Lab5 : Conditional VAE For Video Prediction

310605005 王映勻

1. Introduction

這次 lab 需要實作出 conditional VAE，利用影片前面的 frame 來預測後面的情況。首先要將影片中前一個 frame 輸入 encoder，將其產生的 latent vector h 及利用我們的 condition(action and position) 得到的 z 當作 decoder 的輸入，最後的 output 即為我們要預測的影片的下一個 frame。

AE (AutoEncoder)的架構是由 encoder 和 decoder 組成，encoder 將輸入資料做降維處理得到 latent variable，可以說是主要的特徵，接著在透過 decoder 進行解碼，試著還原出輸入資料。VAE 就是透過限制編碼過程使其生成的 latent variable 能夠遵循標準正態分佈，這樣就能依據這個正態分布進行生成的工作，但也只能產生和原始資料類似的輸出資料。而 CVAE 則是將原始資料和其 condition 共同作為 encoder 的輸入，這樣就可以生成特定類別的輸出資料。

2. Derivation of CVAE

推導 CVAE 的流程基本上和 VAE 差不多，主要是 x 必須限制在條件 c，也就是$p(x) = p(x|c)$。為了求得模型參數 θ，要想辦法最大化 marginal distribution $p(x|c;\theta)$

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

由於計算上的困難，沒辦法利用 maximum likelihood 的原理設計 $p(x|c;\theta)$，因此得透過以下技巧推導：

$$p(X,Z|c;\theta) = p(X|c;\theta)p(Z|X,c;\theta)$$

對等號兩邊同取 log

$$logp(X,Z|c;\theta) = logp(X|c;\theta) + logp(Z|X,c;\theta)$$
$$\rightarrow logp(X|c;\theta) = logp(X,Z|c;\theta) - logp(Z|X,c;\theta)$$

將等號兩邊同乘 $q(Z)$並對 Z 做積分

$$\int q(Z)\,logp(X|c;\theta)dZ = \int q(Z)\,logp(X,Z|c;\theta)dZ - \int q(Z)\,logp(Z|X,c;\theta)dZ$$

$$= \int q(Z)\,logp(X,Z|c;\theta)dZ - \int q(Z)\,logq(zZ)dZ$$

$$+ \int q(Z)\,logq(Z)dZ - \int q(Z)\,logp(Z|X,c;\theta)dZ$$

$$\because \int q(Z)\,logq(Z)dZ - \int q(Z)\,logp(Z|X,c;\theta)dZ = KL(q(Z)||p(Z|X,c;\theta))$$

$$\therefore logp(X|c;\theta) = \mathcal{L}(X,q,\theta) + KL(q(Z)||p(Z|X,c;\theta))$$

因為 $KL \geq 0$，要最大化 $logp(X|c;\theta)$ 就等於要最大化 $\mathcal{L}(X,q,\theta)$

$$\mathcal{L}(X,q,\theta) = \int q(Z)\,logp(X,Z|c;\theta)dZ \quad (\mathbf{1})$$

$$-\int q(Z)\,logq(Z)dZ \quad (\mathbf{2})$$

$$(\mathbf{1}) = E_{Z\sim q(Z)}[logp(X,Z|c;\theta)] \;\; ; \;\; (\mathbf{2}) = E_{Z\sim q(Z)}[logq(Z)]$$

$Let\ q(Z) = q(Z|X,c;\theta')$

$$\mathcal{L}(X,q,\theta) = E_{Z\sim q(Z|X,c;\theta')}[logp(X,Z|c;\theta)] - E_{Z\sim q(Z|X,c;\theta')}[logq(Z|X,c;\theta')]$$

又 $p(X,Z|c;\theta) = p(X|Z,c;\theta)p(Z|c)$

$$\mathcal{L}(X,q,\theta) = E_{Z\sim q(Z|X,c;\theta')}[p(X|Z,c;\theta)] + E_{Z\sim q(Z|X,c;\theta')}[logp(Z|c) - logq(Z|X,c;\theta')]$$

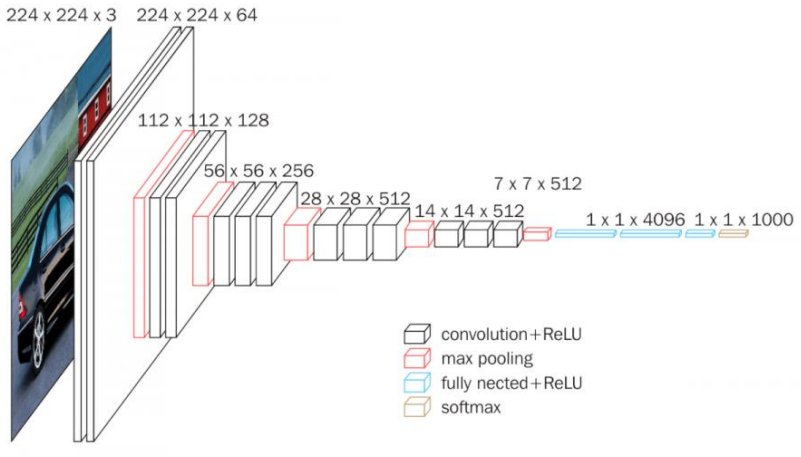
$$\because E_{Z\sim q(Z|X,c;\theta')}[logp(Z|c) - logq(Z|X,c;\theta')] = -KL(q(Z|X,c;\theta')||p(Z|c))$$

$$\therefore \mathcal{L}(X,q,\theta) = E_{Z\sim q(Z|X,c;\theta')}[p(X|Z,c;\theta)] - KL(q(Z|X,c;\theta')||p(Z|c))$$

3. Implementation details

A. Describe how you implement your model (encoder, decoder, reparameterization trick, dataloader, etc.)

(1) Encoder, Decoder



本次 VAE 模型是採用 VGG64 的架構，常見的 VGG16 架構如上圖所示，而 VGG64 和 16 的差別只有網路的深度，原理都是相同的，主要就是透過比較尺寸 3x3 的卷積核來堆疊神經網絡，加深神經網絡的深度，以提升模型的性能。

每個 VGG 層的定義如下，先利用 nn.Conv2d 按照設定的輸入和輸出大小做大小為 3*3 的二維卷積，透過 nn. BatchNorm2d 對數值做正規化處理，最後經由 LeakyReLu activation function 輸出。

```
class vgg_layer(nn.Module):
    def __init__(self, nin, nout):
        super(vgg_layer, self).__init__()
        self.main = nn.Sequential(
                nn.Conv2d(nin, nout, 3, 1, 1),
                nn.BatchNorm2d(nout),
                nn.LeakyReLU(0.2, inplace=True)
                )
```

Encoder 的部分是將一個 64*64 的 3 通道彩色圖片逐層慢慢轉換成 1*1 的輸出，經由池化層提取重要特徵後，透過 decoder 進行上採樣，逆向復原成 64*64*的圖片。

(2) Reparameterization trick

由於從 VAE 參數化分佈的採樣過程是不可微的，這樣會造成梯度更新的問題，因此需要使用 reparameterization 的技巧。利用高斯分布的特性，我們可以對一個標準正態分布進行採樣，加上原本均值再乘上標準差，這樣就可以將隨機元素與學習參數分開。

$$z = \mu + \sigma * \epsilon$$

實際做法如下，算出標準差之後，生成一個標準正態分布，依照上面公式操作就可將 latent vector 轉成正態分布的形式。

```
def reparameterize(self, mu, logvar):
    #raise NotImplementedError
    logvar = logvar.mul(0.5).exp_()
    eps = Variable(logvar.data.new(logvar.size()).normal_())
    return eps.mul(logvar).add_(mu)
```

(3) Dataloader

Dataloader 主要可以分為 get_seq 和 get_csv 兩個部分。

get_seq 是要得到訓練的輸入 image sequence，利用 self.ordered 這個參數決定是否要依照順序讀取，train 使設定為 false，而 test 跟 validate 為了方便驗證效果因此每次都會依照固定順序讀取。將讀取到的圖片轉成(1, 64, 64, 3)，並除以 255 正規化數值，將整個 sequence 的圖片拼接起來後再轉成 tensor，且利用 permute 函數將 tensor 調整成(batch size, channel, width, height)的形式。

每個資料夾中除了有 30 張圖片之外，還有兩個 csv 檔，分別記錄目標物的 actions 和 positions 的資訊，利用 get_csv 將數值讀進來後轉成 np.array 的形式，以方便後續作為 conditional input。

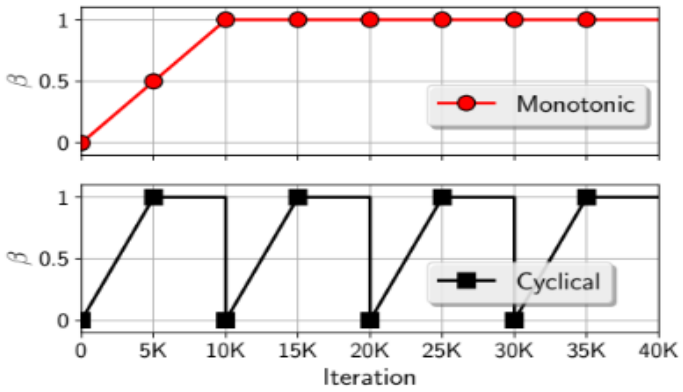
(4) KL annealing

當 decoder 足夠強大，即使依賴很少的 history 信息也可以讓 reconstruction errors 降得很低，也就是說其不依賴 encoder 提供的 z 了，這樣就會導致 KL-vanishing 的問題。

為了解決這個問題，這邊使用了 KL annealing 的技巧，即引入一個權重來控制 KL 項，且權重從 0 開始逐漸增大，讓模型一開始學會 encode 更多信息到 z 里，然後隨著權重增大再 smooth encodings。

其分為 monotonic 和 cyclical 兩種模式。如下圖所示，monotonic 的做法即是讓權重從 0 開始逐漸增大直到 1 為止，而 cyclical 是在訓練中反覆循環 monotonic 的作法。

```
if self.mode == 'monotonic':
    return (1./(self.time))*(epoch) if epoch<self.time else 1.
else:
    epoch %= self.time
    KL_weight = (1./(self.time/2))*(epoch) if epoch<self.time/2 else 1.
    return KL_weight
```



B. Describe the teacher forcing (including main idea, benefits and drawbacks.)

Teacher Forcing 是一種用於序列生成任務(seq-to-seq)的訓練技巧這種訓練方法。正常訓練過程會將前一個時間點的輸出當成下一個時間點的輸入，但若是前一個時間點的結果是錯誤的，那往後所有的時間點都會受到這個錯誤影響。

而 Teacher Forcing 就是不使用前一個時間點的輸出當成下一個時間點的輸入，而是直接將前一個時間點所對應訓練數據的標準答案(ground truth) 作為下一個時間點的輸入。

這樣做的優點是使模型更穩定快速收斂，且能夠在訓練的時候矯正模型的預測，避免在序列生成的過程中誤差進一步放大。而反之其缺點有 Exposure Bias、Overcorrect 和 No diversity 三種：Exposure Bias 為最常見的問題，在 Teacher Forcing 的介入下，模型訓練和推論時的輸入一個是 ground truth，一個是模型前一個輸出，這樣不一致的情況會造成推論的誤差累積。而因為其干涉也可能導致整個序列結果不具邏輯性，也就是 Overcorrect。No diversity 則是指訓練時過度受到 ground truth 的約束，因此其多樣性也受到限制，也就是如果測試數據集與訓練數據集來自不同的領域，模型的 performance 就會變差。

```python
if epoch >= args.tfr_start_decay_epoch:
    ### Update teacher forcing ratio ###
    if args.tfr > args.tfr_lower_bound+args.tfr_decay_step:
        args.tfr = 1. - (1./(args.niter))*(epoch-args.tfr_start_decay_epoch)
    else:
        args.tfr = args.tfr_lower_bound
```

這次 teacher forcing ratio 的設計方法如上，給定一個 lower bound，在設定的 start_decay_epoch 之後，teacher forcing ratio 會開始慢慢下降到 lower bound，而下降的幅度這邊是將 1-lower bound 的差值除以整體訓練的 epoch 數。

4. Results and discussion

A. Show your results of video prediction

(1) Make videos or gif images for test result

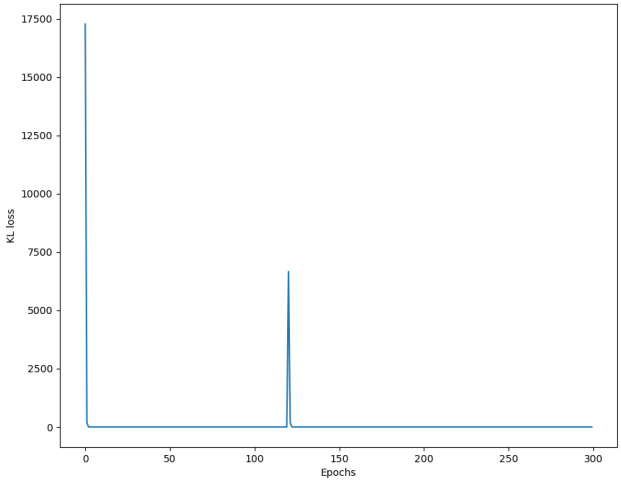
結果請見 result_compare.gif

(左邊為 ground truth，右邊為預測結果)

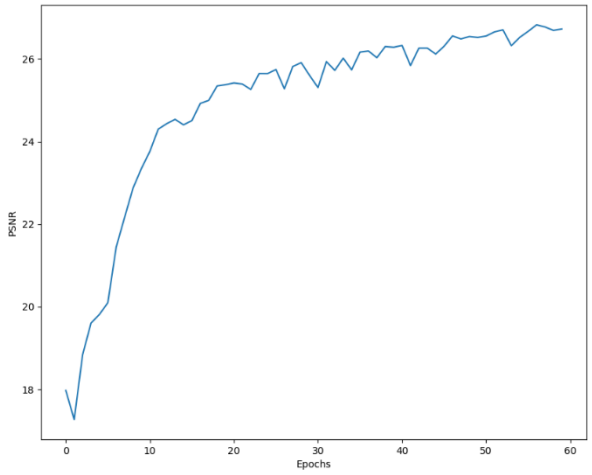
(2) Output the prediction at each time step

B. Plot the KL loss and PSNR curves during training

KL loss：



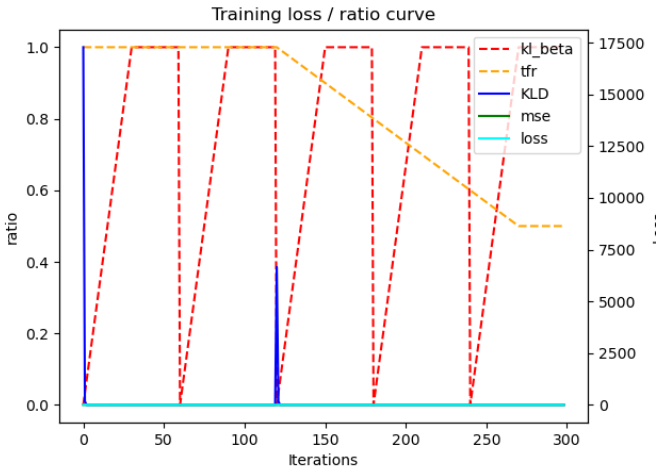
PSNR：



(因為計算 PSNR 的部分是每 5 個 epoch 做一次，因此 x 軸的 epoch 數應該要再乘以 5 才是正確的 epoch 數)

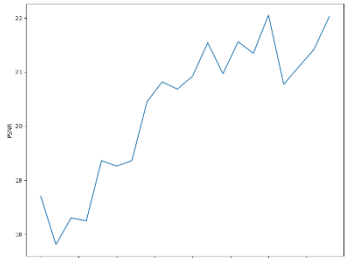
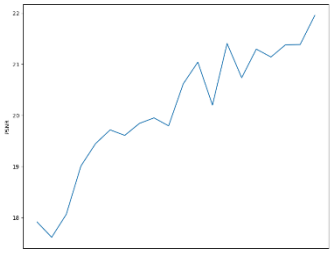
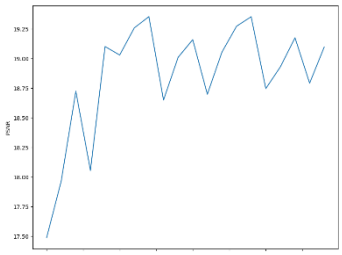
C. Discuss the results according to your setting of teacher forcing ratio, KL weight, and learning rate

下圖為本次訓練重要的參數及 loss 變化圖表，可以發現在 teacher forcing ratio 開始下降，且正好是 KL weight 變化的時刻，造成 KL loss 有不小的起伏，有成功對訓練產生刺激。對照上面的 PSNR 圖表，可以發現 epoch=120 (24)時，有就是 teacher forcing ratio 開始下降後，PSNR 開始有上下起伏，主要原因應該是開始有 ground truth 作為輸入介入，一開始可能會造成訓練分數降低，但長期來看這個方法是能夠提升整體模型的表現的。

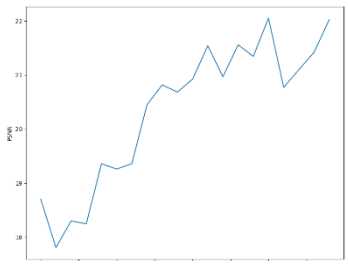
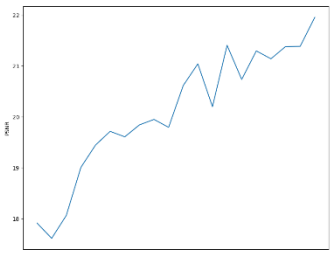
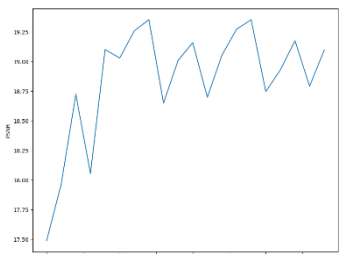


本次實驗我也有針對 teacher forcing ratio, KL weight 和 learning rate 三個參數做測試。

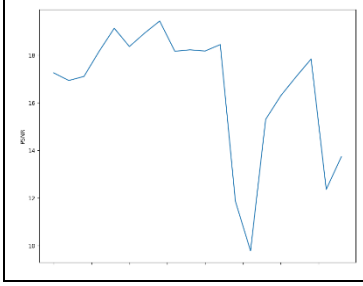
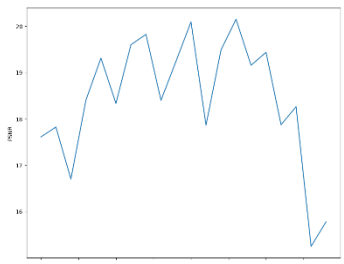
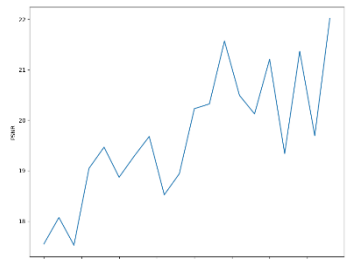
首先，learning rate 預設為 0.002，以 PSNR 的分數來看，將其下調至 0.0002 的表現和原本差不多，但若是調大至 0.02 表現就明顯較差，原因應該就是因為當 learning rate 太大時就容易產生震盪，難以收斂。

Learning rate		
0.0002	0.002	0.02
		

而 KL 我有將 cycle 數分別以 3, 5, 7 去做測試，可以觀察到其實三者 PSNR 曲線大致相同，數值有變化的時機點都差不多，只是 7 的表現略微劇烈一點，推測可能是因為 KL 變化太頻繁，但影響也沒有到很顯著。

Kl annealing cycle number		
3	5	7
		

從這次實驗來看，三個參數中對訓練影響最大是 teacher forcing ratio，設定不同的 start decay epoch，對整個訓練的表現影響是非常明顯的，前兩者應該是因為太早介入訓練，導致整個模型效果變得很差，而 70 的趨勢就較為合理，但震盪還是非常大，如果等模型訓練較為穩定後再介入效果可能會比較好。

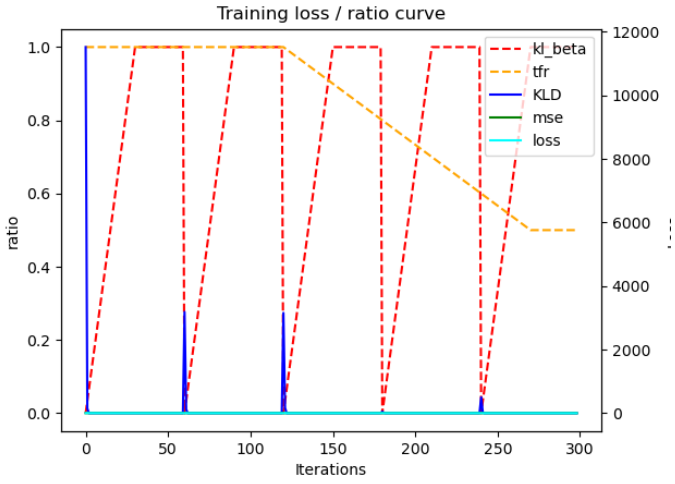
Teacher forcing ratio (start decay epoch)		
30	50	70
		

5. Extra

A. Implement learned prior

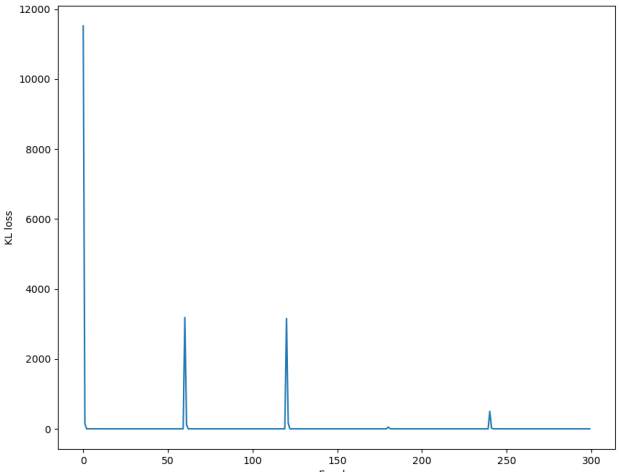
從結果來看，表現幾乎和 fixed prior 的表現一樣，看原本論文對這個資料集的測試結果，PSNR 的分數表現其實也非常相近，但在其他資料集上的表現就會有落差，所以在此資料集測試可能比較看不出來 fixed prior 和 learned prior 對模型的影響。

Hyperparameter & loss:

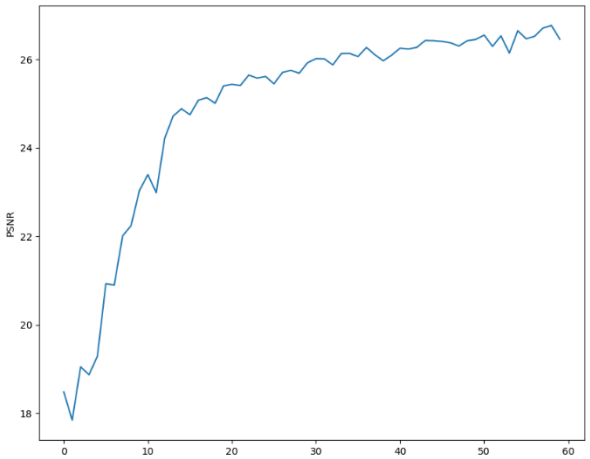


Prediction at each time step：

KL loss :


KL loss
12000
10000
8000
6000
4000
2000
0
0
50
100
150
200
250
300

PSNR :


PSNR
26
24
22
20
18
0
10
20
30
40
50
60