- Introduction

本次作業是想要我們透過給定的video frame和其他資訊，訓練一個CVAE，來預測未來的video frame。在這份作業中，我們可以學到如何訓練一個CVAE，和調整teacher forcing和KL weight來進一步提升效能。

- Derivation of CVAE

推導如下:

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

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

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

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

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

只要 maximize ELBO，就能提高 $\log p(x|c;\theta)$

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

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

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

$$= E_{z \sim q(z|x,c;\phi)}[\log p(x|z,c;\theta)] - KL(q(z|x,c;\phi) \| p(z|c))$$

跟VAE的推導過程差不多，只差在說CVAE在條件機率的部分多了variable c，也就是附帶的資訊，像這次作業中的兩個.csv。

- Implementation details

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

首先需要dataloader讀資料，在初始化的過程中，我將train/test/validate的資料夾目錄全存起來。

```
class bair_robot_pushing_dataset(Dataset):
    def __init__(self, args, mode='train', transform=default_transform):
        assert mode == 'train' or mode == 'test' or mode == 'validate'
        self.seed_is_set = False
        self.seq_len = args.n_past + args.n_future
        self.transform = transform
        self.getitem_dir = ''
        self.count = 0
        self.mode = mode
        self.root_path = args.data_root
        self.dir_list = []
        for d1 in os.listdir('%s/%s' % (self.root_path, mode)):
            for d2 in os.listdir('%s/%s/%s' % (self.root_path, mode, d1)):
                self.dir_list.append('%s/%s/%s/%s' % (self.root_path, mode, d1, d2))
```

然後在gititem時，我return了sequence:[frame num, 3, 64, 64]和condition:[frame num, 7]，而condition又分別是action: [frame num, 4]和position: [frame num, 3]合在一起。

```
def __getitem__(self, index):
    self.set_seed(index)
    seq = self.get_seq()
    cond = self.get_csv()
    return seq, cond
```

然後在get_seq和get_csv時，如果是test/validation，就照資料夾目錄順序output，否則就隨便抽一個資料夾output data。

```
def get_seq(self):
    img_list = []
    if (self.mode== 'test') or  (self.mode=='validate'):
        self.getitem_dir = self.dir_list[self.count]
        self.count = (self.count+1)%len(self.dir_list)
    else:
        self.getitem_dir = self.dir_list[np.random.randint(len(self.dir_list))]
    for i in range(self.seq_len):
        pre_img = Image.open("%s/%d.png" % (self.getitem_dir, i))
        img = self.transform(pre_img)
        img_list.append(img)
    img_list = torch.stack((img_list))
    return img_list
```

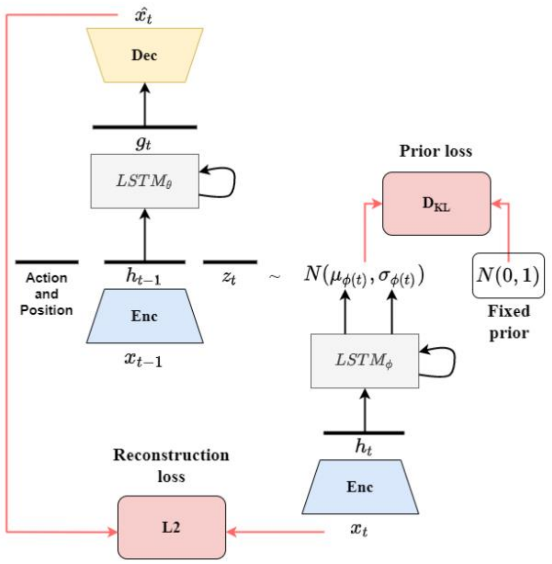
```python
def get_csv(self):
    csv_list=[]
    with open("%s/%s.csv" % (self.getitem_dir, 'actions'), newline='') as csv_file:
        actions_rows = csv.reader(csv_file)
        actions_rows = list(actions_rows)
    with open("%s/%s.csv" % (self.getitem_dir, 'endeffector_positions'), newline='') as csvfile:
        endeffector_positions_rows = csv.reader(csvfile)
        endeffector_positions_rows = list(endeffector_positions_rows)
    for i in range(self.seq_len):
        a_float = []
        for a in actions_rows[i]:
            a_float.append(float(a))
        e_float = []
        for e in endeffector_positions_rows[i]:
            e_float.append(float(e))
        cond = torch.Tensor(a_float + e_float)
        csv_list.append(cond)
    csv_list = torch.stack(csv_list,0)
    return csv_list
```

處理完dataloader後，接下來要準備encoder,decoder,lstm和gaussian_lstm。而助教很好心，model唯一需要實作的部分只有reparameterize。Reparameterize的原因是因為直接對高斯z分布取值會無法做back propagation。因此要用上reparameterization trick，變成對一個平均值=0變異數=1的隨機變數取值，再乘上變異數加上平均值，實作如下:

```python
def reparameterize(self, mu, logvar):
    var = torch.exp(logvar/2)
    eps = torch.normal(0,1,size=logvar.size()).to("cuda")
    return eps*var+mu
```

有了模型，接下來是訓練過程。訓練過程基本上照著下圖跑。

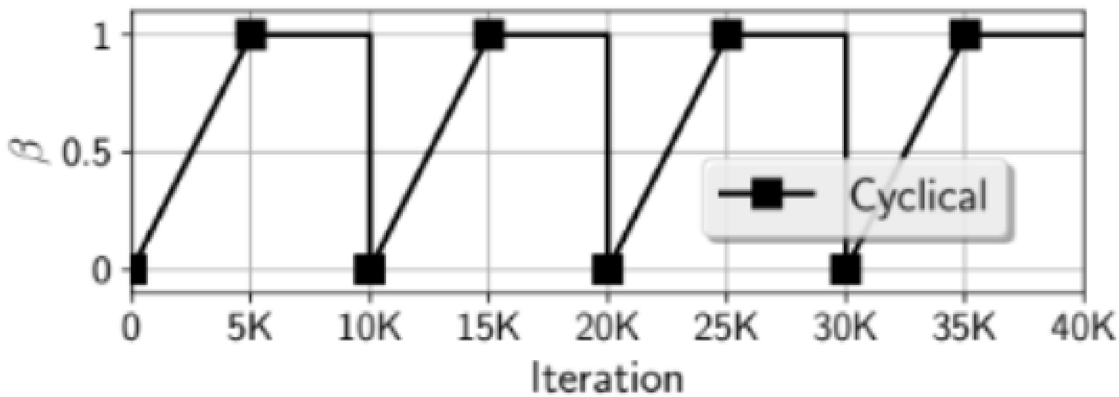
先讓t-1和t時間的x經過encoder分別得到h和h_target，然後讓h_target經過gaussian_lstm得到抽樣後的z，再讓z和h和action,position丟進lstm和decoder中，得到預測pred_x，最後再拿pred_x和t時間x下去算MSE_Loss，然後平均數變異數也要拿去算KL_divergence。最後再做back propagation和gradiant descent。



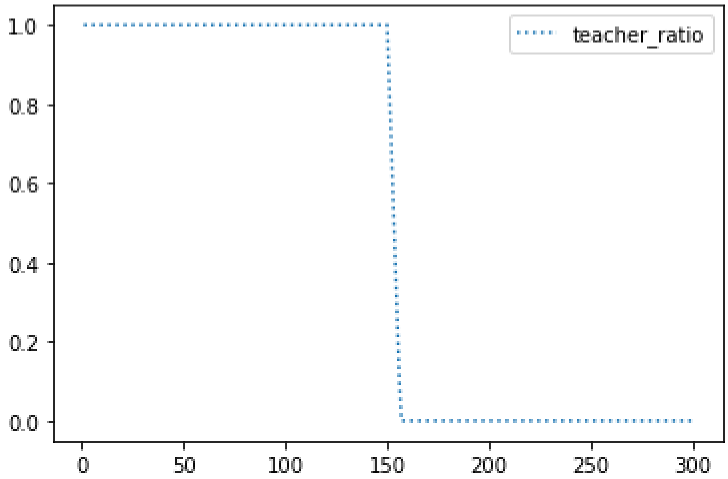
實作如下:

```
x_pred = x[0]

for i in range(1, args.n_past + args.n_future):
    if random.random() < args.tfr:
        use_teacher_forcing = True
    else:
        use_teacher_forcing = False
    h_target = modules['encoder'](x[i])[0]

    if args.last_frame_skip or i < args.n_past:
        if use_teacher_forcing == True:
            h, skip = modules['encoder'](x[i-1])
        else:
            h, skip = modules['encoder'](x_pred)
    else:
        if use_teacher_forcing == True:
            h = modules['encoder'](x[i-1])[0]
        else:
            h = modules['encoder'](x_pred)[0]

    z_t, mu, logvar = modules['posterior'](h_target)
    h_pred = modules['frame_predictor'](torch.cat([cond[i-1], h, z_t], 1))
    x_pred = modules['decoder']([h_pred, skip])
    mse += nn.MSELoss()(x_pred, x[i])
    kld += kl_criterion(mu, logvar, args)

beta = kl_anneal.get_beta()
loss = mse + kld * beta
loss.backward()

optimizer.step()
```

KL annealing的部分，我就照著pdf上的圖，讓KL annealing有cyclical，至於有cyclical的效果好不好，留到後面討論。



Teacher forcing的部分，因為niter我設300，而我在niter=150，直接將teacher forcing ratio從1關成0，至於這種作法好不好，留到後面討論。



hyperparameter的部分，我只讓lr變成5e-4，剩下都照的sample code的參數設。

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

Teacher forcing的想法就像說學生考卷寫個題組題，如果每寫一題，老師都檢討一次，那效果肯定比做完全部題組題後再檢討還好。因為基本上題組題前面錯了，後面題目接續著前面的答案，所以後面題目的答案也不會對。

而teacher forcing也是遵循著這種想法，如果當次epoch有開teacher forcing，那有關上個epoch的output當這個epoch的input時，要把其改成ground truth。

```
if random.random() < args.tfr
    use_teacher_forcing = True
else:
    use_teacher_forcing = False
h_target = modules['encoder'](x[i])[0]

if args.last_frame_skip or i < args.n_past:
    if use_teacher_forcing == True:
        h, skip = modules['encoder'](x[i-1])
    else:
        h, skip = modules['encoder'](x_pred)
else:
    if use_teacher_forcing == True:
        h = modules['encoder'](x[i-1])[0]
    else:
        h = modules['encoder'](x_pred)[0]
```

Teacher forcing的好處就是在teacher forcing情況下收斂的很快，而且只要teacher forcing練的夠好，直接關掉teacher forcing效果也會很好，就好比題組題都做對，不管老師是每寫一題檢討一遍，還是全部寫完再檢討，應該都是對的。

Teacher forcing的缺點就是如果一直開著teacher forcing，那在validation時因為沒有ground truth的支持，會讓模型變得脆弱，導致預測結果和最佳結果有一定的落差。

- Results and discussion

－ Show your results of video prediction

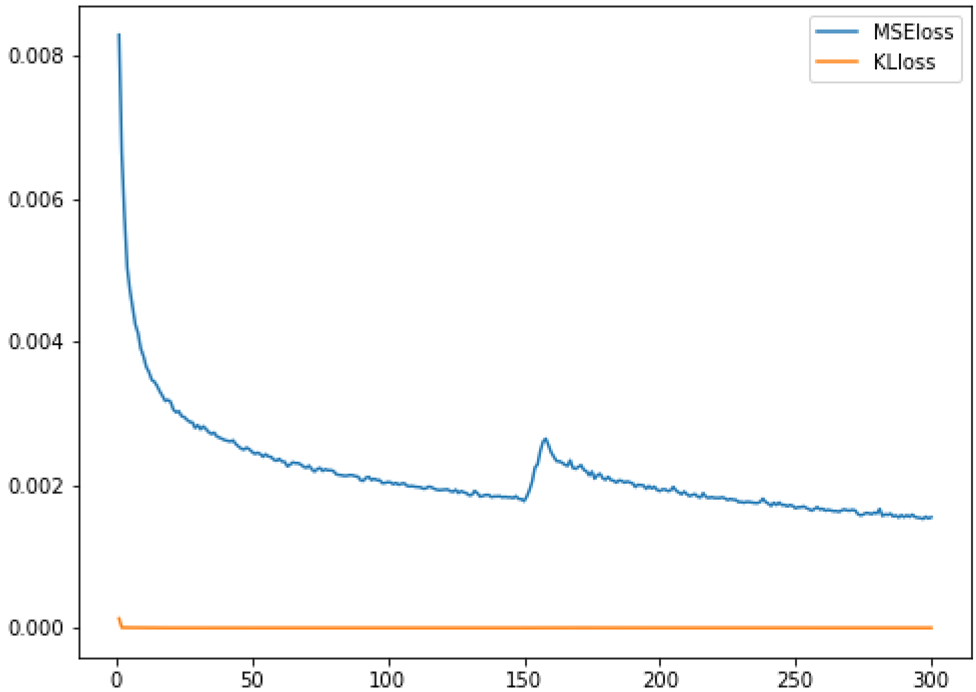
(a) Make videos or gif images for test result (select one sequence)

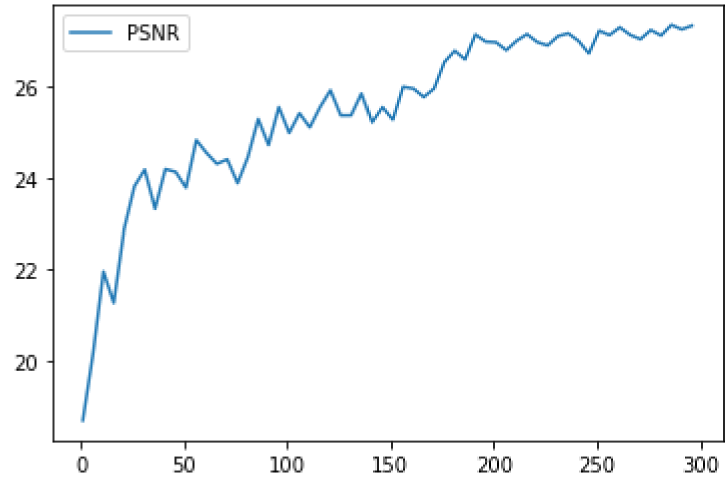
PDF我也不知道怎麼上傳gif，所以就麻煩助教自己上去看了，感恩。

https://drive.google.com/file/d/1G11l74JeRPZ47GJU4e6_1EojFB4B9TT5/view?usp=sharing

(b)Output the prediction at each time step (select one sequence)

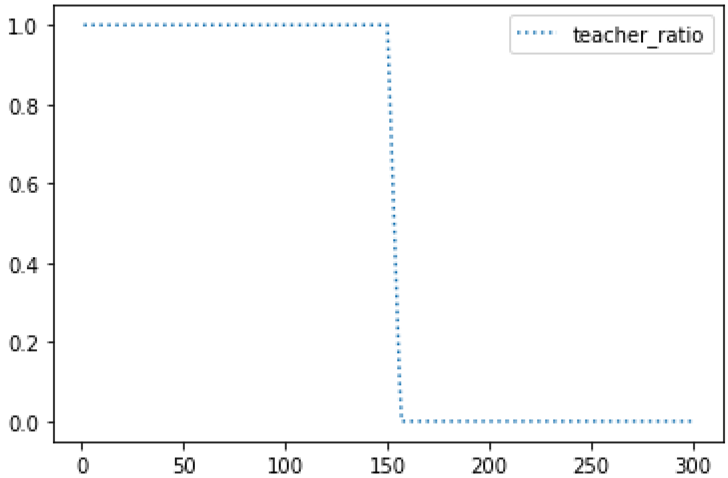
－ Plot the KL loss and PSNR curves during training





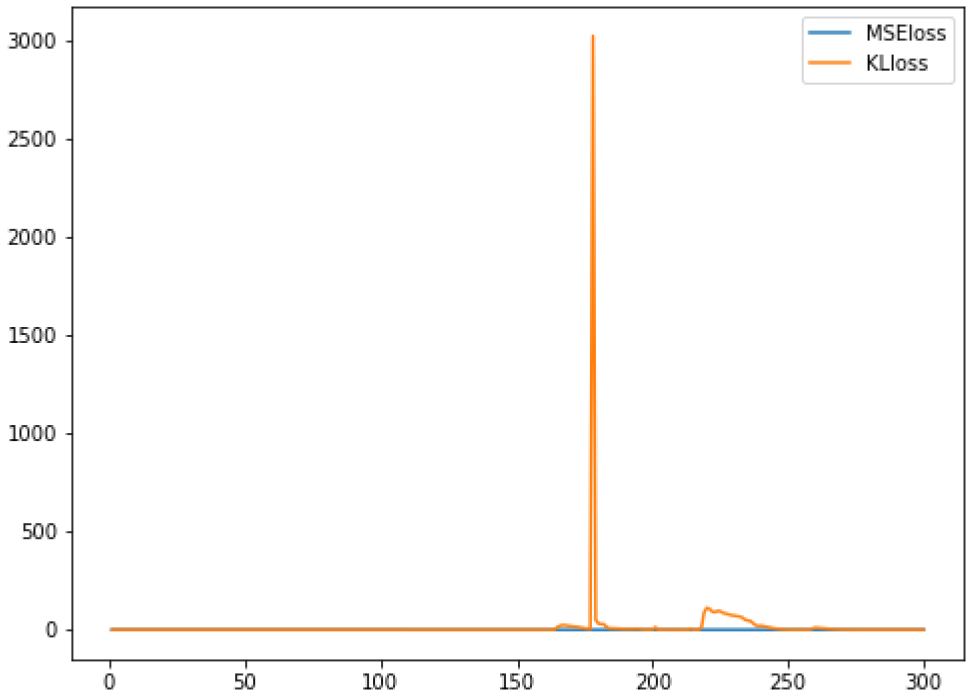
－ Discuss the results according to your setting of teacher forcing ratio, KL weight, and learning rate.

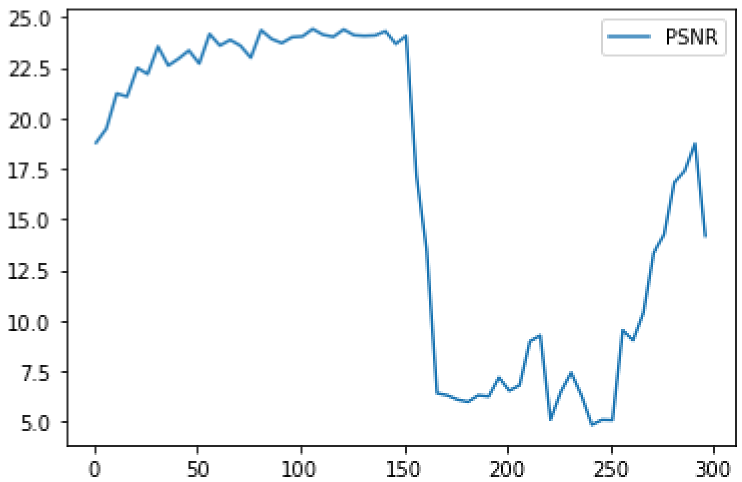
(1)其實上面的結果滿神奇的，前面提到，我的teacher forcing ratio是直接在第150niter時，從1直接降到0。



但關掉teacher forcing後卻沒練爛。我認為這個主要是因為當lr=5e-4時，在niter=150的情況下，PSNR有25。光在全開teacher forcing的情況下，就能練的夠好了，教授上課有提到說，有沒有teacher forcing的情況下，最佳化的那個點應該是相同的。所以結論就是，只要teacher forcing的情況下練的夠好，直接關掉tf應該也能練得還不錯。

當然如果teacher forcing的情況下練的不夠好，直接關掉teacher forcing的話，後面就很容易爛掉。像當lr=2e-3時，在niter=150的情況下，PSNR不到25，結果會如下:





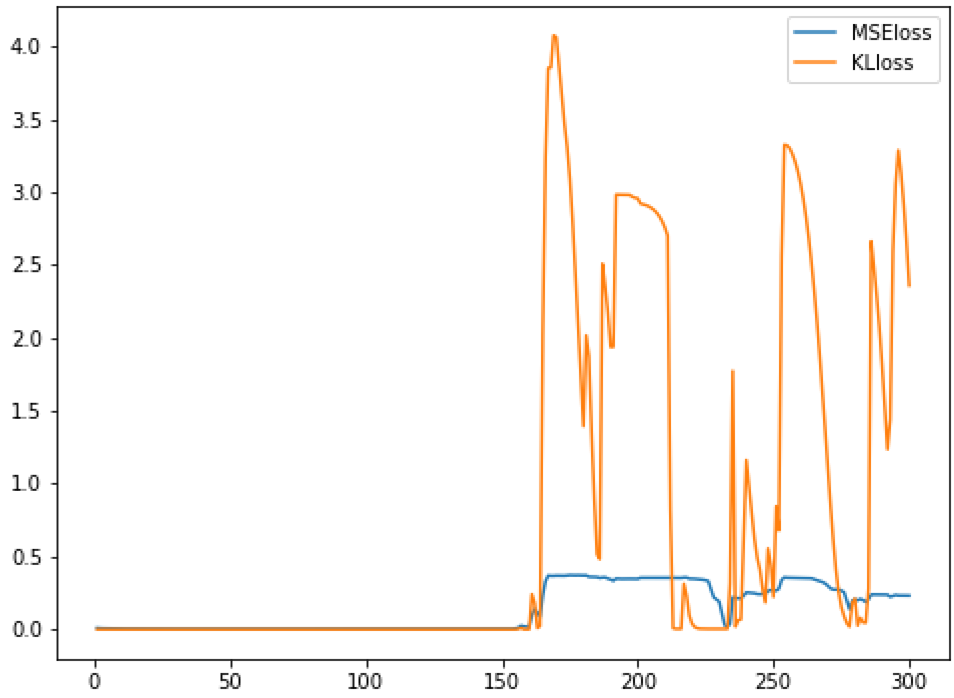
解決方法就是應該把teacher forcing ratio慢慢降低，而不是讓teacher forcing ratio驟降。

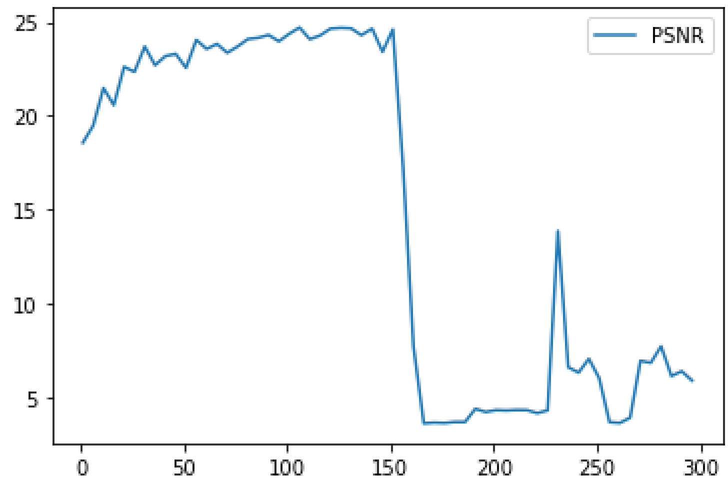
(2)

另外，KL annealing可以有效改善kl vanishing的問題。

在lr=2e-3時，打開KL annealing cyclical，結果就會如discussion(1)的圖片，發現雖然突然關掉teacher forcing會讓PSNR掉的很低，不過最後因為有改善kl vanishing，後期PSNR還是有慢慢練回去。

但如果關掉KL annealing cyclical，使其成為monotonic，這樣變相代表說只有在初期才有KL annealing的效果，後期會關掉KL annealing。最後會發現說關掉teacher forcing後，根本就沒辦法練起來。就如下圖。


MSEloss
KLloss
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
50
100
150
200
250
300


PSNR
25
20
15
10
5
0
50
100
150
200
250
300