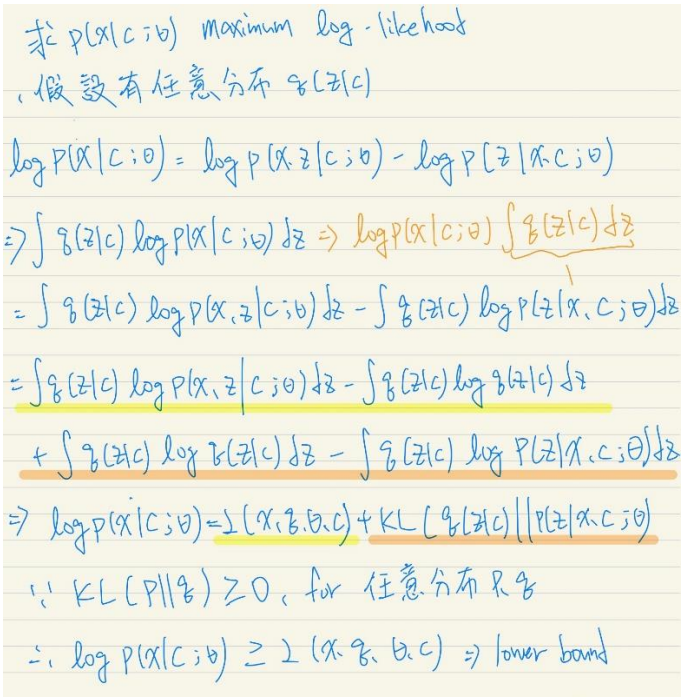
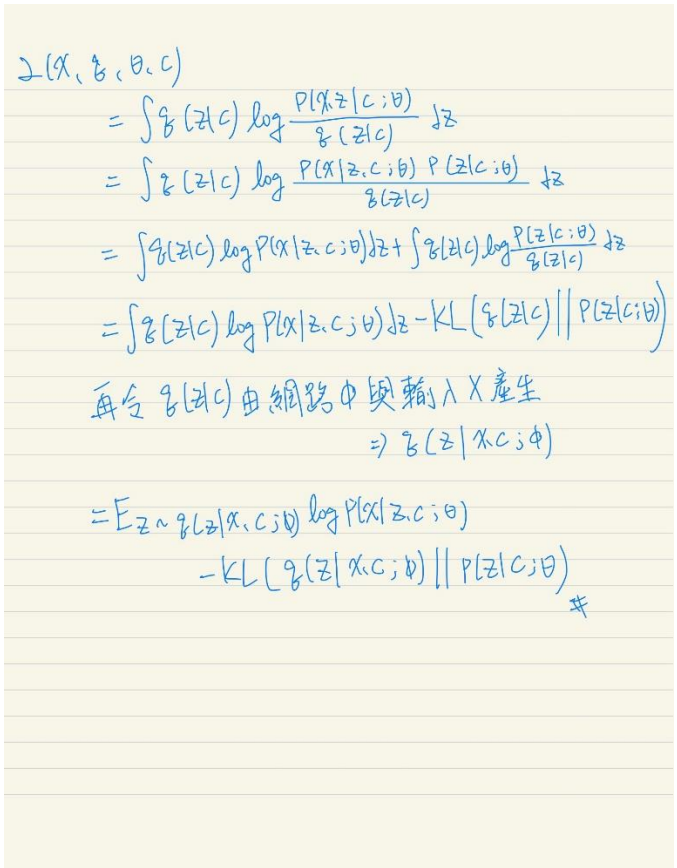
NYCU DL Lab4 – Conditional VAE for Video Prediction

Student ID: 313553014 Name: 廖怡誠

1. Derivate conditional VAE formula





2. Introduction

在 Lab4 中，我們需要實作 Conditional VAE model 在 video prediction 的任務上，我們會對模型輸入前一個 frame (f_{i-1})與想要生成的動作圖片(p_i)，模型會根據f_{i-1}與p_i輸出下一個 frame 的圖片(f_i)，圖片中的人物會有著與p_i相同的動作，並且會使用 PSNR 來評估輸出圖片與期望圖片的相似度，透過這樣的方式，希望可以給定一張圖片與多個連續的動作，讓模型生成有連續指定動作的影片。在這個作業中，我們需要了解 VAE model 中的 reparameterization trick 以及幫助模型訓練的 KL annealing 與 teacher forcing 方法。

3. Implementation Details

A. How do I write my training/testing protocol?

在 training 的程式碼中，因為需要將圖片一幀一幀進行計算，因此要將輸入的維度從(Batch, Time step, Channel, Height, Width)轉換成(Time step , Batch, Channel, Height, Width)，接著開始訓練 Encoder 與 Decoder，首先，以 Gaussian Predictor 為主的 module 屬於 Encoder，目標是輸入f_i與p_i並輸出能產生 normal distribution 上的 μ 與log^{Var}，使用 μ 與log^{Var}計算 KL loss，期望產生的 μ 與log^{Var}能夠越接近 N(0, 1)，而在 Gaussian Predictor 中使用 Reparametrize trick 的技術，產生z。接著以 Generator 為主的 module 屬於 Decoder，目標輸入z、f_{i-1}與p_i並輸出$\hat{f}_i$，使用$\hat{f}_i$與 Ground Truth f_i計算 MSE loss，重複此循環直到 time step 到最後一幀，而在訓練的過程中，為了避免 Generator 產生的圖片效果不佳而影響後續 Encoder loss 變化，因此使用 Teacher force 方法，在前期訓練還不穩定的時候，通過此方法將 Generator 輸出的圖片$\hat{f}_i$替換成 Ground Truth f_i。最後平均 KL loss 與 MSE loss 並結合 KL annealing 的 beta 計算 loss 更新。

```python
def training_one_step(self, img, label, adapt_TeacherForcing):

    mse_loss , kl_loss = 0, 0
    sequence_PSNR = list()
    self.optim.zero_grad()

    frame = 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)

    pred_frame = frame[0]
    for t in range(1, self.train_vi_len):

        prev_frame = frame[t - 1] if adapt_TeacherForcing else pred_frame

        trans_prev_frame = self.frame_transformation(prev_frame)
        trans_cur_label = self.label_transformation(label[t])
        trans_cur_frame = self.frame_transformation(frame[t])
        z, mu, logvar = self.Gaussian_Predictor(trans_cur_frame, trans_cur_label)
        kl_loss += kl_criterion(mu, logvar, self.batch_size)
        G_input = self.Decoder_Fusion(trans_prev_frame, trans_cur_label, z)
        pred_frame = self.Generator(G_input)
        mse_loss += self.mse_criterion(pred_frame, frame[t])
        sequence_PSNR.append(Generate_PSNR(pred_frame, frame[t]))

    beta = self.kl_annealing.get_beta()
    mse_loss /= (self.train_vi_len - 1)
    kl_loss /= (self.train_vi_len - 1)
    loss = mse_loss + beta * kl_loss

    loss.backward()
    self.optimizer_step()

    return loss, sum(sequence_PSNR) / len(sequence_PSNR)
```

Testing 的程式碼與 Validation 的程式碼差不多，主要是少了 Encoder 的部分，直接在 N(0, 1)中 random 出 z 值，與f_{i-1}、p_i輸入至 Decoder 中，並輸出有p_i動作的f_i，將圖片串接起來，輸出 gif。

```python
def val_one_step(self, img, label, idx=0):

    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)
    assert label.shape[0] == 630, "Testing pose sequence should be 630"
    assert img.shape[0] == 1, "Testing video sequence should be 1"
    # decoded_frame_list is used to store the predicted frame seq
    # label_list is used to store the label seq
    # Both list will be used to make gif
    decoded_frame_list = [img[0].cpu()]
    label_list = []
    cur_frame = img[0]
    for t in range(self.val_vi_len - 1):
        trans_cur_frame = self.frame_transformation(cur_frame)
        trans_next_label = self.label_transformation(label[t + 1])

        z = torch.randn((1, self.args.N_dim, self.args.frame_H, self.args.frame_W), device= self.args.device)
        input = self.Decoder_Fusion(trans_cur_frame, trans_next_label, z)
        out_next_frame = self.Generator(input)
        cur_frame = out_next_frame

        decoded_frame_list.append(out_next_frame.cpu())

    # Please do not modify this part, it is used for visulization
    generated_frame = stack(decoded_frame_list).permute(1, 0, 2, 3, 4)
    # label_frame = stack(label_list).permute(1, 0, 2, 3, 4)

    assert generated_frame.shape == (1, 630, 3, 32, 64), f"The shape of output should be (1, 630, 3, 32, 64), but your output

    self.make_gif(generated_frame[0], os.path.join(self.args.save_root, f'pred_seq{idx}.gif'))

    # Reshape the generated frame to (630, 3 * 64 * 32)
    generated_frame = generated_frame.reshape(630, -1)

    return generated_frame
```

B. How do I implement reparameterization tricks?

因為 Gaussian Predictor 產生的 mu 與 logvar 產生的高斯分布沒辦法進行梯度計算，因此需要使用 reparameterization trick 的技巧，將高斯分布表達為均值與標準差的線性組合，使其可以計算梯度。而在 reparametrize 的函式中，會先傳入均值與 log 變異數，透過以下式子可以將此推導成標準差，接著隨機產生一個常態分佈的 noise (epsilon)，進行式子 $epsilon * std + mu$計算。

$$\sigma = \sqrt{Var}$$

$$\sigma = \sqrt{e^{logVar}}$$

$$\sigma = 0.5 * e^{logVar}$$

```python
def reparameterize(self, mu, logvar):

    std = torch.exp(0.5 * logvar)
    # sample from Normal distribution (0, I) which shape is same with std
    epsilon = torch.randn_like(std)
    return  epsilon * std  + mu
```

C. How do I set my teacher forcing strategy?

Teacher forcing strategy 是為了避免 RNN 等模型前後輸入有關連性所設計的方法，通過 teacher forcing strategy 可以在模型前期還不穩定時，使用 ground truth 做為下一次的輸入。根據助教提供的參數說明，每次的訓練

中，會先隨機一個數與 teacher forcing ratio(tfr)比較，若該數小於 tfr，就會於該次 epoch 中使用 teacher forcing，而 tfr 的更新方法是先進行幾個 epoch，再開始 decay，decay 的方式是直接減少一個固定的數值 (tfr_d_step)，為了避免數值變成負數，若 tfr 小於就會設為 0，即代表不會再使用 tfr。

```python
def teacher_forcing_ratio_update(self):

    if self.current_epoch >= self.tfr_sde:
        self.tfr -= self.tfr_d_step
        self.tfr = max(0, self.tfr)
```

D. How do I set my KL annealing ratio?

在 KL annealing 的實作中，我直接參考原始論文的 source code。首先，在初始化的地方會先決定要使用甚麼 beta 變化方式，有週期性的 Cyclical、只有一次變化的 Monotonic 與沒有變化的 Constant，而在 Cyclical 和 Monotonic 中，我使用線性變化來更新 beta 值，使用總訓練 epoch 數除以 n_cycle 計算變化週期的數量，並使用 ratio 控制每個週期中，beta 值從 0 變化至 1 的斜率 y (stop – start) / x (ratio * period)作為每次的更新的變化量 step，最後一次將每個 epoch 的 beta 數值紀錄到 L 中，在訓練時，只需要根據對應到的 epoch 存取即可。

```python
class kl_annealing():
    def __init__(self, args, current_epoch=0):
        # kl_anneal_cycle: number of cycle during the whole training process (M)
        # kl_anneal_ratio: proportion used to increase beta within a cycle (R)
        # beta: beta control the strength of regularization
        self.current_epoch = current_epoch

        if args.kl_anneal_type == "Cyclical":
            self.L = self.frange_cycle_linear(n_epoch= args.num_epoch,
                                              n_cycle= args.kl_anneal_cycle,
                                              ratio= args.kl_anneal_ratio)
        elif args.kl_anneal_type == "Monotonic":
            self.L = self.frange_cycle_linear(n_epoch= args.num_epoch,
                                              n_cycle= 1,
                                              ratio= args.kl_anneal_ratio)
        elif args.kl_anneal_type == "constant":
            self.L = np.ones(args.num_epoch)

    def update(self):
        # update epoch
        self.current_epoch += 1

    def get_beta(self):
        return self.L[self.current_epoch]

    def frange_cycle_linear(self, n_epoch, start=0.001, stop=1.0,  n_cycle=1, ratio=1):
        """
        n_iter: the iteration number
        n_cycle: number of cycles (M)
        ratio: proportion used to increase beta within a cycle (R)
        """
        # https://github.com/haofuml/cyclical_annealing/blob/master/plot/plot_schedules.ipynb
        L = np.ones(n_epoch)
        period = np.ceil(n_epoch / n_cycle)
        # y / x, x is total period * ratio, step is slope
        step = (stop - start) / (period * ratio)

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

        return L
```

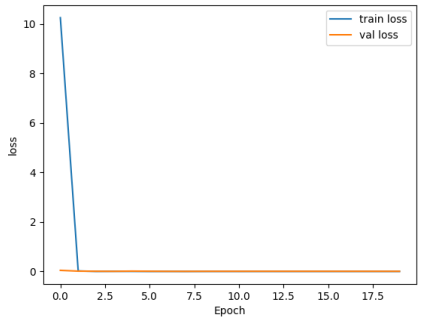
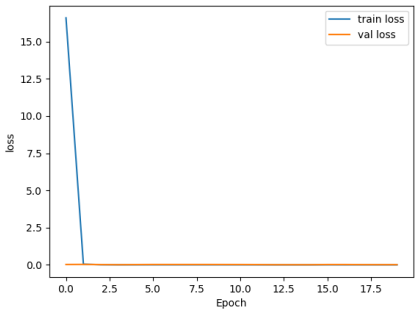
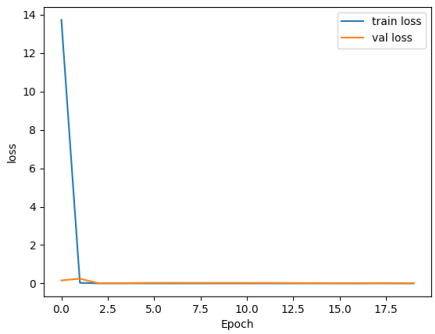
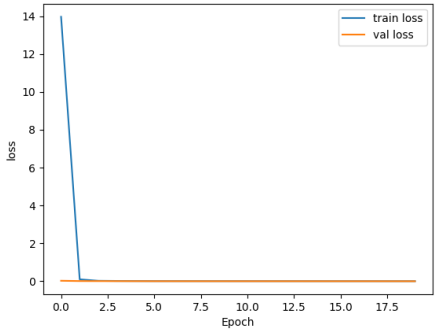
4. Analysis & Discussion

由於我起始的 loss 數值都很高，所以呈現出的圖表可能沒辦法那麼清楚的表示結果，但還是可以觀察到一些細微的變化，另外我也會分析訓練過程的 PSNR。

A. Plot teacher forcing ratio

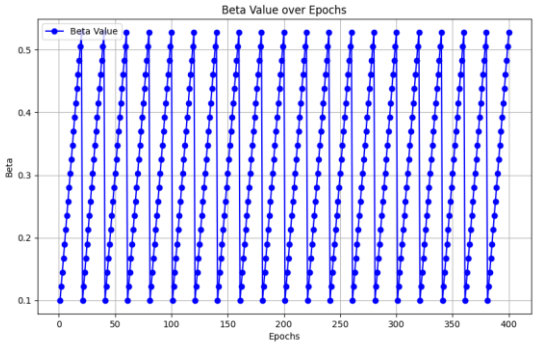
i. Analysis & compare with the loss curve

在分析 teach forcing ratio 的實驗中，我主要進行了兩種實驗，第一種是固定 KL type 為 cyclical 與 teacher forcing ratio(tfr)為 1，並調整 tfr 的 start decay epoch(sde)為 0, 10 和 15，主要想要實驗，在不同的 epoch 開始減少 ratio，也就是提早停止使用 teacher forcing，對 loss 的影響，實驗結果可以看出 tfr_sde 為 10 和 15 的差別不大，PSNR 大概也都落在 20 左右，反而是 tfr_sde 為 0，除了一開始的 loss 比較低，PSNR 大概在 23 左右；第二種實驗是固定 tfr_sde 為 10，並調整 tfr 為 0 和 1，一樣是觀察不使用 teacher forcing 與使用的比較，tfr 為 0 的 PSNR 為 26.8 而 tfr 為 1 的 PSNR 為 20.4，這個實驗中可以說明 teacher forcing 的幫助不大，設置有可能讓模型誤以為自己的能力很好而卡在 local minimum。

KL type: cyclical, tfr: 1, tfr_sde: 0	KL type: cyclical, tfr: 1, tfr_sde: 10
	
KL type: cyclical, tfr: 1, tfr_sde: 15	KL type: cyclical, tfr: 0, tfr_sde: 10
	

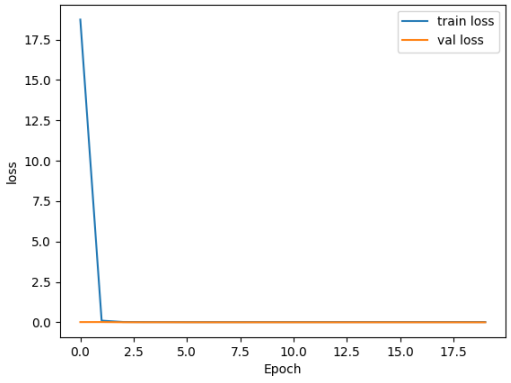
B. Plot the loss curve while training with different setting. Analyze the difference between them.

如前一個實驗所說，使用 Teacher Forcing 對於訓練的效果幫助不是很大，因此在這個實驗中，我將 Teacher forcing ratio 設為 0，並進行以下三中 KL annealing 方法，此外，設定 kl anneal cycle 為 20 與 kl anneal ratio 為 0.9，下圖是訓練於 400epoch 的 beta 變化示意圖。

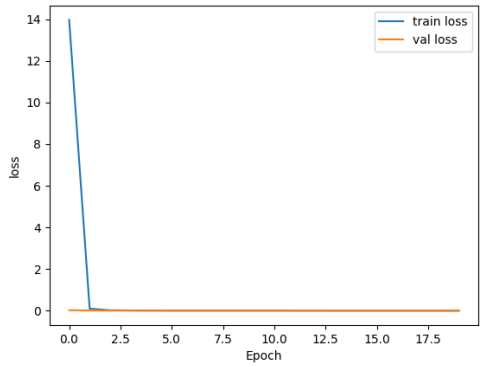


從 KL annealing 三個不同模式的 loss 曲線分析，可以觀察到沒有 KL annealing 在一開始的 loss 大約到 80 與其他兩個模式相差甚遠，代表沒有使用 KL annealing 會使得模型很不穩定，但還是可以降下來。三種模式 Monotonic、Cyclical、Constant 的 PSNR 分別為 26.8、25.7 和 27.2，反而是 Constant 也就是沒有 KL annealing 的效果最好，總結來說，不使用 teacher forcing 與 KL annealing 對於本次的 task 最好。

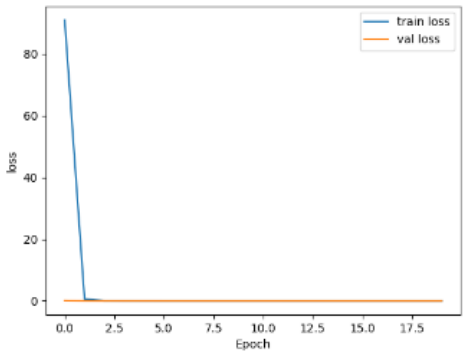
i. With KL annealing (Monotonic)



ii. With KL annealing (Cyclical)

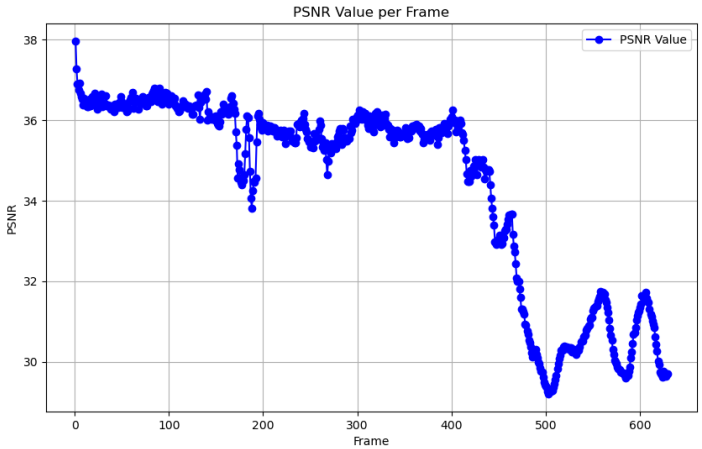


iii. Without KL annealing (Constant)



C. Plot the PSNR-per frame diagram in validation dataset

我最好的模型平均的 PSNR 是 34.4 左右，在大約 400 frame 之前的 PSNR 大約都有 36 以上，但是 400 frame 便開始急遽下降到 PSNR 30 左右。



D. Other training strategy analysis (Bonus)

在訓練上，我除了對 teacher forcing 與 KL annealing 的參數進行實驗之外，我也有嘗試做 data augmentation，對資料做高斯模糊或 normalization，在這邊需要避免連同 label 都一起進行操作，但實驗的結果並不是很好，首先，使用 normalization 輸出的圖片，由於我直接使用 normalization 過的圖片進行 Loss 繼續，導致最後模型 inference 輸出的圖片也是 normalization 過的，經過 denormalization 會有點模糊。使用高斯模糊也是差不多的情況，原本希望減少圖片細節讓模型好學習，但 PSNR 算出來的效果不太好，且每次需對圖片處理使得計算時間加長。除此之外，我也有修改不同的 optimizer 與 scheduler，最後選用有 weight decay 效果的 AdamW，而 scheduler 的效果都不太好，便使用預設的 MultiStepLR。

```
self.label_transform = transform
self.img_transform = transform
transform_list = list(self.img_transform.transforms)
transform_list.append(transforms.GaussianBlur(kernel_size=3))
self.img_transform = transforms.Compose(transform_list)
```