Lab4 Report

Lucie Sénéclauze Bertolo

April 14, 2025

1 Introduction

In this lab, I explored a model for video frame prediction using a VAE-based approach. The goal was to generate videos of a person moving starting from a initial picture of them and a succession of poses. I trained and tested the model using different strategies like teacher forcing, KL annealing, and varying the noise dimension to see how they impact the performance. The main metric I used to evaluate the model was PSNR. This report goes over the implementation details, the effect of different training settings, and how I chose the final hyperparameters.

2 Implementation details

2.1 Training/Testing protocol

2.1.1 Training protocol

To implement the training protocol, I used the method presented in the lab description and summarized in the following diagram :

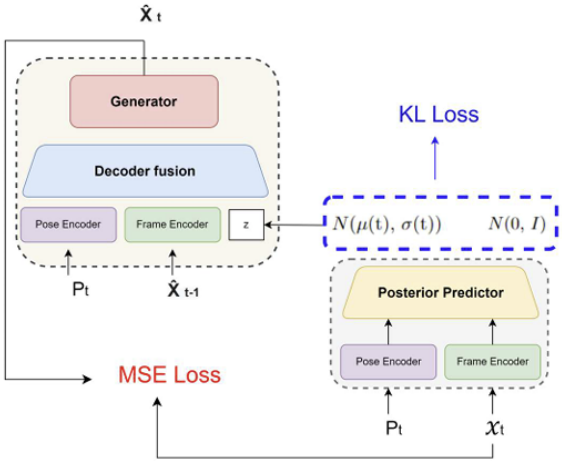

Figure 1: Training overview

Source: Lab4 description

To implement that, for each epoch I run each batch of the dataloader inside a function called training_one_step. This function will then predict the video by passing each frame through the forward pass to get the following frame. Depending on wether or not this epoch is using teacher forcing, the previous frame that is fed to the forward pass will either be the previously generated frame (no teacher forcing) or the ground truth frame (teacher forcing).
As for the forward pass, I start by encoding the pose with the RGB encoder that was provided with the code then I encode the label with the label encoder that was also provided. I then feed them to the gaussian predictor to get the tensor z as well as the estimated variance and mean of the distribution.

The next step is to concatenate the encoded pose, the encoded previous frame and z through the decoder fusion. Finally I pass the tensor through the generator to obtain the predicted frame.
The next step is to compute the loss. To do that, I combine the reconstruction (MSE loss between the predicted video and the ground truth video) and the KL loss. The weight that the KL loss will have is dependent on the importance given to kl annealing on this particular epoch. Finally, I use the loss the update the gradients through the backpropagation using the optimiser Adam.
At the end of each epoch, I update the hyperparameters (the teacher forcing ratio and the kl annealing) and save the weights.

2.1.2 Testing protocol

For the testing, I used the following method :

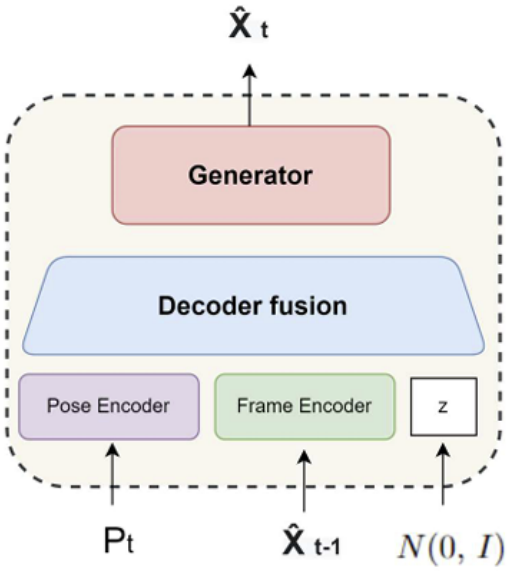

Figure 2: Enter Caption

Source: Lab4 description

As it is the testing step, there is no more teacher forcing so as we can see we simply feed the decoder fusion the encoded pose and frame as well as a tensor z that follows a normal law with mean 0 and variance I. I then generate the next frame that I will feed again to the frame encoder until I have generated all the frames. To evaluate the performance of the model, I use the Peak Signal to Noise ratio (PSNR).

2.2 Reparametrization trick

In the reparametrisation trick, instead of sampling directly z $\sim \mathcal{N}(\mu, \sigma^2)$, we sample $\epsilon \sim \mathcal{N}(0, 1)$ and then define $z = \mu + \sigma \cdot \epsilon$. That way z becomes a differentiable function of μ and σ which is useful for the backpropagation.
In my code, the reparameterisation trick is done inside the gaussian predictor :

```
def reparameterize(self, mu, logvar):
    std = torch.exp(0.5 * logvar) #standart deviation computed from the log of the variance (logvar)
    eps = torch.randn_like(mu) #generate epsilon following a N(0,1) and having the same shape as mu
    z = mu + std * eps #reparametrisation
    return z
```

Figure 3: Reparametrisation trick

2.3 Teacher forcing strategy

Teacher forcing is the concept of feeding the model the ground truth instead of the previously predicted image. This is useful at the beginning of the training when the model is still to "weak" to produce coherent images so the errors would propagate through the whole video and the final output would be really bad, which would make the model learn slower at the beginning. However as the training goes on, we diminish the teacher forcing so that the model can learn to generate images on its own as there will be no ground truth during the testing phase.
The teacher forcing ratio starts at one and is then decreased by the teacher forcing step at each epoch starting from a given epoch :

```
def teacher_forcing_ratio_update(self):
    if self.current_epoch >= self.args.tfr_sde:
        self.tfr = max(0.0, self.tfr - self.tfr_d_step) #Not going below zero
```

Figure 4: Teacher forcing ratio update

3 Analysis and discussion

3.1 Teacher forcing ratio

In figure 5, I show the evolution of the teacher forcing ratio next to the loss curve. In this example I did not use KL annealing as to only see the impact of the teacher forcing ratio.

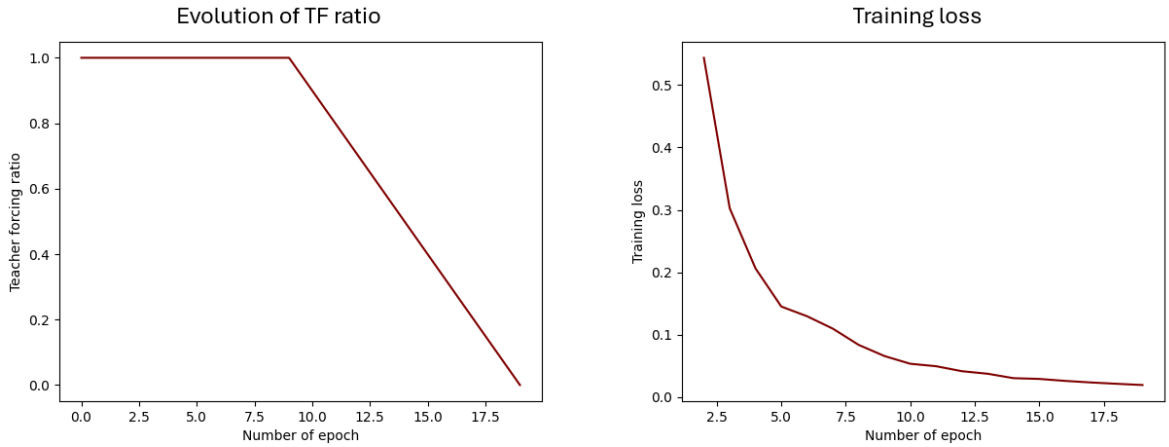

Figure 5: Impact of teacher forcing on the loss

As we can see, the impact of the teacher forcing on the loss curve is not very visible. Hence from now on I will comment the model's performance based on the evolution of the PSNR score on the validation data.

As we can see in figure 6, at the beginning while the teacher forcing ratio is still 1 (ie. there is 100 % chance of doing teacher forcing) the PSNR score starts high but decreases. It is because the model is taking the "bad habit" to rely on the ground truth image during training and start to perform poorly on the validation dataset where there is no ground truth image. Around the 10th epoch, we can see the PSNR score is starting to go up again as the model is letting go of the teacher forcing and becomes more performant on the validation dataset.
In order to better see the impact of the teacher forcing, I trained my model using no teacher forcing as shown in figure 7.

Indeed, there is no more crease around the 10th epoch. We can also notice that the performance is way better without teacher forcing, hence in my final prediction I did not use teacher forcing.

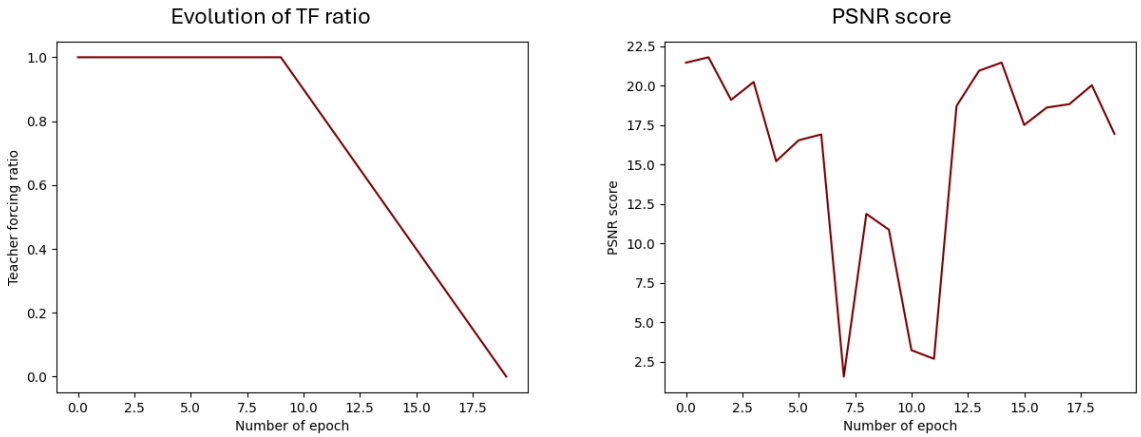

Figure 6: Impact of the teacher forcing on the PSNR score

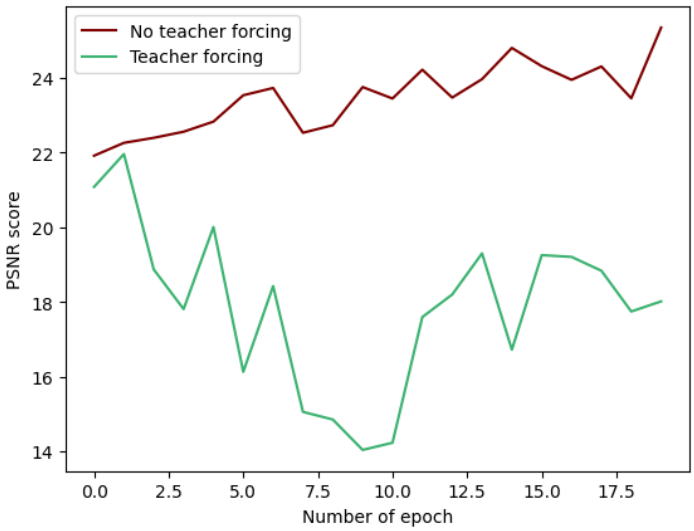

Figure 7: Evolution of the PSNR score with no teacher forcing

3.2 KL annealing

In this section, I will explore different type of KL annealing : monotonic KL annealing, cyclical KL annealing, no KL annealing and a hybrid method.
The idea of KL annealing is to vary the weight of the KL loss as the training goes on. Indeed, as the representation of the training data at the beginning is bad, the decoder might learn to ignore it to get better performances. While this will work at the beginning of the training, it will lead to the data representation being ignored when the posterior predictor becomes good, thus it might worsen the final performance. In order to avoid that, we give a small importance to the KL loss at the beginning and increase its importance through the epochs. Since the training loss is the sum of the reconstruction error and the KL loss weighted with a parameter β, we can achieve KL annealing by defining β as a function of the current epoch.

3.2.1 Monotonic KL annealing

In monotonic KL annealing, the importance of the KL loss, ie. β linearly increases.
In figure 8 is the evolution of β as well as the evolution of the PNRS score for the validation data across one run of 20 epoch.

As we can see the model learns really fast at the beginning but then get stuck at a PSNR score of about 22 then start learning again around epoch n°18.

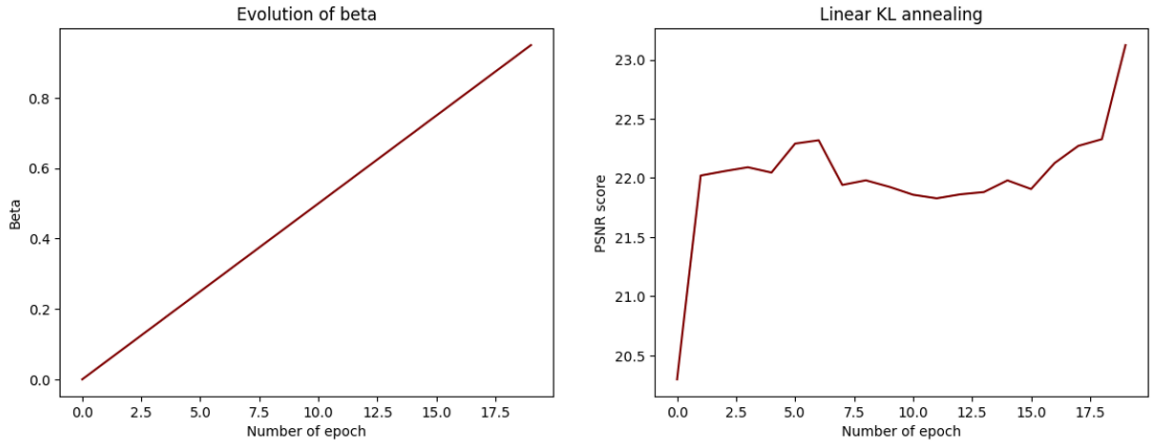

Figure 8: Linear KL annealing

3.2.2 Cyclical KL annealing

In cyclical KL annealing, β increases linearly over a given number of epoch then goes back to zero. This cycle is then repeated a given number of time.

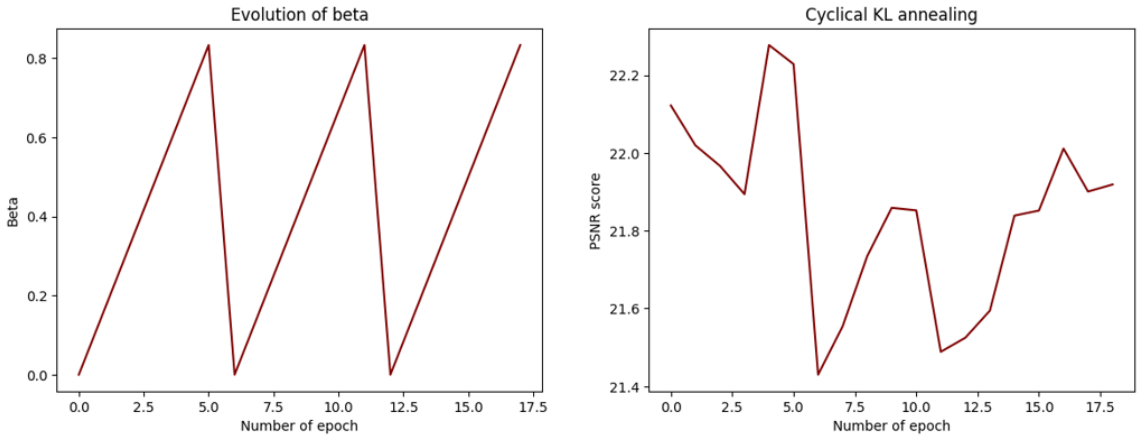

Figure 9: Cyclical KL annealing

As we can see in figure 9, the evolution of the PSNR score roughly follows the evolution of the parameter β with, unlike the linear model which was rather smooth.

3.2.3 No KL annealing

In figure 10 is the PSNR score for the case where there is no KL annealing. As we can see it is pretty similar to the linear case but without the stagnation stage in the middle, suggesting that it might be a better choice to not use annealing for this problem.

3.3 Hybrid methods

I wanted to explore what would happen if the KL annealing initially followed a cyclical β but then stayed at a high value. The result is in figure 11.

As we can see, no only is the PSNR score not really increasing, the performance in overall worse than for all the previous types of KL annealing. Hence I didn't pursue this idea.

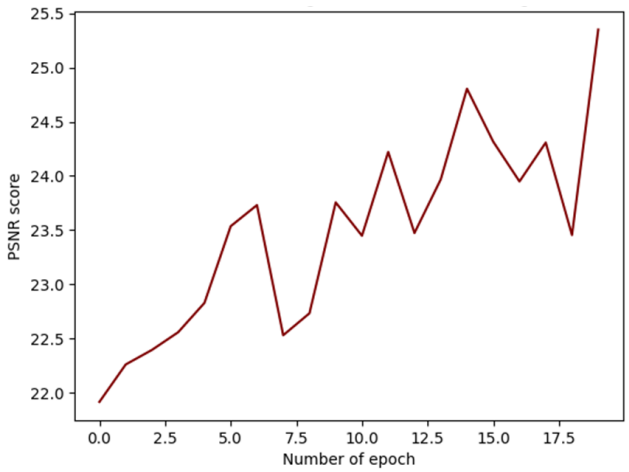

Figure 10: No KL annealing

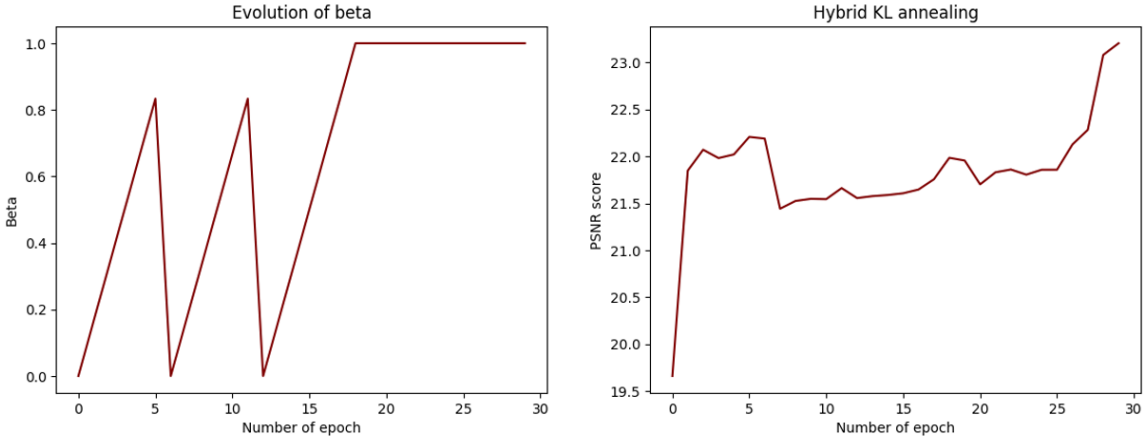

Figure 11: Hybrid KL annealing

3.3.1 Comparison of performance

In order to select the best KL annealing strategy, I compared all of them on a given number of epoch as shown in figure 12

3.4 PSNR per frame

The figure 13 plot the PSNR per frame for one of the video of the validation dataset. As expected, the PSNR score decreases as the frame number increases. This makes sense as, as the video goes by, we predict the next frame based on the previously generated frame which might be faulty. Hence the errors "propagate" and the performance decreases.

3.5 Dimension of the noise

In the model's parameter in the pre given code, the default value for the dimension of the noise was 12. I wanted to see if the performances would increase or decrease if I varied N. Indeed, a small dimension could help compress the data more efficiently but a bigger dimension could also help encode the data in a more complex way, hence both had their advantages.
Figure 14 is the comparison figure for the PSNR score for different values of the dimension :

As we can see a bigger dimension (here 32) yields better results thought the difference is not really significant.

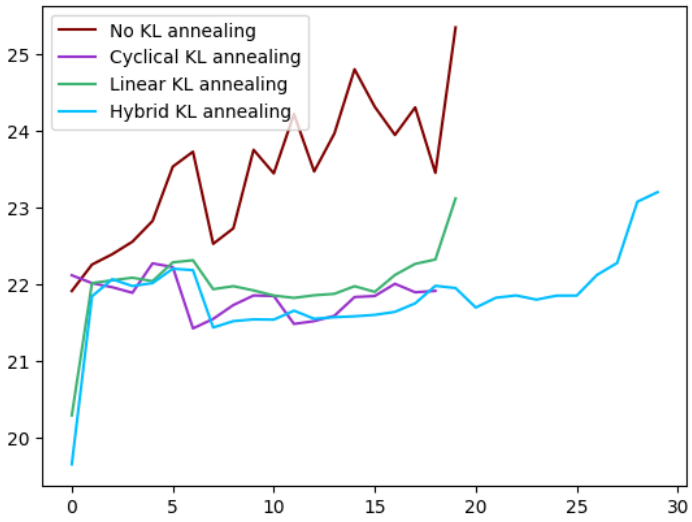

Figure 12: KL annealing comparison

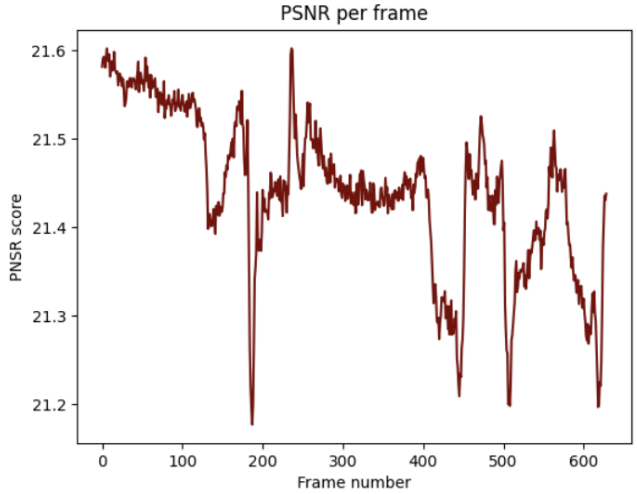

Figure 13: PSNR per frame

4 Conclusion

Throughout this report, I explored the impact that different hyperparameters - the KL annealing, the teacher forcing and the dimension of the noise - impact the model performance. Through that, I selected the model that gave me the best results : no KL annealing, no teacher forcing and a noise dimension of 32. Below is the command line for my final model :
python Lab4_template/Trainer.py - -DR LAB4_Dataset - -save_root result –batch_size 8 - -num_epoch 70 - -fast_train - -kl_anneal_type none - -per_save 1 - -tfr 0 - -tfr_sde 80 - -N_dim 32

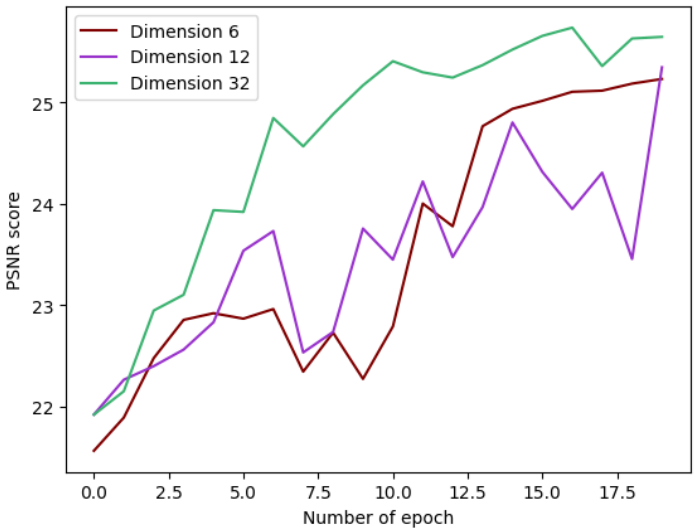

Figure 14: Impact of the dimension