1. Derivate conditional VAE formula

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

aribitrary distribution $q(z|x,c;\varphi)$

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

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

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

$$\Rightarrow E_{z\sim q(z|x,c;\varphi)}\log p(x|z,c;\theta)$$

$$= \mathcal{L}(x,c,q,\theta)$$

$$+ D_{KL}(q(z|x,c;\varphi)\,\|\,p(z|c;\theta))$$

$$\Rightarrow \mathcal{L}(x,c,q,\theta) = E_{z\sim q(z|x,c;\varphi)}\log p(x|z,c;\theta) - D_{KL}(q(z|x,c;\varphi)\,\|\,p(z|c;\theta)) \quad ※$$

$$D_{KL}(P\|Q) = \int p(x)\log\left(\frac{p(x)}{q(x)}\right)dx$$

$p(x|z,c;\theta)$ → Decoder

$q(z|x,c;\theta)$ → Encoder

圖.1、Derivate conditional VAE

2. Introduction

在這次實驗報告中，設計了一個 CVAE 的模型來做 Seq-to-Seq 類型的影像產生器，透過提供上一偵的圖形(或者預測圖形)和人體骨架(pose)來去預測下一個動作的圖片生成，此外因為是 Seq-to-Seq 類型的，因此在 Training 上採用跟 RNN 架構很像的方式，去做 forward 和 backward。

在訓練的實驗上有特別針對 teacher forcing 的訓練，也有使用 KL annealing 的方式來設定 VAE 模型中的β參數，在實驗數據中發現在不以 teacher forcing 的情況下訓練得到的效果是最好的，此外也有做其他類型的訓練，像是針對不同 Sequence 大小、不同 KL Annealing、甚至是透過 finetune 的方式去進一步的優化訓練。

3. Implementation details

How do you write your training/testing protocol

在 training 中，會先將輸入的圖片和標籤從 [Batch_size,Seq,C,H,W]轉成 [Seq,Batch_size,C,H,W]，這樣做的原因是直接對 Sequence 的每一項去 forward，也就是在一個 training 的 step 中會全部的 Sequence 全部 forward 完畢之後才會去做後續的 back-propagation 和 optimizer 的動作，在 optimizer 更新時為了防止梯度爆炸的問題，有使用到梯度裁減(clip_grad_norm)的動作，但是此動作無法有效的防止梯度消失的情況發生[1]。

在 training 的實作過程中 forward 的動作如圖.2 所示，上述有提到先對輸入的圖片和標籤(pose)做轉換，依序輸入每一個 frame 做 forward，在一開始會先對內部所有的網路重製梯度(zero_grad)，在程式中紀錄了一個產生出來的圖片串列，可以利用此串列不管是使用(a)Teacher-Forcing：在 Decoder fusion 階段直接以 ground-truth 的方式去做後續的 generate(b)Non-Teacher-Forcing：在 Decoder fusion 階段以上一張辨識出來的圖片做後續的 generate，此串列也可以用於後續儲存成一個 gif 的檔案。此外在每一次的 Sequence forwarding 中都會去比較在這過程中是否出現了梯度爆炸的情況，也就是說是否有任何在通過各個網路時出現 Nan 的情況，如果有的話那則是需要做跳過的情況，避免直接影響到整體的網路訓練。在 Training

中也會去計算每個 Sequence 下來的 PSNR 與 Loss 值，Loss 值的算法是以 $\beta - VAE$ 的方式去計算，由 KL-annealing 退火算法得到 β，再乘上 KL-Divergence 的值去算出最終的 Loss，最後會將 PSNR 和 Loss 回傳到 training_stage 的 function 中，在 training_stage 的 function 中如果有跳過的 Sequence(可能會出現梯度爆炸，而略過)則不會去計算它的平均 PSNR 與 Loss。

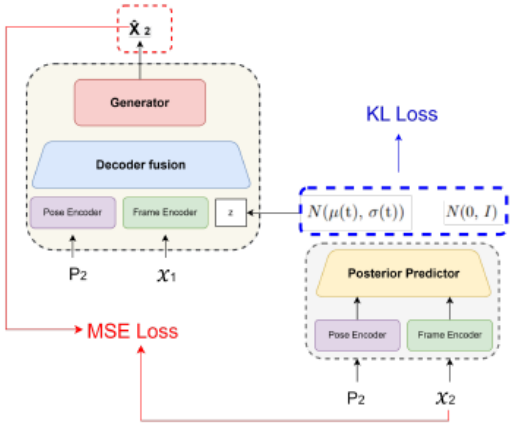


圖.2、training 時的步驟

此外，此實驗可以視作為一種 RNN 類的模型，在 Sequence 中的每一項都是一個小的網絡，而在 forward 時有先前提到的(a)、(b)兩種 Teacher-Forcing 的機制，如果有採用 Teacher-Forcing 的話則是直接以 Ground Truth 作為 Sequence 中下一個網絡的輸入，如圖.3 所示。

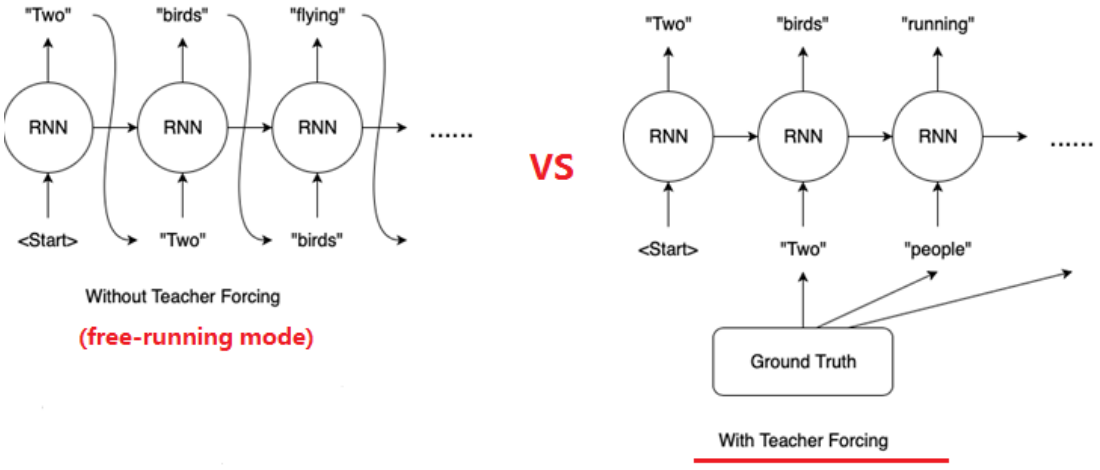


圖.3、在 RNN 中的 Teacher Forcing[2]

在 Testing 的階段則是以圖.4 所示，同樣會先將圖片轉成[Seq,B,C,H,W]的形式，再去對 Sequence 中每一個 frame 做 predict 生成，和在 Trainer.py

程式中的 validation 使用同樣的方法，但唯一不同的地方是 validation 因為有 ground truth 的關係，所以可以去算圖片的 PSNR 值，而 Tester 的程式則是直接給第一張圖片和 pose 的 label 去產生後續的連續動作圖片。

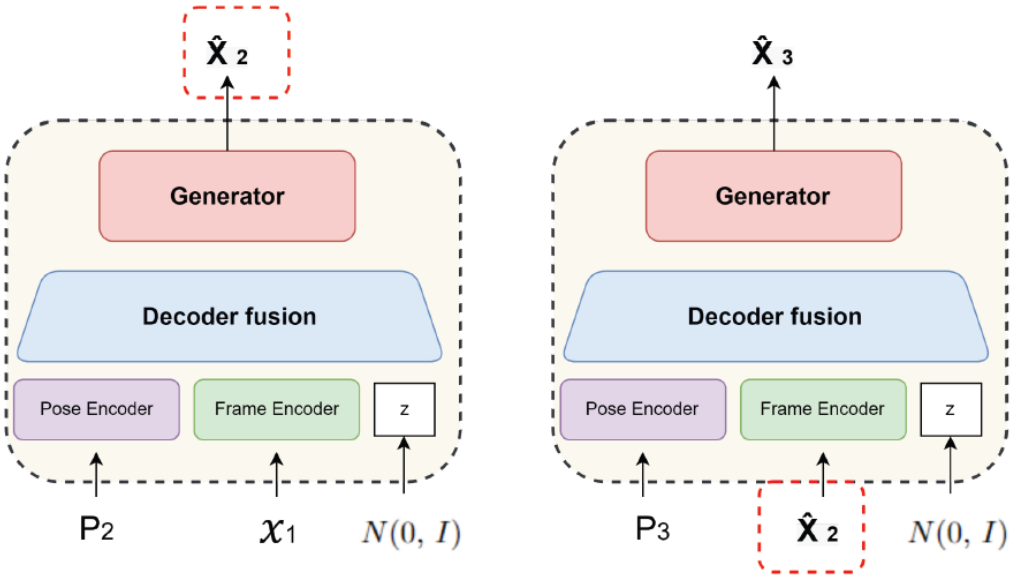


圖.4、Tester 程式的 Inference 步驟

圖.4 說明了 Tester 程式和 Trainer 程式中 Validation 的實作步驟，在這步驟中只針對整體網絡的 Decoder 的部分給資料，在資料集中第一張圖片 X_1 是固定提供的，接著生成後續的 Sequence($\hat{X}_2$ 、$\hat{X}_3$)等，而在 latent space 中則是給隨機的 normal distribution，且種子碼都是固定的，因此可以再每次使用時都產生相同的結果。

在 training_stage 時還有額外寫一些自動化的功能，和其他的 training strategy，其中自動化功能有自動儲存最佳的結果(--auto_save)，如果啟用的話，程式會去判斷此 validation 出來的結果是否達到(1.)PSNR 最高、Loss 最低、(2)PSNR 最高、(3)Loss 最低等情況，如果有的話則是無視每幾個 epoch 儲存(--per_save)的功能。

其他另類的 training strategy 則是會在下一章節做介紹。

How do you implement reparameterization tricks

在 VAE 架構中，使用 reparameterization trick [3, 4]可以簡單表示為針對 latent space 的值 z 進行取樣。在架構中 Encoder 會輸出平均值 μ 和對數平方差 $\log(\sigma^2)$，透過 reparameterization 的技巧，z 的值可以表示為 $z = \mu + \sigma \cdot \epsilon$，其中 ϵ 是來自標準正態分佈的隨機噪音 $\epsilon \sim N(0,1)$ [4]。

假設一個圖片x從 Encoder 中拿到了平均值$\mu(x)$和平方標準差$\sigma^2(x)$，但是 z 本身為 normal distribution $z \sim N(\mu, \sigma^2)$，想要從中抽樣並使用 reparameterization 的技巧，則需要重新計算他的標準差 σ，但是由程式中我們所拿到的是帶有 log 的值($\log(\sigma^2)$)，因此可以透過使用自然對數的方式將 log 取消掉並且乘上 0.5 可以將次方數移除，得到單一個標準差σ，因此程式的實作上如下方所示。

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

How do you set your teacher forcing strategy

在 teacher forcing strategy 時則是有用兩種，一種是透過每次扣除衰減值(--tfr_d_step)的部分，另一種則是判斷在 validation 時是否趨近收斂情況，如果趨近收斂的話則是會重新啟用 teacher forcing，持續訓練模型嘗試達到一個更好的結果。

在 Trainer 中有額外寫一個 TF_detector 的類別，用來判斷前 N 個(--detector_patience)validation 的 Loss 和 PSNR 差異是否小於一個設定值(--detector_threshold)，如果小於的話，代表可能趨近收斂的情況，則將 teacher forcing ratio 設置為 1，接著以原先給一個衰減值緩慢扣掉。

How do you set your kl annealing ratio

KL 退火(annealing)的實作方式則是以[5]為主，可調的參數有，在特定的訓練次數(--num_epoch)下重複多少次(--kl_anneal_cycle)圖形和在一次 cycle 中佔的比例(--kl_anneal_ratio)為多少，和退火的種類(--kl_anneal_type)等，圖.5 表示了論文中所提出的兩種種類 Monotonic 與 Cyclical，其中兩者的 num_epoch 都為 40K，Monotonic 的 cycle 占比(kl_anneal_ratio)為 0.25，cycle 次數(kl_anneal_cycle)為 1 次；而 Cyclical 則是占比為 0.5，次數為 4 次[5]。

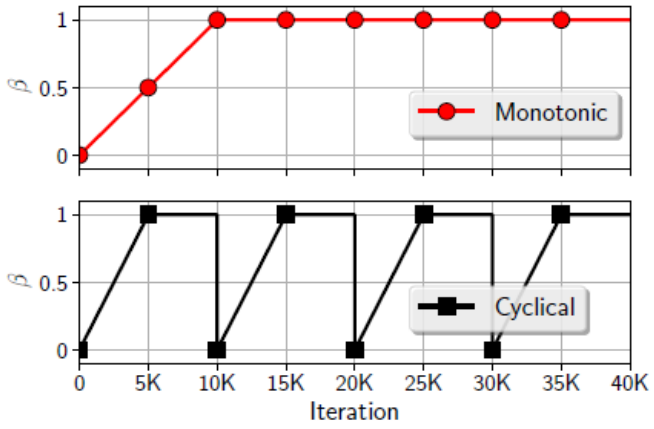


圖.5、Monotonic 與 Cyclical 比較圖[5]

4. Analysis & Discussion

Plot Teacher forcing ratio

在圖.6 中使用的訓練指令為

6.a
Python Trainer.py --batch_size 1 --lr 0.001 --device cuda --optim Adam --gpu 1 --DR ../LAB4_Dataset/LAB4_Dataset/ --save_root . --num_epoch 100 **--train_vi_len 32 --tfr 1.0** --tfr_sde 10 --tfr_d_step 0.1 --kl_anneal_type Cyclical --kl_anneal_cycle 10 --kl_anneal_ratio 1 --auto_save
6.b
Python Trainer.py --batch_size 1 --lr 0.001 --device cuda --optim Adam --gpu 1 --DR ../LAB4_Dataset/LAB4_Dataset/ --save_root . --num_epoch 100 **--train_vi_len 32 --tfr 0.0** --tfr_sde 10 --tfr_d_step 0.1 --kl_anneal_type Cyclical --kl_anneal_cycle 10 --kl_anneal_ratio 1 --auto_save

在使用 Sequence 為 32 且 kl annealing 為 Cyclical 的方法去訓練，從途中可以很明顯地看到，不使用 teacher forcing 的情況下，模型在自己 train 的過程能比一開始使用 teacher forcing 的情況還要來的好。

此圖中分四個區塊，分別是左上角(Training 和 Validation 的 PSNR 比較)、右上角(Training 和 Validation 的 Loss 比較)、左下角(同個 epoch 中 Beta、Teacher forcing ratio 和是否啟用 Teacher forcing(TF status))、右下角(learning_rate)，可以很清楚的看出在同樣 epoch 下不同訓練參數的差異。

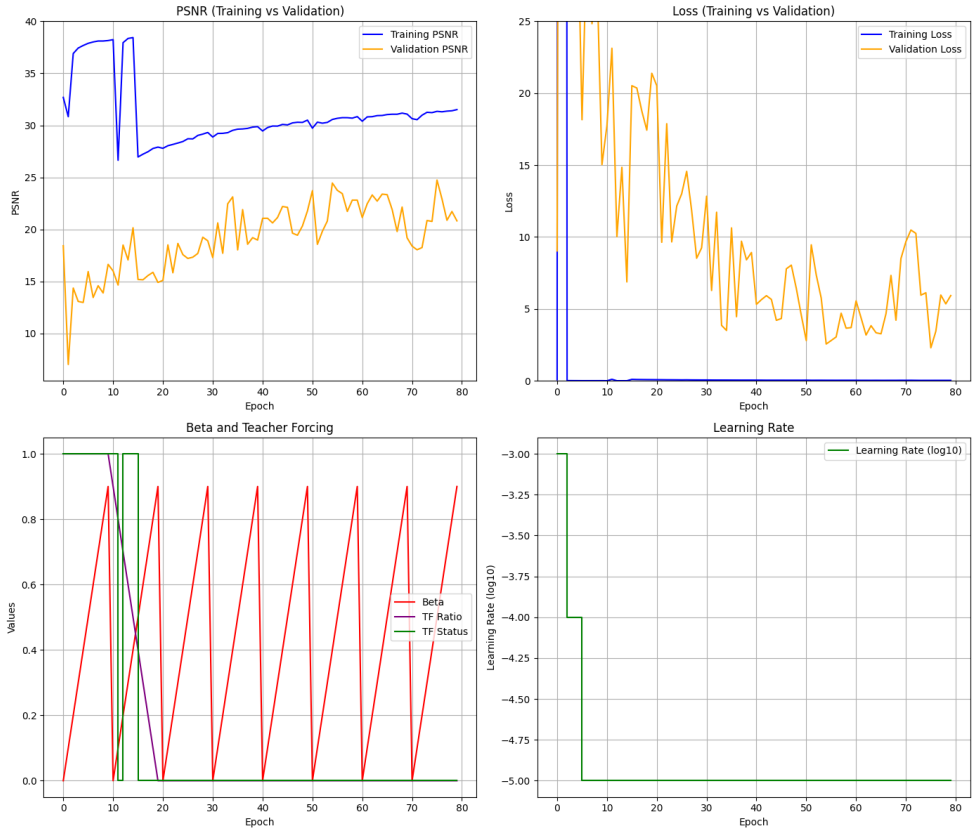


圖 6.a、有 Teacher Forcing 訓練 70 個 Epoch 圖

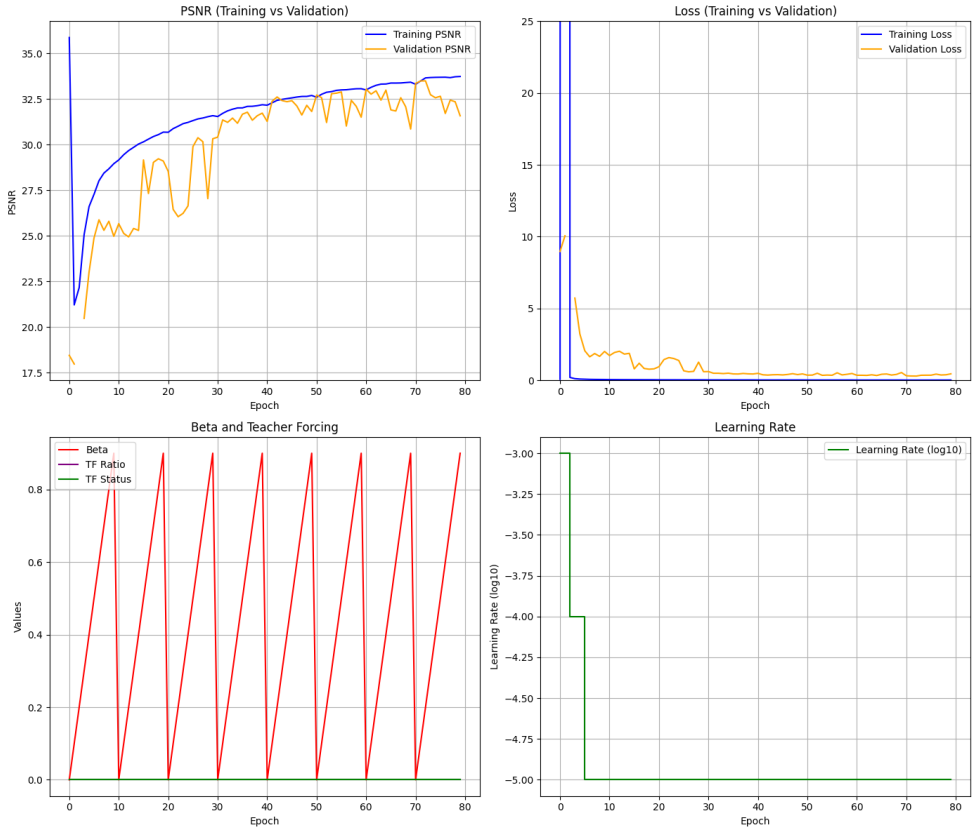
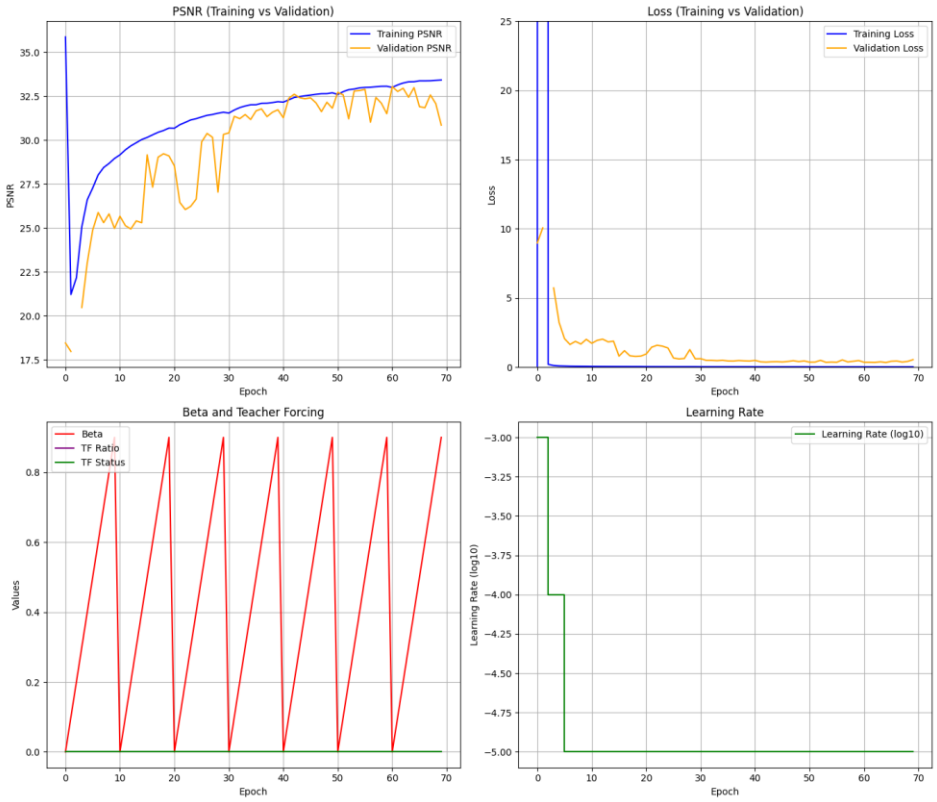


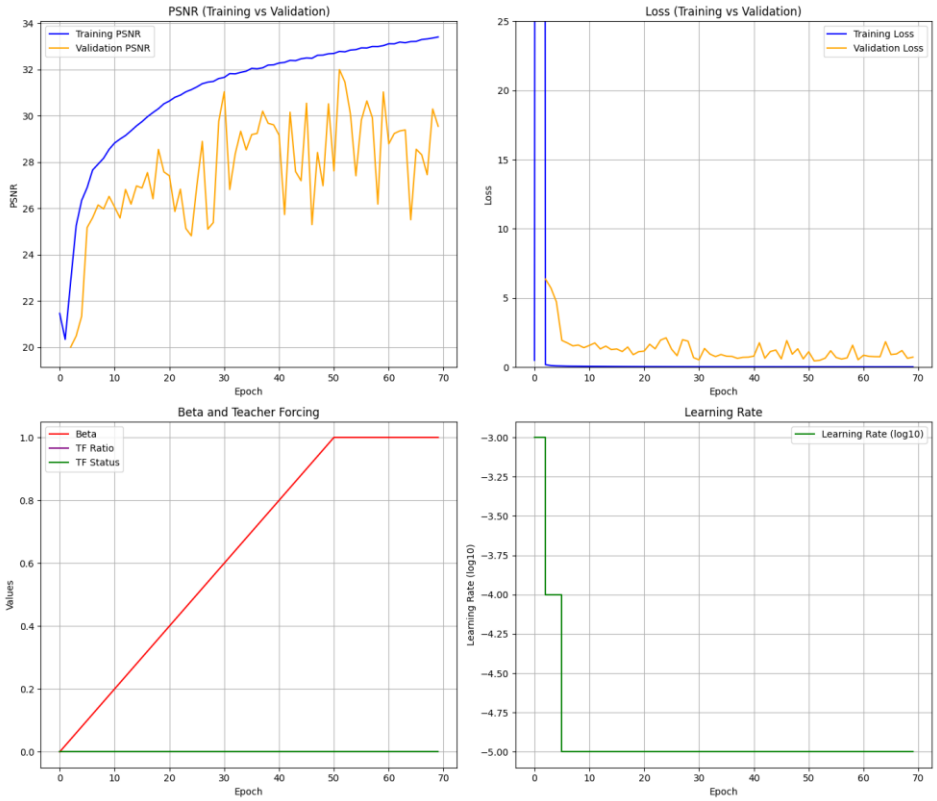
圖 6.b、無 Teacher Forcing 訓練 70 個 Epoch 圖

Plot the loss curve while training with different settings.

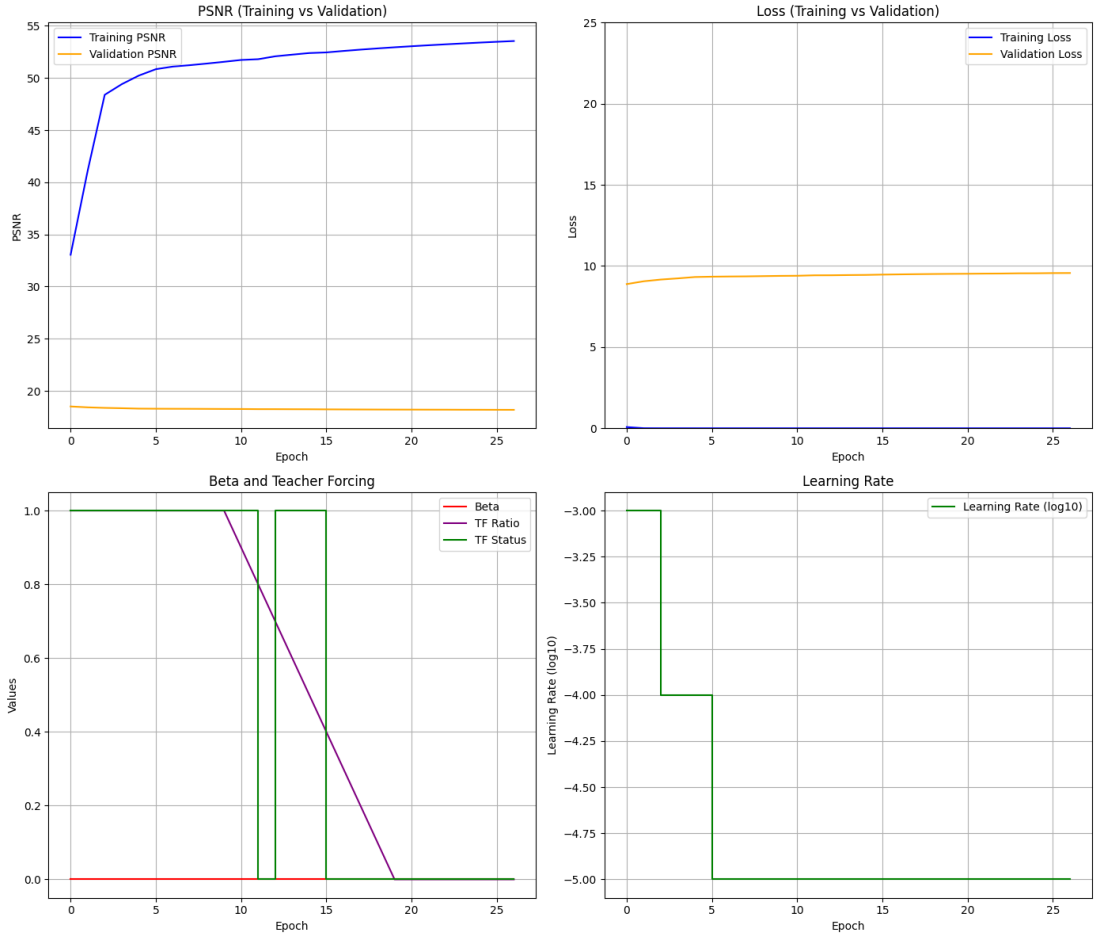
Cyclical (without teacher forcing)



Monotonic (without teacher forcing)



Without KL annealing ($\beta = 0$, with teacher forcing)



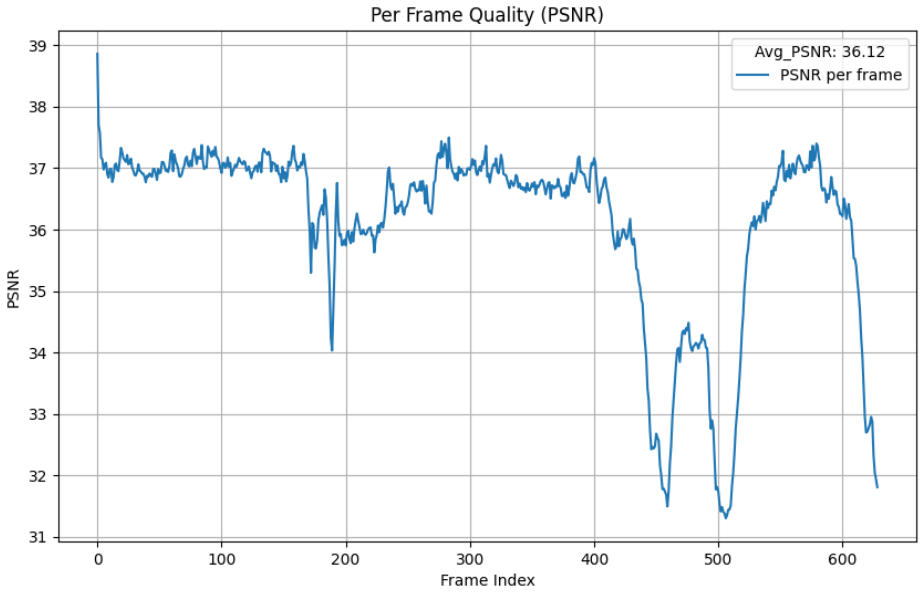
在這些訓練中發現，使用 Cyclical 的效果是比較好的，從圖中可以看到，為了避免跟沒有 annealing 的情況 overfitting 出現，導致在 Training 時 PSNR 很高但是 Validation 都無上升的情況，此外也發現了在 Monotonic 的情況下雖然 Training 和 Validation 的 PSNR 都很高，但是 Monotonic 的 validation 變動很大，不像 Cyclical 一樣，共同往上升。

Plot the PSNR-per frame diagram in validation dataset

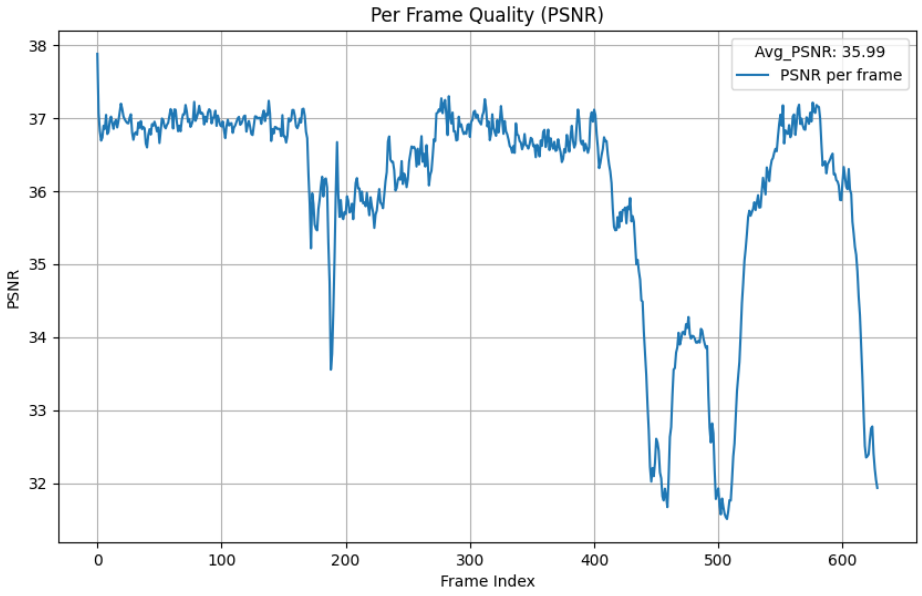
在一定數量的訓練次數和不同種類的訓練結果，發現彼此之間有很多相同的共通點，幾乎都是在特定的 frame 下會有明顯的下降，以下為由大到小的 PSNR 值和其訓練方式做比較：

AVG PSNR	KL annealing	Teacher Forcing	train_vi_len
36.12	Cyclical	X	32
35.99	Cyclical	X	32
31.99	Monotonic	X	32
31.88	Cyclical	X	64
28.49	Cyclical	O	64
27.38	Cyclical	O	32

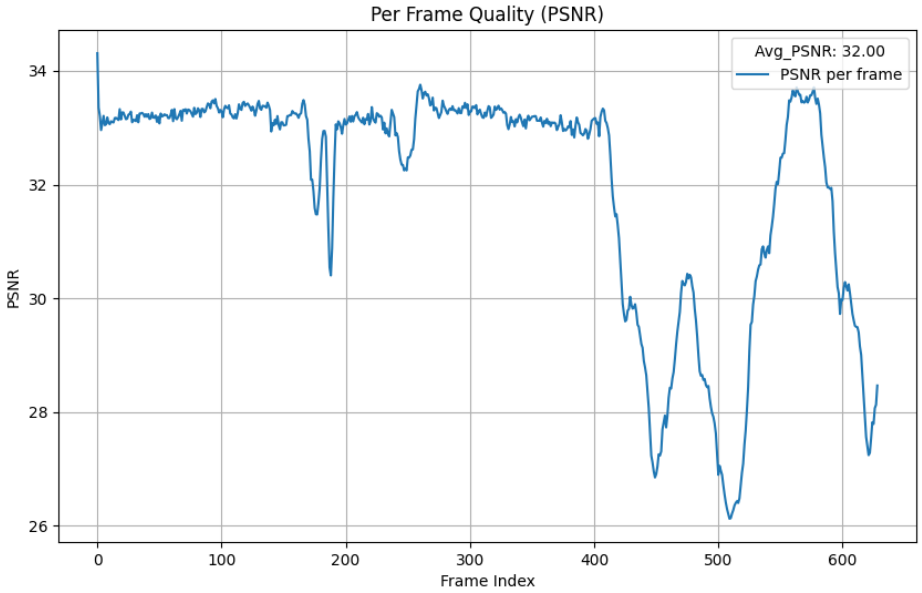
AVG PSNR 36.122 (epoch=205, cyclical, without tf,learning rate finetune)



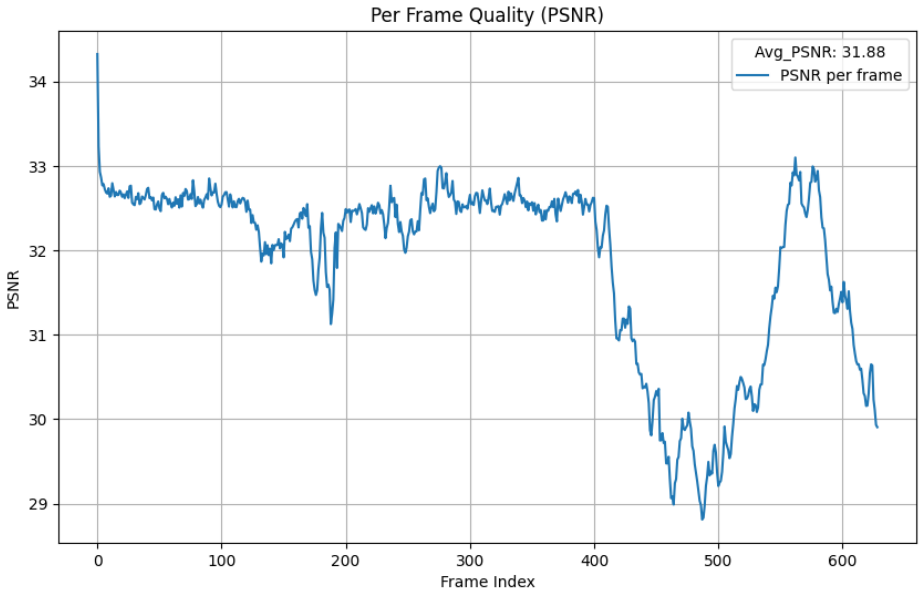
AVG PSNR 35.99 (epoch=99, cyclical, without tf, learning_rate finetune)



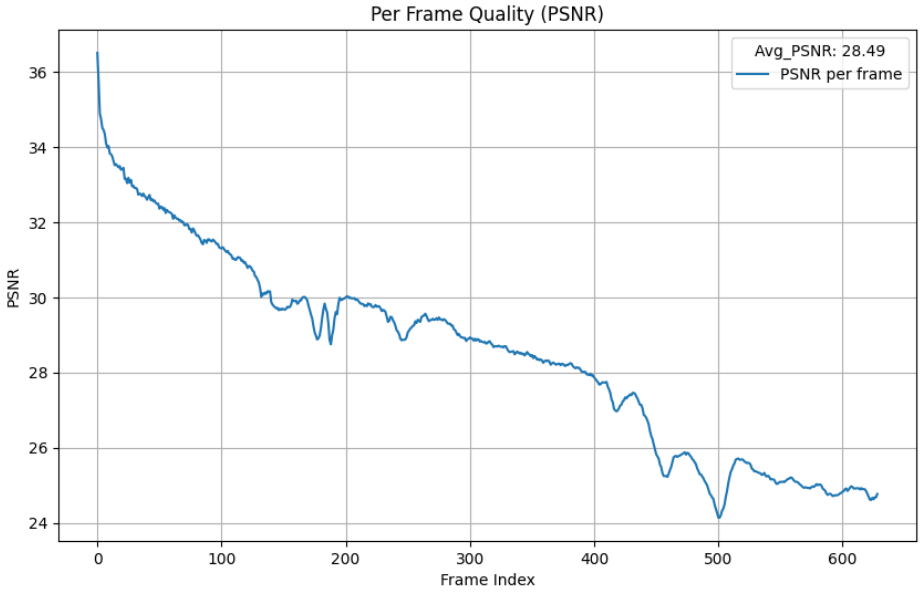
AVG PSNR 31.99 (epoch=51,monotonic, without tf)



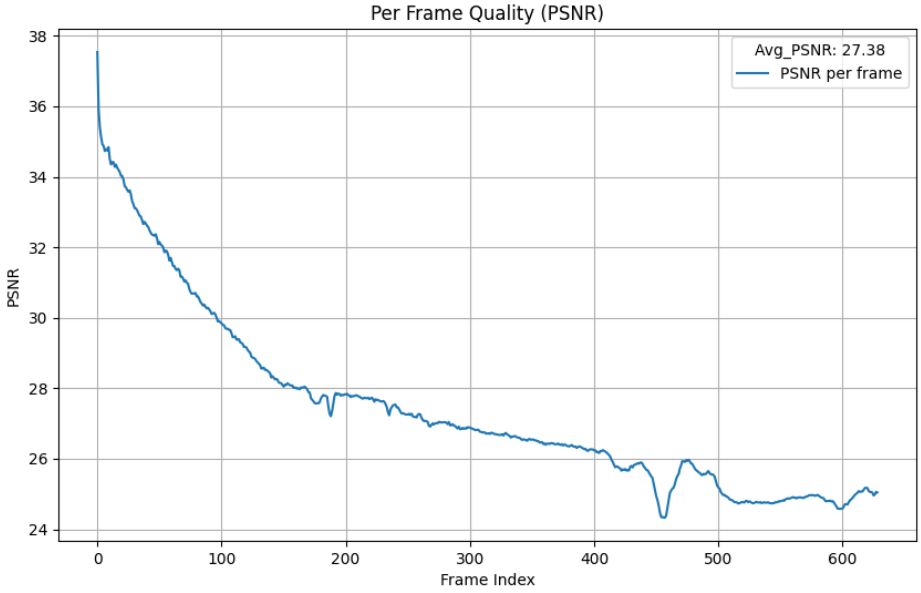
AVG PSNR 31.88 (epoch=92, cyclical, without tf)



AVG PSNR 28.49 (epoch=129, cyclical, with tf)



AVG PSNR 27.38 (epoch=96, cyclical, with tf)



Other training strategy analysis

除了修改參數之外，還有對已經訓練好的模型繼續做 finetune，發現透過調整學習率可以進一步的提升在 validation 時的效果，如圖.7 所示，在 epoch80 左右調整了 learning rate 使得模型往 loss 更低的最小值移動，達到更好的效果，此模型訓練參數為 (train_vi_len,cyclical,without teacher forcing)。

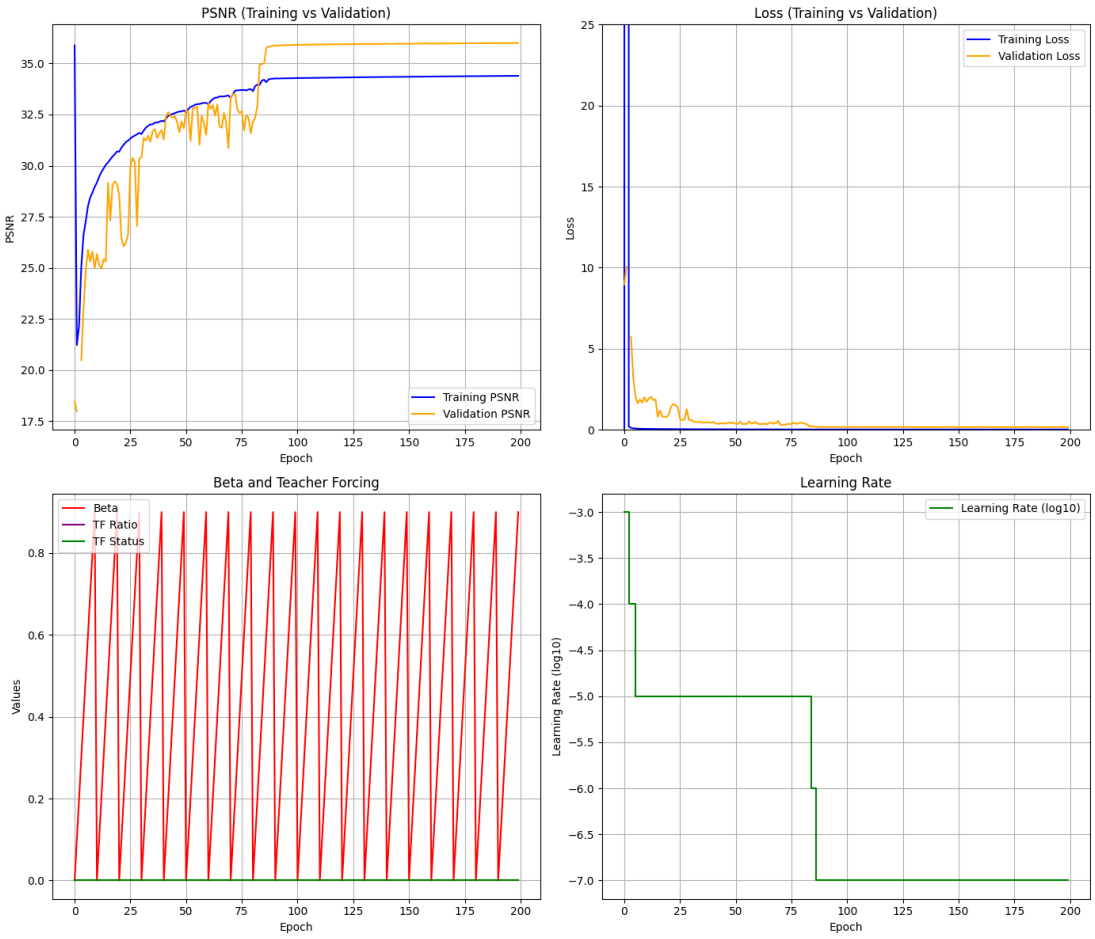


圖.7、調整 learning rate 達到更好效果的 finetune

5. Reference

1. Mikeyboi. *梯度剪裁: torch.nn.utils.clip_grad_norm_()*. 2021-08-08; Available from: https://blog.csdn.net/Mikeyboi/article/details/119522689.
2. MissHsu. *Teacher forcing 是什么？*. 2021-04-22; Available from: https://www.cnblogs.com/dangui/p/14690919.html.
3. JianJie. *DL、ML 筆記(12):Variational AutoEncoder (VAE)*. May 7, 2021; Available from: https://jianjiesun.medium.com/dl-ml%E7%AD%86%E8%A8%98-12-variational-autoencoder-vae-6d74bf83daa.
4. Jayakody, D., *The Reparameterization Trick - Clearly Explained.* Dec 19, 2023
5. Fu, H., et al., *Cyclical annealing schedule: A simple approach to mitigating kl vanishing.* 2019.