Lab4 Report

Deep Learning

Conditional VAE for Video Prediction

姓名：陳科融

學號：314551010

July 26, 2025

1. Introduction

在本實驗中，我們實現了一個基於條件變分自動編碼器（Conditional Variational Autoencoder, CVAE）的模型，用於影片預測。模型的目標是根據前一個影像以及一系列的姿態（pose）標籤，生成後續的影片影像。

這個任務在許多領域中具有重要應用，例如影片壓縮、影片補幀以及人機互動等。我們的模型結構主要結合了變分自動編碼器（VAE）與長短期記憶網路（LSTM）的概念，透過循環神經網路（RNN）的架構生成影片序列。

2. Implementation detail

本研究採用條件變分自編碼器（Conditional Variational Autoencoder, CVAE）架構，根據過去影像與姿勢生成未來影像。整體模型由五個模組組成：

- Frame Encoder（RGB_Encoder）輸入：RGB 影像（彩色畫面）
  功能：將影像轉換為特徵向量 輸出特徵維度：F_dim = 128
- Label Encoder（Label_Encoder）輸入：姿勢圖（如骨架圖）
  功能：將姿勢資訊轉換為特徵向量 輸出特徵維度：L_dim = 32
- Gaussian Predictor：預測潛在空間的後驗分布（均值與變異數）
  用途：VAE 抽樣潛在向量（latent vector）
  潛在向量維度：N_dim = 12
- Decoder Fusion：將下列三個特徵融合起來，用於影像生成前一影像的特徵（來自 Frame Encoder）下一幀的姿勢特徵（來自 Label Encoder）抽樣潛在向量（來自 Gaussian Predictor）
- Generator：根據融合後的特徵，產生下一幀的預測影像。

2.1 How do you write your training/testing protocol

Train

在訓練過程中，模型針對每段影片序列進行逐幀預測。每個時間步中，模型以前一幀影像與當前姿態標籤作為輸入，嘗試重建當前影像。損失函數由兩部分組成：重建損失（MSE）與 KL 散度（KLD），後者透過 β 值進行加權以實現 KL Annealing。過程中可根據 Teacher Forcing 策略決定是否使用真實影像作為下一幀輸入。最後累加損失並進行參數更新。

```
# 從第二個影格開始
for time_step in range(1, timesteps):
    # 根據 use_teacher_forcing 決定解碼器的輸入是真實前一影格還是模型生成的影格
    if use_teacher_forcing:
        previous_frame_input = image_sequence[:, time_step - 1, ...]  # 使用
    else:
        previous_frame_input = predicted_frame  # 使用模型生成的影格

    current_ground_truth_frame = image_sequence[:, time_step, ...]
    current_ground_truth_label = label_sequence[:, time_step, ...]
    encoded_current_frame = self.frame_transformation(current_ground_truth_f
    encoded_current_label = self.label_transformation(current_ground_truth_l

    # 使用高斯預測器從目前影格和標籤預測潛在變數 z 的分佈
    latent_z, posterior_mu, posterior_logvar = self.Gaussian_Predictor(encod

    # 將前一影格編碼為特徵，並使用 .detach() 避免梯度回傳
    encoded_previous_frame = self.frame_transformation(previous_frame_input)

    # 解碼器融合前一影格特徵、目前標籤特徵和潛在變數 z
    decoded_features = self.Decoder_Fusion(encoded_previous_frame, encoded_c

    predicted_frame = self.Generator(decoded_features)
    predicted_frame = nn.functional.sigmoid(predicted_frame)

    # 計算重建損失 (MSE) 和 KL 散度損失
    reconstruction_loss = self.mse_criterion(predicted_frame, current_ground
    kl_divergence_loss = kl_criterion(posterior_mu, posterior_logvar, batch_

    # 結合兩種損失，並使用 beta 加權 KL 損失
    step_loss = reconstruction_loss + kl_beta_weight * kl_divergence_loss
    total_training_loss += step_loss
    total_recon_loss += reconstruction_loss
    total_kl_loss += kl_divergence_loss

# 正規化總損失
total_training_loss /= (timesteps - 1)
total_recon_loss /= (timesteps - 1)
total_kl_loss /= (timesteps - 1)

# 梯度清零、反向傳播和更新模型參數
self.optim.zero_grad()
total_training_loss.backward()
self.optimizer_step()
```

程式碼 1. Train 的程式架構

Test

對於第 t 幀影像，模型依據前一幀影像與當前姿勢標籤進行推論。首先，前一幀影像經由 frame_transformation 模組轉換為影像特徵，姿勢標籤則透過 label_transformation 模組轉換為標籤特徵。

接著，利用標準模擬標準常態分佈，並隨機抽樣潛在變數 $z = \mathcal{N}(0, 1)$。模型將影像特徵、標籤特徵與潛在變數 z 融合後，依序通過 Decoder_Fusion 與 Generator 模組生成下一幀影像。

```python
def val_one_step(self, img, label, idx=0):
    img = img.permute(1, 0, 2, 3, 4)  # change tensor into (seq, B, C, H, W)
    label = label.permute(1, 0, 2, 3, 4)  # change tensor into (seq, B, C, H, W)
    assert label.shape[0] == 630, "Testing pose seqence should be 630"
    assert img.shape[0] == 1, "Testing video seqence should be 1"

    # decoded_frame_list is used to store the predicted frame seq
    # label_list is used to store the label seq
    # Both list will be used to make gif
    decoded_frame_list = [img[0].cpu()]
    label_list = []

    # TODO
    for i in range(1, label.shape[0]):
        # 取得前一個生成的影格
        previous_frame = decoded_frame_list[-1].to(self.args.device)
        # 取得目前的標籤
        current_label = label[i, ...].to(self.args.device)

        # 將前一影格與當前標籤轉換為特徵向量
        encoded_previous_frame = self.frame_transformation(previous_frame)
        encoded_current_label = self.label_transformation(current_label)

        # 產生潛在空間的分佈（均值和對數方差）
        latent_z, mu, logvar = self.Gaussian_Predictor(encoded_previous_frame, encoded_currer
        # 從標準正態分佈中採樣噪聲，用於生成過程
        noise = torch.randn_like(latent_z)

        # 解碼器融合模組，結合前一影格特徵、標籤特徵和噪聲來生成解碼後的特徵
        fused_features = self.Decoder_Fusion(encoded_previous_frame, encoded_current_label, r

        # 生成器根據融合後的特徵生成新的影格
        generated_frame = self.Generator(fused_features)
        generated_frame = nn.functional.sigmoid(generated_frame)  # 將數值縮放到 [0, 1] 範圍
        # generated_frame = nn.functional.Tanh(generated_frame)

        # 將生成的影格和標籤儲存起來
        decoded_frame_list.append(generated_frame.cpu())
        label_list.append(current_label.cpu())
```

程式碼 2. Test 的程式架構

2.2 How do you implement reparameterization tricks

在 VAE 中，我們希望從一個高斯分佈中取樣：

$$z \sim \mathcal{N}(\mu, \sigma^2)$$

但是這個過程無法直接進行反向傳播（因為取樣是隨機的，沒辦法對隨機取樣結果微分）。**Reparameterization trick** 的關鍵是把隨機性從模型參數中分離出來，改寫成：

$$z = \mu + \sigma \cdot \epsilon, \quad where\ \epsilon \sim \mathcal{N}(0, I)$$

這樣，μ和σ是可微分的，ϵ的隨機性與它們無關，訓練就能進行。

模型設定潛在變數 z 的平均值與對數變異數為全零張量，模擬標準常態分佈。使用重參數化技巧：

$$z = \mu + exp\left(\frac{log\sigma^2}{2}\right) \cdot \epsilon$$

```python
def reparameterize(self, mu, logvar):
    # TODO

    # 重參數化技巧
    std = torch.exp(0.5 * logvar) # 計算標準差
    eps = torch.randn_like(std) # 從標準常態中採樣噪聲
    return eps.mul(std).add_(mu) # 平均 + 標準差 × 標準常態噪聲
```

程式碼 3. Reparameterize 架構

2.3 How do you set your teacher forcing strategy

為了提升序列生成模型的穩定性與泛化能力，本模型在訓練階段採用了 Teacher Forcing 機制，並配合一種稱為排程衰減（Scheduled Sampling）的方法進行控制。以下是其具體實作方式：

2.3.1 機率性（Stochastic Teacher Forcing）：

- 在每個訓練中，模型會根據目前的 Teacher Forcing，tfr 生成一個隨機數。
- 若隨機數小於 tfr，則本批次會使用 Teacher Forcing，即接收真實的影格（ground truth）作為輸入。模擬實際場景，避免模型過度依賴訓練資料。

2.3.2 排程衰減（Scheduled Decay）：

- 每個訓練 epoch 結束時會呼叫 teacher_forcing_ratio_update 函數。
- 若當前 epoch 數 ≥ 閾值 tfr_sde，則 tfr 值將減去一固定步長 tfr_d_step。

2.3.3 Teacher Forcing 策略結合了穩定學習與錯誤自我修正的優點：

- 在訓練初期：模型依賴真實影格，可快速學習資料分佈。
- 在訓練後期：逐漸轉向自我生成輸入，有助於學習如何處理自己的預測誤差，進而提升長期預測的穩健性。

```python
def teacher_forcing_ratio_update(self):
    # 檢查目前的 epoch 是否已達到開始衰減 Teacher Forcing 比例的 epoch
    if self.current_epoch >= self.args.tfr_sde:
        # 將 Teacher Forcing 比例減去指定的衰減步長
        self.tfr -= self.args.tfr_d_step
        self.tfr = max(self.tfr, 0) # 確保比例不會小於 0
```

程式碼 4. Teacher forcing 調整數值

2.4How do you set your kl annealing ratio

在變分自編碼器（VAE）的訓練過程中，我們的目標是最大化證據下界（ELBO），這等價於同時最小化 KL 散度項和最大化重建項。然而，在實際訓練中，常會遇到所謂的 KL-vanishing 問題。

KL-vanishing 指的是 KL 散度項趨近於零的現象，這代表的後驗分布 q(z|x) 幾乎退化成為先驗分布 p(z)，也就是說潛在變數 z 與輸入 x 幾乎沒有關聯。這種情況下，模型幾乎不利用潛在空間來生成數據，導致學習的表達能力不足。

```python
class kl_annealing:
    def __init__(self, args, current_epoch=0):
        self.args = args
        self.current_epoch = current_epoch

        # 確保參數正確
        assert args.kl_anneal_type in ["Cyclical", "Monotonic", "None"]

        # 根據不同的退火類型，生成對應的 beta 值序列
        if args.kl_anneal_type == "Cyclical":
            # 週期性退火：beta 值在多個週期內從 start 線性增加到 stop
            self.betas = self.frange_cycle_linear(args.num_epoch, n_cycle=args.k
        elif args.kl_anneal_type == "Monotonic":
            # 單調退火：beta 值在單個週期內（通常是整個訓練過程）從 start 線性增加到 stop
            self.betas = self.frange_cycle_linear(args.num_epoch, n_cycle=1, rat
        elif args.kl_anneal_type == "None":
            # 不使用退火：beta 值在所有 epoch 中都保持為 1
            self.betas = np.ones(args.num_epoch)
        else:
            raise NotImplementedError

    def update(self):
        self.current_epoch += 1 # 更新目前的 epoch 計數。

    def get_beta(self):
        return self.betas[self.current_epoch] # 獲取目前 epoch 對應的 beta

    def frange_cycle_linear(self, n_iter, start=0.1, stop=1.0, n_cycle=1, ratio=
        # 生成一個週期性線性變化的 beta 值序列。

        betas = np.ones(n_iter) * stop # 初始化所有 beta 值為 stop
        period = n_iter // n_cycle # 計算每個週期的長度

        # 為每個週期生成線性增加的 beta 值
        for i in range(n_cycle):
            start_ = period * i
            end_ = start_ + int(period * ratio)
            betas[start_:end_] = np.linspace(start, stop, int(period * ratio))
        return betas
```

程式碼 5. KL-annealing 程式架構

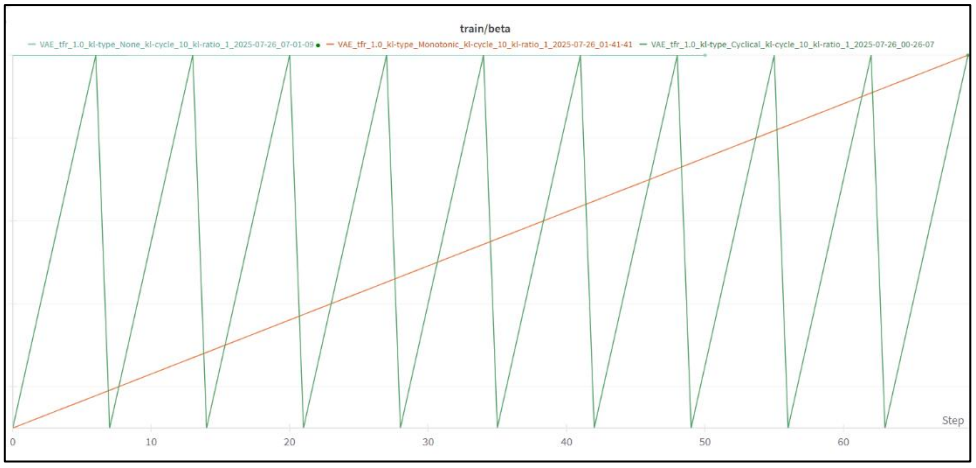


圖 1. KL beta 變化

3. Analysis & Discussion

3.1 Plot Teacher forcing ratio

在訓練過程中，模型在預測下一個時間步時，會使用真實的輸入（Ground Truth）而不是它先前的預測結果，這種方式稱為（Teacher Forcing）。

然而，若模型長期依賴 Ground Truth，會導致其在推論階段無法有效處理預測誤差的累積，因此通常會逐步減少 teacher forcing 的比例，這個過程稱為 Teacher Forcing Ratio（TFR）衰減（Decay）。

在預設 tfr_sde 設定為 10，代表從第 10 個 epoch 開始進行 TFR 衰減；之後每經過 1 個 epoch（即每次 tfr 更新），TFR 會以 tfr_d_step = 0.1 的速率進行遞減，直到最小為 0 為止。

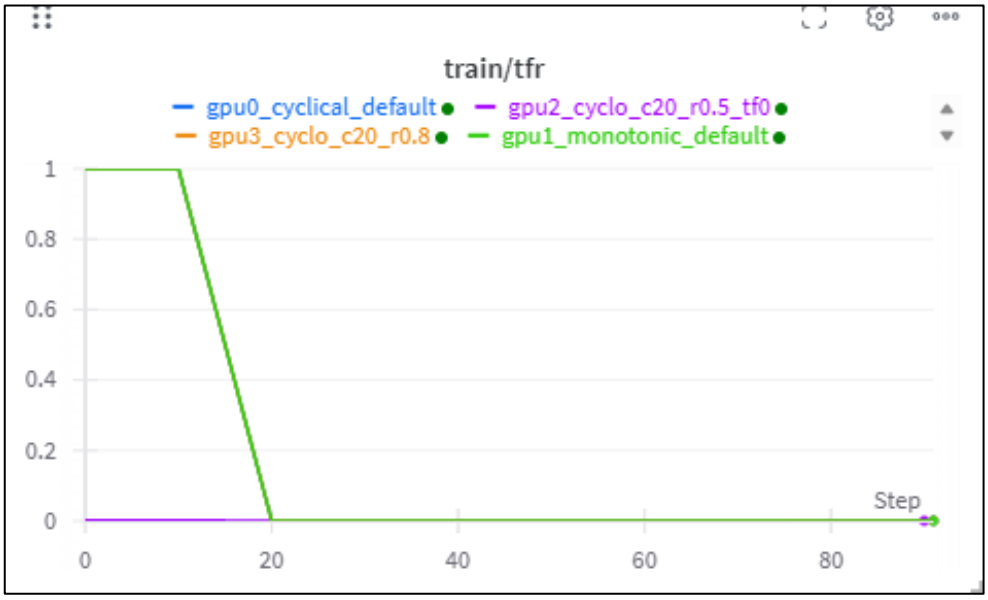


圖 2. Teacher forcing 變化圖表

3.2 Plot the loss curve while training with different setting.

本實驗比較了三種 KL Annealing 方法：**週期性退火**、**單調性退火**與**無退火**。以下為各策略的說明與實驗觀察：

1. 週期性退火（Cyclical Annealing）β 在訓練反覆從 0 增至 1

- **優點**：可幫助模型跳出局部最小值，探索更完整的潛在空間。
- **缺點**：訓練過程中產生明顯損失尖峰，穩定性較差。
- **觀察**：Loss 曲線呈現週期性起伏，每個週期初期因重建主導，損失下降，週期後段因 KL 增強，損失短暫上升。

2. 單調性退火（Monotonic Annealing）β 由 0 緩慢增至 1

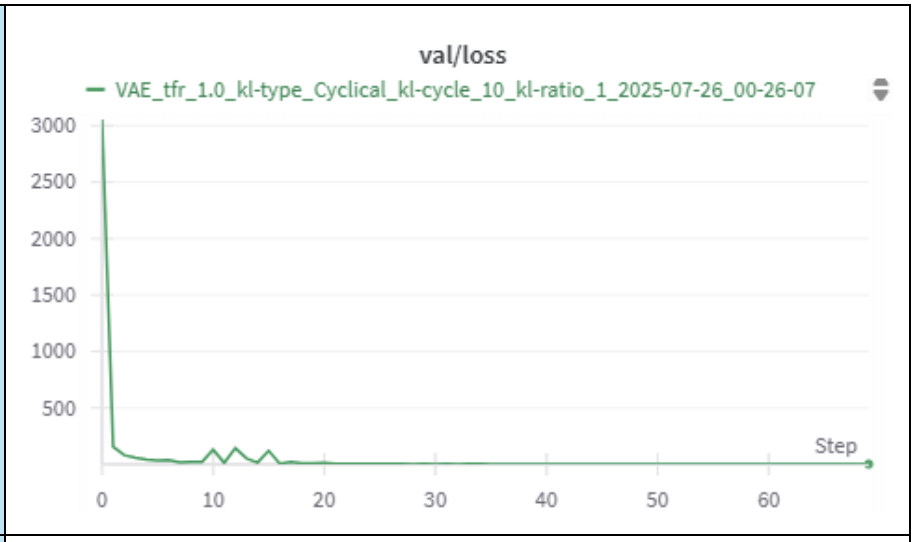
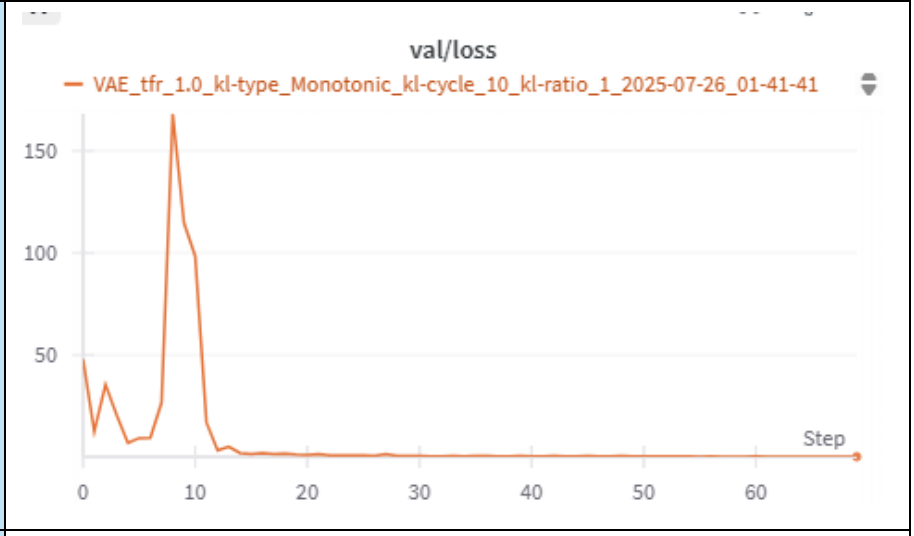
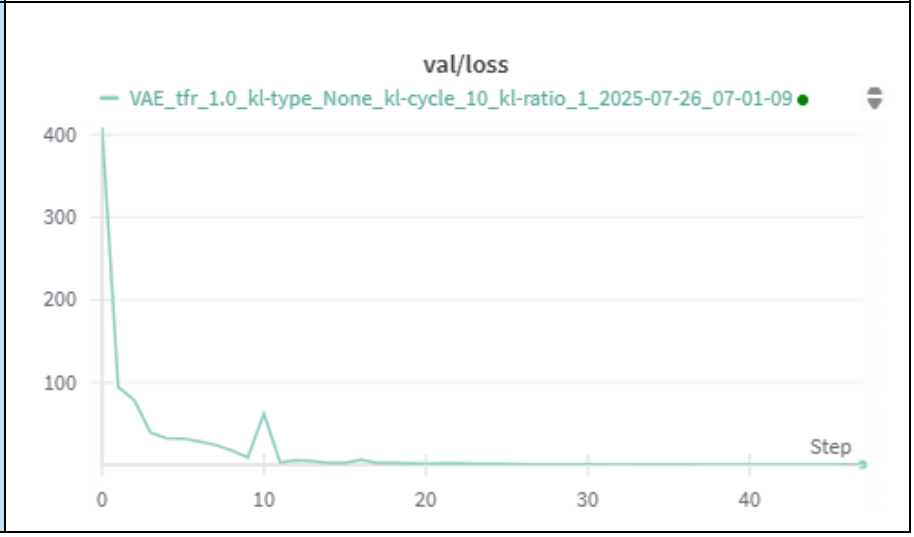
- **優點**：訓練穩定避免 posterior collapse，最終重建與潛在空間表現均衡。
- **缺點**：需設計合理的增長曲線，避免過快或過慢。
- **觀察**：Loss 曲線平滑下降，收斂效果最佳，損失值最低。

3. 無退火（Without Annealing）β 固定為 1，整個訓練期間不調整

- **優點**：實作簡單，訓練流程固定。
- **缺點**：初期學習困難，重建與正規化目標競爭，容易陷入次優解。
- **觀察**：Loss 下降後波動較多。

從 Lab 結果可知，單調性退火策略在穩定性與最終表現上最為優秀。週期性退火雖具探索能力，但訓練不穩；無退火策略在初期受限較多，難以快速收斂。綜合而言，適當設計 KL 權重變化，對提升 VAE 訓練品質具有關鍵影響。

Wandb 圖表數據

Cyclical	
Monotonic	
Without KL annealing	

3.3 Plot the PSNR-per-frame diagram in the validation dataset and analyze it

為比較不同 KL Annealing 策略對模型生成圖像品質的影響，本 Lab 使用 wandb 記錄了訓練過程中驗證集的 PSNR（峰值訊噪比）變化。

1. 單調性退火（Monotonic Annealing） – 效果最佳

- 模型初期可專注於重建任務，隨後逐步引入 KL 約束，讓模型有時間學習潛在空間結構，最終達成平衡。

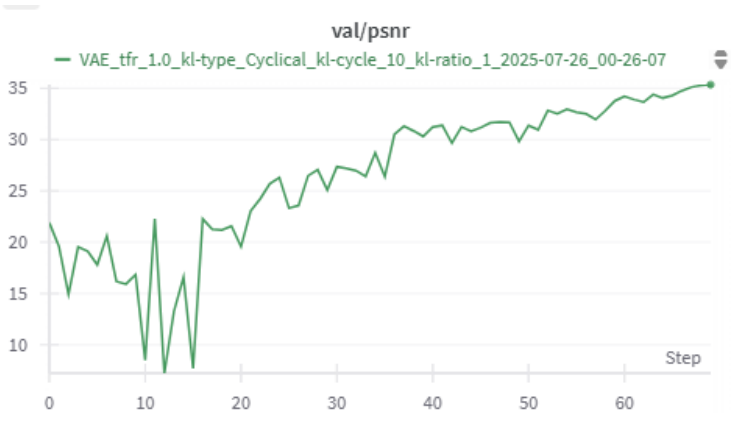
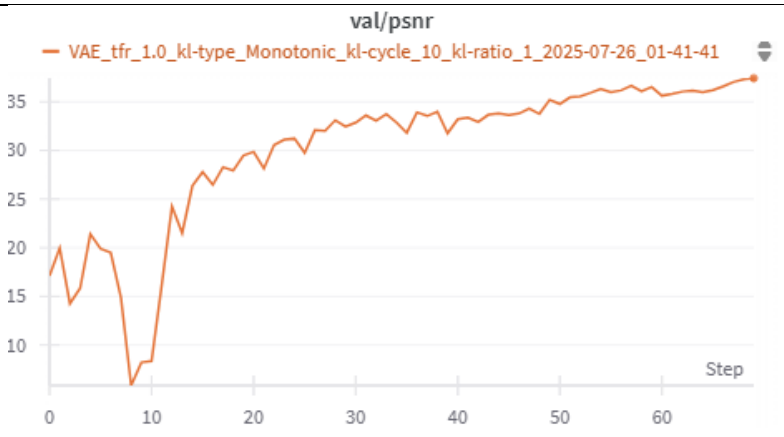
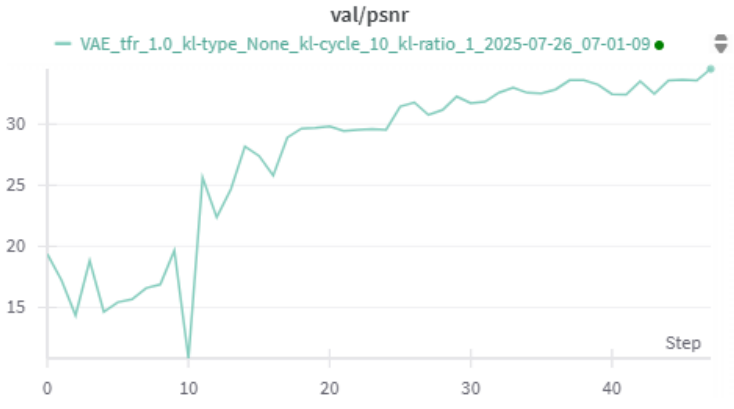
2. 無退火（Without Annealing） – 次佳表現

- PSNR 成長穩定但略低於單調退火。
- 由於 KL 約束自始至終存在，模型初期難以專注學習重建細節，導致最終效果略受限制。

3. 週期性退火（Cyclical Annealing） – 效果最差

- PSNR 呈現週期性劇烈波動，每當 KL 快速增大時，重建質量會顯著下降。
- 雖有助於潛在空間探索，但嚴重破壞重建能力，造成 PSNR 表現不穩定。

圖 3. PSNR 比較圖表

Cyclical	
Monotonic	
Without KL annealing	

3.4 Other training strategy analysis (Bonus 10%)

Evidence Lower Bound (ELBO)

令 x 為觀察變數，z 為潛在變數。目標是最大化資料的對數近似函數$log\ p(x)$。

其可被拆解並以變分下界（ELBO）估計：

$$\log p\,(x) = \log \int p(x,z)dz \geq E_{z\sim q(Z|X)}[\log p\,(x,z) - \log q\,(z|x)]$$

$q_x(z|x)$: 編碼器，近似真實後驗分布 $p(z|x)$，$p_x(x|z)$: 解碼器，生成模型

ELBO 形式為：

$$\text{ELBO} = E_{z\sim q(Z|X)}[\log p\,(x,z) - \log q\,(z|x)]$$

$$E_{z\sim q(Z|X)}[\log p\,(x|z)] - D_{\text{KL}}\big(q(z|x)|p(z)\big)$$

VAE 的目標是最大化 ELBO，即：

- **最大化重建機率 $E[\log p(x|z)]$**
- **最小化 KL 散度 $D\_KL(q(z|x)\,||\,p(z))$**

The VAE Objective with Gaussian Assumptions

通常我們假設先驗與後驗皆為高斯分布：

$$q(z|x) = N(\mu(x), \Sigma(x))$$

$$p(z) = N(0, I)$$

KL Divergence Term (Encoder Objective)

兩個多變量高斯分布的 KL 散度公式為：

$$D_{\text{KL}}(\mathcal{N}_0|\mathcal{N}_1) = \frac{1}{2}\left[tr(\Sigma_1^{-1}\Sigma_0) + (\mu_1-\mu_0)^{\top}\Sigma_1^{-1}(\mu_1-\mu_0) - K + log\frac{de\,t(\Sigma_1)}{de\,t(\Sigma_0)}\right]$$

針對 VAE 的情況，其中 $N_1 = N(0,I)$，化簡得：

$D_{KL}(q(z|x)\,||\,p(z)) = 1/2\,[\,tr(\Sigma(x)) + \mu(x)\text{^}T\,\mu(x) - K - log(det\,\Sigma(x))\,]$，這是 **Encoder** 的一部分損失函數。

Reconstruction Term (Decoder Objective)

假設解碼器輸出分布為 $N(x;\,\mu(z),\sigma^2\,I)$，其對數似然為：

$$log\,p(x|z) = -1/(2\sigma^2)\,||x - \mu(z)||^2 - D/2 * log(2\pi\sigma^2)$$

因此最大化 $log\,p(x|z)$ 等同於最小化重建誤差 $||x - \mu(z)||^2$，此為 **Decoder** 的損失函數。

完整重建項為期望： $E_{z\sim q(Z|X)}[logp(x|z)]$，由於不能對採樣的 z 反向傳播，因此使用重參數化技巧： $z = \mu + \sigma * \varepsilon$

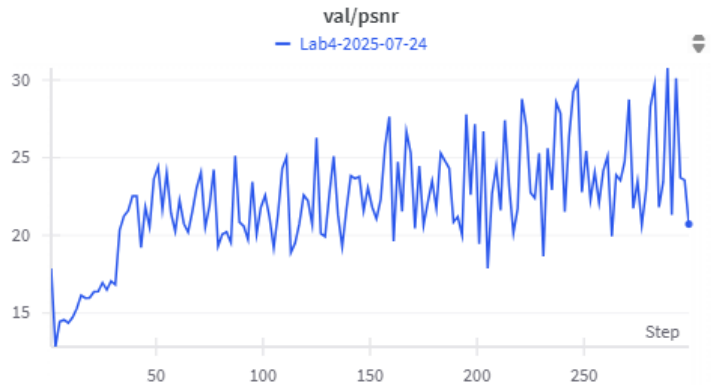
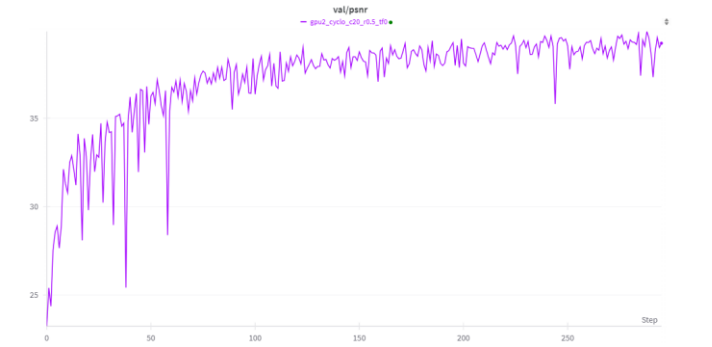
program instructions

train	python Trainer.py --DR ../LAB4_Dataset --save_root checkpoints --num_epoch 500 --per_save 10 --num_workers 6 --batch_size 8 --wandb
train (best)	python Trainer.py --DR ../LAB4_Dataset --save_root checkpoints --num_epoch 500 --num_workers 8 --batch_size 16 --kl_anneal_cycle 20 --kl_anneal_ratio 0.5 --tfr 0.0 --wandb
test	python Tester.py --DR ../LAB4_Dataset --save_root ./results --ckpt_path checkpoints/(ckpt 名稱)

Parameter(best)

參數	設定	說明
epochs	340	訓練場次
batch_size	16	每次訓練所使用的樣本數量，有助平衡記憶體使用與收斂速度。
num_workers	8	用於資料載入的背景執行緒數，有助於加速 I/O 效率。
learning_rate	0.0001	更新學習速率，數值越小，學習越穩定。
optimizer	AdamW	使用帶權重衰減 Adam，改善 L2 正則化效果。
scheduler	Cosine Annealing	幫助模型以較平滑的方式收斂。
loss function (Reconstruction)	Mean Squared Error (MSE)	評估輸出與真實值的差異。
loss function (Regularization)	KL Divergence KL	散度控制潛在空間的分布。
output activation	Sigmoid	將結果壓縮至 0 到 1 的區間。
teacher forcing	off	預設是初始使用 teacher forcing，從第 10 輪開始，每輪遞減 0.1，有助於穩定訓練初期的學習效果。
KL annealing (cyclical)	Cycles: 20 Anneal ratio: 0.5	KL 損失週期性遞增策略共設定 20 個循環，每個循環內部逐步增加 KL 權重，以防止模型一開始忽略重建任務。

Improvement

舊的模型	
新的模型	

為提升模型性能，針對訓練流程、模型架構與學習策略進行優化，最終使 PSNR 分數從原始分數的二十分左右，顯著提升至三十幾。以下為三項主要優化內容與其關鍵效益：

1. 訓練迴圈優化：批次向量化運算取代樣本逐一處理

原始作法對每個樣本個別計算 loss 並更新參數，無法發揮 GPU 並行優勢，且導致梯度不穩定。優化後改以整個 batch 為單位計算與反向傳播，大幅提升運算效率與梯度穩定性，顯著加快收斂速度。

2. 模型輸出層調整：加入 Sigmoid 激活函數

原模型輸出未加限制，導致預測值與真實影像在數值範圍上不一致。透過加入 sigmoid 函數將輸出壓縮至 [0,1]，與目標圖像一致，讓 MSE loss 更有效聚焦於影像內容學習，並穩定梯度。

3. 學習率策略更換：由 MultiStepLR 改為 CosineAnnealingLR

原先使用階梯式學習率調整，導致學習率驟降、收斂不穩。新策略以餘弦函數平滑下降學習率，讓模型能更細緻地探索最佳解，有效提升生成品質與泛化能力。

Experiment

實驗名稱	KL 週期	KL 比率	Teacher Forcing	說明
gpu0_cyclical_default	無	無	啟用	預設 cyclical 模式，無調整 KL 週期與比率，作為基準組。
gpu1_monotonic_default	無	無	啟用	改為單調遞增的 KL 調整方式，觀察其穩定訓練影響。
gpu2_cyclo_c20_r0.5_tf0	20	0.5	關閉	Cyclical，設定明確 KL 週期與比率，並關閉教師強制，模擬最少指導的學習情況。
gpu3_cyclo_c20_r0.8	20	0.8	啟用	Cyclical ，KL 強度提高至 0.8，觀察更強正則化對潛在空間學習影響。

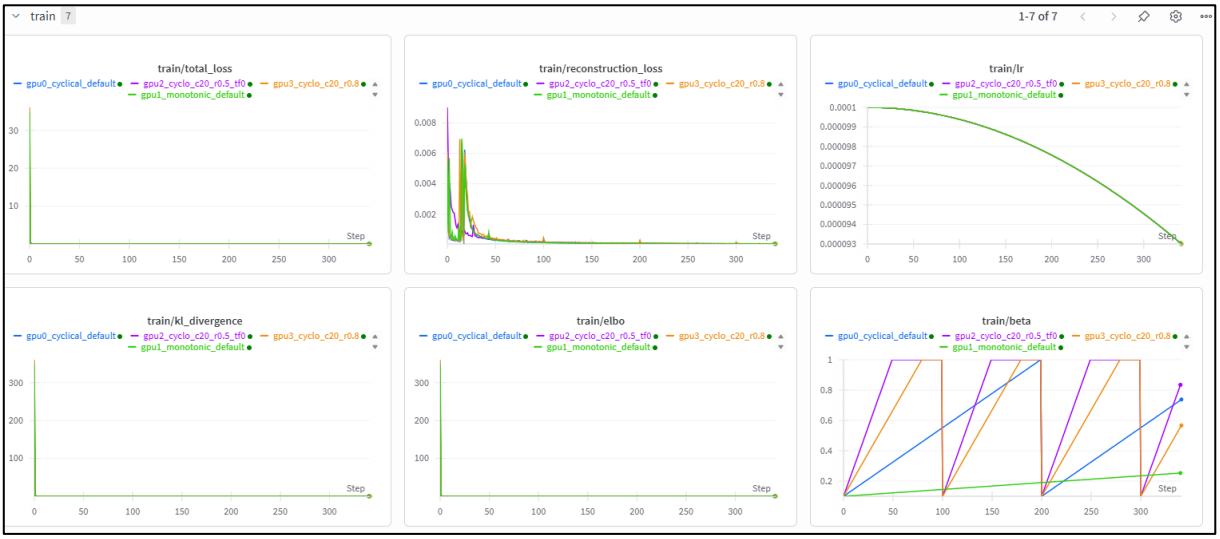


圖 4. 訓練數據圖表

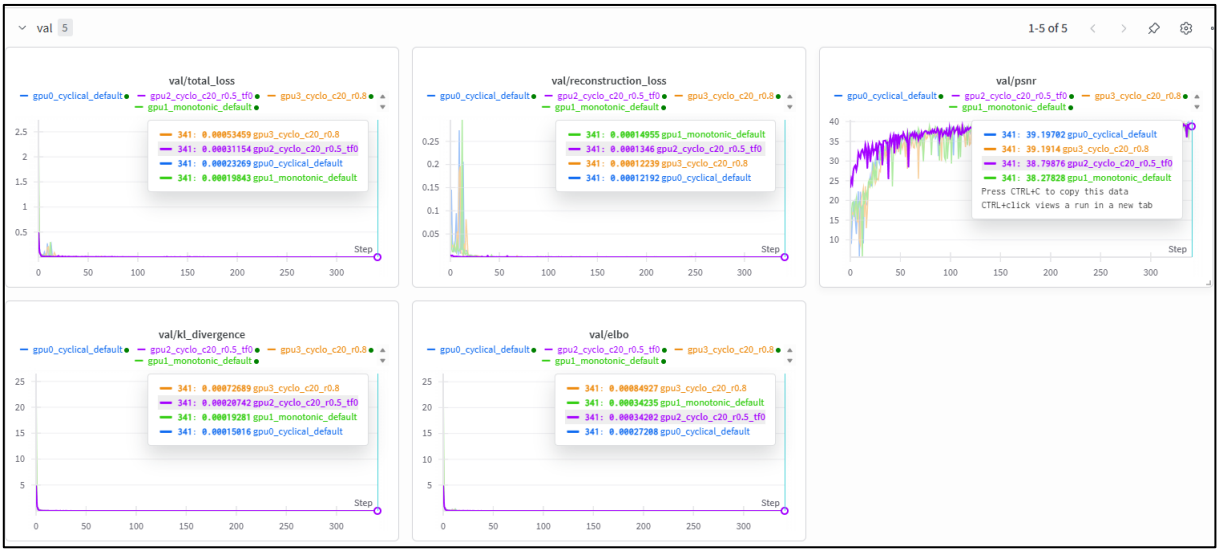


圖 5. 驗證數據圖表

4. 參考資料

[1] J. Yin, “NYCU Deep Learning Course Code,” GitHub Repository, [Online]. Available: https://github.com/jayin92/NYCU-deep-learning
[2] alu98753, “NYCU Deep Learning 2025,” GitHub Repository, [Online]. Available: https://github.com/alu98753/NYCU-Deep-Learning-2025
[3] Part-time-Ray, “DLP: Deep Learning Practice,” GitHub Repository, [Online]. Available: https://github.com/Part-time-Ray/DLP
[4] c1uc, “2025 Spring Deep Learning Labs,” GitHub Repository, [Online]. Available: https://github.com/c1uc/2025_Spring_Deep-Learning-Labs
[5] 哔哩哔哩視頻, “Transformer Self-Attention 與 Multi-head Attention 機制講解,” Bilibili, [Online Video]. Available:https://www.bilibili.com/video/BV1xx411c7A3
[6] yangweipeng708, “重参数化（Reparameterization）的原理,” CSDN Blog, Apr. 2022. [Online]. Available:https://blog.csdn.net/yangweipeng708/article
[7] E. Denton and R. Fergus, “Stochastic Video Generation with a Learned Prior,” International Conference on Machine Learning (ICML), 2018.
[8] C. Chan, S. Ginosar, T. Zhou, and A. A. Efros, “Everybody Dance Now,” IEEE International Conference on Computer Vision (ICCV), 2019.
[9] H. Fu, C. Li, X. Liu, J. Gao, A. Celikyilmaz, and L. Carin, “Cyclical Annealing Schedule: A Simple Approach to Mitigating KL Vanishing,” Conference on Empirical Methods in Natural Language Processing (EMNLP), 2019.
[10] qq_15821487, “解讀 Cycle Annealing Schedule：一種簡單但有效避免 KL 消失的方法,” CSDN Blog, [Online]. Available: https://blog.csdn.net/qq_15821487/article/details/119757207
[11] lawrencetech, “論文探討：Everybody Dance Now,” Medium Blog, [Online]. Available:https://lawrencetech.medium.com/%E8%AB%96%E6%96%87%E6%8E%A2%E8%A8%8E-everybody-dance-now-385ca9bb61b

使用的 AI 工具：

✧ **ChatGPT-4o**（由 OpenAI 提供）

用於問答、報告草稿撰寫、英文轉中文及論文說明。

✧ **Google AI Studio**

作為補充問答工具，用於模型概念、架構或函式撰寫思路輔助。

✧ **VSCode + Google Gemini 2.5 Pro**

協助撰寫與補全 Python 程式碼，並支援簡易 debug。

✧ **VSCode GitHub Copilot（GPT-4o）**

用於實作過程中進行即時程式補全、語法建議與除錯輔助。