Report

• Introduction (5%)

In this lab, we need to implement a conditional VAE (cVAE) for video prediction. For every 2 input frames, we need to use the trained model to predict the following 10 frames. And the “conditional” mean there are also some label / condition be a part of input.

Compare to original paper, this Lab add a teacher forcing way to may the result converge more quickly.

• Derivation of CVAE (Please use the same notation in Fig.1a)(10%)

We regard the action / position as condition c.

In the slide of VAE, we can get the following function:

$$\mathcal{L}(\boldsymbol{X}, q, \boldsymbol{\theta}) = \int q(\boldsymbol{Z}) \log p(\boldsymbol{X}, \boldsymbol{Z}; \boldsymbol{\theta}) d\boldsymbol{Z} - \int q(\boldsymbol{Z}) \log q(\boldsymbol{Z}) d\boldsymbol{Z}$$

Covert it to conditional form.

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

And we regard z and c have no directly relation, and the

$p(Z|c; \theta)$ would equal to $p(Z; \theta)$

then,

$$L(X,c,q,\theta) = -\,KL(q(Z|X,c;\phi)||p(Z|c)) + E_{Z\sim q(Z|X,c;\phi)} \log p(X|Z,c;\theta)$$

• Implementation details (15%)

– Describe how you implement your model (encoder, decoder, reparameterization trick, dataloader, etc.). (10%)

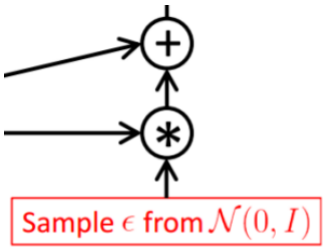
About my encoder and decoder, I use the given code directly. So, the structure is vgg in vgg_64.py. Adding the extra layer in the model not effect the result obviously.

```
# ------------ build the models  --------------

if args.model_dir != '':
    frame_predictor = saved_model['frame_predictor']
    posterior = saved_model['posterior']
else:
    frame_predictor = lstm(args.g_dim+args.z_dim, args.g_dim, args.rnn_size, args.predictor_rnn_layers, args.batch_size, device)
    posterior = gaussian_lstm(args.g_dim, args.z_dim, args.rnn_size, args.posterior_rnn_layers, args.batch_size, device)
    frame_predictor.apply(init_weights)
    posterior.apply(init_weights)

if args.model_dir != '':
    decoder = saved_model['decoder']
    encoder = saved_model['encoder']
else:
    encoder = vgg_encoder(args.g_dim)
    decoder = vgg_decoder(args.g_dim)
    encoder.apply(init_weights)
    decoder.apply(init_weights)
```

About reparameterization trick:



To implement the part in the figure. The only thing is sample and multiple to the logvar and the mu.

```
def reparameterize(self, mu, logvar):
    logvar = logvar.mul(0.5).exp_()
    eps = Variable(logvar.data.new(logvar.size()).normal_())
    return eps.mul(logvar).add_(mu)
```

About dataloader:

According the folder name, we can know the data is from the TensorFlow's TFRecords file. We can get the image directly, so I use the double glob the store all image in the dataset.

```
assert mode == 'train' or mode == 'test' or mode == 'validate'

self.ordered = False
if mode == 'test':
    self.ordered = True

self.seed_is_set = False

self.data_dir = os.path.join('/DATA/2022DLP/LAB5/data/processed_data', mode)
self.filenames = glob.glob(os.path.join(self.data_dir, '*'))

more_filenames = []
for name in self.filenames:
    tmp = glob.glob(os.path.join(name, '*'))
    for tmp2 in tmp:
        more_filenames.append(tmp2)

self.filenames = more_filenames
self.seq_len = args.n_past + args.n_future
self.seed_is_set = False
self.d = 0
```

– Describe the teacher forcing (including main idea, benefits and drawbacks.) (5%)
In the normal situation, we use the previous state as next state's input. And the main idea of teacher forcing is using the ground truth as next state's input.
In the way, the model can learn the feature and converge more quickly.
But the drawbacks are that in the test stage, we don't have the ground truth anymore. So, the model generate output could not have a good performance in the such different situation.(Exposure Bias)

• Results and discussion (30%)
– Show your results of video prediction (10%)
(a) Make videos or gif images for test result (select one sequence)
Because I can't put gif in the word file, I add the website.

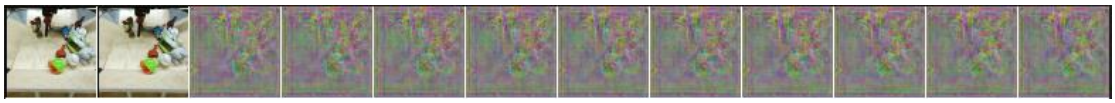
Not trained:
https://i.imgur.com/0QjlFPe.gif

Trained:
https://i.imgur.com/HBKrBC0.gif

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

The 2 past frames + 10 future frames (take randomly)

NOT trained:



Epoch 0:

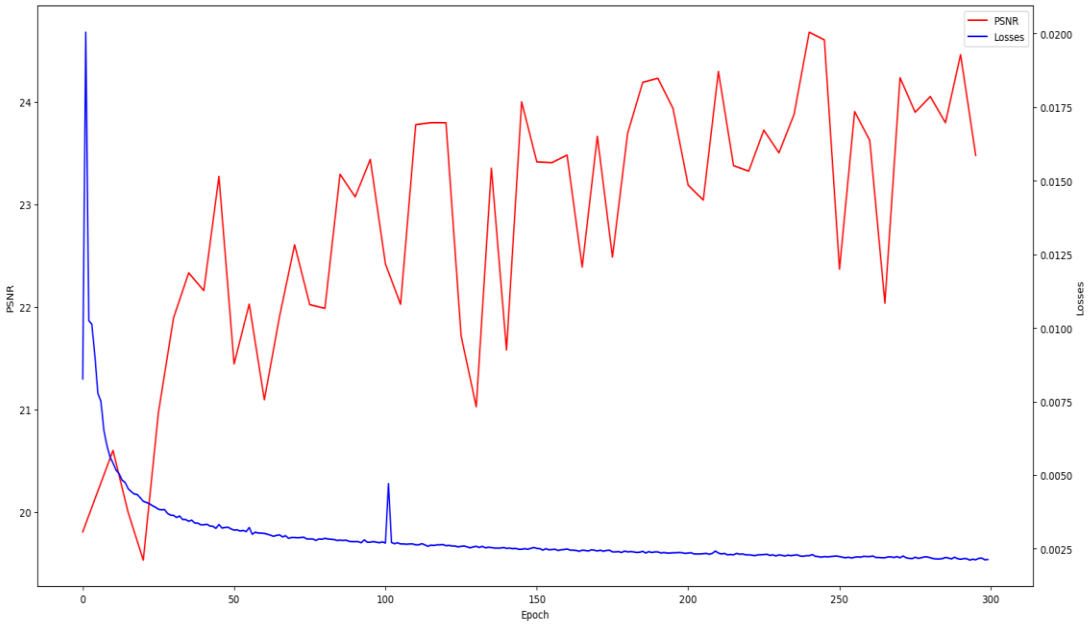
Epoch 60:

Epoch 120:

Epoch 180:

Epoch 240:

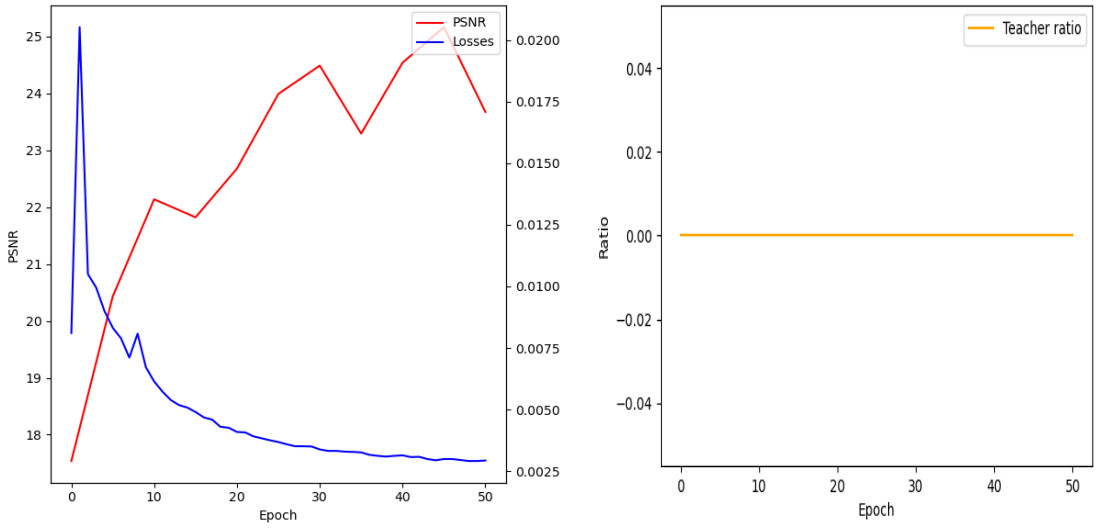
– Plot the KL loss and PSNR curves during training (5%)



Red: PSNR Blue: loss

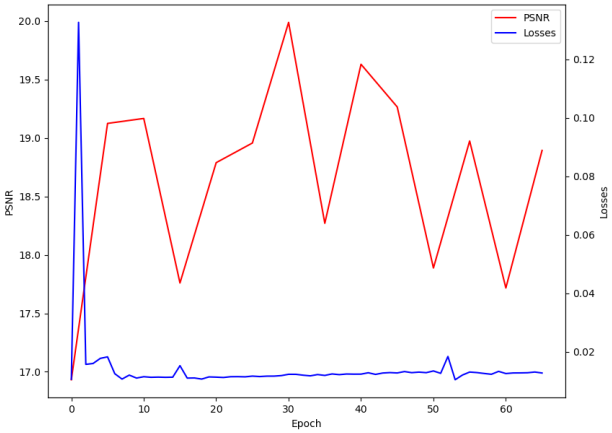
– Discuss the results according to your setting of teacher forcing ratio, KL weight, and learning rate. Note that this part mainly focuses on your discussion, if you simply just paste your results, you will get a low score. (15%)

Because of the training time is too long, I can only do some simple test. In the following case, without teacher forcing, it reaches psnr 25 within 50 epochs. This is sort of weird. Theoretically using teacher forcing would converge more quickly.




```
========================= test psnr = 24.47583 ==========================
```

And use the model, it actually get more than 24 in test stage (the upper one used for demo only about 23)



If we set the lr = 0.01, obviously, the model cant not find and converge to an optimal solution that is good enough.