1. Derivate conditional VAE formula

一張含有 文字, 筆跡, 字型, 螢幕擷取畫面 的圖片

自動產生的描述

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值的算法是以的方式去計算，由KL-annealing退火算法得到，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。

一張含有 文字, 螢幕擷取畫面, 圖表, 字型 的圖片

自動產生的描述

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。

一張含有 圖表, 文字, 行, 寫生 的圖片

自動產生的描述

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。

一張含有 文字, 螢幕擷取畫面, 圖表, 字型 的圖片

自動產生的描述

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是固定提供的，接著生成後續的Sequence()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 會輸出平均值和對數平方差 ，透過reparameterization的技巧，z 的值可以表示為 ，其中 ϵ 是來自標準正態分佈的隨機噪音 [4]。

假設一個圖片從Encoder中拿到了平均值和平方標準差，但是 本身為normal distribution ，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 ，但是由程式中我們所拿到的是帶有log的值()，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，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]。

一張含有 文字, 圖表, 行, 繪圖 的圖片

自動產生的描述

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。

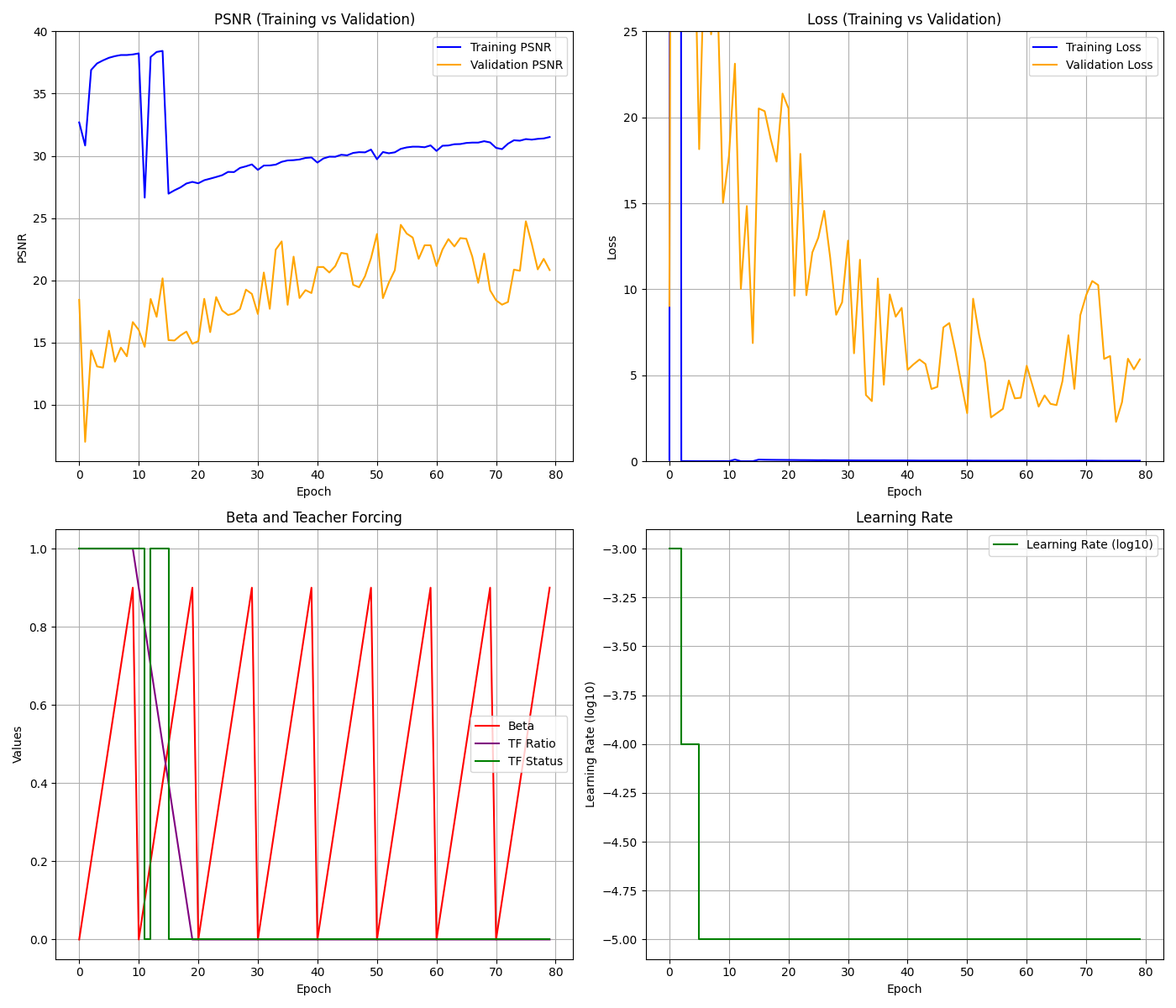


圖6.a、有Teacher Forcing訓練70個Epoch圖

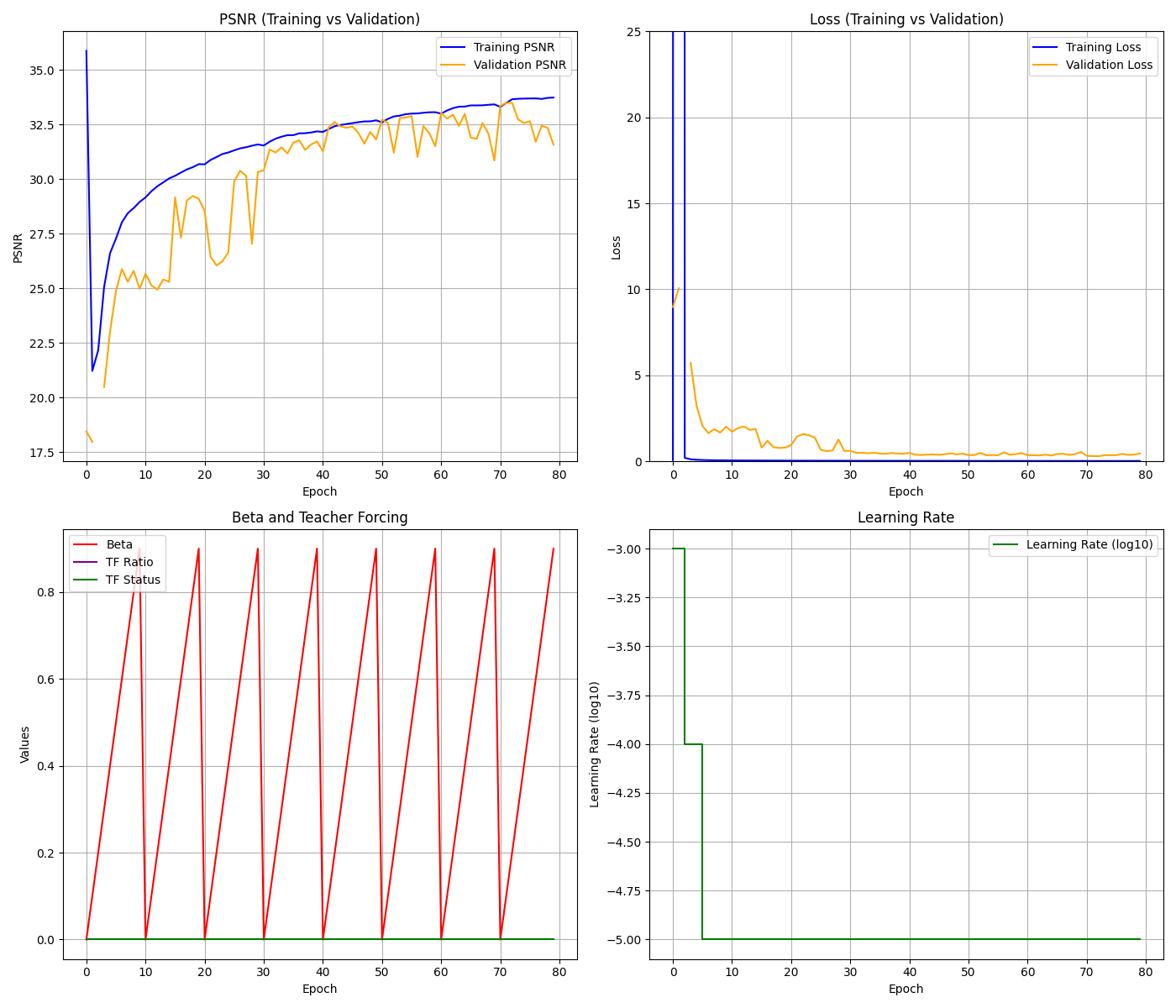


圖6.b、無Teacher Forcing訓練70個Epoch圖

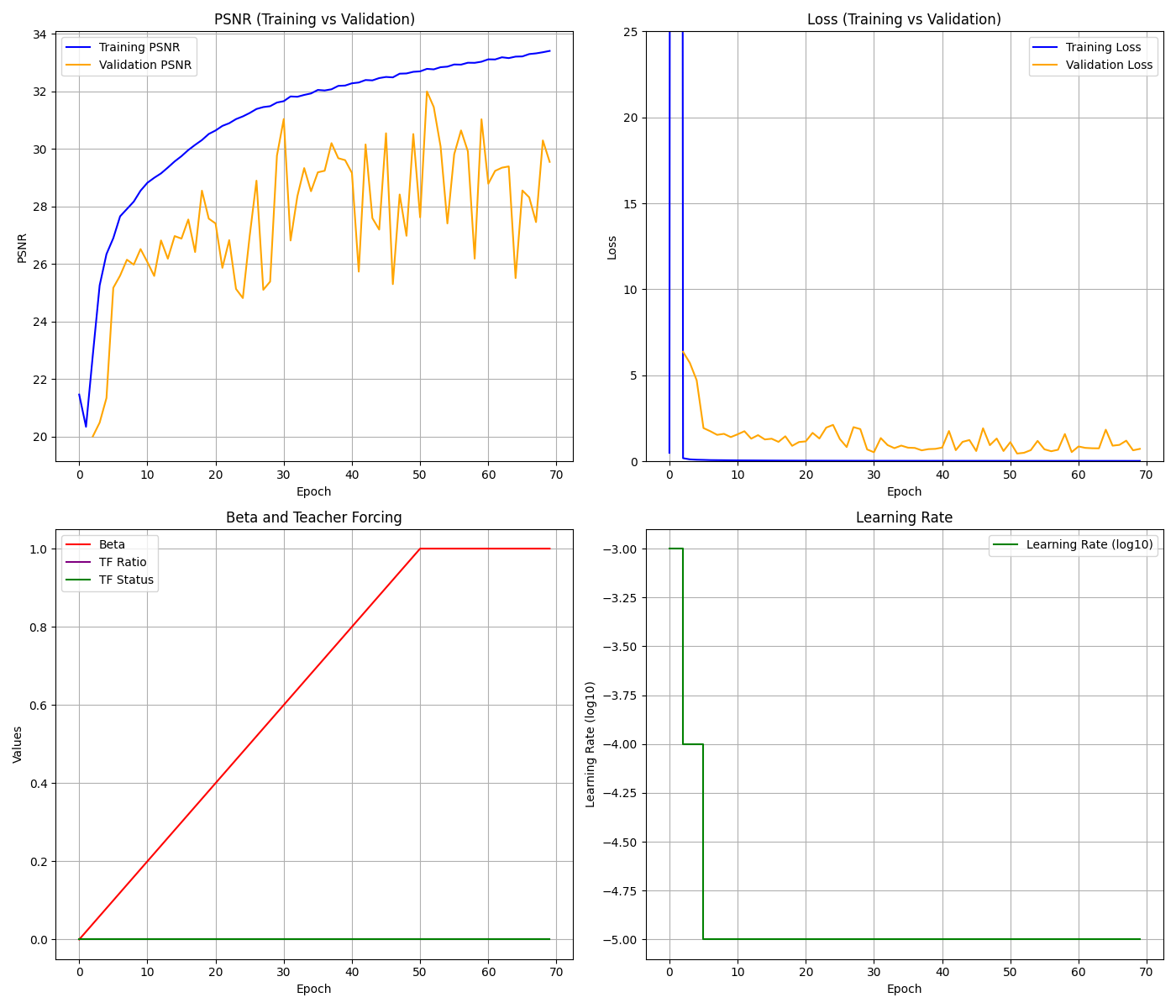
**Plot the loss curve while training with different settings.**

Cyclical (without teacher forcing)

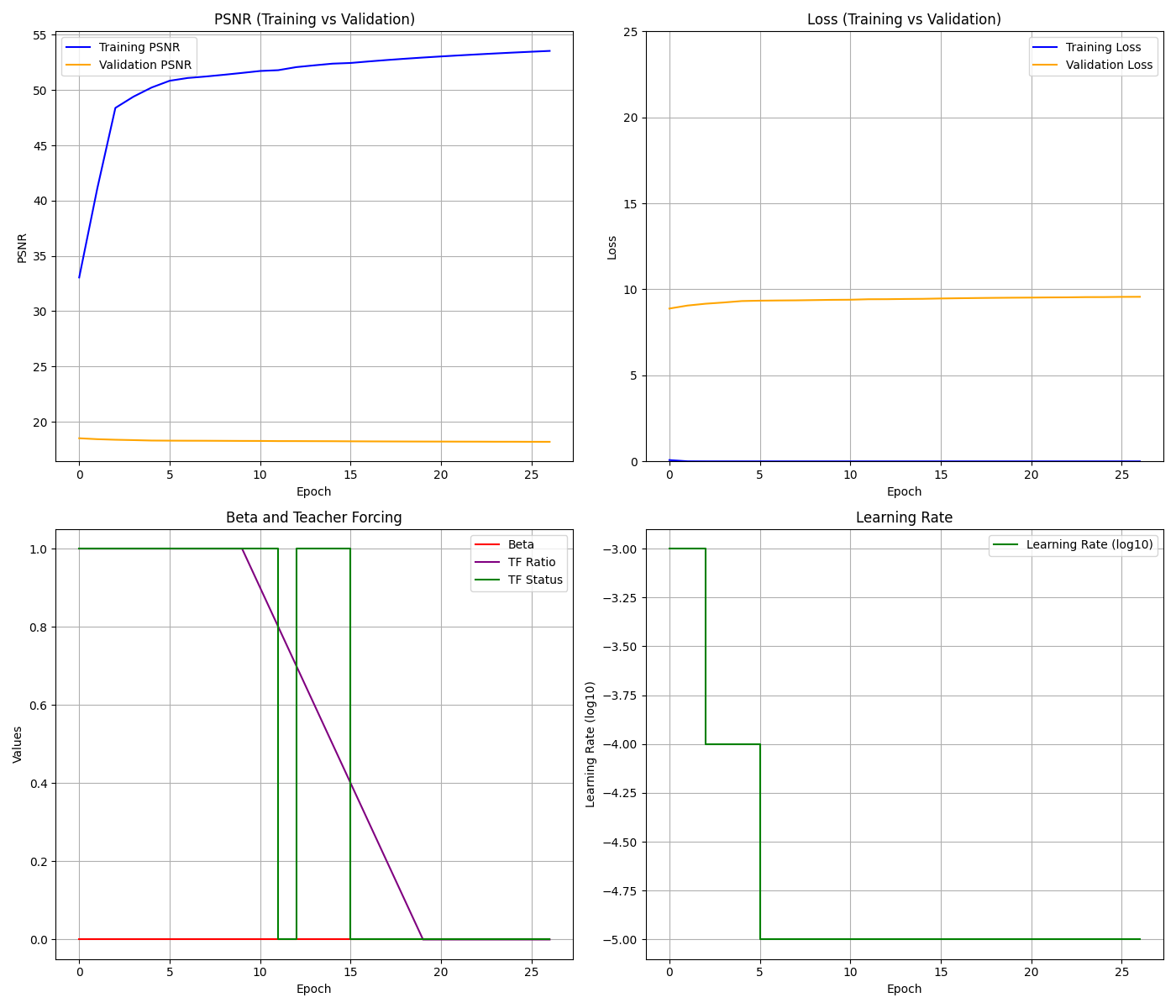
一張含有 文字, 圖表, 繪圖, 行 的圖片

自動產生的描述

Monotonic (without teacher forcing)



Without KL annealing (, with teacher forcing)



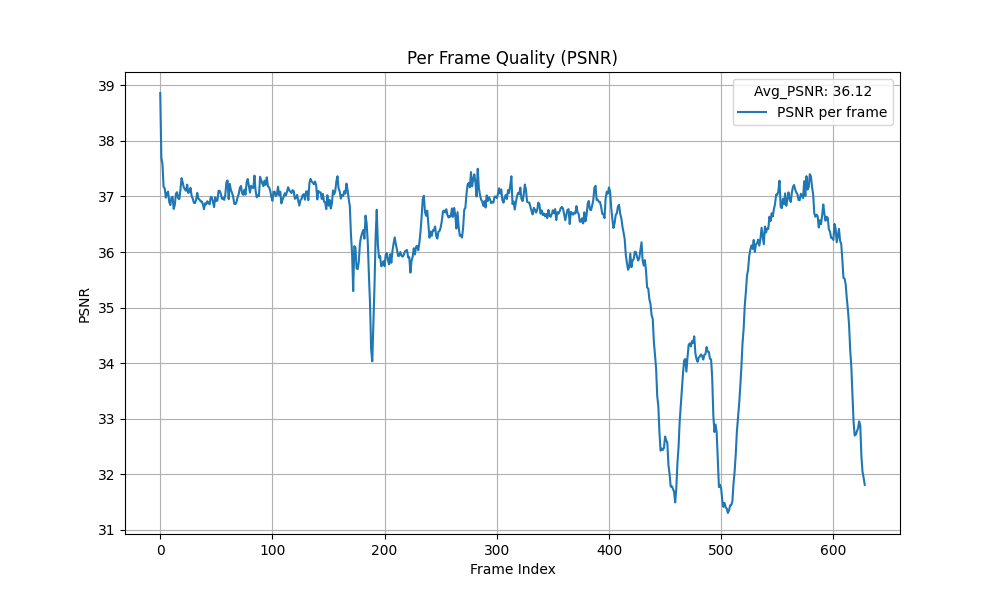
在這些訓練中發現，使用Cyclical的效果是比較好的，從圖中可以看到，為了避免跟沒有annealing的情況overfitting出現，導致在Training時PSNR很高但是Validation都無上升的情況，此外也發現了在Monotonic的情況下雖然Training和Validation的PSNR都很高，但是Monotonic的validation變動很大，不像Cyclical一樣，共同往上升。

**Plot the PSNR-per frame diagram in validation dataset**

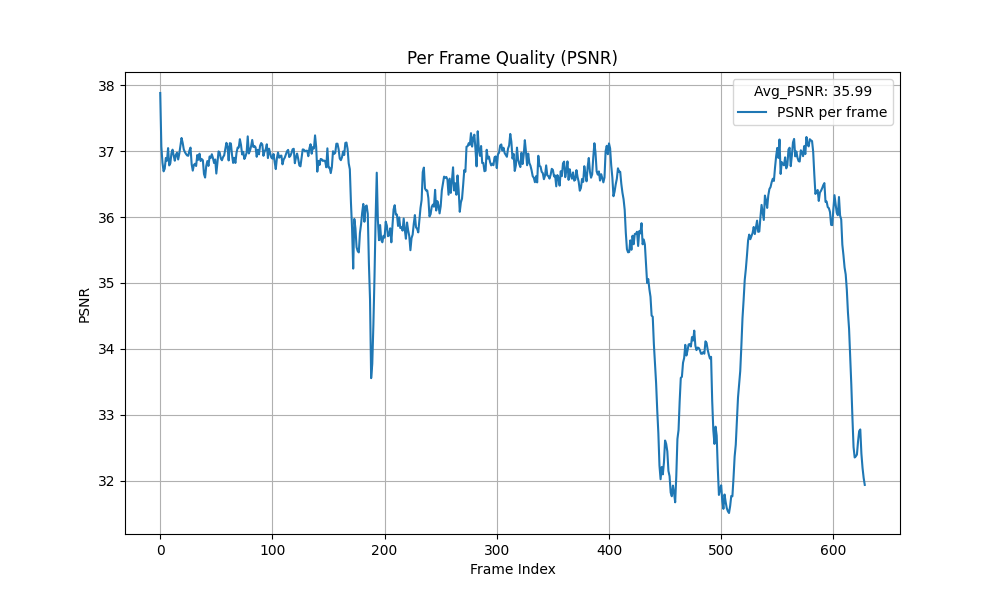
在一定數量的訓練次數和不同種類的訓練結果，發現彼此之間有很多相同的共通點，幾乎都是在特定的frame下會有明顯的下降，以下為由大到小的PSNR值和其訓練方式做比較：

|  |  |  |  |
| --- | --- | --- | --- |
| 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)

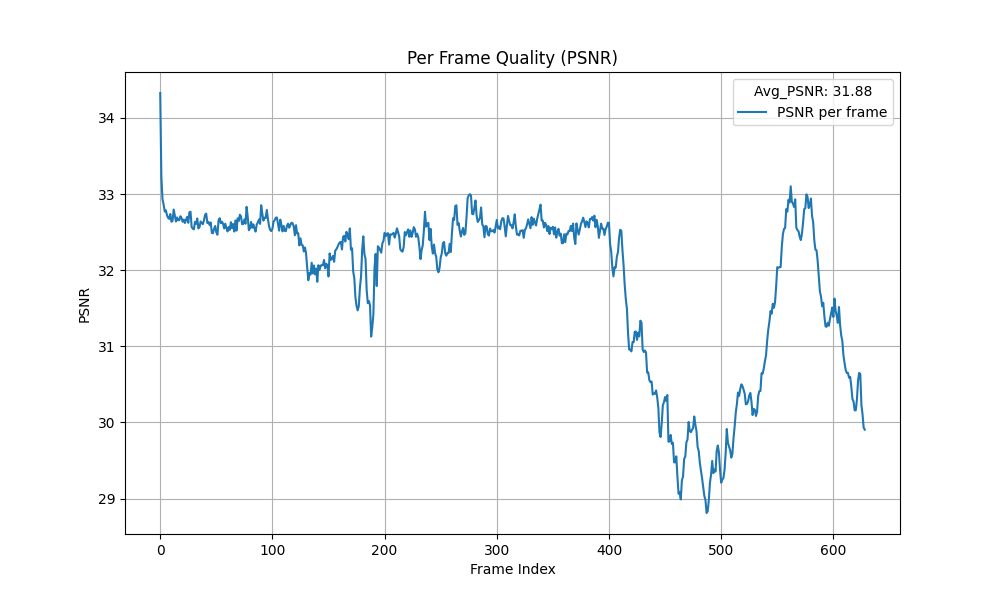


AVG PSNR 31.99 (epoch=51,monotonic, without tf)

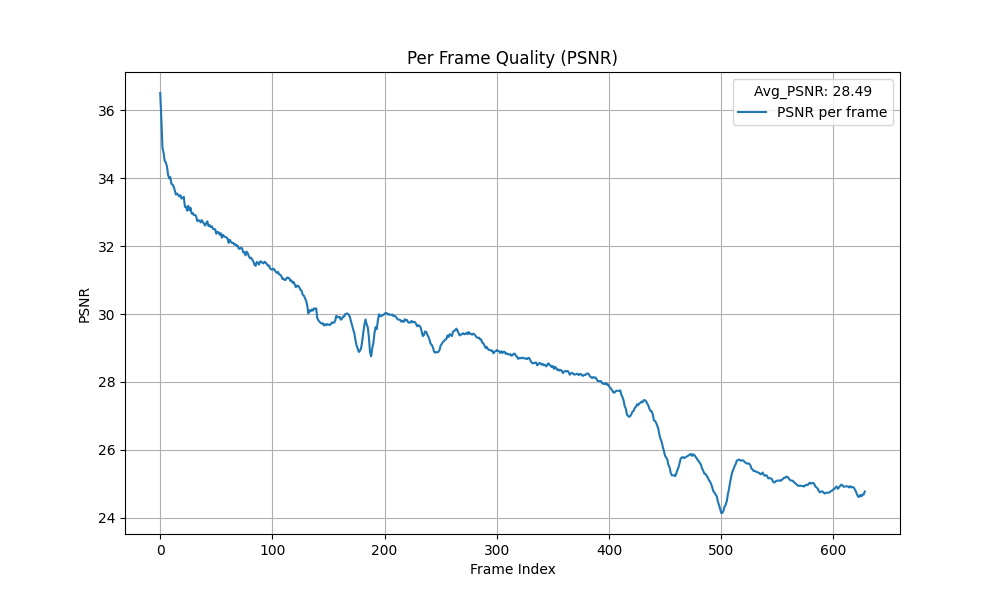
一張含有 文字, 行, 圖表, 繪圖 的圖片

自動產生的描述

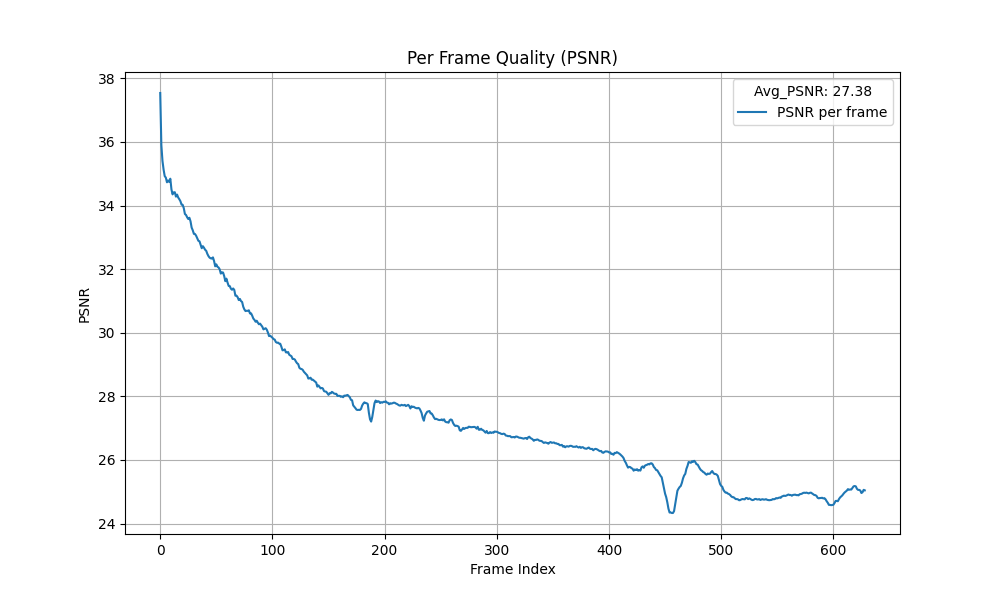
AVG PSNR 31.88 (epoch=92, cyclical, without tf)



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)。

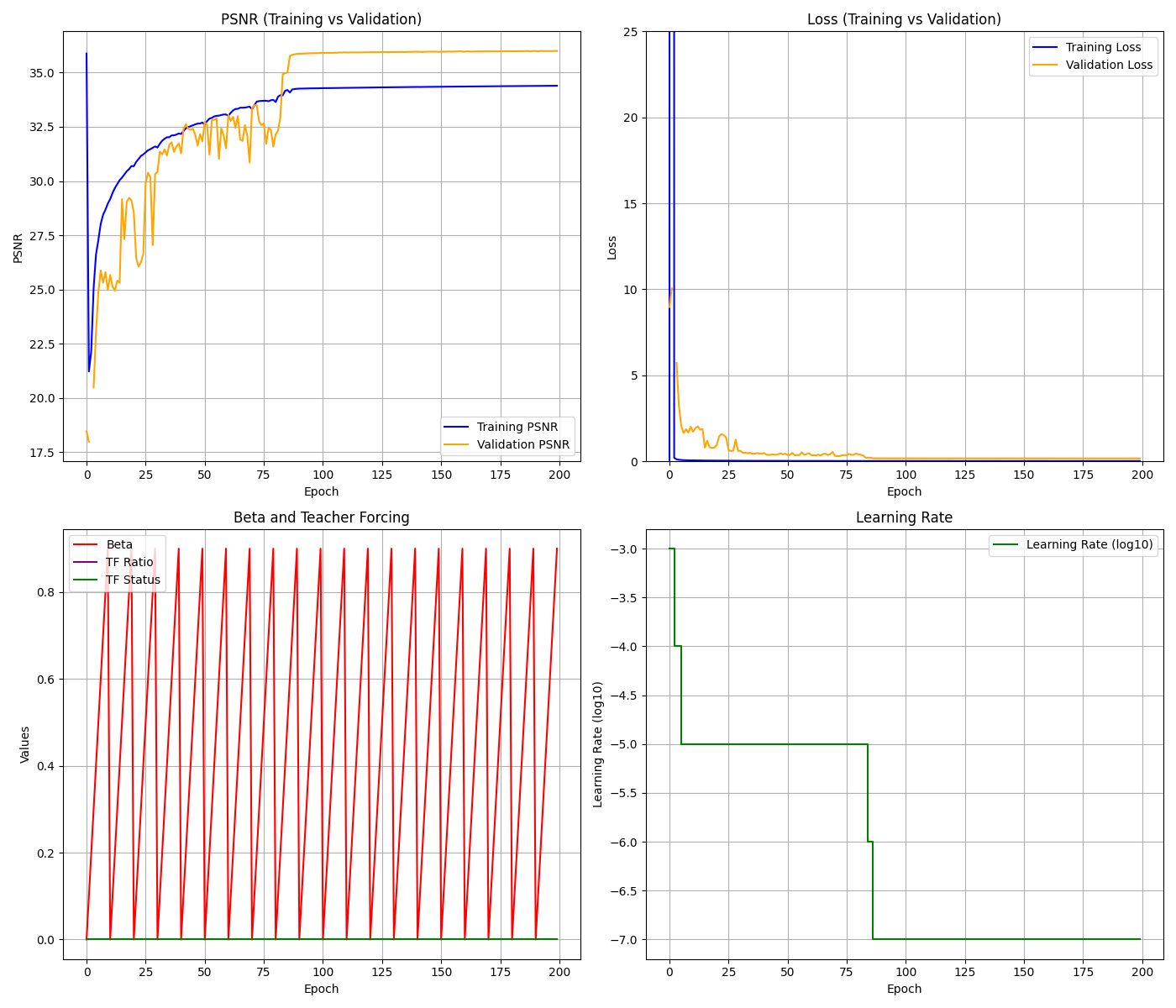


圖.7、調整learning rate達到更好效果的finetune

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.