Task 1

What is the difference between model-based and model-free?

To help us explain this question, firstly we define the MDP with the (S, A, R, T). S is the State set; A is the set of actions; R(s,a) is a function records the rewards; T(s'|s,a) is the transition probability function. Our target is to find the policy which can maximize the expectation reward.

Model-based RL

In Model-based, the agent would learn a model which observes and describes the environment in its own way, and make a policy based on the model. And once the model is good enough, the agent can use a planning algorithm with the model to find a policy.

Model-free RL

In Model-free, the agent would not model the environment to find a good policy. For example, the Q-learning estimates the Q(s,a) and finds a policy based on this. For the approaches do not learn a model of the environment, so they are model-free.

The key to check whether the model is model-based or model-free depends on whether the agent can make a prediction of the rewards and the state before it takes action. If the agent can, it is a model-based RL. If not, it is a model-free RL.

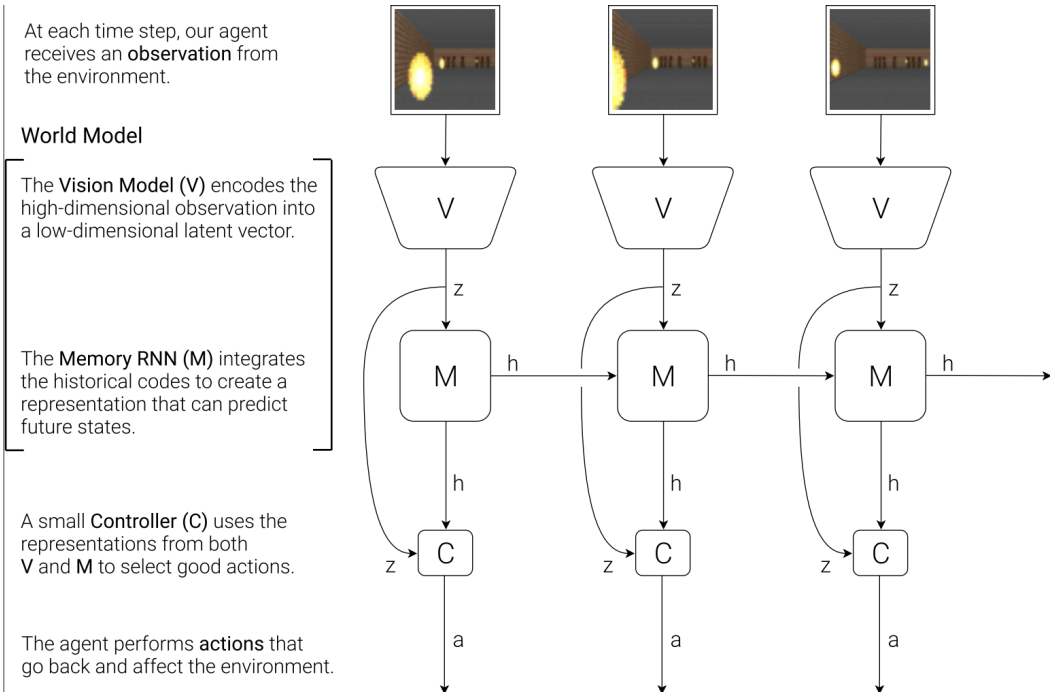
The advantage of the Model-free RL is that it is easier to implement. For the Model-based RL, we need to model the environment and then use neural networks to model the states, rewards, and terminal states. After this, the hyperparameters

should also be adjusted. But Model-based RL has a high data utilization which helps us solve some problems because they may have high cost in real scenarios.

What is the model architecture in the *World Model* ?

In model-based reinforcement learning, the agent would model the environment. On the one hand, to learn the environment more effectively, a more powerful and expressive model is needed. On the other hand, it is difficult to learn the models with millions of weights, because the search space for the reward distribution is too large.

To overcome the disadvantages, we can split the model into two parts. A larger and complex model would be responsible for learning the environment, while using a smaller model to perform tasks. As a result, the larger model would not affect the search space of reward distribution, and the smaller model can obtain an understanding of the environment by the larger model to learn better.



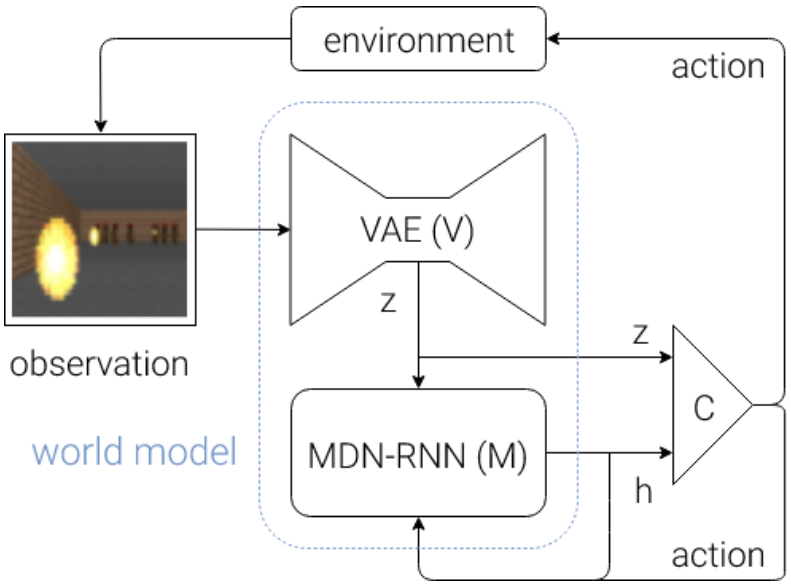
In the *World Model*, the authors use a big world model, consisting of Vision Model**(V)** and Memory RNN**(M)**, to model the environment and a small controller**(C)** to learn performing tasks.

Specifically, the Vision Model**(V)** receives a 2D image frame as the input and encodes it into a small *latent vector* ***z*** to compress the information as the output.

The MDN-RNN(M) model aims to predict the future by predicting the z vectors. It receives the action the agents took, the hidden state of the RNN, and the z vector as the input, and we can get the formula $P(z_{t+1}|a_t, z_t, h_t)$. It would output a probability density function p(z) instead of a deterministic prediction of z, considering the randomness problem.

The controller**(C)** decides the actions the agent would take to maximize the rewards. It receives the vector z, and the hidden state as the input, and it is a simple linear model with the formula $a_t = W_c[z_t h_t] + b_c$.

These three components would work in this way: after the agent getting the observation, the V would encode it into ***z***; then the C would decide to take which action according to the z and the hidden state h, and the agent would take the action in the environment which may change its state; at last, the M would receive the z, the action to update the hidden state h. The detail is also shown in the flow diagram.



Example: Using the VMC model in Car Racing

In the Car Racing game, each time the environment would generate random tracks. The aim of the agent is to get more rewards in limited time, and the agent can take four actions: steering left/right, acceleration and brake.

Firstly, the agent uses a random action policy to explore the environment multiple times, and record the actions taken and the observations. Then the VAE can reconstruct the image by vector z, and obviously, some information would be lost in the new image. After getting the dataset, we use it to train the V model to learn a latent space. Then the M model could use the data which is preprocessed by the V model to model the $P(z_{t+1}|a_t, z_t, h_t)$. Lastly, we can train the controller model.

After the training, we can find the parameter count of each model that V model has 4,348,547, M model has 422,368 and C model has 867. And this meets our expectations that the world model is complexing and the controller model is a simple one.