REPRODUCIBILITY CHALLENGE-LATENT IMAGE ANIMATOR

Ankana Mukherjee, Keniel Peart , Supritha Konaje
School of Electronics and Computer Science
University of Southampton
(am3e21, krp1n21 , ssk1n21)@soton.ac.uk

ABSTRACT

In this report, we assess the reproducability of "Latent Image Animator: Learning to animate images via latent space navigation " Wang et al. (2022b). The paper proposes a novel self supervised technique to transfer motion from videos to images by learning to linear transformation in the latent space. We detail our re-implementation of their proposed model, and the testing methodology used to compare our results with those presented in Wang et al. (2022b). Our results show that, despite re-implementing the details in the paper, our results were far worse than those achieved in the original paper. We found that the published findings were still valid however, as they provide better transferability over existing methods.

1 INTRODUCTION

Latent Image Animator is a novel self supervised technique that transfers the motion of a driving video onto source images by learning linear transformation in the latent space.The authors tries to eliminate the previously used concepts of structure representation (for instance, semantic maps, human key points, 3D meshes and optical flows). As structural representation limits sufficient disentanglement of motion from other attributes such as shape and also situations where the source image and driving video encompass large appearance variation.

The empirical evidence presented in the paper shows that LIA outperforms existing techniques, including state-of-the-art (FOMM) methods, in standard both benchmark metrics such as perceptual loss between the reconstructed image and the driver image (LPIPS) and Frechet Inception Distance (FID) and tasks such as Cross-video generation and Same-identity reconstruction.

In this reproducibility project, we attempt to implement from scratch [1] the model proposed and reproduce baseline experiments of Same-identity reconstruction described by Wang et al. (2022b).

2 IMPLEMENTATION DETAILS

The models, losses and other methods proposed in the paper were implemented from scratch as code was not available. These were implemented per specification, however while there was some amount of detail on the models, there still existed alot of gaps in details. Where details were not specified, we made assumptions using information from other papers written by the same author or by other authors.

The datasets used in the paper are Taichi-HD, VoxCeleb and TED-talk. We however limited our re-implimentation to the Taichi-HD and VoxCeleb datasets. These two were chosen as they provide unique representations of motion. The Taichi dataset represents articulated human body movements (Taekwondo) while the Voxceleb dataset represents movement of facial features. We pre-processed the datasets by converting the videos to frames and resizing the frames per the specification in the paper (256x256 pixels). The size of the datasets were also culled because we had access to limited compute time.

[1] https://github.com/COMP6248-Reproducability-Challenge/Latent-Image-Animator

2.1 Model Architecture

The LIA architecture consists of a self-supervised auto-encoder , the linear motion decomposition technique and a discriminator. (Figure 1)

The general idea of the technique is to encode the driving and source images into latent codes assuming there is a reference image by which motion can be modelled from ie. we encode the source image into a representation of $Z_s \rightarrow_r$ where Z represents the latent code and $s \rightarrow_r$ represents a movement from source image x_s to reference image x_r. The same is done for driving image x_d ie. when we encode the driving image we will receive a result of $Z_d \rightarrow_r$. The result of encoding the driving image is then fed into a 5 layer MLP to obtain a vector A that represents the magnitudes of the motion directions in the latent space.

The cornerstone algorithm for the method proposed by the authors is Linear Motion Decomposition. The LMD approach involves learning independently a dictionary of orthogonal (enforced by applying QR decomposition at each forward pass) motion directions D_m (where each vector represents a basic visual transformation) and combining them with the learnt motion magnitudes A to create a linear latent path $w_r \rightarrow_d$ (Equation 1). This path is used to navigate $Z_s \rightarrow_r$ to a target latent code $Z_s \rightarrow_d$ (Equation 2).

The target latent code is then fed into the Generator which decodes the latent code and uses it to warp the source features into reconstructing the target image. The Generator consists of two components, a flow field network G_f and a refinement network G_r. The flow field generator uses a StyleGan inspired block to upsample and apply the latent code to the image,and return a feature map with 3 channels, which, two channel are used for flow fields (ϕ) and the other channel as the corresponding mask (m_i) (regions required to be inpainted). We perform the Hadamard product ($\odot$) result of warping the and m_i. The output is then converted to RGB and merged with an upsampled map from the previous layer.

$$w_r \rightarrow_d = \sum_{i=1}^{M} a_i d_i \tag{1}$$

$$Z_s \rightarrow_d = Z_s \rightarrow_r + w_r \rightarrow_d \tag{2}$$

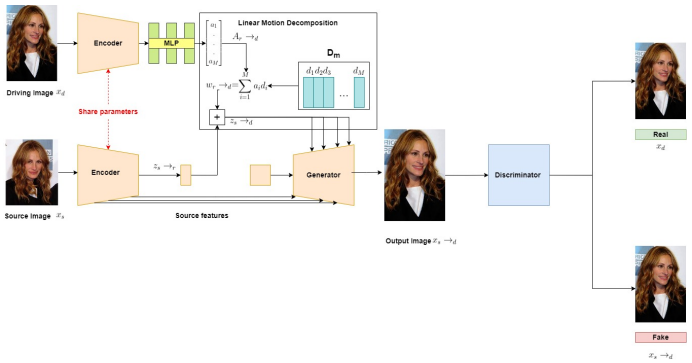


Figure 1: Latent Image Animator Architecture

2.2 Loss Functions

The loss function that is proposed by the paper is a sum of 3 other functions, namely: Reconstruction, Adversarial, Perceptual.

- Reconstruction Loss ($\mathscr{L}_{recon}$):It is used to reduce the pixel wise loss from by calculating the distance between

- Perceptual Loss ($\mathscr{L}_{vgg}$):To minimise this loss on multi-scalar feature maps, the VGG19-based
- Adversarial Loss ($\mathscr{L}_{adv}$):To minimise the losses during generation of photo-realistic results, we used the non-saturating adversarial loss $\mathscr{L}_{adv}$. This loss includes implementing a discrminator (Figure 1).

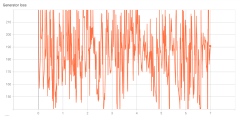
Total loss is given by the sum of all three losses with λ as the balance hyperparameter used to penalize more the perceptual loss.

$$\mathscr{L}(x_{s\to d}, x_d) = \mathscr{L}_{recon}\,(x_{s\to d}, x_d) + \lambda\,\mathscr{L}_{vgg}\,(x_{s\to d}, x_d) + \mathscr{L}_{adv}\,(x_{s\to_d}, x_d)$$

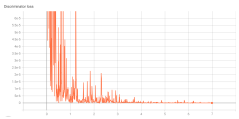
2.3 TRAINING

We train the implemented LIA model to transfer motion from driving images to source images in batches of Voxceleb and Taichi data. Our Pytorch model was trained using with Python 3.6 on the module's gpu server (4 GTX 1080Ti GPUs). Wang et al. (2022b) trained their model on four 16G NVIDIA V100 GPUs with a batch size of 32 with 8 images per GPU. The authors also used the Adam Optimizer with a learning rate of .002 and trained their model for an undisclosed amount of epochs. We tried to stay as close as possible to the specification of the original model however we had to reduce the batch size because of gpu memory allocation issues. We used a batch size of 15.

Figure 2 and 3 shows the training with the described loss criterion for the generator and an assumed criteria for the discriminator. It can be noted that the plots are quite noisy. We expected the generator loss to be noisy as the loss function designed by the author heavily penalises perceptual loss which is a measure of how good the generated image is. This loss will initially be bad until the model overtime learns to generate high quality images. We ran the training for 60 epochs in total; the authors while they didn't specify the number of epochs they trained for, wrote that training took 150 hours.

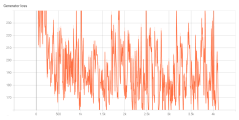


(a) Generator Loss Graph for TaiChai Dataset

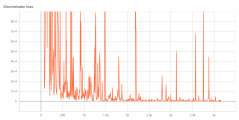


(b) Discriminator Loss Graph for TaiChai Dataset

Figure 2: Generator and Discriminator Loss graph for TaiChai Dataset



(a) Generator Loss Graph for VoxCeleb Dataset



(b) Discriminator Loss Graph for VoxCeleb Dataset

Figure 3: Generator and Discriminator Loss graph for VoxCeleb Dataset

2.4 ASSUMPTIONS AND CHALLENGES

We encountered some difficulties while reproducing this task due to the ambiguous descriptions of the methodology. Firstly, the model architectures were not described in full detail. Certain parts had to be assumed from estimated from the image sizes listed in the paper, hence our resulting models were not an exact reproduction of the originals. Secondly, integral to the Linear Motion Decomposition algorithm is the motion dictionary, Wang et al. (2022b) failed to provide details on how this was initialized. We however found reference to LMD in a previous paper Wang et al. (2022a) by one of the authors and assumed the initialization approach was the same.

Although the authors made reference to the result of a discriminator as apart of the adversarial loss, however a description of the architecture of that discriminator was not given. The author also

failed to mention specific details re activation functions (we assumed RelU and LeakyRelu - for discriminator) and the convolution layers; we assumed padding and stride based on the input and output image size.

3 Results and Analysis

Despite computing constraints and information gaps in breadth of the model trained we noticed some amount of adoption by the model on the task of transferring motion. From a baseline comparison however with the paper; our model failed to reproduce the evaluation metrics used in the paper. Results in table 1 and table 2 shows the difference in performance of our trained model and the model described in the paper. it is worth nothing that our model was trained at a marginal fraction of time that the model in the paper was trained for and with less computing power. We also had to make modifications to parameters such as the batch size to execute our training. These factors are likely to play a part in the results displayed.

Methods	L_1	**AED**	**LPIPS**
LIA	0.041	0.138	0.123
Ours	0.2315	48.874	0.4731

Table 1: Comparision of VoxCeleb dataset

Methods	L_1	**AED**	**LPIPS**
LIA	0.057	0.431	0.180
Ours	0.3315	35.874	0.3562

Table 2: Comparision of TaiChai dataset

4 Discussion

The paper was generally well written, and many (although not all) experimental aspects were detailed. The report structure could benefit from a clearer flow and better clarity around the specific of the networks and the Linear motion Decomposition algorithm. A table of parameters would have assisted with reproduction.

There are a couple concerns/gaps we have in our understanding of the approach. Particularly we believe that the model would not perform well on transferring large and complex motions as the latent representation being used is highly compact. For example the model would suffer from trying to represent an intricate dance motion scene into a motion representation of 20 directions. This hypothesis is also compounded by the fact that the paper uses linear navigation which is restrictive. A non-linear function would be better able to capture complex motion and transitions.

Also it is noticed that the model does poorly on transferring motion when there exist some amount of occlusion. While we focused on absolute transfer of motion ie. Same Video Reconstruction we noticed that there were some artifacts in the generated images. This may not be the case for relative transfer but that is yet to be determined.

5 Conclusion

We have reproduced the "Latent Image Animator" from Wang et al. (2022b) to animate images by linear navigation in the latent space. We have trained our model on TaiChai and VoxCeleb dataset. The paper was a bit abstract while describing the generator and the discriminator, because of which we had to assume few parameters. Since, we have used only a subset of the main dataset and less number of epochs to train our model and for less number of epoch, the output and analytical results are not the exact same copy of what is shown in the paper.

References

Yaohui Wang, Francois Bremond, and Antitza Dantcheva. Inmodegan: Interpretable motion decomposition generative adversarial network for video generation, 2022a. URL https://arxiv.org/abs/2101.03049.

Yaohui Wang, Di Yang, Francois Bremond, and Antitza Dantcheva. Latent image animator: Learning to animate images via latent space navigation. In *International Conference on Learning Representations*, 2022b. URL https://openreview.net/forum?id=7r6kDq0mK_.