Lab4 report

1. Introduction

In this assignment, I need to implement conditional video prediction in a VAE-based model. There are the following requirements:

(1) Implement Video prediction protocol in (train) stage

(2) Implement reparameterization tricks

(3) KL annealing implementation. (a) Cyclical. (b) Monotonic

(4) Implement Teacher forcing strategy

Dataset:

a. Training dataset

i. train_img: 23410 png files

ii. train_label: 23410 png files

b. Valadition dataset

i. val_img: 630 png files

ii. val_label: 630 png files

c. Testing dataset

i. 6 video sequences are given. Each video sequence contains one first frame and 630 label frames.

2. Implementation details

(1) How do you write your training protocol

```python
def training_one_step(self, img, label, adapt_TeacherForcing):
    img = img.permute(1, 0, 2, 3, 4) # change tensor into (seq, B, C, H, W)
    label = label.permute(1, 0, 2, 3, 4) # change tensor into (seq, B, C, H, W)
    decoded_frame_list = [img[0].cpu()]
    human_feat_hat_list = []
    label_list = []
    seq_loss = 0
    for i in range(0, self.train_vi_len):
        human_feat_hat = self.frame_transformation(img[i])
        human_feat_hat_list.append(human_feat_hat)

    for i in range(1, self.train_vi_len):
        label_feat = self.label_transformation(label[i])
        z, mu, log_var = self.Gaussian_Predictor(human_feat_hat_list[i], label_feat)
        if not adapt_TeacherForcing:
            human_feat_hat = self.frame_transformation(self.last_generated_frame)
        else:
            human_feat_hat = human_feat_hat_list[i-1]
        parm = self.Decoder_Fusion(human_feat_hat, label_feat, z)
        generated_frame = self.Generator(parm)
        reconstruction_loss = self.mse_criterion(generated_frame, img[i])
        self.last_generated_frame = generated_frame.detach()
        beta = self.kl_annealing.get_beta()
        loss = reconstruction_loss  + beta * kl_criterion(mu, log_var, self.batch_size)
        seq_loss = seq_loss + loss

    return seq_loss / self.train_vi_len
```

圖 1

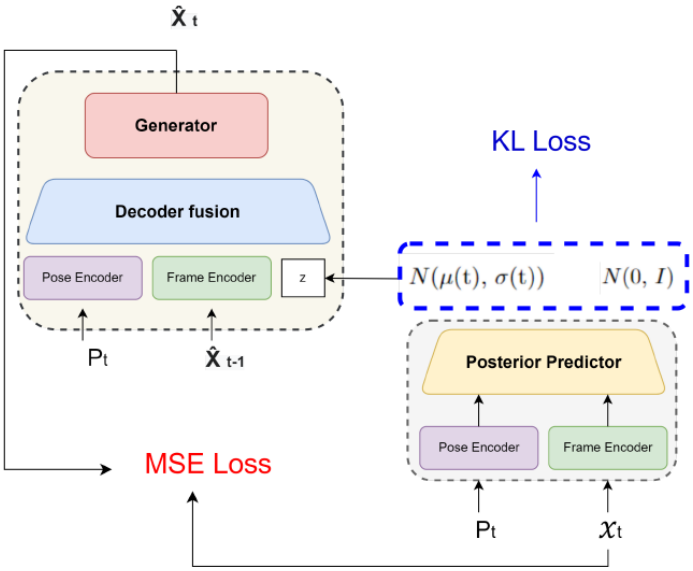


圖 2

從圖 1 的 code 中可以知道圖 2 就是我的訓練架構。在訓練的部分，我用 adapt_TeacherForcing 來調控 TeacherForcing 的開跟關，如果 TeacherForcing 是開著的，因為模型要輸入 Xt 跟 Xt-1，所以我設計先讓所有的 X 都進 frames encoder 一次，然後將輸出儲存成 list 需要時再取出，這樣可以避免每一個 iteration，Xt-1 都要重新進 frames encoder 一次，可以減少不少計算量。從圖一可以看出接下來是讓圖二的 Pt 進入 pose encoder，並且將輸出與之前儲存在 list 中已經 encoder 的 Xt，一起輸入進圖二的 posterior predictor，算出 Zt、平均值、變異數，家下來如果 TeacherForcing 是開著的話，就是讓已經 encoder 的 Xt-1 跟 Pt 進入圖二的的 decoder fusion，在進入 generator 就可以生成圖片了。圖 1 中 Loss 的部分，我一樣是按照圖 2 先算 kl diverse 再用預測的圖片跟原始的輸入 Xt 算 mse，唯一跟圖 2 不太一樣的地方是我的 loss 最後是將整個 seq 的 loss 加起來取平均再 optimize，這樣可以讓模型權重不用每張圖輸入就 optimize 一次，減少計算量的同時，模型效能也不太會減弱。

(2) How do you implement reparameterization tricks

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

第一行是從 log variance 算出標準差，第二行是從 normal distribution sample noise 也就是 epsilon，第三行是將平均值（mu）與縮放的隨機 noise（epsilon）

結合，生成來自目標分布的 sample。這個操作使得 sample 過程可微分，這對於 gradient 的優化非常重要。透過使用這種 reparameterization tricks，允許 gradient 在模型中 back propagation，同時仍然在 sample 過程中引入隨機性。這有助於訓練有 laten variable 的生成模型。

(3) How do you set your teacher forcing strategy

```python
def teacher_forcing_ratio_update(self):
    if self.current_epoch % 5 == 0:
        self.tfr = max(self.tfr - 0.05, 0)
```

我的 teacher forcing strategy 就是讓模型訓練時，每 5 個 epoch teacher forcing ratio 就下降 0.05，讓模型能夠逐漸適應自己生成的圖片。

(4) How do you set your kl annealing ratio

```python
class kl_annealing():
    def __init__(self, args, current_epoch=0):
        self.strategy = args.strategy
        self.beta = 0.0
        self.max_beta = 1.0
        self.anneal_rate = args.anneal_rate
        self.anneal_interval = args.anneal_interval
        self.current_epoch = current_epoch
        self.cycle_length = args.cycle_length if self.strategy == 'cyclical' else None

    def update(self):
        if self.strategy == 'cyclical':
            cycle_position = self.current_epoch % self.cycle_length
            self.beta = min(self.max_beta, cycle_position / (self.cycle_length / 2))
            if cycle_position > self.cycle_length / 2:
                self.beta = self.max_beta - (cycle_position - self.cycle_length / 2) / (self.cycle_length / 2) * self.max_beta
                linear_schedule = self.frange_cycle_linear(self.cycle_length, start=0.0, stop=self.max_beta)
                self.beta = linear_schedule[cycle_position]

        elif self.strategy == 'monotonic':
            self.beta += self.anneal_rate
            self.beta = min(self.beta, self.max_beta)

        elif self.strategy == 'none':
            self.beta = self.max_beta

        self.current_epoch += 1
```

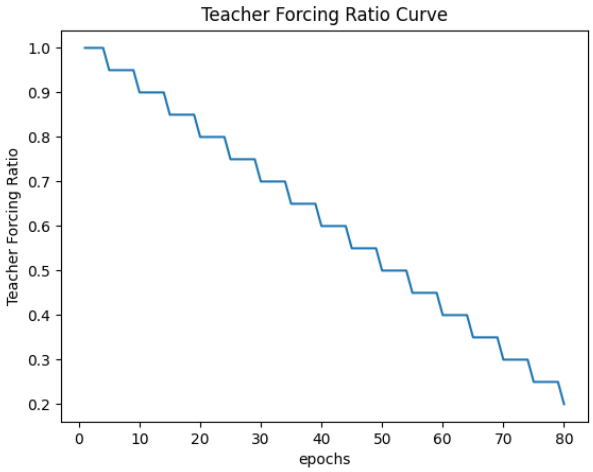
```python
def get_beta(self):
    return self.beta

def frange_cycle_linear(self, n_iter, start=0.0, stop=1.0, n_cycle=4, ratio=0.5):
    L = np.ones(n_iter) * stop
    period = n_iter / n_cycle
    step = (stop - start) / (period * ratio)  # linear schedule

    for c in range(n_cycle):
        v, i = start, 0
        while v <= stop and (int(i + c * period) < n_iter):
            L[int(i + c * period)] = v
            v += step
            i += 1
    return L
```

從上面兩張圖可以看出我實作了 Cyclical、Monotonic 這兩種 KL annealing，我的 Monotonic 是讓 beta 一開始是 0，每個 epoch 上升 0.01，直到 1 就停下來，Cyclical 是週期性的變化，我的 beta 值是以 10 個 epoch 為一個週期，下圖是他的週期變化，會一直重複。

Epoch	Beta Value
0	0.0
1	0.2
2	0.4
3	0.6
4	0.8
5	1.0
6	0.8
7	0.0
8	0.8
9	1.0

3. Analysis & Discussion

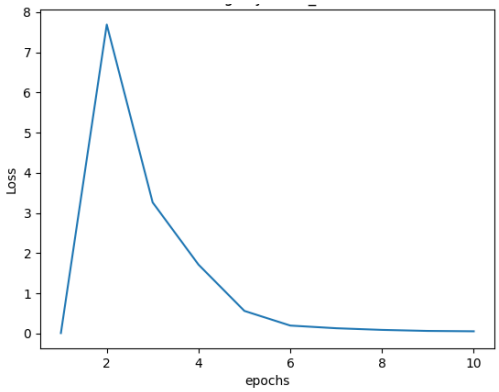
(1) Plot Teacher forcing ratio



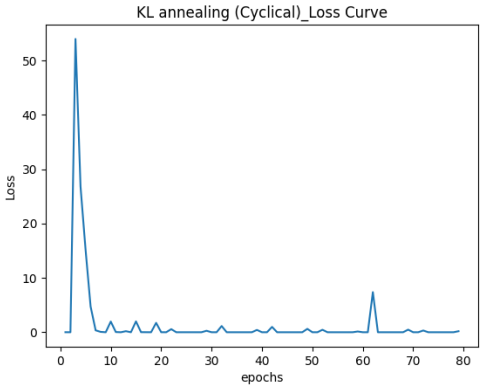
我每隔 5 個 epoch 下降 0.05 的 teacher forcing ratio，從下面 Cyclical loss 圖可以看出前期接近每 5 個 epoch，都會突然上升一點，然後又下回去，後期要接近每 10 個 epoch，才會突然上升一點，可以看到模型有逐漸不在依賴原始的訓練資料。

(2) Plot the loss curve while training with different settings

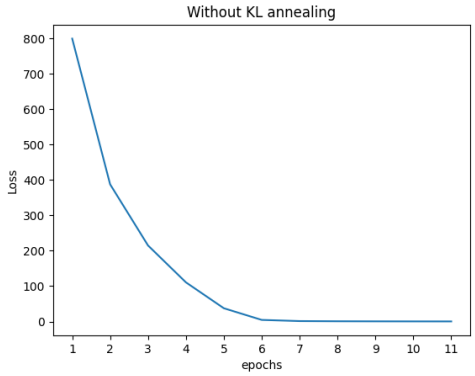
a. With KL annealing (Monotonic)



b. With KL annealing (Cyclical)



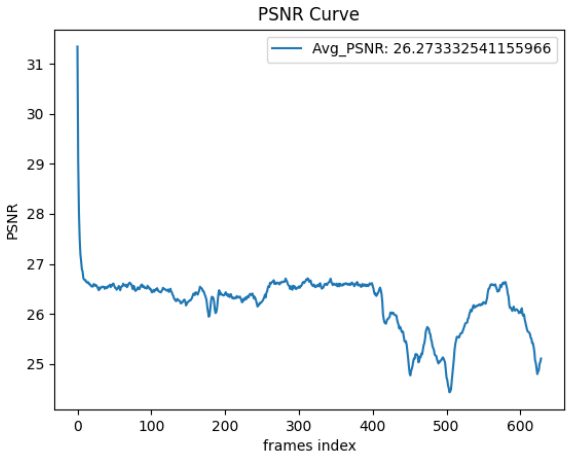
c. Without KL annealing



從上面三張圖的前 10 個 epoch 可以發現，Without KL annealing 會讓一開始的 beta 就是 1，導致模型對 laten space 的限制很強，模型可能完全忽視 laten variable，造成生成的圖片質量較低，loss 變很大， KL annealing 用 Monotonic 的話，因為一開始 beta 是 0，我的設定是每一個 epoch 增加 0.01，所以可以讓模型對 laten space 的限制較弱，生成的圖片可能會更接近訓練數據，使的 loss 可以比其他方法低，但是生成的圖片在 beta 還很小時，多樣性可能不太夠，造成 valid 時模型性能較差，等到 beta 變成 1 時，又有可能因為 beta 都不會動了，導致模型過早收斂到局部最優解，降低模型的泛化能力，最後如果用 Cyclical，因為我每個 epoch

增加 0.2，可以讓 loss 雖然一開始比 Monotonic 高，可是也因為 beta 不會太小，valid 時模型性能反而較好，而且因為他的 beta 值是週期性的，所以可以讓他可以在訓練過程中反覆調整 beta 值，從而讓模型在不同階段關注不同的 loss function，可能有助於防止模型過早收斂到局部最優解，提高模型的泛化能力。

(3) Plot the PSNR-per frame diagram in validation dataset



上圖我是用 KL annealing (Cyclical)，跑 77 個 epoch，然後再 validation dataset 算 psnr 得到的結果。

(4) Derivate conditional VAE formula

$\therefore p(x,z,c;\theta) = p(x,c;\theta)\, p(z|x,c;\theta)$

$\therefore \log p(x,z,c;\theta) = \log p(x,c;\theta) + \log p(z|x,c;\theta)$

$\Rightarrow \log p(x|c;\theta) = \log p(x,z|c;\theta) - \log p(z|x,c;\theta)$

We next introduce an arbitrary distribution $q(z|c)$ on both sides and integrate on z

$$\int q(z|c) \log p(x|c;\theta)\, dz$$

$$= \int q(z|c) \log p(x,z|c;\theta)\, dz - \int q(z|c) \log p(z|x,c;\theta)\, dz$$

$$= \int q(z|c) \log p(x,z|c;\theta)\, dz - \int q(z|c) \log q(z|c)\, dz + \int q(z|c) \log q(z|c)\, dz - \int q(z|c) \log p(z|x,c;\theta)\, dz$$

$$= \int q(z|c) \log p(x,z|c;\theta)\, dz - \int q(z|c) \log q(z|c)\, dz + \int q(z|c) \log \frac{q(z|c)}{p(z|x,c;\theta)}\, dz$$

$$= \int q(z|c) \log p(x,z|c;\theta)\, dz - \int q(z|c) \log q(z|c)\, dz + KL(q(z|c) \,\|\, p(z|x,c;\theta))$$

$$\geq \int q(z|c) \log p(x,z|c;\theta)\, dz - \int q(z|c) \log q(z|c)\, dz$$

$$= \int q(z|c) \log \frac{p(x,z|c;\theta)}{q(z|c)}\, dz$$

$$= \int q(z|c) \log \frac{p(z|x,c;\theta)\, p(x|c;\theta)}{q(z|c)}\, dz$$

$$= \int q(z|c) \log p(x|c;\theta)\, dz + \int q(z|c) \log \frac{p(z|x,c;\theta)}{q(z|c)}\, dz$$

$$= \log p(x|c;\theta) - KL(q(z|c) \,\|\, p(z|x,c;\theta))$$

$$\Rightarrow \log p(x|c;\theta) \geq \int q(z|c) \log \frac{p(x,z|c;\theta)}{q(z|c)}\, dz$$

$$= \int q(z|c) \log p(x,z|c;\theta)\, dz - \int q(z|c) \log q(z|c)\, dz$$

$$= \int q(z|c) \log p(x|z,c;\theta)\, dz + \int q(z|c) \log p(z|c)\, dz - \int q(z|c) \log q(z|c)\, dz$$

$$= \int q(z|c) \log p(x|z,c;\theta)\, dz - KL(q(z|c) \,\|\, p(z|c))$$

$$= E_{z \sim q(z|x,c;\theta)} [\log p(x|z,c;\theta)] - KL(q(z|c) \,\|\, p(z|c))$$

$$= E_{z \sim q(z|x,c;\theta)} [\log p(x|z,c;\theta)] + \int q(z|c) \log p(z|c)\, dz - \int q(z|c) \log q(z|c)\, dz$$

$$= E_{z \sim q(z|x,c;\theta)} [\log p(x|z,c;\theta)] + \int q(z|c) \log \frac{p(z|c)}{q(z|c)}\, dz$$

$$= E_{z \sim q(z|x,c;\theta)} [\log p(x|z,c;\theta)] - KL(q(z|c) \,\|\, p(z|c))$$

$$= L(x,c,q;\theta)$$

In other words, $L(x,c,q;\theta)$ is a lower bound on $\log p(x|c;\theta)$ with equality if and only if

$$q(z|c) = p(z|x,c;\theta)$$

where

$$L(x,c,q;\theta) = \int q(z|c) \log p(x|z,c;\theta)\, dz - KL(q(z|c) \,\|\, p(z|c))$$

$$= \int q(z|c) \log p(x|z,c;\theta)\, dz + \int q(z|c) \log p(z|c)\, dz - \int q(z|c) \log q(z|c)\, dz$$

$$= \int q(z|c) \log p(x,z|c;\theta)\, dz - \int q(z|c) \log q(z|c)\, dz$$

$$= \int q(z|c) \log \frac{p(x,z|c;\theta)}{q(z|c)}\, dz$$

$$= \int q(z|c) \log \frac{q(z|x,c;\theta)\, p(x|c;\theta)}{q(z|c)}\, dz$$

$$= \int q(z|c) \log p(x|c;\theta)\, dz + \int q(z|c) \log \frac{q(z|x,c;\theta)}{q(z|c)}\, dz$$

$$= \log p(x|c;\theta) - KL(q(z|c) \,\|\, p(z|x,c;\theta))$$

$$\leq \log p(x|c;\theta)$$

$$= \int q(z|c) \log p(x,z|c;\theta)\, dz - \int q(z|c) \log q(z|c)\, dz$$

$$= \int q(z|c) \log p(x|z,c;\theta)\, dz + \int q(z|c) \log p(z|c)\, dz - \int q(z|c) \log q(z|c)\, dz$$

$$= \int q(z|c) \log p(x|z,c;\theta)\, dz - KL(q(z|c) \,\|\, p(z|c))$$

$$= E_{z \sim q(z|x,c;\theta)} [\log p(x|z,c;\theta)] - KL(q(z|c) \,\|\, p(z|c))$$

$$= L(x,c,q;\theta)$$