Lab4 - Conditional VAE For Video Prediction

- Student Info
  1. Student ID: 310555024
  2. Student Name: 林廷翰

- Introduction (20%)

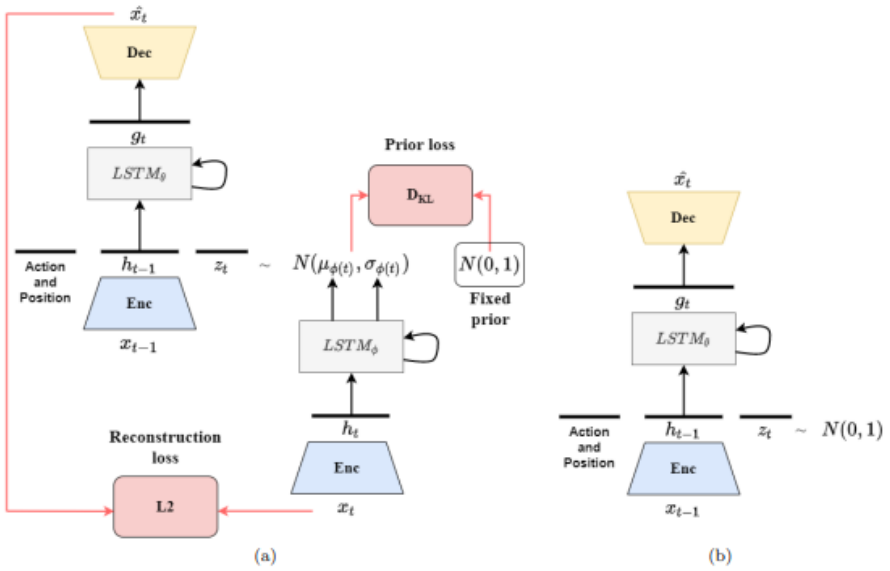


Figure 1: The illustration of overall framework. (a) Training procedure (b) Generation procedure

此次lab主要是基於RNN及auto encoder來實現連續的照片預測，再基於PSNR來計算產出與真實資料之間的差距，以優化網絡參數。

在網路中，encoder的輸出是一個mean及variance，因此不是單純的autoencoder而是variational autoencoder，且網路中會使用的RNN，RNN的特性即在於可以記錄連續照片之間的特性，做為下一張照片輸出的參考。

最後，再把 decoder 所產生的十張照片整合起來成 GIF 或者是影像檔案，看起來就會像是一段影像的預測或者說是一段影像的產生。

- Derivation of CVAE (10%)

$$p(x,z|c) = p(x|c)p(z|x,c)$$

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

左右同乘 $q(z)$ 並積分

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

$$= \int q(z)\log p(x,z|c) - \underbrace{\int q(z)\log q(z)\,dz}_{\text{extra}}$$

$$+ \underbrace{\int q(z)\log q(z)\,dz}_{\text{extra}} - \int q(z)\log p(z|x,c)\,dz$$

$$\rightarrow \log p(x|c) = \mathcal{L}(x,q|c) + KL(q(z|x,c)\,\|\,p(z|x,c))$$

$$\mathcal{L}(x,q|z) = \int q(z|x,c)\log p(x,z|c)\,dz - \int q(z|x,c)\log q(z)\,dz$$

$$\rightarrow E_{z\sim q(z|x,c,\theta')}[\log p(x|z,c,\theta)]$$

$$+ \underbrace{E_{z\sim q(z|x,c,\theta')}[\log p(z|c) - \log q(z|x,c}_{-KL(q(z|x,c;\theta)\,\|\,p(z|c))}$$

- Implementation details (15%)

  1. Encoder

     首先會將大小為 [n_past + n_future, epoch_size, 3, 64, 64] 的一組資料送進去我們的 encoder，通過五層的轉換之後將最後一層的輸出當成我們的 h 向量（LSTM 的輸入），encoder 剩餘前四層輸出會進一步送進去 decoder 當成 decoder 的輸入，如下圖:

```
def forward(self, input):
    h1 = self.c1(input)  # 64 -> 32
    h2 = self.c2(self.mp(h1))  # 32 -> 16
    h3 = self.c3(self.mp(h2))  # 16 -> 8
    h4 = self.c4(self.mp(h3))  # 8 -> 4
    h5 = self.c5(self.mp(h4))  # 4 -> 1
    return h5.view(-1, self.dim), [h1, h2, h3, h4]
```

  2. Decoder

     經過 decoder 之後會輸出跟原始輸入一樣大小的矩陣[n_past + n_future, epoch_size, 3, 64, 64]，並且透過 MSE Loss計算跟原始輸入差，此項就稱為 reconstruction term，透過此 MSE Loss 就可以去執行 Backpropagation 來跟新 decoder 以及encoder 的網路參數。

```
def forward(self, input):
    vec, skip = input
    d1 = self.upc1(vec.view(-1, self.dim, 1, 1))  # 1 -> 4
    up1 = self.up(d1)  # 4 -> 8
    d2 = self.upc2(torch.cat([up1, skip[3]], 1))  # 8 x 8
    up2 = self.up(d2)  # 8 -> 16
    d3 = self.upc3(torch.cat([up2, skip[2]], 1))  # 16 x 16
    up3 = self.up(d3)  # 8 -> 32
    d4 = self.upc4(torch.cat([up3, skip[1]], 1))  # 32 x 32
    up4 = self.up(d4)  # 32 -> 64
    output = self.upc5(torch.cat([up4, skip[0]], 1))  # 64 x 64
    return output
```

  3. Reparameterization Trick

     在上課中有提到，做reparameterization trick的目的在end-to-end的優化，如果不在架構中加入加及乘法運算，會導致無法透過objective function的結果優化encoder的參數，因此網路架構會變成下面的方式及實作:

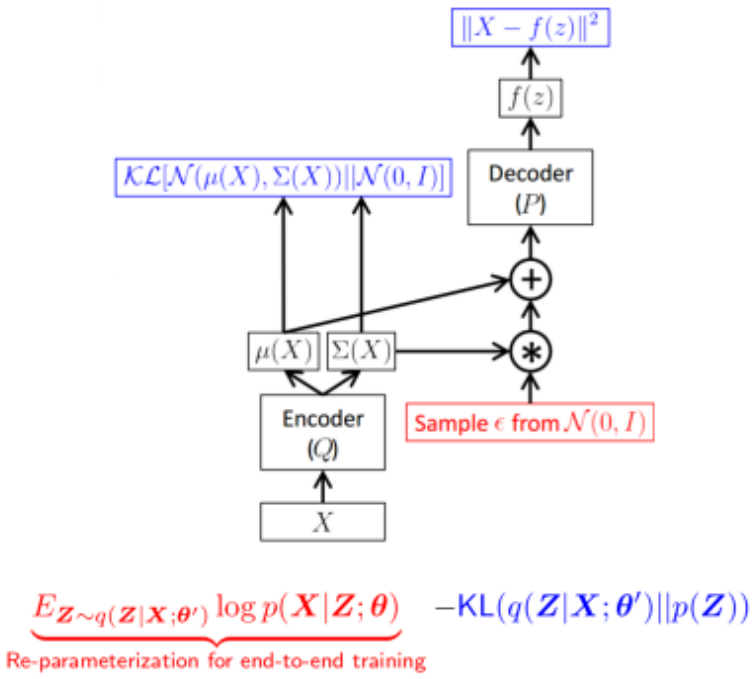


Figure 3: The illustration of reparameterization trick.

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

4. Dataloader

   每一個資料夾都代表是一段影片，影片由 30 張圖片組成，而我們只需要用前 2 張照片來預測後面 10 張照片，所以只需要將資料夾裡前 12 張照片讀進來，每張照片大小為[3, 64, 64]讀完 12 張之後維度會變成 [12, 3, 64, 64] 這邊要記得先用 To_Tensor 將圖片轉成 Tensor 的資料型態，如下:

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

    image_seq = []
    for i in range(self.seq_len):
        fname = os.path.join(self.cur_dir, f'{i}.png')
        im = self.transform(Image.open(fname)).view(1, 3, 64, 64)
        image_seq.append(im)
    image_seq = torch.cat(image_seq, dim=0)
    return image_seq
```

再來需要將 condition 也讀進來，裡面包含位址跟移動的一些資訊，這邊會直接將整份CSV資料讀進來，這邊也記得要將 nprray 的資料型態轉成 Tensor，但我們只需要將CSV 裡的前12 row 傳回去主程式就可以了，維度會是 [30, 7] à [12, 7]，如下:

```
def get_csv(self):
    d = self.cur_dir
    csv_seq = []
    actions = list(csv.reader(open(os.path.join(d, 'actions.csv'), newline='')))
    endeffector = list(csv.reader(open(os.path.join(d, 'endeffector_positions.csv'), newline='')))
    for i in range(self.seq_len):
        row_list = actions[i]
        row_list.extend(endeffector[i])
        csv_seq.append(row_list)
    csv_seq = torch.tensor(np.array(csv_seq).astype(np.float), dtype=torch.float)
    return csv_seq
```

5. Teacher Forcing

- Main idea

  這個技巧是在RNN當中很常使用的到一個技巧，最原始的網路架構在產生第 t 個時間點的輸出時，只會知道網路本身在前一刻時間點 t-1 所產生的輸出，並拿此拿來當成第 t 個時間點的輸入，藉此就能夠循序地往下產生輸出，但原始的這個方法可能會讓模型學到非常不理的特徵，舉例來說，在訓練時如果第 t 個時間點模型所產生的輸出已經是錯誤的結果，這時如果再拿此輸出來當成第 t+1 個時間點的輸入，有可能會讓模型“一步錯、步步錯”。

- Benefits

  在training初期，透過teacher forcing的方式可以加入訓練的流程，避免在訓練初期網路還不穩定時，就把錯誤的推測結果下傳。

- Drawbacks

  在實際prediction環境，產出是具有bias的，因此如果從頭到尾皆基於ground truth來訓練的話，會和實際場景具有差異，因此teacher forcing ratio應根據epoch數量增加而下降，以在實際場景predict能夠順利。

- Implementation

```
# Update teacher forcing ratio.
if epoch >= args.tfr_start_decay_epoch:
    total_delta = 1.0 - args.tfr_lower_bound
    total_decay_epoch = args.niter - args.tfr_start_decay_epoch
    slope = total_delta / total_decay_epoch

    tfr = 1.0 - (epoch - args.tfr_start_decay_epoch) * slope
    args.tfr = min(1, max(args.tfr_lower_bound, tfr))
```

- Result and Discussion

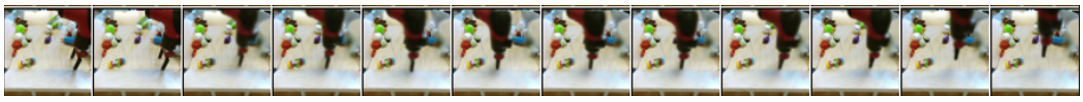
1. Show your results of video prediction (10%)

   - Make videos or gif images for test result

     下圖是GIF的截圖，夾帶於zip檔案中。 圖中顯示了每個時間步的基本事實和預測，預測序列的前2 frame是ground truth，接下來的10 frame是過去2 frame的未來預測。

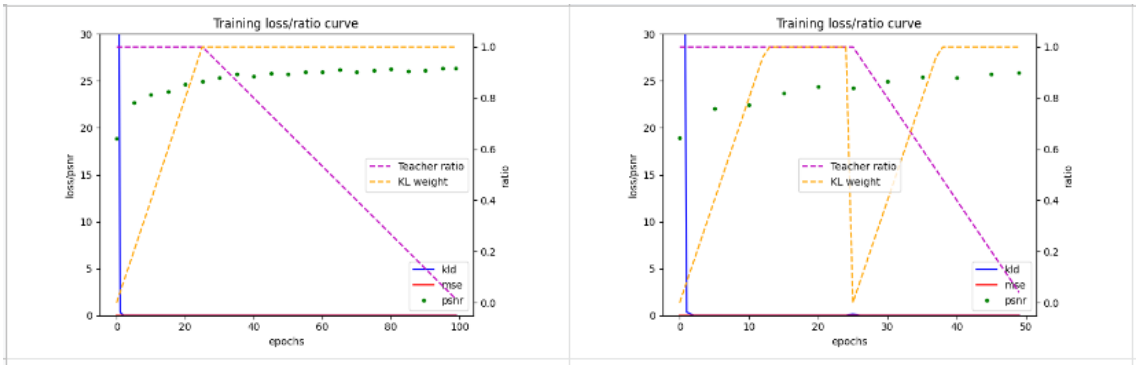
   - Output the prediction at each time step

     下圖基於前兩張image預測後面十張image的結果:



2. Plot theKL loss and PSNR curves during training (5%)

   下圖是training curve及對應的，基於兩種mode - monotonic和cyclical:



   - Parameters

lr=0.002, beta1=0.9, batch_size=32, optimizer='adam', niter=100, epoch_size=600, tfr=1.0, tfr_start_decay_epoch=25, tfr_decay_step=0, tfr_lower_bound=0.0, kl_anneal_cyclical=False, kl_anneal_ratio=2, kl_anneal_cycle=2, seed=1, n_past=2, n_future=10

3. Discuss the results (15%)

- Teacher forcing ratio

  此lab是使用前兩個真實frame來預測10個未來frame。因此，我們需要使用我們的預測frame來預測下一個frame。但是在訓練模型的時候，不能一開始就讓teacher force為0，原因是預測的輸入數據可能有偏差或可能過於模糊，這將影響我們的模型重建正確的frame，為了避免這種情況，我在前 25 個 epochs 中將teacher force ratio 設置為 1，然後線性衰減到 0。 這種方式可以讓模型在第一階段在給定地面實況輸入數據的情況下學習正確的重建。 然後在給定偏差輸入數據的情況下學習重建。

- KL weight

  cyclical的部分我只有train 50個epoch，不過PSNR的數據穩定上升，因此可預期當epoch數量更多實，應該可以再提高一些PSNR。

  但如果單就輸出結果來看，monotonic的方式相對起cyclical有更穩定上升的趨勢。