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。

1. 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; 𝜃)

由於z有無限多種，我們沒有辦法利用maximum likelihood的原理設計，因此我們需要透過以下方法推導

對兩邊同取log

將兩邊同乘𝑞(Z) 後對 Z積分:

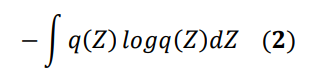
=

= KL(q(Z)||p(Z|X,c;

因為KL 0， 是定值，所以最小化KL，就是最大化X,q,

一張含有 文字 的圖片

自動產生的描述



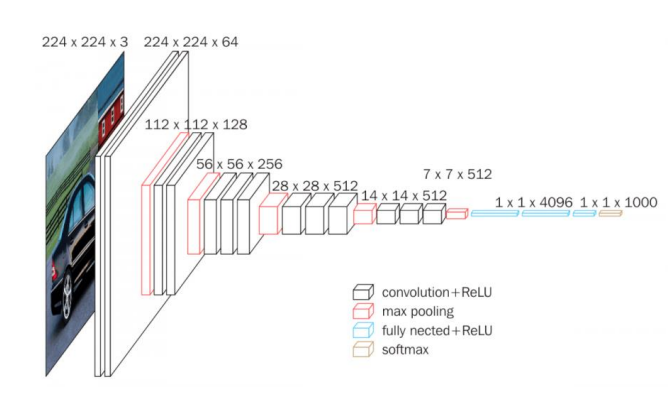


一張含有 文字, 室內, 文件, 螢幕擷取畫面 的圖片

自動產生的描述

1. Implementation details
2. Describe how you implement your model (encoder, decoder, reparameterization trick, dataloader)
3. 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。

一張含有 文字 的圖片

自動產生的描述

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。

1. 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。

一張含有 文字 的圖片

自動產生的描述

1. 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。

1. 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。

1. 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。

1. Result and discussion
2. Show your results of video prediction



Prediction of KL\_Anneal\_monotonic



Gound Truth of KL\_Anneal\_monotonic

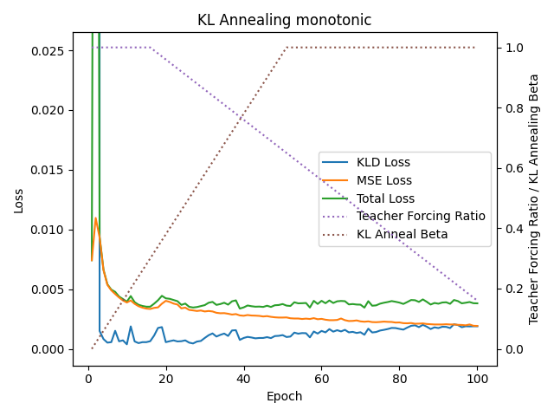
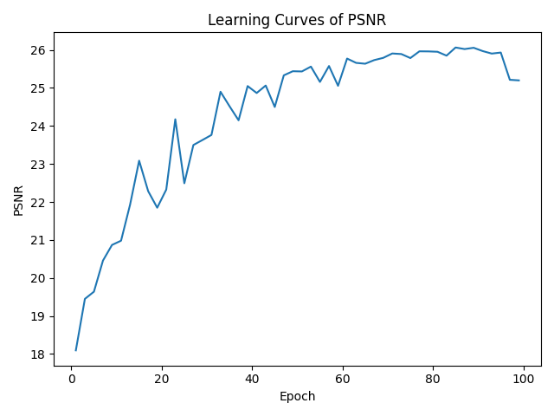


Prediction of KL\_Anneal\_cyclical

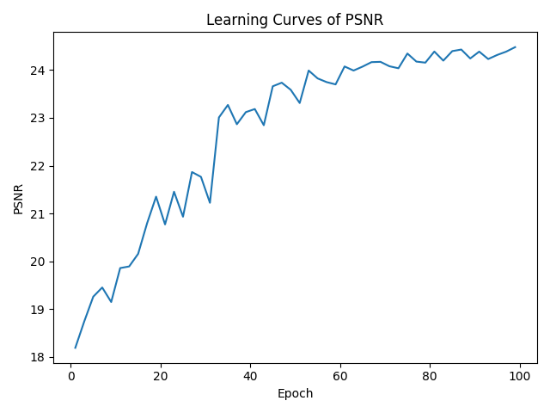
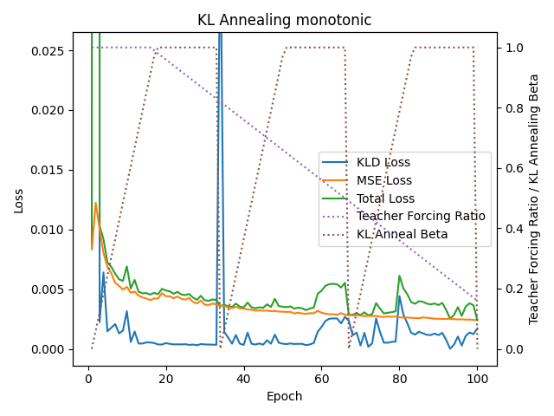


Gound Truth of KL\_Anneal\_cyclical

1. Plot the KL loss and PSNR curve during training
2. KL anneal monotonic



1. KL anneal cyclical



1. 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

1. 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。

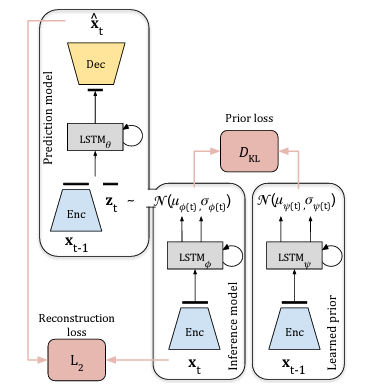
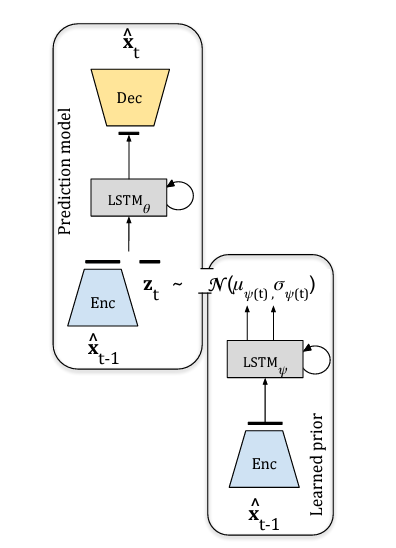
1. Comparisons between TFRs

|  |  |  |  |
| --- | --- | --- | --- |
| 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。

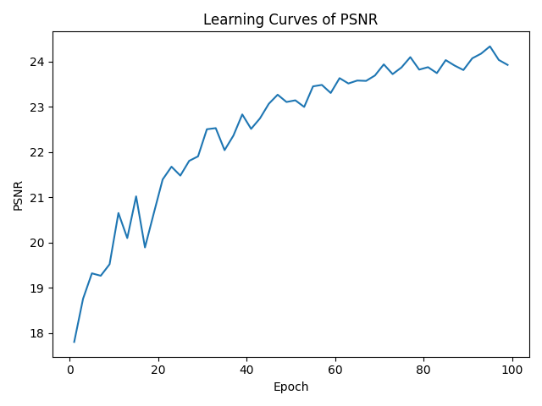
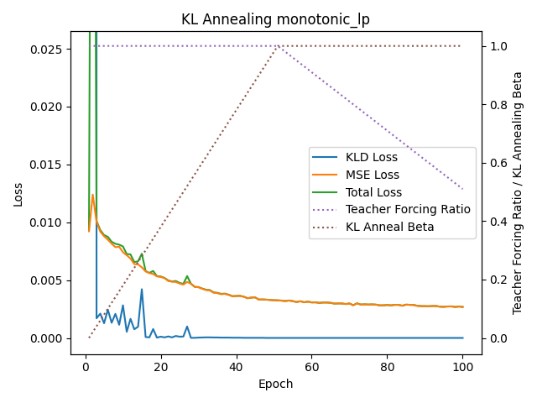
1. Extra

(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。

1. 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。