Lab 4 - Conditional VAE for Video Prediction

312551133
鄧長軒

Derivate conditional VAE formula

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

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

$$\int q_\phi(z|x,c;\phi) \log p(x|z,c;\theta)\, d(z|x,c;\phi)$$

$$= \int q_\phi(z|x,c;\phi) \log p(x,z|c;\theta)\, d(z|x,c;\phi) - \int q_\phi(z|x,c;\phi) \log p(z|c;\theta)\, d(z|x,c;\phi)$$

$$= \int q_\phi(z|x,c;\phi) \log p(x,z|c;\theta)\, d(z|x,c;\phi) - \int q_\phi(z|x,c;\phi) \log q_\phi(z|x,c;\phi)\, d(z|x,c;\phi)$$

$$+ \int q_\phi(z|x,c;\phi) \log q_\phi(z|x,c;\phi)\, d(z|x,c;\phi) - \int q_\phi(z|x,c;\phi) \log p(z|c;\theta)\, d(z|x,c;\phi)$$

Derivate conditional VAE formula

$$\rightarrow \int q(z|x,c;\phi)\log p(x|z,c;\theta)\,d(z|x,c;\phi) = E_{q(z|x,c;\phi)}\log p(x|z,c;\theta)$$

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

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

$$E_{q(z|x,c;\phi)}\log p(x|z,c;\theta) = L(x,c,q,\theta) + KL(q(z|x,c;\phi)\,||\,p(z|c;\theta))$$

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

Introduction

在Lab4中，我實作了VAE模型，將圖片與骨架encode到分布空間，再從裡面decode產生出圖片，其中用到reparameterization tricks, teacher forcing strategy 與 kl annealing ratio。使用的訓練資料為連續的23410張圖片與骨架label，可以合成出影片，目標是用1張初始圖片與630張骨架label，預測出完整的影片。最後在kaggle上拿到24.04214的分數。

Implementation details

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)
    label = label.permute(1, 0, 2, 3, 4)
    self.train()
    total_loss=0
    prev_img = img[0]
    for i in range(self.train_vi_len-1):
        if adapt_TeacherForcing:
            frame_t1=self.frame_transformation(img[i])
        else:
            frame_t1=self.frame_transformation(prev_img)
        frame_t2=self.frame_transformation(img[i+1])
        pose=self.label_transformation(label[i+1])
        z, mu, logvar=self.Gaussian_Predictor(frame_t2, pose)
        decoder=self.Decoder_Fusion(frame_t1, pose, z)
        output_img=self.Generator(decoder)
        prev_img = output_img
        MSE_loss = self.mse_criterion(output_img, img[i+1])
        kl_loss = kl_criterion(mu, logvar, args.batch_size)
        total_loss += MSE_loss + self.kl_annealing.get_beta() * kl_loss
    self.optim.zero_grad()
    total_loss.backward()
    self.optimizer_step()
    return total_loss
```

1. 首先使用Tester裡的做法，將img與label重新排列。

2. prev_img儲存前一張圖片，用於預測下一張圖片，一開始是第一張圖片。

3. 根據adapt_TeacherForcing來決定是否要用正確圖片來預測，是的話就用img，否的話用prev_img。

4. 通過模型得到預測出的圖片output_img，將它存入prev_img，以供下一張預測。

5. 最後算出MSE loss + beta * KL loss，更新參數。

How do you implement reparameterization tricks

用torch.randn_like建立N(0,1)高斯分布，形狀為mu。
乘上logvar，加上mu後回傳。

```
def reparameterize(self, mu, logvar):
    nd = torch.randn_like(mu)
    z = mu + nd * logvar
    return z
```

How do you set your teacher forcing strategy

如果當前的epoch大於tfr_sde，代表需要decay，所以減去trf_d_step。
用max跟min來讓ratio保持在0~1之間。

```
def teacher_forcing_ratio_update(self):
    if self.current_epoch >= self.args.tfr_sde:
        self.tfr = max(0.0, self.tfr - self.args.tfr_d_step)
    self.tfr = min(1.0, self.tfr)
```

How do you set your kl annealing ratio

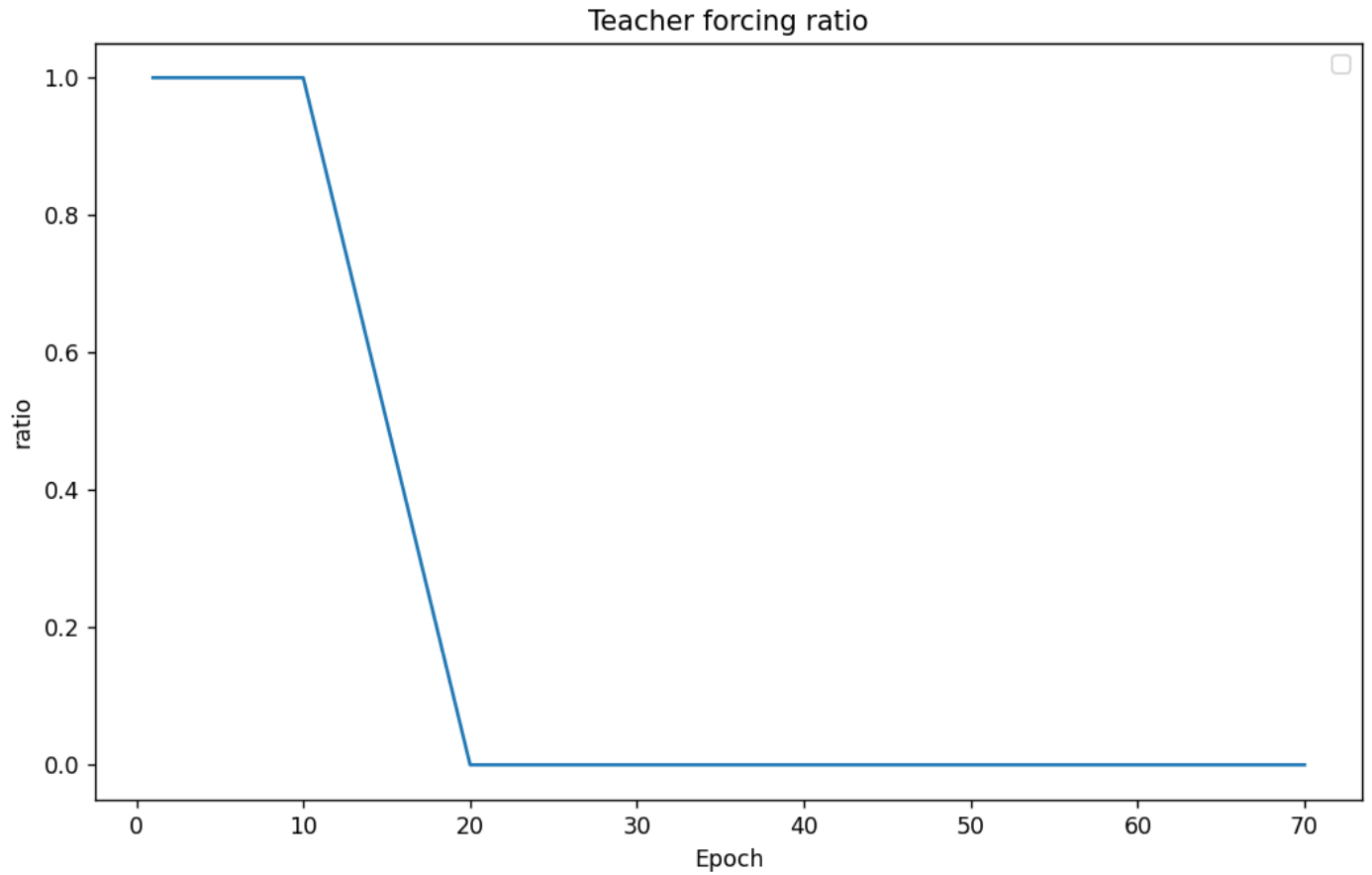
```python
class kl_annealing():
    def __init__(self, args, current_epoch=0):
        self.args = args
        self.current_epoch = current_epoch
        self.beta = 0.0

    def update(self):
        self.current_epoch += 1
        if self.args.kl_anneal_type == 'Cyclical':
            self.beta = self.frange_cycle_linear(self.current_epoch, n_cycle=self.args.kl_anneal_cycle, ratio=self.args.kl_anneal_ratio)
        else:
            self.beta = min(1.0, self.current_epoch / self.args.kl_anneal_cycle)

    def get_beta(self):
        return self.beta

    def frange_cycle_linear(self, n_iter, start=0.0, stop=1.0,  n_cycle=1, ratio=1):
        step = n_cycle*ratio
        iter = n_iter%n_cycle
        return min(1.0, iter/step)
```

How do you set your kl annealing ratio

- __init__：初始化beta為0。
- update：根據current_epoch來算出beta，如果是Cyclical，就由另一個函式來算；如果是monotonic，則當epoch小於cycle時，beta等於epoch/cycle。
- get_beta：return beta。
- frange_cycle_linear：先算出在cycle內第幾個epoch之後beta會是1，再把當前的epoch除以cycle取餘數，如果iter大於step，beta就為1，iter小於step，beta為iter/step。

Analysis & Discussion

Plot Teacher forcing ratio



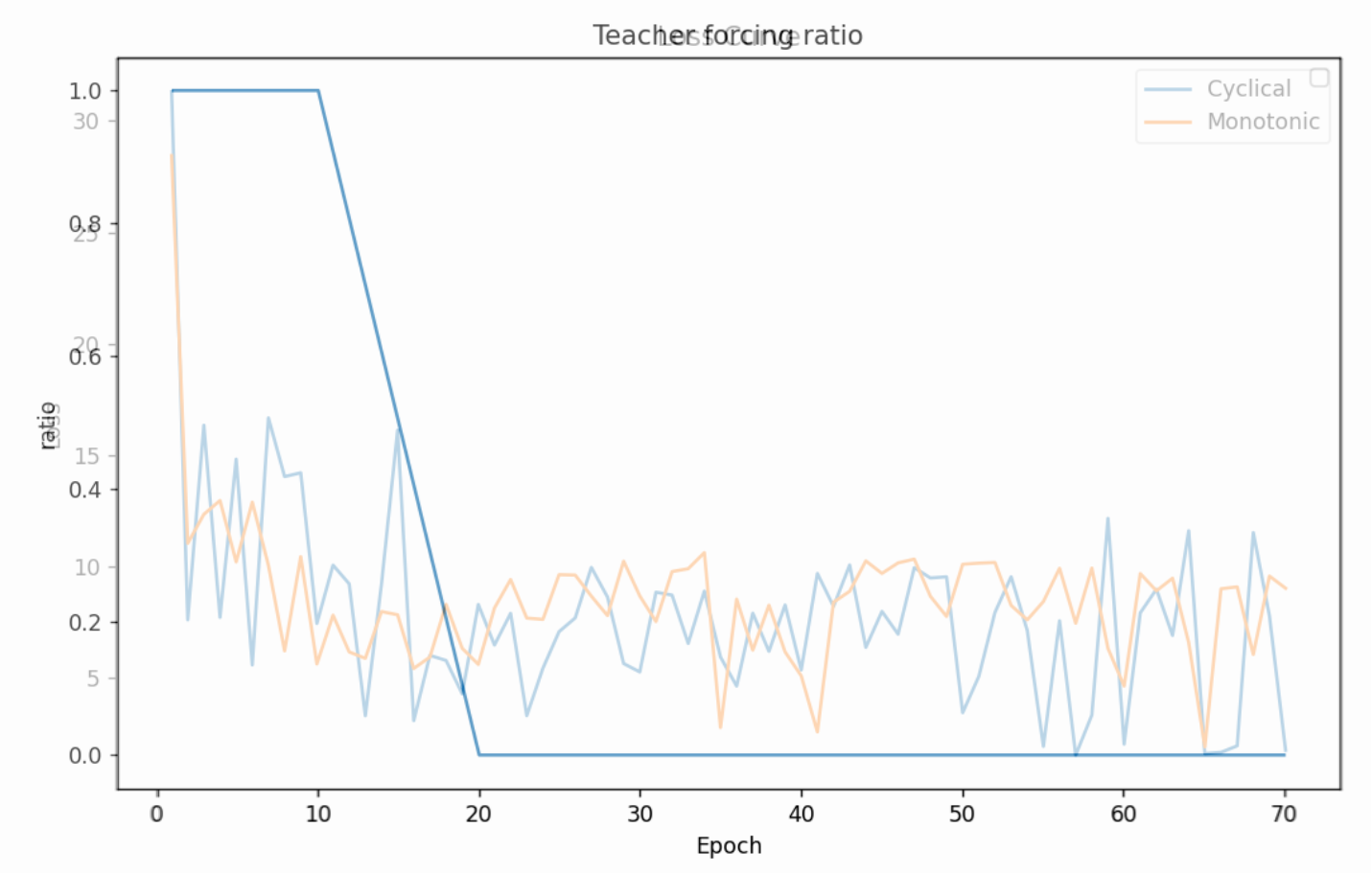
Teacher forcing ratio

在epoch<10時等於1。

在epoch=10~20時，以decay=0.1下降。

在epoch>20時等於0。

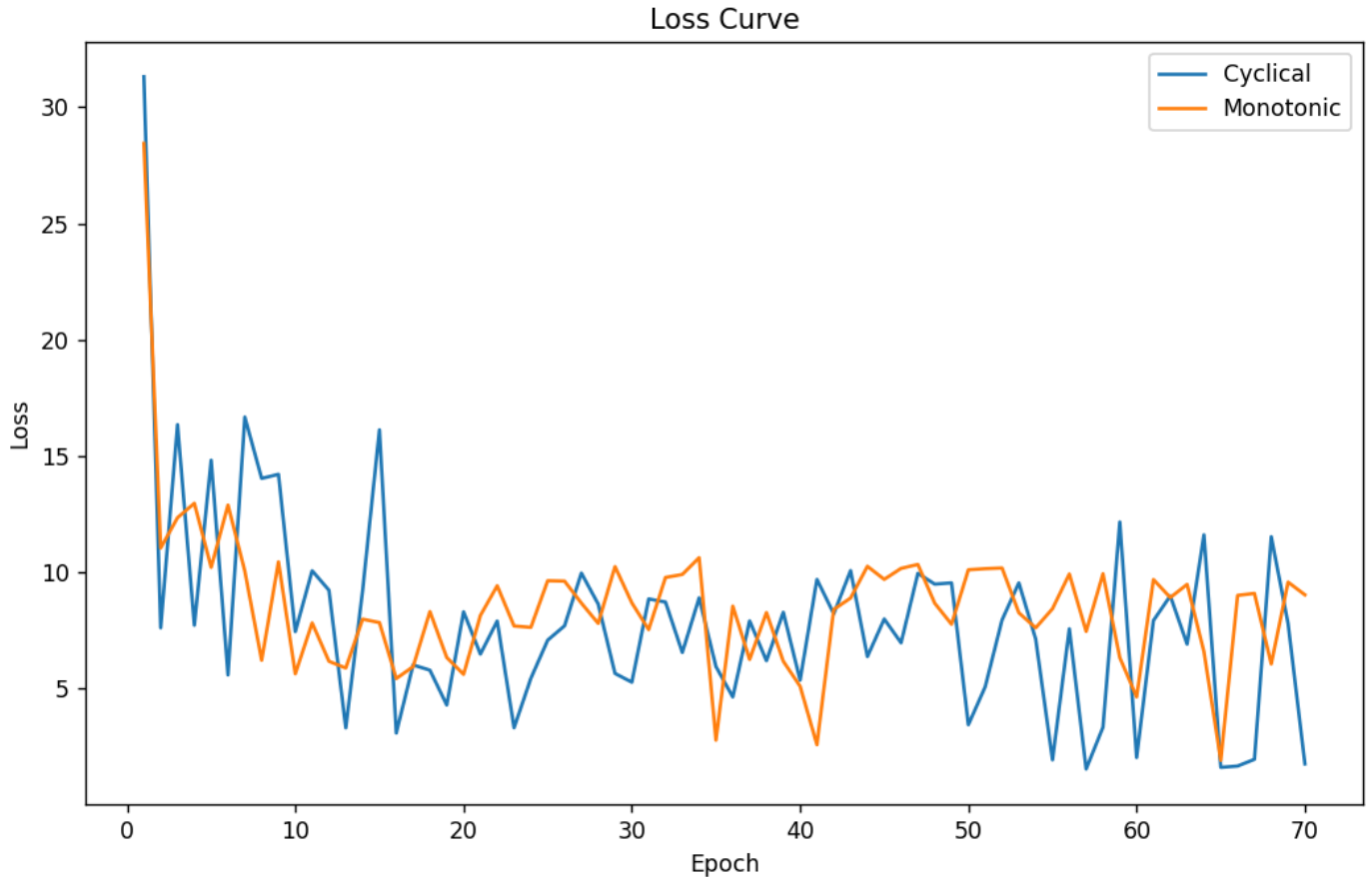
Analysis & compare



Teacher forcing ratio在epoch=10~20時下降，而loss在epoch=20~30有上升的趨勢。

推測是因為在epoch=10時，模型學到用正確圖片預測正確圖片，但當epoch=20時，變成要用不一定正確的圖片來預測，導致預測結果與正確圖片差更多，因此loss上升。

Plot the loss curve while training with different settings.

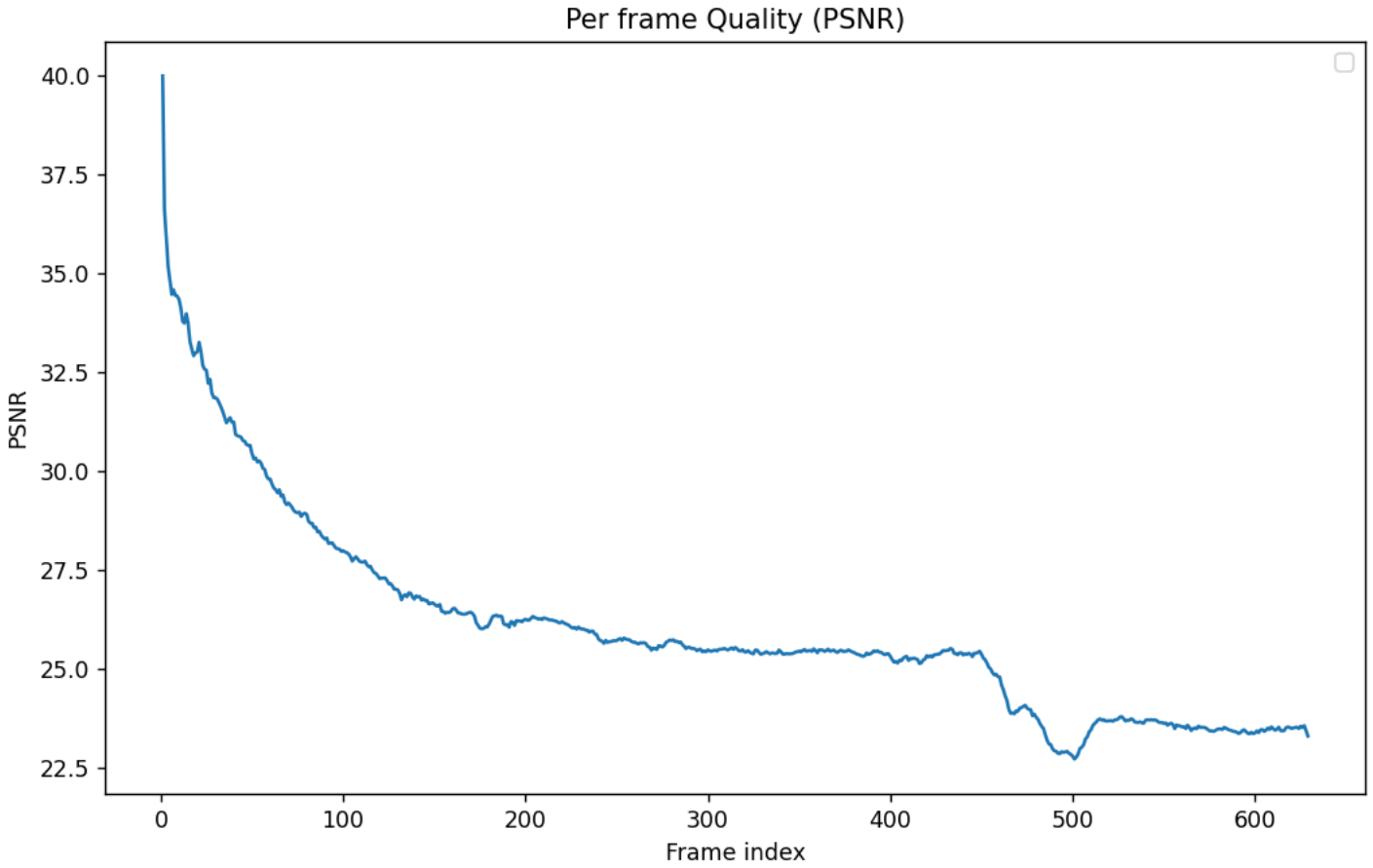


Monotonic：
曲線較平緩，因為beta大部分時候保持為1，算出的總loss不會有太大的浮動。

Cyclical：
曲線較崎嶇，因為beta會一直變，同樣的圖片在不同epoch，算出的loss會不一樣，導致模型可能無法學習正確答案。但由於震盪幅度大，有可能震到loss更小的地方。

Without KL annealing：
loss會算出nan，因為一開始就把全部的kl_loss拿去更新參數，數字太大了。

Plot the PSNR-per frame diagram in validation dataset



Other training strategy analysis

我使用了以下的方法

1. 調整learning rate：結果不變。
2. 調整num_epoch：結果不變。
3. 調整Kl annealing：結果變差。
4. 使用gradient accumulate：結果人變得像黑猩猩。
5. 調小\train_vi_len：背景產生雜訊。
6. 把Adam更換成SGD：訓練時loss無法下降。
7. 調整Teacher Forcing：結果不變。

Result

- 使用了很多調整方法，最後還是預設的參數表現最好。
- Monotonic的平均loss較小，Cyclical的平均loss較大，但由於Cyclical震盪幅度大，所以它的最小loss比Monotonic還小。