Advanced Computer Vision Resit Coursework

dflc39

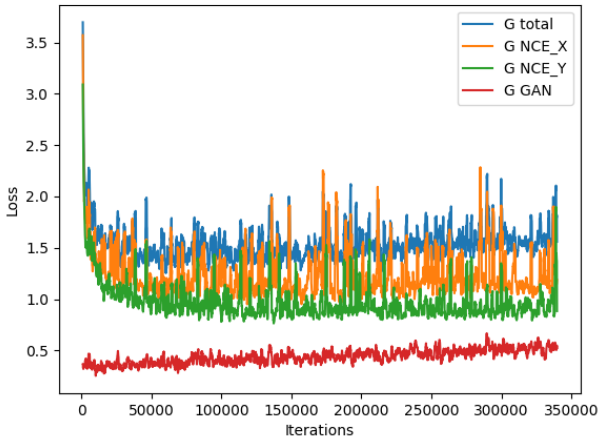

Fig. 1: Graph of the generator losses during the training of the baseline CUT model (games → movies). We can see that loss decreases sharply at the beginning, but plateaus soon after.

1 Preprocessing

We considered randomly sampling extracted frames to create a testing set. However, this approach is unsuitable as consecutive frames are nearly identical; hence, the testing set would contain similar data to the training set, and we would not test for overfitting. Instead, we use the images extracted from the final 30% of each video to test our models; thus, we avoid the problem of consecutive frames, despite a minor overlap.

2 Frame-to-frame Models

Hyperparameter	Value
Resolution	512×512
Number of epochs with the initial learning rate	50
Number of epochs to linearly decay learning rate to zero	50

TABLE 1: Hyperparameter configuration for our frame-to-frame models.

We use Contrastive Unpaired Translation (CUT) [1]. Park et al. train a CUT model on the '-Cityscapes dataset using 512×512 crops; we use the same resolution in our experiments. We train our models over 100 epochs and choose the learning rate configuration shown in Table 1.

Park et al. [1] do not explore how the number of negative samples affects performance[1]. Hence, we train models with the following number of patches: 256, 128, and 64. Finally, we explore the use of triplet loss as an alternative to PatchNCE [2]: we use the query patch as the anchor, the target patch as a positive sample, and the others as negatives.

Our models perform significantly better in the movie-to-game direction, as observed in Table 2. Game frames lack the detail present in movies. Hence, we observe a loss of detail when transferring in this direction, as seen in Figure 2. The model simplifies the lighting, reduces colour variety, and removes edges. However, certain parts of the image become too blurry and pixelated. Additionally, we observe artifacts and colour aberrations in the output, and the changes in lighting are frequently erroneous. These failure cases are especially evident when we examine faces in the output.

In the game-to-movie direction, the model attempts to add detail to the input to make the image more "movie-realistic", which is a challenging task. It changes the texture, the colours, and the lighting. It also softens the edges, which occasionally results in distortions in the output. Again, we observe artifacts and erroneous changes in lighting, which are most noticeable on faces.

3 Face-to-face Models

In the following subsections, we discuss the face-to-face models implemented and our results.

3.1 Face Detection

We use the *face_recognition* package [3] to detect faces. We create two datasets: one with all the detected faces and the other with high-quality face images. We determine image quality by examining image resolution and blurriness. We use the Fast Fourier Transform algorithm in NumPy to detect blur [4], [5], [6].

We use the *face_recognition* package [3] to encode faces and calculate face distances. We create the testing set by randomly sampling an image and selecting the closest faces. Hence, we ensure that the test data is distinct from our training set and that we can test for overfitting.

1. Park et al. [1] explore the use of 15 negatives in combination with other modifications; however, they do not conduct an ablation study exclusively on the number of negative samples. Furthermore, they fail to explain how they arrived at their default number of 255.

TABLE 2: FID scores for trained models.

Model	Testing dataset FID			
	Frame-to-frame		Face-to-face	
	Games → Movies	Movies → Games	Games → Movies	Movies → Games
CUT	262.7	**151.6**	218.6	351.5
CUT (64 patches)	262.6	229.9	196.6	204.7
CUT (128 patches)	**255.8**	162.4	222.0	363.2
CUT (L2)	308.2	293.5	272.6	280.3
CUT (triplet loss)	266.8	241.7	251.9	263.9
CUT (faces)	296.4	284.8	132.3	**132.5**
CUT (hq-faces)	266.9	292.0	**128.2**	133.8
CUT (hq-faces-aug)	267.7	274.9	158.6	230.8

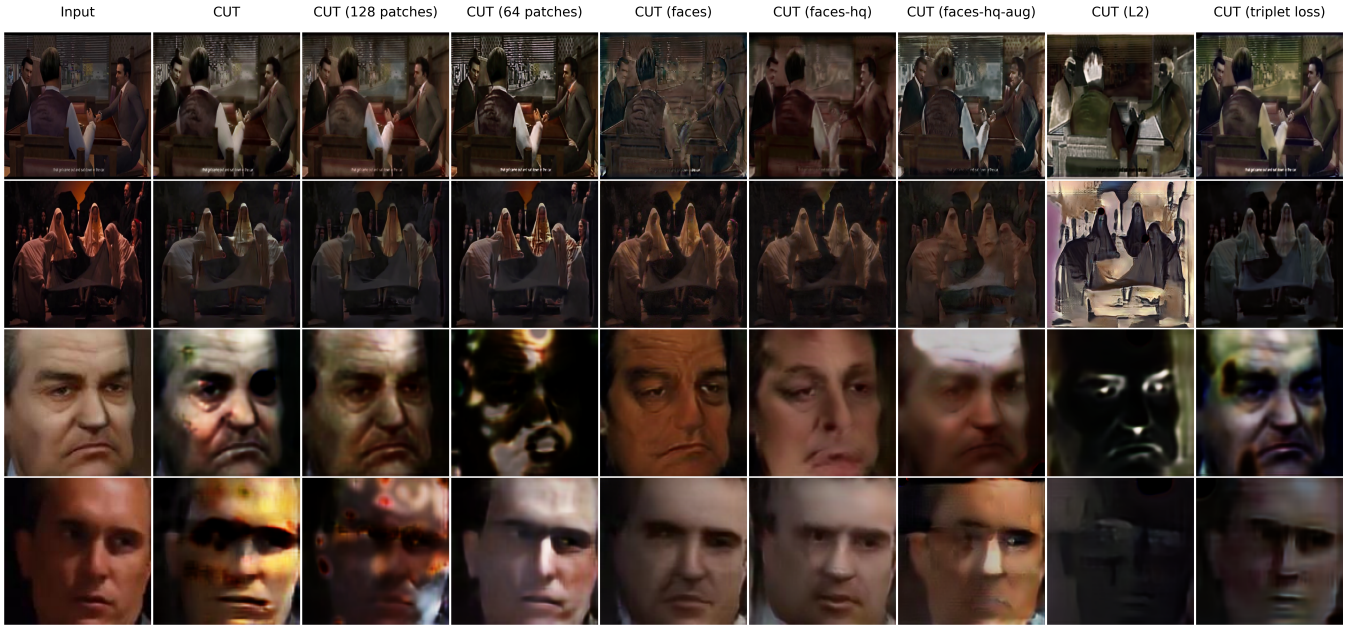

Fig. 2: Output comparison of trained models.

3.2 Model Analysis and Comparison

We adjust the resolution of our training data to 128×128 as the images in our new dataset are smaller. Next, we train two CUT models, one with each of the face datasets[2]. The results in Table 2 demonstrate no significant difference in performance between the two models. The model performs better transforming faces than entire frames. The face images contain fewer details than whole frame images, and patches are more similar throughout the image. We believe this encourages the generator to create fewer artifacts and more "movie-realistic" data.

3.3 Data Augmentation on Faces

Wang et al. [10] identify three types of transformations to augment face data: generic, component, and attribute. Research has demonstrated that augmenting data using a combination of different types of transformations improves the performance of face recognition models [11], [12], [13].

2. We refer to the model trained with the high-quality dataset as "CUT (hq-faces)" in figures and tables.

(a)

(b)

Fig. 3: Image augmentation using Albumentations [7].

Thus, we attempt to augment the high-quality dataset using Albumentations, PA-GAN, and Att-GAN [7], [8], [9].

Figure 4 demonstrates our attempt at data augmentation using PA-GAN and Att-GAN [8], [9]. We consider the out-

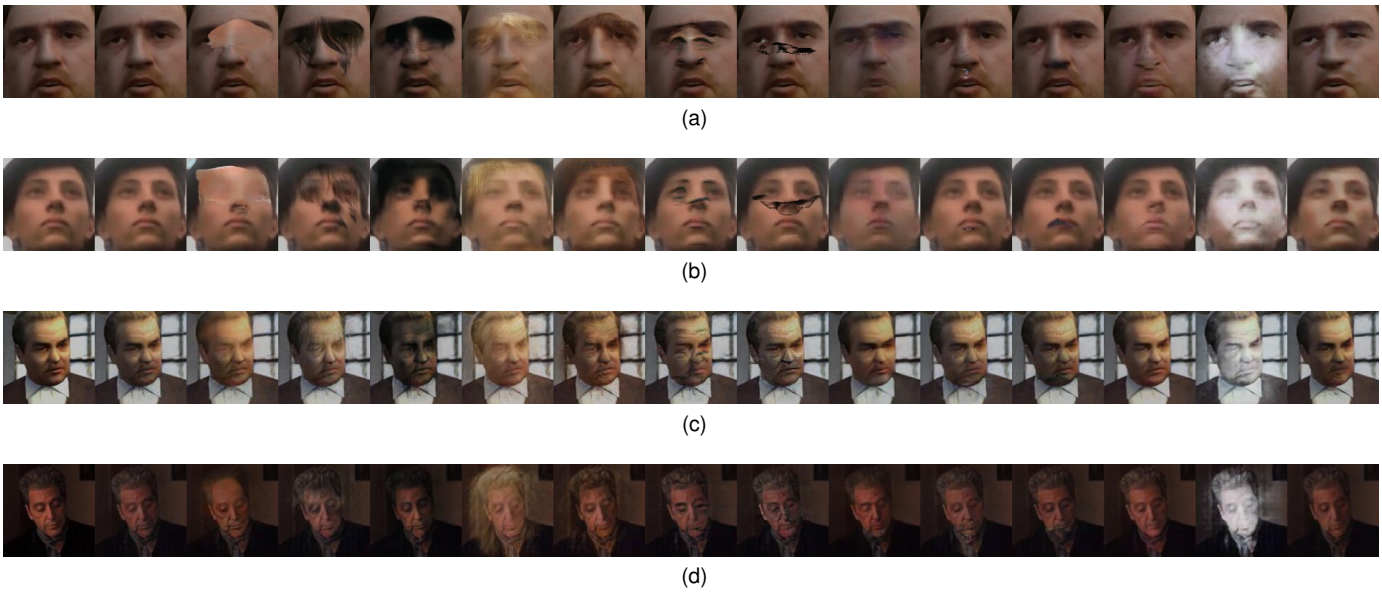

Fig. 4: Figures 4a and 4b were generated using PA-GAN, while Figures 4c and 4d were generated with Att-GAN [8], [9]. In each figure, the leftmost face is the original image. The subsequent faces are the result of various transformations by pretrained models.

put of insufficient quality. Hence, we choose not to use it.

Table 2 demonstrates that training with the augmented dataset results in worse performance.

3.4 Evaluation

Structurally speaking, the videogame faces are generally realistic, with a few exceptions. However, this realism breaks down when the characters do not have a neutral expression. For example, one of the images has an angry character, and the lines that define the eyebrows and the edges of the face are too straight and unnatural. The model attempts to fix this by softening the edges. Additionally, we observe that the model adds detail to the faces; for example, it adds wrinkle lines. The skin is too smooth and lacks detail in the videogame faces.

4 Real-world Applications

We discuss real-world applications of our models in the subsections below.

4.1 Transforming Game Footage

We used the CUT (faces-hq) model to transform the game footage. The textures and tones in the transformed video are similar to those in the movie data. However, the lighting does not look natural, and we observe a loss of fine details. Moreover, we observe motion blur and distortions around moving objects and the surrounding background. Additionally, temporal consistency is poor.

We could stabilize generated videos and reduce motion blur using optical flow, which we could compute using FlowNet [14]. Next, we could use the computed optical flow during training and penalise the generator for creating unstable videos using a loss function [15]. Finally, to scale our system and make it viable for real-time applications, we could train a student network using the trained model, thus eliminating the need for optical flow during inference [16]. However, failure cases could emerge if the student network fails to generalise.

4.2 Using the Rendering Pipeline

If we had access to the entire rendering pipeline, we would construct a two-stage training process to address the problems discussed. Firstly, we would train a model from various perspectives of the same scene to prevent blurry output. For example, we could prioritise close-up views of essential details, such as faces. Then, once the entire scene is transformed from various perspectives, we could average the output for each patch in the scene. Secondly, to address the lighting issue, we could train a second model or finetune the first one and train it with different light sources at different angles.

REFERENCES

[1] T. Park, A. A. Efros, R. Zhang, and J.-Y. Zhu, "Contrastive learning for unpaired image-to-image translation," in *European conference on computer vision*. Springer, 2020, pp. 319–345.

[2] G. Chechik, V. Sharma, U. Shalit, and S. Bengio, "Large scale online learning of image similarity through ranking." *Journal of Machine Learning Research*, vol. 11, no. 3, 2010.

[3] A. Geitgey, "Face recognition," 2020. [Online]. Available: https://github.com/ageitgey/face_recognition.git

[4] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex fourier series," *Mathematics of computation*, vol. 19, no. 90, pp. 297–301, 1965.

[5] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. Fernández del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, p. 357–362, 2020.

[6] A. Rosebrock, "Opencv fast fourier transform (fft) for blur detection in images and video streams," 2020. [Online]. Available: https://pyimagesearch.com/2020/06/15/opencv-fast-fourier-transform-fft-for-blur-detection-in-images-and-video-streams/

[7] A. Buslaev, V. I. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin, and A. A. Kalinin, "Albumentations: Fast and flexible image augmentations," *Information*, vol. 11, no. 2, 2020. [Online]. Available: https://www.mdpi.com/2078-2489/11/2/125

[8] Z. He, M. Kan, J. Zhang, and S. Shan, "Pa-gan: Progressive attention generative adversarial network for facial attribute editing," *arXiv preprint arXiv:2007.05892*, 2020.

[9] Z. He, W. Zuo, M. Kan, S. Shan, and X. Chen, "Attgan: Facial attribute editing by only changing what you want," *IEEE Transactions on Image Processing*, vol. 28, no. 11, pp. 5464–5478, 2019.

[10] X. Wang, K. Wang, and S. Lian, "A survey on face data augmentation for the training of deep neural networks," *Neural computing and applications*, vol. 32, no. 19, pp. 15 503–15 531, 2020.

[11] J.-J. Lv, X.-H. Shao, J.-S. Huang, X.-D. Zhou, and X. Zhou, "Data augmentation for face recognition," *Neurocomputing*, vol. 230, pp. 184–196, 2017.

[12] I. Masi, A. T. Tran, T. Hassner, G. Sahin, and G. Medioni, "Face-specific data augmentation for unconstrained face recognition," *International Journal of Computer Vision*, vol. 127, no. 6, pp. 642–667, 2019.

[13] A. Zhuchkov, "Analyzing the effectiveness of image augmentations for face recognition from limited data," in *2021 International Conference" Nonlinearity, Information and Robotics"(NIR)*. IEEE, 2021, pp. 1–6.

[14] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "Flownet 2.0: Evolution of optical flow estimation with deep networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2462–2470.

[15] J. Yu and R. Ramamoorthi, "Learning video stabilization using optical flow," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 8159–8167.

[16] X. Chen, Y. Zhang, Y. Wang, H. Shu, C. Xu, and C. Xu, "Optical flow distillation: Towards efficient and stable video style transfer," in *European Conference on Computer Vision*. Springer, 2020, pp. 614–630.