Lab5: Conditional VAE For Video Prediction

張千祐
Department of Computer Science
National Yang Ming Chiao Tung University
qianyou.cs11@nycu.edu.tw

1 Introduction

在這次 lab 中，目標是希望能從 VAE 的概念上，增加條件機率的機制，實作 Conditional VAE(CVAE) 以用來作連續影像的預測，也就是說，在訓練及測試時，除了給予訓練的資料，會另外給予前一幀的影像作爲條件輸入，以預測這一幀的影像。資料集是機器手臂推動球體的畫面，每一段影片含有 30 幀的照片，同時 action 和 end-effector position 也包含在資料集當中，並希望能夠透過實作 teacher forcing 和 KL annealing 的機制，來比較及改善模型的預測表現，過程中我們可以自行調整 hyper parameters 來取得更高的準確率，最後畫出 PSNR 對 epoch 曲線來視覺化結果。

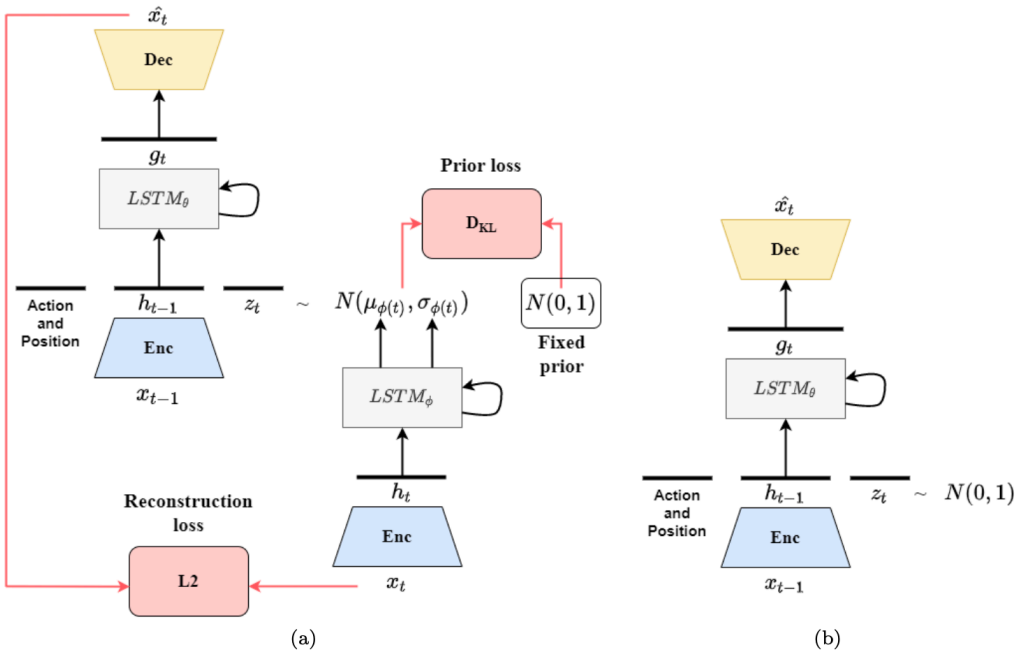


Figure 1: The illustration of overall framework

2 Derivation of CVAE

Fig 2 and Fig 3 show the derivation of CVAE.

From (L13, page 23), we have

$$\log p(x|c;\theta) = \log p(x,z|c;\theta) - \log p(z|x,c;\theta)$$

We next introduce an arbitrary distribution $q(z|c)$ on both side and integrate on z

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

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

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

$$+ \int q(z|c) \log q(z|c)\, dz - \int q(z|c) \log p(z|x,c;\theta)\, dz$$

$$= L(x,c,q,\theta) + KL(q(z|c) \| p(z|x,c;\theta)) \quad \cdots ①$$

where $L(x,c,q,\theta) = \int q(z|c) \log p(x,z|c;\theta)\, dz - \int q(z|c) \log q(z|c)\, dz$

By rearrange ①, we have

$$L(x,c,q,\theta) = \log p(x|c;\theta) - KL(q(z|c) \| p(z|x,c;\theta))$$

Figure 2: Derivation of CVAE(1)

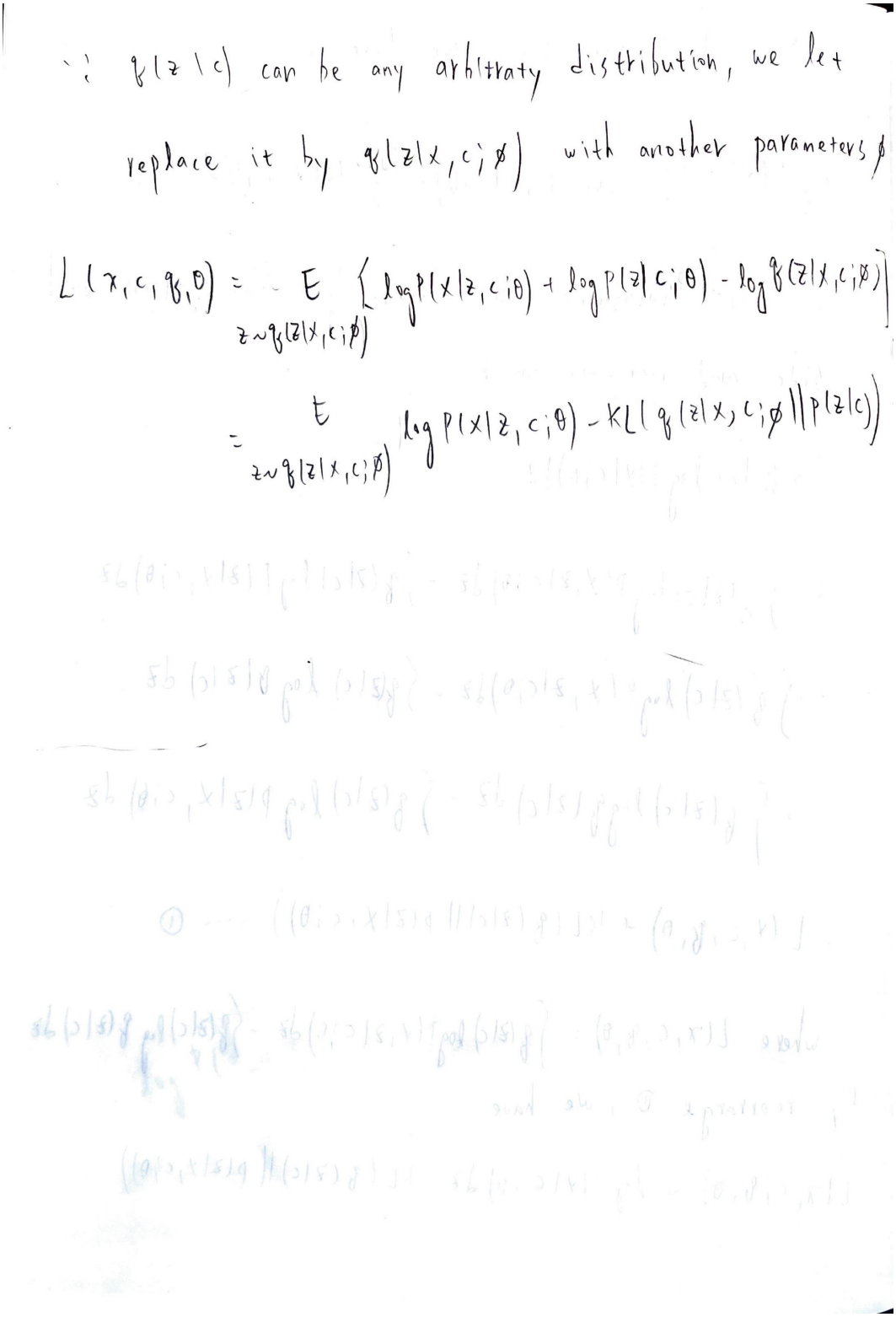


Figure 3: Derivation of CVAE(2)

3 Implementation details

3.1 Describe how you implement your model

因爲 encoder, decoder 的部分是給予的 sample code，不需要改動，因此這兩部分我會簡要的介紹 sample code 裡面是如何實作的。

3.1.1 Vgg Layer

Encoder 和 decoder 的組成單元都是 vgg layer，vgg layer 是由一層 convolution layer 組成，經過 batchnormalization 和 activation function 後輸出。

```
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)
                )

    def forward(self, input):
        return self.main(input)
```

Listing 1: Class vgglayer

3.1.2 Encoder

Encoder 是由數層的 vgg layer 組成，其中層數越深時 hidden size 越大，在最後一層會將 dim 設爲 output dim，其間的每一層最後都有經過 maxpooling layer。

3.1.3 Decoder

Decoder 同樣是由數層的 vgg layer 組成，其中層數越深時 hidden size 越小，在最後一層會將 dim 設爲 output dim，其間的每一層最後都有經過 upsampling layer。

3.1.4 Reparameterization trick

Reparameterization trick 的方法如 Fig 4所示，將原本需要進行積分的運算改爲乘加運算，使得整個模型可以進行 end-to-end 的 training。

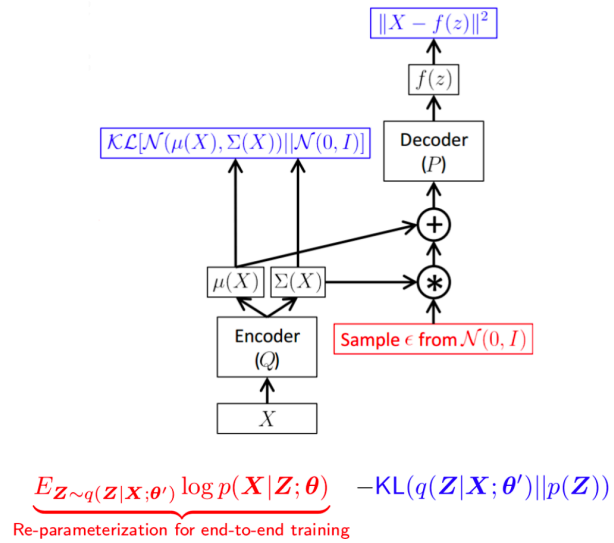


Figure 4: The illustration of reparameterization trick

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

Listing 2: Reparameterization

3.1.5 Dataloader

Dataloader 會分別回傳 image sequence 和 condition，分別由 get_seq 和 get_csv 來實作。

```python
def get_seq(self):
    if self.ordered:
        self.cur_dir = self.dirs[self.idx]
        if self.idx == len(self.dirs) - 1:
            self.idx = 0
        else:
            self.idx += 1
    else:
        self.cur_dir = self.dirs[np.random.randint(len(self.dirs))]

    image_seq = []
    for i in range(self.seq_len):
        fname = '{}/{}.png'.format(self.cur_dir, i)
        img = Image.open(fname)
        image_seq.append(self.transform(img))
    image_seq = torch.stack(image_seq)

    return image_seq

def get_csv(self):
    with open('{}/actions.csv'.format(self.cur_dir), newline='') as
    csvfile:
        rows = csv.reader(csvfile)
        actions = []
        for i, row in enumerate(rows):
            if i == self.seq_len:
                break
            action = [float(value) for value in row]
            actions.append(torch.tensor(action))

        actions = torch.stack(actions)

    with open('{}/endeffector_positions.csv'.format(self.cur_dir),
    newline='') as csvfile:
        rows = csv.reader(csvfile)
        positions = []
        for i, row in enumerate(rows):
            if i == self.seq_len:
                break
            position = [float(value) for value in row]
            positions.append(torch.tensor(position))
        positions = torch.stack(positions)

    condition = torch.cat((actions, positions), axis=1)

    return condition
```

Listing 3: dataloader

3.2 Describe the teacher forcing

Teacher forcing 的機制是在訓練時，將 input 由前一幀的 predicted frame 取代爲 ground truth frame 的方法。這麼做的好處是確保 input 是眞實的影像，因此 input 的分佈是眞

實資料的分佈。然而也有壞處，因爲我們訓練的目標就是讓模型能夠從前一幀的 frame 去預測下一幀的 frame，如果我們都不讓預測出來的 frame 作爲 input，那麼在測試時拿掉 teacher forcing 的機制後表現就會相對的比較差，因爲在訓練時並不是用預測出來的資料而是 ground truth。因此我們會傾向於建立一個隨著訓練 epoch 數遞減的 teacher forcing ratio，使得使用 teacher forcing 的機制的機率隨著訓練而減少。

4 Results and discussion

4.1 Hyper parameters

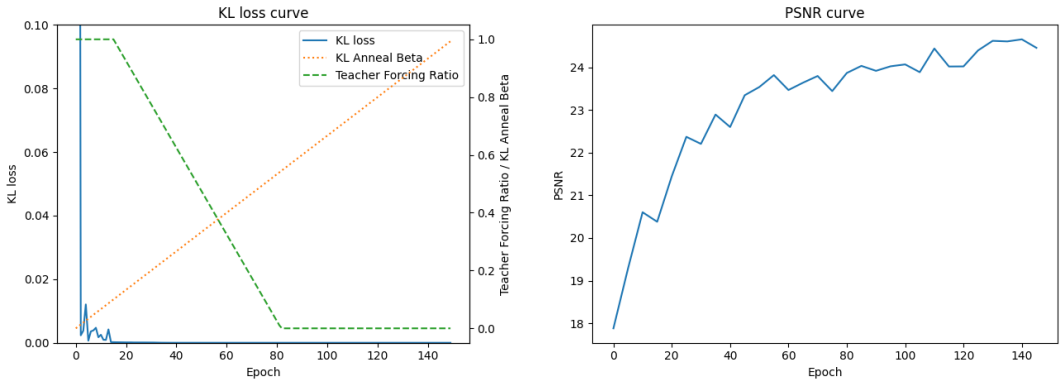
- Batch size=12
- Learning rate=2e-3
- Epochs=150
- Optimizer=ADAM
- tfr=1.0, tfr_start_decay_epoch=15, tfr_decay_step=0.015, tfr_lower_bound=0
- kl_annealing_cyclical=False

4.2 Show your results of video prediction

Figure 5: Prediction

Figure 6: Ground truth

4.3 Plot the KL loss and PSNR curves during training



(a) KL loss

(b) PSNR curve

Figure 7: The KL loss and PSNR curves during training

4.4 Discuss the results according to your setting

4.4.1 Teacher forcing ratio

從 Fig 7可以看到 Teacher forcing ratio 在約 80 個 epoch 後達到最低，訓練的 psnr 表現也差不多在 80 個 epoch 後停止上升，這個結果當然可能是巧合，而另外一個解釋就是 Teacher forcing ratio 確實對訓練表現產生蠻大的影響，因此我推測如果 tfr decay step 再小一點，另外再設個不爲 0 的 lower bound，使得 teacher forcing ratio 下降緩慢一點，或許還能夠有更好的表現。

4.4.2 KL weight

我自己在用比較小的 epoch 數量在做測試時，發現通常都是 cyclical 的 kl annealing 會比 monotonic 的表現還要來得好一點，但因爲最後在實際 training 時忘記設 cyclical 的 flag，所以 Fig 7畫的是 monotonic 下的 KL weight 的表現。若是在 cyclical 的情況下，KL loss 會隨著 KL anneal beta 上升而上升，而在 monotonic 下則是如 Fig 7一樣，在幾個 epoch 後 KL loss 就基本上等於 0。