NYCU DLP Lab4 – Conditional VAE for Video Prediction

312551086 張紀睿

i. Introduction

本次作業中，訓練 VAE 進行影片預測，將輸入的首幀畫面與骨架圖片 encode，並由 Gaussian Predictor 預測 mu, logvar 並進行 reparameterize，再由 Generator 產生下一幀的圖片，重複此步驟以完成預測。程式部分，實作了 Training 與 Evaluation 的基本步驟，Gaussian Predictor 的 Reparameterization，KL annealing 以及 Teacher forcing rate 的調整。

ii. Implementation details

1. How do you write your training protocol

```python
def training_one_step(self, img, label, adapt_TeacherForcing):
    # TODO
    img = img.permute(1, 0, 2, 3, 4)
    label = label.permute(1, 0, 2, 3, 4)
    out = img[0]

    reconstruction_loss = 0.0
    kl_loss = 0.0
    for i in range(1, self.train_vi_len):
        label_feat = self.label_transformation(label[i])
        if self.mode == 1:
            out = img[i-1] * self.tfr + out * (1 - self.tfr)
        elif adapt_TeacherForcing:
            out = img[i-1]
        human_feat_hat = self.frame_transformation(out)

        z, mu, logvar = self.Gaussian_Predictor(human_feat_hat, label_feat)

        parm = self.Decoder_Fusion(human_feat_hat, label_feat, z)
        out = self.Generator(parm)

        reconstruction_loss += self.mse_criterion(out, img[i])
        kl_loss += kl_criterion(mu, logvar, batch_size = self.batch_size)

    beta = torch.tensor(self.kl_annealing.get_beta()).to(self.args.device)
    loss = reconstruction_loss + beta * kl_loss

    self.optim.zero_grad()
    loss.backward()
    self.optimizer_step()

    return loss
```

第一個 frame 會作為 input 輸入 frame_transformation，將其輸出與 label 輸入 label_transformation 產生的輸出傳入 Gaussian Predictor，以取得 mu，logvar 與分布 z。mu 與 logvar 用於計算 kl_loss；z 與 frame, label 一同輸入 Decoder Fusion，再將其輸出由 Generator 產生 output frame。Teacher Forcing 部分我設計了兩種方式:

(1) 若 teacher forcing 為 True，則以訓練集中的原圖做為模型輸入，否則以前一輪 Generator 輸出作為新一輪的輸入。

(2) 根據 tfr 的值乘上原圖加上 1-tfr 乘上前一輪的 output 後相加作為輸入，透過減少原圖佔比漸近式的讓模型學習，而非機率性讓模型直接輸入原圖。

所有 frame 輸入完畢後，從 kl_annealing 中取得 beta 值，將 reconstruction loss 與 kl_loss*beta 總合為最終的 loss 並更新模型權重。

2. How do you implement reparameterization tricks

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

從 logvar 取得 std，用 torch.randn_like 隨機抽樣後，乘上 std 並加上 mu 取得 z，而非直接由 N(mu, std)取樣，以完成 reparameterize。

3. How do you set your teacher forcing strategy

```python
def teacher_forcing_ratio_update(self):
    #TODO
    if self.current_epoch >= self.tfr_sde and self.tfr > 0:
        #self.tfr *= self.tfr_d_step
        self.tfr -= self.tfr_d_step
    self.tfr = max(self.tfr, 0)
```

原先採取再超過 tfr_sde 時每次 epoch 乘上一個 step 的方式，但經過測試後發現使用線性的方式每次減少 tfr_d_step 的訓練結果更為穩定一些，因此最終使用此方法更新 tfr。

4. How do you set your kl annealing ratio

```python
class kl_annealing():
    def __init__(self, args, current_epoch=0):
        self.anneal_type = args.kl_anneal_type
        self.anneal_cycle = args.kl_anneal_cycle
        self.anneal_ratio = args.kl_anneal_ratio
        self.current_epoch = -1
        if self.anneal_type == "Cyclical":
            self.L = self.frange_cycle_linear(args.num_epoch)
        elif self.anneal_type == "Monotonic":
            end_period = False
            L = self.frange_cycle_linear(args.num_epoch)
            for i in range(1, args.num_epoch):
                if end_period:
                    L[i] = 1.0
                    continue
                if L[i] < L[i-1]:
                    L[i] = 1
                    end_period = True
            self.L = L
        else:
            self.L = np.ones(args.num_epoch) * self.anneal_ratio
        self.beta = self.L[0]
        print("kl_annealing schedule:", self.L)

    def update(self):
        # TODO
        self.current_epoch += 1
        self.beta = self.L[self.current_epoch]
```

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

def frange_cycle_linear(self, n_iter, start=0.0, stop=1.0):
    # TODO
    n_cycle=self.anneal_cycle
    ratio=self.anneal_ratio

    L = np.ones(n_iter) * stop
    period = n_iter/n_cycle

    step = (stop - start)/(period * ratio)

    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
```

初始化時，由 frange_cycle_linear 取得包含每個 epoch 的 beta 值的 list，每次 update 就將 current epoch+1 並由此 index 取得對應 beta 值。

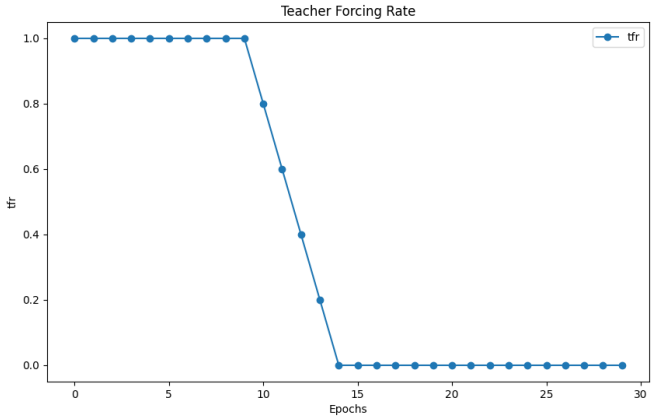
frange_cycle_linear 部分，參考 " Cyclical Annealing Schedule: A Simple Approach to Mitigating KL Vanishing" 論文的 source code 作法，以線性方式更新各 epoch 的 beta 值，由 ratio 調整 linear 的斜率(beta 值上升的速度)，並在每個 period 結束時回到 start 值。由於實測時當 beta 值過大會造成訓練不穩定，因此設定 ratio 為 5，使上升更為平緩。

另外，若選擇 Monotonic 方式，第一個 period 結束後會將後續 beta 值都設為 1；不使用 kl_annealing 時則直接將 anneal_ratio 作為 beta 值。

iii. Analysis & Discussion

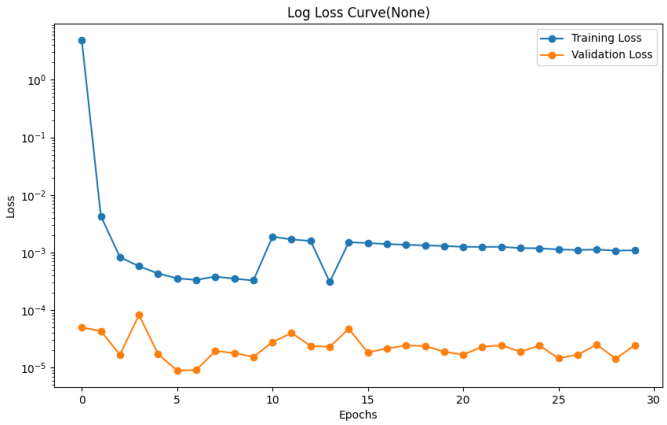
由於 **Part ii-1-(2)**的方式較晚設計完成，來不及測試在所有 model 設定上，因此此處的 loss curve 會是以 **Part ii-1-(1)**的 Teacher Forcing 策略進行訓練。**Part ii-1-(2)**的分析將於 **Teacher Forcing method (2) (additional)**章節補充。

1. Plot Teacher forcing ratio



a. Analysis & compare with the loss curve

為了讓分析結果不被 beta 值的改變影響，此處我以沒有使用 KL_annealing 的 loss curve 來做分析。



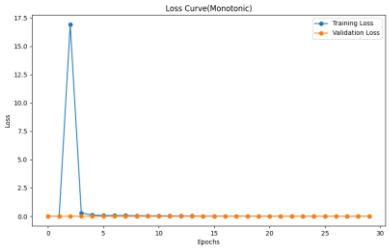
由 tfr curve 可以看到 tfr 在 epoch 10 開始下降，而 train loss 在 epoch 10 的時候突然上升，是因為此時 teacher forcing 被設成 false，模型使用自己預測出的輸出做為 input 而非正確的圖像，到 epoch 13 時 train loss 又有明顯下降，則是 teacher forcing 此時被設為 True 了。

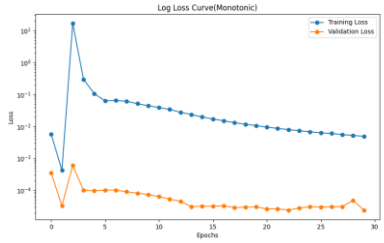
在 epoch 14 以後 tfr 為 0，因此之後都會是由模型 output 作為輸入而非正確圖像，因此之後的 train loss 都在比較高的位置慢慢下降，而沒有出現 epoch 13 時突然下降的狀況。

2. Plot the loss curve while training with different settings. Analyze the difference between them

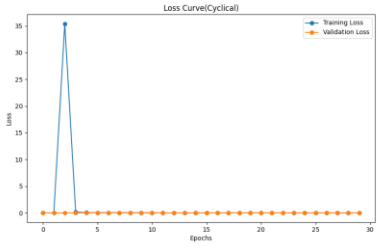
由於 loss 的數值差距較大，因此除了一般的 loss curve 外，也 plot 了 log loss curve 以利觀察 loss 變化趨勢。。

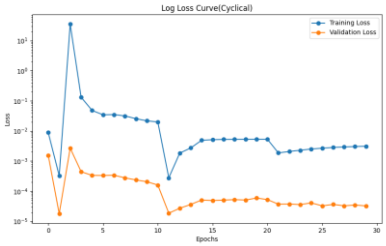
a. With KL annealing (Monotonic)



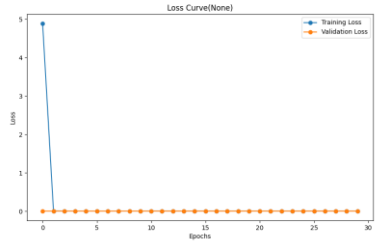


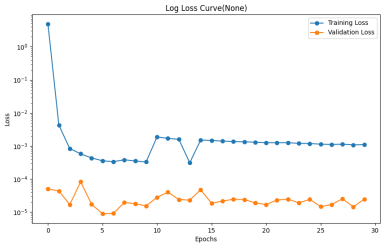
b. With KL annealing (Cyclical)





c. Without KL annealing

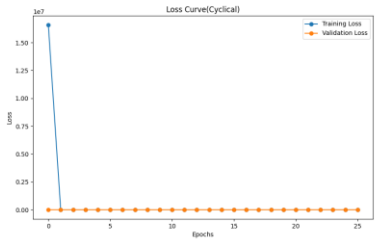


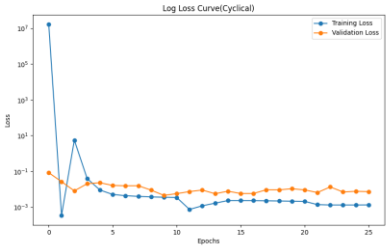


Result Analyze:

兩個 with kl_annealing 的 loss 都在 epoch 3 時突然飆升，而 without kl_annealing 的則是第一個 epoch loss 就是最高的，之後就下降了，這是因為前兩者的 kl loss 在最初占比非常低，要經過兩個 epoch 後 beta 值慢慢上升，kl loss 占比才變大，此時由於初期 kl loss 值非常大，導致在第一次 kl loss 有明顯的佔比時 loss 飆升的狀況。之後，Monotonic 的 loss 就是一直緩緩下降的趨勢，因為 beta 值在慢慢上升到 stop 以後就固定了，而 Cyclical 時每過 10 個 epoch loss 就會先下降再緩緩上升，這是因為 cyclical 將 10 epoch 設為一個 period，period 開始時 beta 被設為 0，之後才慢慢增加。另外可以看到第三個 period 開始時，loss 緩緩上升的幅度沒有第二個 period 那麼大，代表模型的學習是有確實在進步的。

Teacher Forcing method (2) (additional)

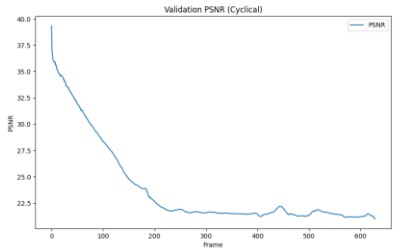


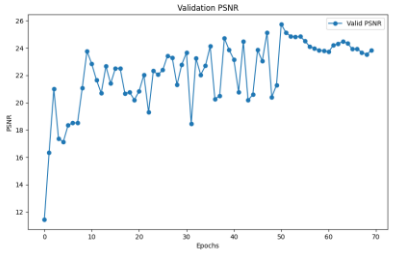


在 Teacher Forcing 的第二個策略上，由於改為漸進式的調整原圖所佔的比例，因此沒有了 **Part iii.1.a** 中 tfr ratio 下降時明顯的 loss 的 gap。

3. Plot the PSNR-per frame diagram in validation Dataset

使用 Teacher Forcing





初期 PSNR 分數非常高，越到後面的 frame 由於輸入與真正的 frame 會有所差異，導致輸出的圖片也會越來越偏離正確圖像，使 PSNR 下降。在訓練時嘗試了許多設定，但 testing 結果都不佳，直到將模型 Dimension 調大為--D_out_dim 256 --F_dim 152 --L_dim 48，並動態調整 lr 後，才在第 50 epoch 達到 24 分的 PSNR。

不使用 Teacher Forcing

之後嘗試一開始就把 tfr 設為 0，結果意外發現平均 PSNR 反而更高，並且只要 train 大約 4 個 epoch 就能達到超越 25 的 testing PSNR，甚至仍在 fast train 階段，模型學習速度非常快。到第七個 epoch 時效果大幅下降，便終止訓練。