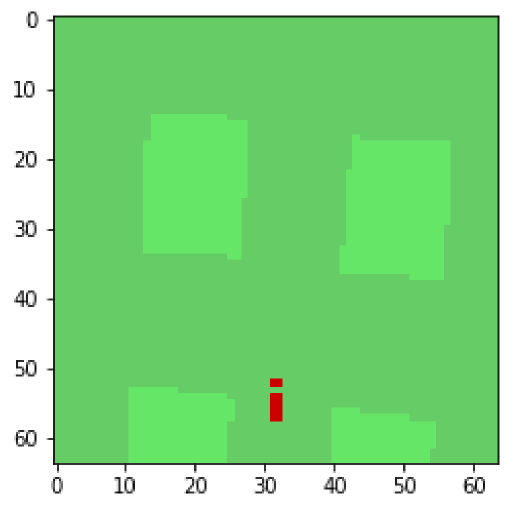
$$\begin{aligned}
\boldsymbol{\theta}_{\text{Enc}} &\overset{+}{\leftarrow} -\nabla_{\boldsymbol{\theta}_{\text{Enc}}}(\mathcal{L}_{\text{prior}} + \mathcal{L}_{\text{llike}}^{\text{Dis}_l}) \\
\boldsymbol{\theta}_{\text{Dec}} &\overset{+}{\leftarrow} -\nabla_{\boldsymbol{\theta}_{\text{Dec}}}(\gamma\mathcal{L}_{\text{llike}}^{\text{Dis}_l} - \mathcal{L}_{\text{GAN}}) \\
\boldsymbol{\theta}_{\text{Dis}} &\overset{+}{\leftarrow} -\nabla_{\boldsymbol{\theta}_{\text{Dis}}}\mathcal{L}_{\text{GAN}}
\end{aligned}$$

As you can see in the updating equation of Decoder, there is an additional parameter. This parameter is used to balance the reconstruction and the ability of generating image that can fool the discriminator.

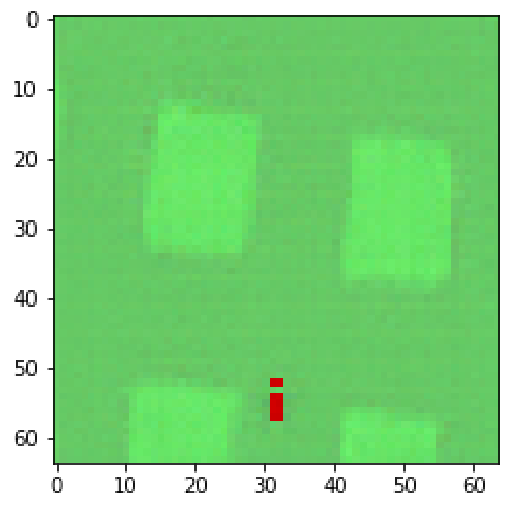
In this particular experiment, the author uses deep convolutional neural network for all three parts: encoder, generator, and discriminator. This is because convolutional neural network is great at generating images from latent space.

Repeat the experiment with VAE/GAN:

Input:

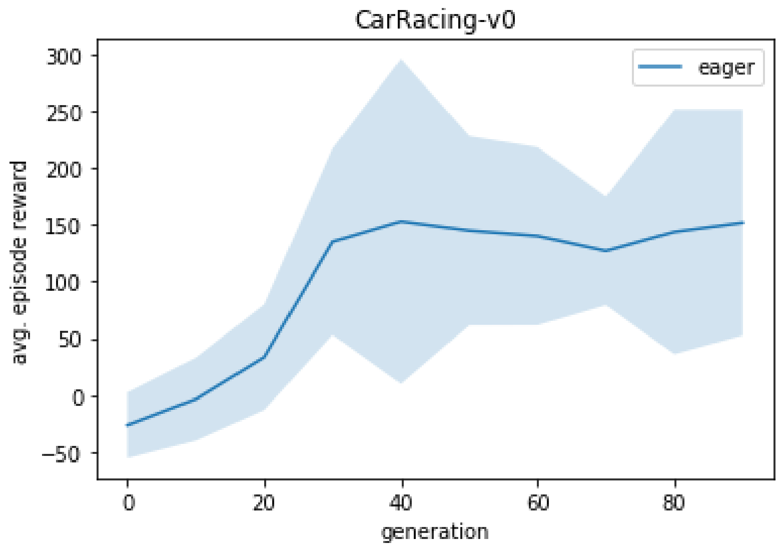


VAE/GAN reconstructed



In my opinion, using VAE/GAN can results better(clearer) reconstructed image than VAE alone.

Score:



In VAE/GAN, the score goes up faster than VAE over time. I did not reach 130 generation here at VAE/GAN.

However, I would imagine the score will be higher than VAE's score when it reaches generation 130.