Lab4 : Conditional VAE For Video Prediction

311511043 李承翰

1. Introduction

這次 lab 需要實作出 conditional VAE，利用影片前面的 frame 來預測 並生成後面的 frame。首先將 frame 輸入進 encoder，接著使用 encode 出來的 latent variable 及利用 condition(action 還有 position) 得到 z 當作 decoder 的輸入，最後就能夠輸出生成我們預測的 frame。

AutoEncoder 的架構是由 encoder 還有 decoder 組成，encoder 將輸入進行降維處理之後得到 latent variale，也就是輸入的主要特徵，接著透過 decoder 將這個主要特徵還原成輸入本身。而 VAE 就是透過限制 encoder 生成的 latent variable，使其能夠遵守 Normal distribution。這樣就能產生與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 的流程和 CVAE 基本相同，主要是需要將 x 限制在條件 C，也就是 p(x) = p(x|c)。而為了要求得模型參數θ，要想辦法最大化$p(x|c;\theta)$

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

由於 z 有無限多種，我們沒有辦法利用 maximum likelihood 的原理設計 $p(x|c;\theta)$，因此我們需要透過以下方法推導

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

對兩邊同取 log

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

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

將兩邊同乘$q(Z)$ 後對 Z 積分:

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

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

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

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

因為 KL≥ 0， $logp(X|c;\theta)$是定值，所以最小化 KL，就是最大化$\mathcal{L}$(X,q,θ)

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

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

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

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

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

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

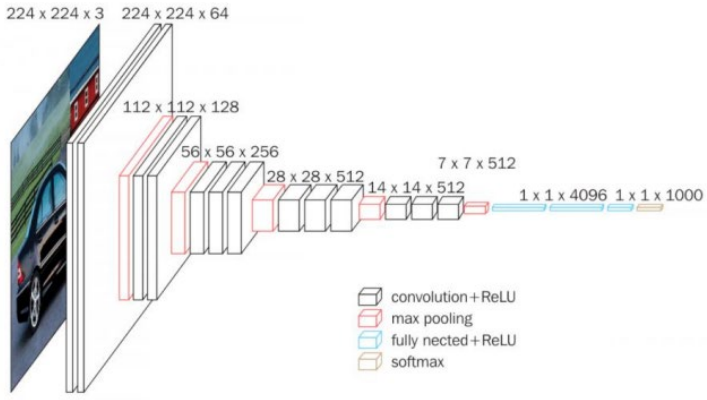
$$\mathcal{L}(X, q, \theta) = E_{Z\sim q(Z|X,c;\theta')}[p(X|Z, c; \theta)] + E_{Z\sim q(Z|X,c;\theta')}[logp(Z|c) - logq(Z|X, c; \theta')]$$

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

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

3. Implementation details
   A. Describe how you implement your model (encoder, decoder, reparameterization trick, dataloader)
      1. Encoder、decoder

         這次模型當中的架構我採用助教給予的 sample code 而並沒有去改變他。



         這次模型採用的是 VGG64 的架構，而常見的 VGG16 架構就如上圖。這兩種網路的差別只在於深度不一樣。透過多個捲積層來堆疊網路，以加深深度。

         而在實作上則是透過推疊多個 vgg_layer 來達成。

         下圖是程式當中的定義，也就是說當我們疊加了多個 vgg_layer，就是堆疊了多個捲積層。

```
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，提取重要特徵後，透過 decoder 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 Normal distribution 進行採樣，加上原本均值再乘上標準差。

```
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 ordered 這個參數決定是否要依照順序讀取，有點類似我們使用 pytorch 的 dataloader 時設定的一個參數，shuffle，而在 train 使設定為 false，而在 test、validate 當中設為 true。將整個 sequence 的圖片使用 pytorch 的 stack 串在一起之後再轉成 tensor，而之後在我們將資料 input 到模型之前，我們需要利用 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，而這兩個資訊會在 get_csv 當中被我們讀取，並且我們將 action 與 position 的資訊全部串接在一起成為一個 7 維的資料，而為了要將 action 及 position 也放入 lstm 當中，所以在 lstm 的 input feature 當中，我們需要將 cond_dim 這一項也加入。

4. KL annealing
   當 decoder 足夠強大，讓它可以自行模擬出 model 的情況下，encoder 所提供的採樣 z 就不再重要了，也就是說就算 decoder 不依賴 encoder 提供的 z ，reconstruction error 也不會太大，這樣就會出現 KL-vanishing 的問題。為了解決這個問題，就需要 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，如此一來 KL 項帶來的影響就會比較晚出現，讓模型能夠多花一點時間從 encoder 那邊多學習一點時間。

B. Describe the teacher forcing (including main idea, benefits and drawbacks)
   Teacher forcing 是一種訓練上常見的技巧，也就是將 input 到 LSTM 層的

資料，從我們自行預測出的 frame 改為使用 ground truth。這麼一來的好處就是在模型剛開始被訓練的時候，因為我們使用 ground truth 來進行訓練，所以模型進步的會比較快，同時也可以避免當我們訓練過程中一次預測出現問題的時候，造成後續的所有預測都是依照著那一次錯誤的預測。

而這個方法雖然優點很多，但它還是有缺點存在的，因為在訓練當中使用 ground truth 來進行訓練，所以如果在 testing data 當中與 training data 有較大的差異存在，那我們的模型就有可能因為過度依賴 ground truth 而沒辦法有較好的表現。

4. Result and discussion

A. Show your results of video prediction

Prediction of KL_Anneal_monotonic

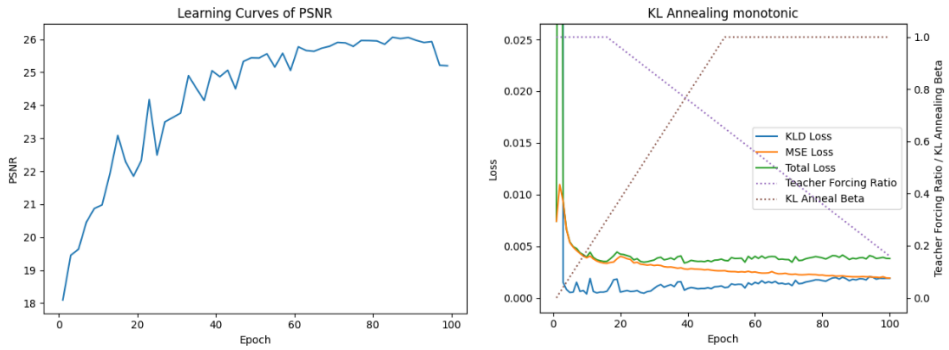
Gound Truth of KL_Anneal_monotonic

Prediction of KL_Anneal_cyclical

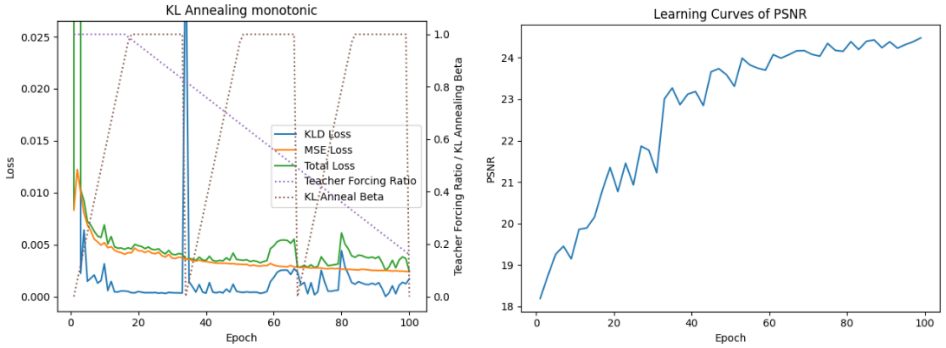
Gound Truth of KL_Anneal_cyclical

B. Plot the KL loss and PSNR curve during training

(a) KL anneal monotonic



(b) KL anneal cyclical



C. Discuss the results according to your settings

在這邊我統一使用 KL_ANNEAL_MONOTONIC 的實驗數據來進行比對

(a) Comparisons between LRs

lr	0.0002	0.002	0.005
Highest Psnr	21.31	25.73	24.76

可以看出在 lr 過小的情況下，模型收斂的速度會慢很多，所以在同樣的 hyper-parameters 的情況下，就沒有辦法達到差不多的 psnr，而當 lr 調大之後，也可能因為 overfit 相關的問題，讓 PSNR 不如預期。

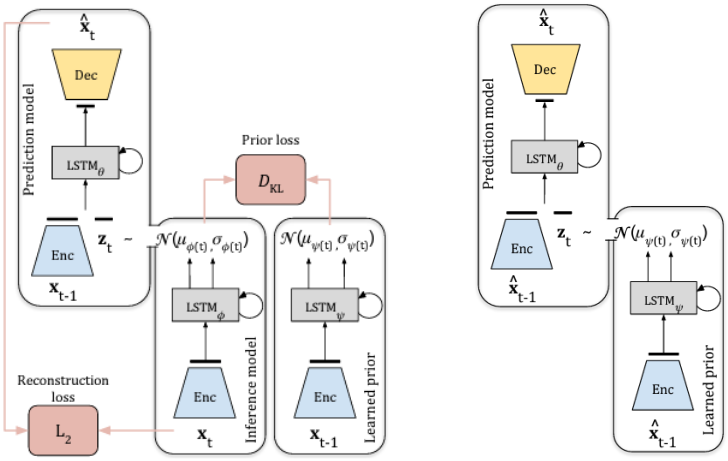
(b) Comparisons between TFRs

tfr_start_decay_epoch	0	15	50
Highest Psnr	18.15	25.73	26.11

從比較可以看到，當我們在不同的時機點開始減少 teacher forcing 的出現時，對於我們訓練的結果是有很大的影響的，如果太早減少 teacher forcing，那模型將會很難訓練完成，也就造成了表現不佳，而適當的延長 teacher forcing，在不 overfit 的前提下也能夠帶來更好的表現。

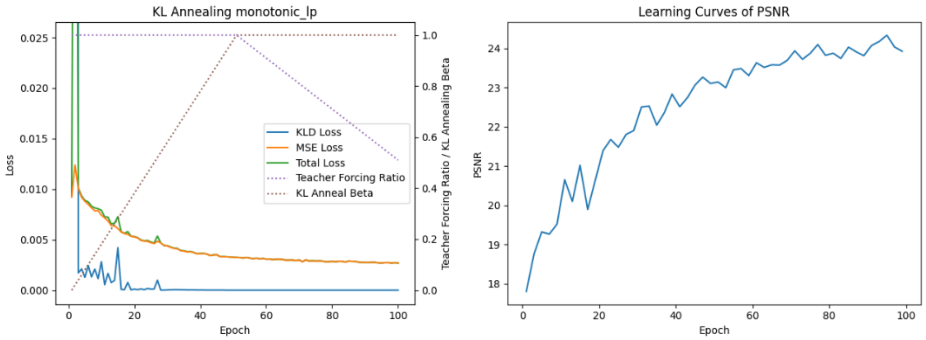
D. Extra

(a). learned prior



左、右圖分別為 learned prior 的 train 、 prediction 架構

以往我們的 prior 是固定的，也就是 N(0,1)，而現在我們採用的 prior 改採用學習的方式，使用一個 LSTM 來學習 ground truth 的 pattern，將其作為 KL divergence 的輸入，使我們的採樣可以趨近於 ground truth 的 pattern，而 generation 的時候使用這個學習過的 LSTM 作為我們採樣的來源，理論上就可以產出更好的結果。



在這一個資料集當中，模型的表現並沒有高出多少，不過這也跟論文當中使用此資料集所實驗出的表現差不多。

5. Discussion

這次我們將資料丟進模型當中的方式跟以往不同:

以往我們使用 for idx, (input,label)的方式來完成一個 epoch，而這次我們使用的是 iter(data_loader) 以及 next(iterator)的方式進行，而為了確保每一個 epoch 當中我們可以經歷所有的 training data，所以我們需要計算 batch_size * epoch_size 的數值，讓其乘積是 43008，這樣才可以做到在每一個 iter 的訓練當中我們都有經歷過所有訓練資料。

6. Testing 生成的 git

(a) Ground truth gif

https://drive.google.com/file/d/1beTp9SqEB1rPBS5HOoZwCKhxwFmyexox/view?usp=sharing

(b) Prediction gif

https://drive.google.com/file/d/1ANySnKxTdqPdAkEhAlw1mQPIxYIypDas/view?usp=sharing