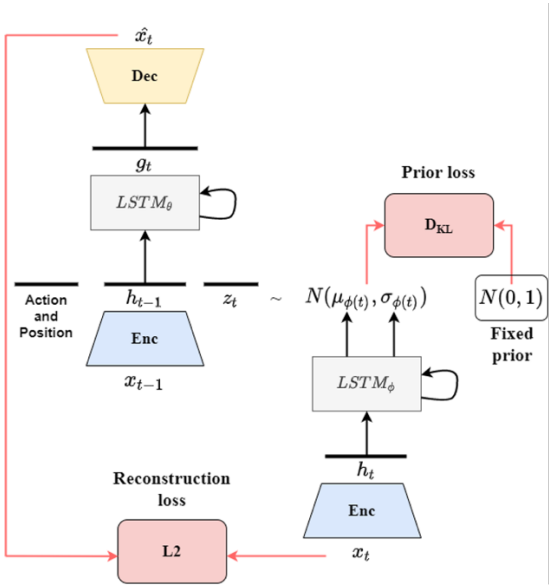
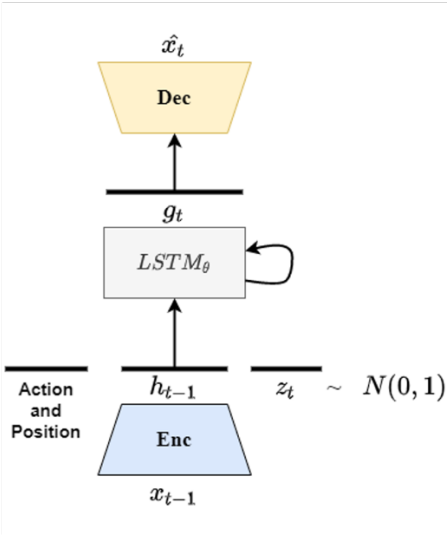
311551180_楊佳誠

Introduction

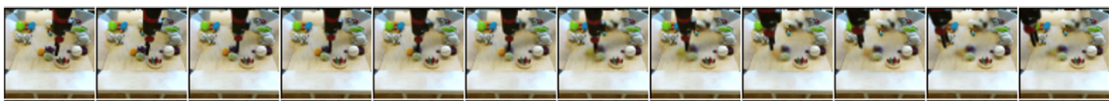
In this lab, the goal is to implement a conditional VAE to predict the next frame of video. The video is bair robot pushing small dataset, which includes 30 frames and “action and position” information in each data.



(a) Training procedure



(b) generating procedure



Derivation of CVAE

$$P(x) = \int_z P(z)\,P(x|z)\,dz$$

$$L = \sum_x \log P(x)$$

$$\log P(x) = \int_z q_\phi(z|x) \log P(x)\,dz$$

$$= \int_z q_\phi(z|x) \log\left(\frac{P(z,x)}{P(z|x)}\right) dz$$

$$= \int_z q_\phi(z|x) \log\left(\frac{P(z,x)}{q_\phi(z|x)} \frac{q_\phi(z|x)}{P(z|x)}\right) dz$$

$$= \int_z q_\phi(z|x) \log \frac{P(z,x)}{q_\phi(z|x)} dz + \int_z q_\phi(z|x) \log \frac{q_\phi(z|x)}{P(z|x)} dz$$

$$= \underbrace{\int_z q_\phi(z|x) \log \frac{P(x|z)\,P(z)}{q_\phi(z|x)}}_{L_b} + \underbrace{KL(q_\phi(z|x) \,\|\, p(z|x))}_{\geq 0}.$$

$$\log P(x) \geq L_b$$

$$L_b = \int_z q_\phi(z|x) \log \frac{P(x|z)\,P(z)}{q_\phi(z|x)} dz = \int_z q_\phi(z|x) \log \frac{P(z)}{q_\phi(z|x)} dz + \int_z q_\phi(z|x) \log p(x|z)\,dz$$

$$= -KL(q_\phi(z|x) \,\|\, P(z)) + \underbrace{\int_z q_\phi(z|x) \log p(x|z)\,dz}_{E_{z\sim q(z|x)}[\log p(x|z)]}$$

Goal: maximize L_b.

① → minimize $KL(q_\phi(z|x) \,\|\, P(z))$

② → maximize $E_{z\sim q(z|x)}[\log p(x|z)]$ → $L = E_{z\sim q(z|x)}[\log P(x|z)] - KL(q_\phi(z|x) \,\|\, P(z))$

CVAE 為加入 "given c" 的條件 → $L = E_{z\sim q(z|x,c)}[\log p(x|z,c)] - KL(q_\phi(z|x,c) \,\|\, P(z|c))$

① : $-\frac{1}{2}(\log\sigma^2 - \sigma^2 - \mu^2 + 1)$

② : $MSE(x, g_\phi(x|(p(z|x,c),c))$

Implementation details

- Encoder

  Use VGG16 architecture as an encoder, which can reduce the dimension of images. This encoder has a special technique, skip connection. It preserves the layer output of encoder and can be used as the input of decoder. This technique can significantly improve the model's ability to learn, because of knowing the features of each layer.

- Decoder

  The decoder architecture is mirror symmetry to the encoder, while the decoder is aim to raise the dimension to the original image dimension.

- Reparameterization trick

  The sample from z distribution process is not differentiable, which results in back-propogation can't be directly applied. The reparameterization trick can perfectly deal with this problem by treating random sampling as noise term.

A: Without Reparameterization

B: With Reparameterization

My implementation refers to the work shown bellowed:

[1] Vae reference code. https://github.com/pytorch/examples/tree/master/vae.

```
def reparameterize(self, mu, logvar):
    std = torch.exp(0.5*logvar)
    eps = torch.randn_like(std)
    return mu + eps*std
```

The formula of transfer logvar to std:

$$e^{\frac{1\log\sigma^2}{2}} = e^{\log\sigma} = \sigma$$

We can know sample z from a known distribution by reparameterization trick.

.

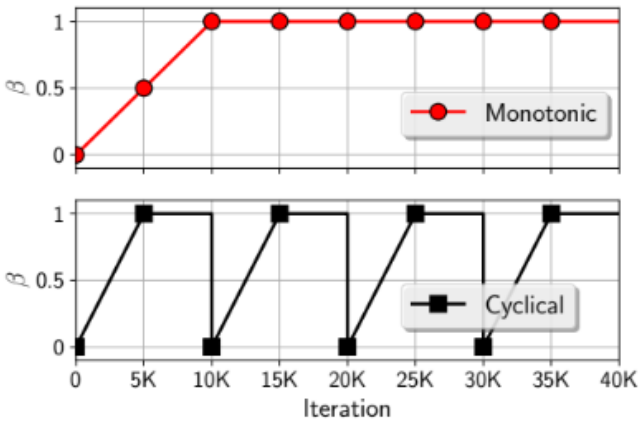
- Dataloader

  The dataloader is already implemented, it returns the seq(the batch data) and cond(action and position).

- KL annealing

  KL annealing give model more time to learn how to encode z. Specifically, if the model focuses too much on minimizing the KL divergence term, which encourages the latent variables to have a distribution close to a standard normal distribution. As a result, the latent space becomes over-compressed, and the representation of each sample in the latent space becomes less informative. Moreover, the latent variables become increasingly independent, which means that even if two samples are very similar in the original data space, their representations in the latent space may be very different, leading to poor modeling of the data distribution.

  To implement KL annealing, just pre-compute all the beta value based on the args.



- Describe the teacher forcing

(including main idea, benefits and drawbacks.) (5%)

The main idea of teacher forcing is to give the ground-truth sequence as input to the model during training. The benefits of teacher forcing are it can improve the stability. Without using teacher forcing, if an output from model is not satisfy, this output will still treat as the input of next frame, result in awful affect in training.

The drawback is that the model only learn the true input. So, when the output may lack of diversity. Or, if the unseen test data is far different from train data, it might not perform well.

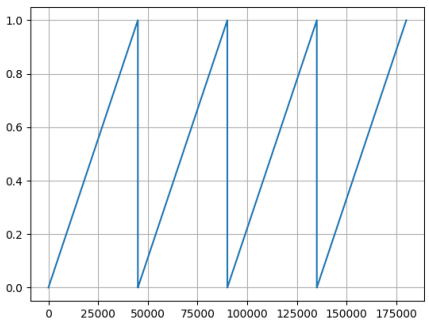
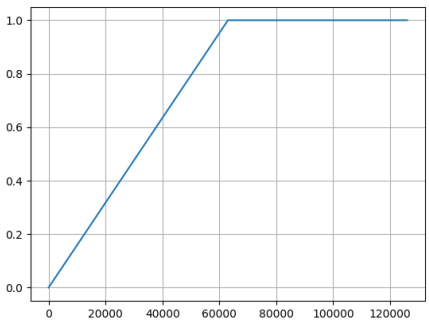
Results and discussion

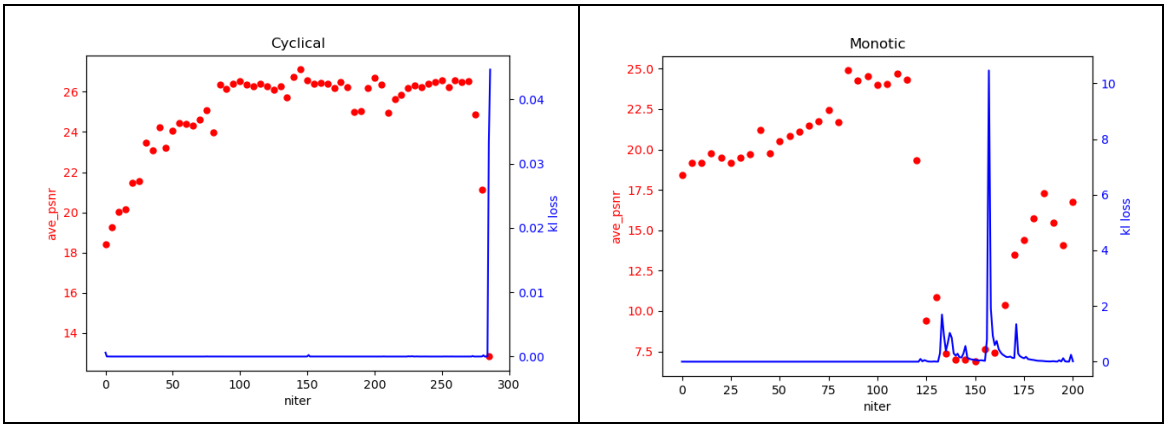
- Show your results of video prediction

  Pred:

  Truth:

- Plot the KL loss and PSNR curves during training

Cyclical	Monotic
niter = 290 epoch_size = 600	niter = 200 epoch_size = 420
KL_annealing beta curve	
	



- Discuss the results according to your setting of teacher forcing ratio, KL weight, and learning rate.

  It is obvious that whenever the KL_loss raise, then the average psnr will significantly decrease.

  In right picture, when KL's beta reach the peak, the model try to modify its parameter to ease the KL_loss. The performance gradually raise again during the train process.

  I set tfr_decay_step larger(100) so that the model will train better because it give more time to learn from teacher. When the tfr ratio start decay, the loss will raise.

  I didn't modify the learning rate. Learning rate is set to 0.002, and use default Adam optimizer to train the model.