An Interesting Fighting Game using Style-based Generative Adversarial Network and Real-time Human Pose Estimation

Lin-Yung Hsieh
National Tsing Hua University
s111061553@m111.nthu.edu.tw

Jin-Cheng Jhang
National Tsing Hua University
s111061540@m111.nthu.edu.tw

Yu-Cheng Hsieh
National Tsing Hua University
s111061517@m111.nthu.edu.tw

Yi-Shan Lee
National Tsing Hua University
s110061623@m110.nthu.edu.tw

You-Ze Huang
National Tsing Hua University
s111033545@m111.nthu.edu.tw

Abstract

We implement an interesting two-player fighting game, which can be played on our laptop without the need of GPU. One goal of this work is to present real-time human action detection using the front-facing camera, and we use ***mediapipe*** *(a real-time human pose detection network) to achieve this goal.*

Besides, the other purpose is to gradually change the players' faces to ugly faces during the game, we utilize ***StyleGAN*** *to reach this goal. Our quantitative and qualitative results demonstrate the excellent end-to-end performance of this game.*

Our source code is released at https://github.com/LeoTheBestCoder/CV-Final-Project, and our complete demo is published at https://youtu.be/0Zamq8Xk2AM.

1. Introduction

Many computer vision techniques incorporate forms of machine learning and have been applied to various video games, such as Minecraft and Super Mario. These application of computer vision focuse on interpreting game events using visual data.

Based on those existing computer-vision-based games, we develop a two-player fighting game. There are some game rules in the following:

(1) When entering the game, two players will take a selfie respectively as their initial player avatar in the game.

(2) The camera will capture the players' human pose, reflecting the human pose in the game. There are five actions defined by the relative positions of key points. For example, when the player raises his/her right hand, the avatar goes right.

(3) The players' faces (source face) will change to another face (target face) step by step.

(4) The more health points a player loses, the more his/her face will change.

(5) The player should try his/her best in order to keep his/her face from turning into other people's faces.

To achieve the goal above, we use StyleGAN [5] to generate a sequence of images that are style mixed between the source face and the target face. For real-time human action detection, we use mediapipe [7] to implement it. To combine these parts into a game, the game logic is needed.

Our contributions are combining all above elements and implementing into a real-time fighting game without the need of GPU, and achieving excellent performance (some actions even achieve 100% accuracy) on human action detection.

2. Related Work

2.1. StyleGAN

When it comes to generating images from style aware latent codes, StyleGAN-based work must first come to our mind. There are several styleGAN based works, such as

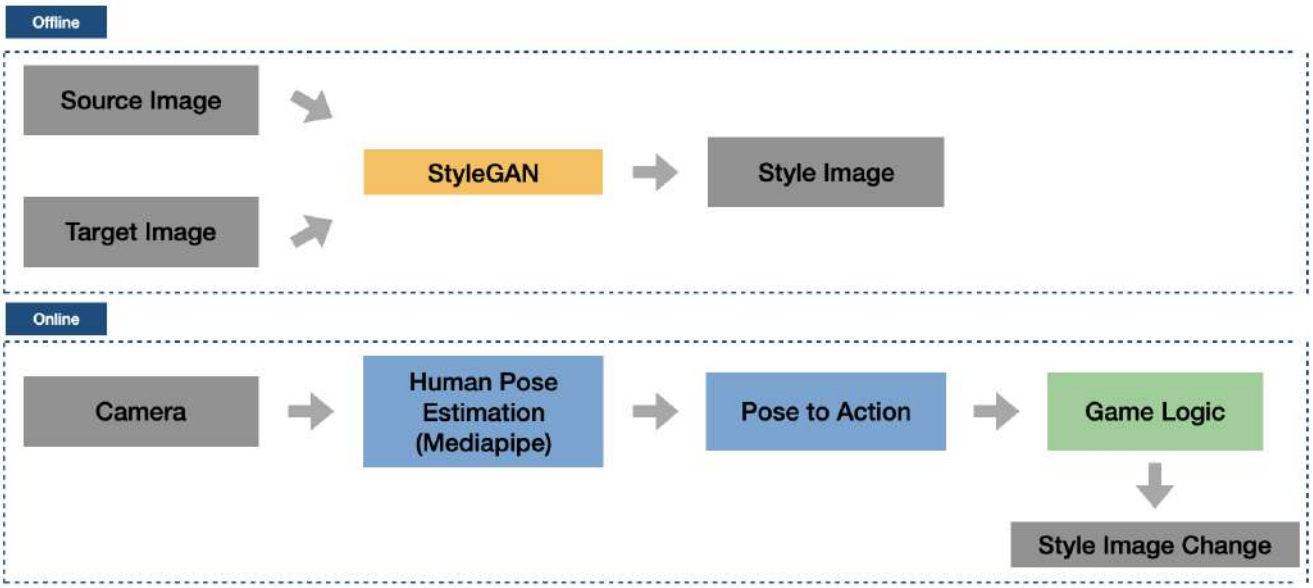

Figure 1. Players will take a selfie respectively as their player avatar before the game starts. During the game, the camera will capture the players' human pose, reflecting the action in the game. The faces in the game will also change by time according to the health points of each player.

original styleGAN [5] and styleGAN2 [6]. These works focus on diversity and quality on the generated images. However, the ultimate goal of these works is using random latent code to generate diverse and plausible images, while our goal is to use specific latent code to reconstruct the same images and apply latent code interpolation to mix the style of two given faces, which is totally different from original styleGAN based works.

To address this issue, we finally find some similar works called style encoding. Style encoding generates corresponding latent codes based on input images. We find two works focusing on style encoding tasks [4] [8]. Finally we decide the latter one because its style encoder outperforms the former one.

2.2. Human Pose

There are many existing work focusing on human pose estimation, such as YOLO [3] [9] [2]. However, we find that if we inference on our laptop, it'll run very slow. The FPS of the game will drop to 2 or 3, which is unsuitable for our application. What we need is a simple backbone that can perform human pose estimation real-time, instead of a heavy backbone that can predict very accurately.

Therefore, we search for a simple network that satisfies our goal, **mediapipe** [7]. We find that it can perform human pose estimation in real-time, mainly due to two reasons. First, Figure 2 and Figure 3 shows the architecture of YOLOv7 [9] and mediapipe respectively. We can easily find that the architecture of YOLOv7 [9] is far more complicated than that of mediapipe [7], which implies that the number of parameters in mediapipe [7] is far less than that in YOLOv7 [7], so the inference speed will be faster.

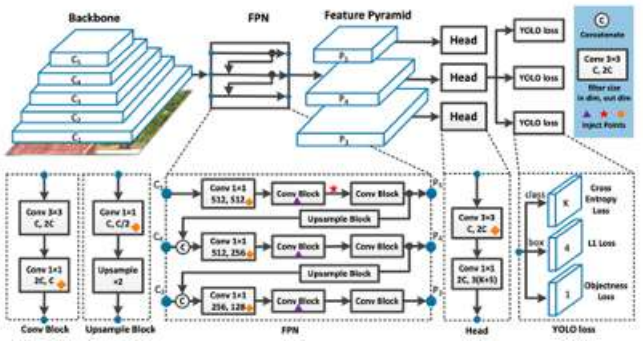

Figure 2. Architecture of YOLOv7 [9].

Second, unlike most human pose backbones that perform detection at every single frame, mediapipe only performs human detection at the very beginning frame as Figure 4 shows. This approach greatly reduces the computational cost. Combined with 2 advantages mentioned above, mediapipe seems to be the best choice for our application.

The original version of mediapipe, however, only supports human pose estimation for single person, but our game definitely requires the simultaneous huamn pose estimation on two people. Therefore, we make some modifications to

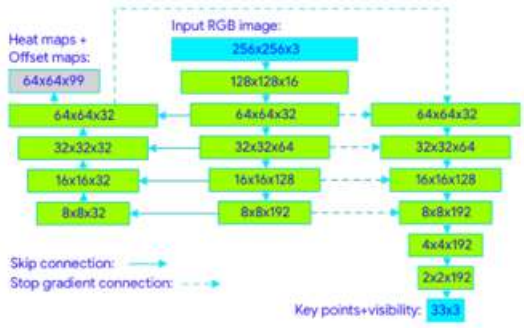

Figure 3. Architecture of Mediapipe [7].

Figure 4. Mediapipe only performs human detection at the very beginning frame, which greatly reduces the computational cost.

it, which will be elaborated in Section 3.2.

3. Technical Part

In this section, we present the framework for our game, which is depicted in Figure 1. There are methods of detailed implementation.

3.1. StyleGAN

In order to mix the style of two real faces, we utilize the style encoding [8], style mixing, latent code interpolation and StyleGAN [5] [6] inference. Firstly, Style encoding aims to find a corresponding latent code w from a real image I such that the generated image I' from w is very similar to the original real image I (you can regard this task as a reconstruction task). We use Style encoder to transform source images I_{source} and target image I_{source} to obtain corresponding latent vectors w_{source} and w_{target} and then use these latent vectors to do latent code interpolation. our style encoder architecture are shown in Figure 5.

Secondly, the main idea of latent code interpolation is to get a sequence of latent codes $[w_1, ..., w_n]$ by interpolating different proportion of latent code w_{source} and w_{target}, where n is the length of sequence (e.g. w_1 consists of 95% of w_{source} and 5% of w_{target} ; w_2 consists of 90% of w_{source} and 10% of w_{target}...)

Finally, we will put results from latent code interpolation $[w_1, ..., w_n]$ into StyleGAN model and the model outputs a sequence of images $[I_1, ..., I_n]$ that mixed the style of w_{source} and w_{target}. From the left part of the sequence $I_1, I_2, ...$ preserves more information of w_{source} while the right part of the sequence $I_n, I_{n-1}, ...$ preserves more information of w_{target}. In our game, we will change the player's face base on image sequences $[I_{source}, I_1, I_2, ..., I_n, I_{target}]$ whenever the player loses health points.

3.2. Human pose estimation

In order to reflect the human action in the game, we need a model that can predict real-time human pose. Besides, due to the immediacy we require in our game and the laptop we use without the high Compute Capability GPU, the model cannot be too computationally expensive. Therefore, we discard some state-of-the-art human pose estimation frameworks such as YOLOv7 [9].

Instead, we select **mediapipe** [7] developed by **Google** to achieve real-time human pose estimation. It captures the real-time image from the camera and performs real-time single person human pose estimation, which can predict 33 key points as Figure 6 shows.

In order to predict two people's key points in one image, we crop the image into two halves. One person stands on the left side and the other stands on the right side. We perform their pose estimation individually. The position matrix of the left-side person is denoted as $M_{left} = [[x_1, y_1], [x_2, y_2], ..., [x_{33}, y_{33}]]$ and the position matrix of the right-side person is denoted as $M_{right} = [[x_1, y_1], [x_2, y_2], ..., [x_{33}, y_{33}]]$, where x and y are the positions of each key point. We can use these 33 key points to predict actions such as "raise right hand." Take "raising right hand" for example. When the position y of key point 15, the right wrist, is larger than the position y of key point 0, the nose, we can define the action as "raising right hand." This is how the pose-to-action step work.

3.3. Game interface

In this part, we will need to create the GUI of this game. Besides, it is important to integrate the StyleGAN 3.1 and human pose estimation 3.2 into this game.

3.3.1 Basic GUI

We use `pygame` module to implement this game and the main framework is refer to [1]. In the game GUI shown in Figure 7, we have several features such as health bar, background, music, character, etc. Two players can use keyboard to control the actions of the game characters. The actions include move, jump, attack, and defense. When the character is being attack, his health point will minus 10.

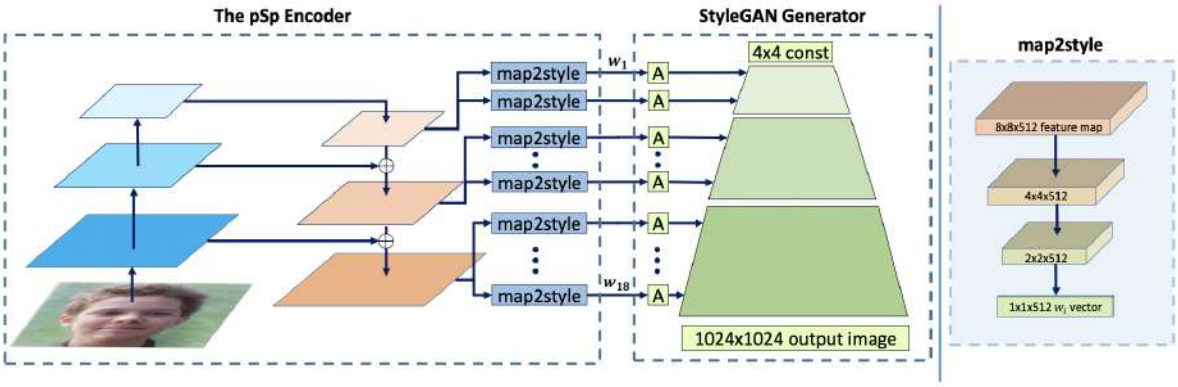

Figure 5. Style Encoder architecture

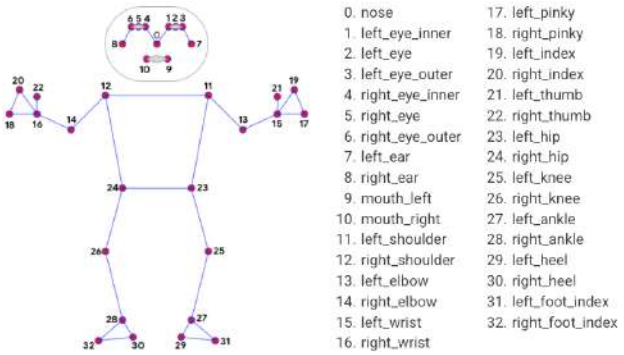

Figure 6. 33 key points predicted by mediapipe [7].

However, if the character successfully defend the attack, the health point will only drop by 2. Both the attack and the defense have a cool-down time for 1 second. Whenever the health drops to zero, the game will terminate and restart a new match.

3.3.2 Sprite

In order to let the game characters do an action on the screen, we drew a stickman sprite shown in Figure 8 on our own. In the sprite, each row represents an action. For example, the fifth row stands for "attack". When the player presses the attack button on the keyboard, the image in the fifth row will sequentially change from the first frame to the last frame rapidly. Therefore, we can see the attack action showing on the GUI.

3.3.3 Integration with human pose estimation

In this part, we need to replace the original keyboard input with the result from the human pose estimation. For the implementation detail, the actions come from the players will be stored in a boolean list and sent to a function in the game interface. When any element of the list equals to True, we will regard it as the same meaning of having a keyboard input. In this way, we can successfully integrate with the human pose estimation.

3.3.4 Integration with StyleGAN

Before the game starts, we will receive a sequence of style images pre-calculated by StyleGAN. With these images, we will sequentially put them on the game character's face. Throughout this game, whenever the health point of the character drop by 10, the image will switch to the next one. Therefore, we achieve the effect that the game character's face will gradually change from the player's selfie to the target image.

4. Experiment

4.1. StyleGAN

In this section, we will present the result of the style fusion of the source image and target image in Figure 9. The fusion image is present in Figure 10. According to this result, there are several points that we can discuss.

As shown in Figure 10, the style interpolation of the reconstructed source image and the reconstructed target image is very successful. We can actually see the hairstyle, faces and background of the reconstructed source image gradually fuse into those of the reconstructed target image with smoothness. The result proves that interpolation of latent code in latent space actually leads to style fusion of image space.

According to Figure 9 and 10, the reconstruction of the source image and the target image is quite a failure. After some discussion, we find out some points that may lead to this undesirable result. These points are listed below:

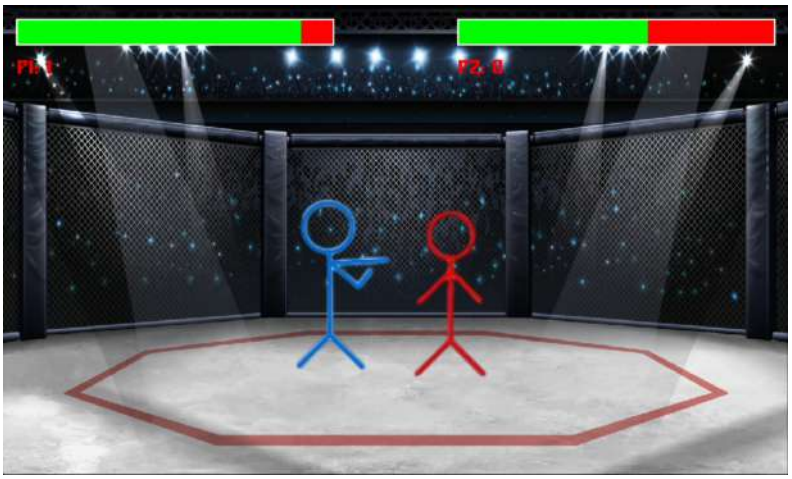

Figure 7. GUI of our fighting game

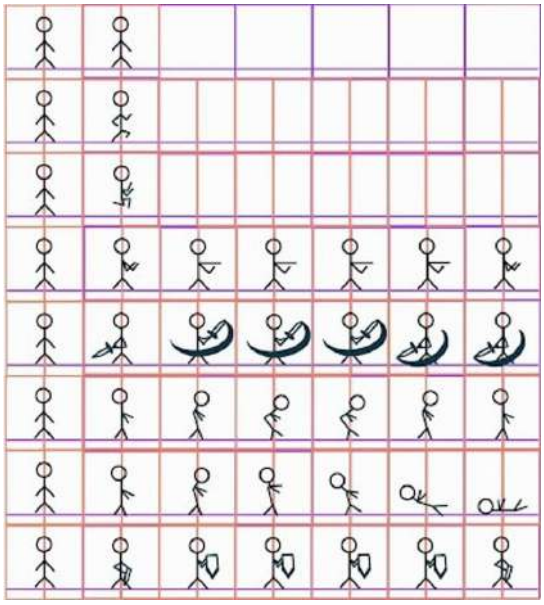

Figure 8. Sprite for animation

4.1.1 Face position problem

During implementation, we find that the whole reconstructed image will change greatly if we slightly scale the original image. Figure 11 can fully explain this problem. We divide the four images in Figure 11 into two groups. The leftmost two pictures are one group, and the rightmost two pictures are another group. Both groups of image contains original image(left one) and reconstructed image(right one). The two images of left part of Figure 11 is the same as left part of Figure 9 and upper left of Figure 10. We just cut a little fraction below the neck on the original image Figure 9, the reconstructed image varies a lot (shown in right part of Figure 11).

To be more specific, compared to the reconstructed image that does not cut, the reconstructed image has more hair and wrinkles. The hair and wrinkles, however, have nothing to do with the neck (cut place). We attribute this result to the fact that GAN-based models are inherently very hard to control the generated results.

4.1.2 Origin goal problem

The main goal of StyleGAN-based model is to generate diverse results using latent code, while our goal is to reconstruct original image using latent code, which is totally opposite. This might be a potential reason of our bad result.

4.1.3 Domain gap problem

The style encoder is trained on FFHQ dataset, which consists mostly of Caucasian and African. Our source and target images are both Asian, which may lead to a domain gap problem.

4.1.4 Latent code space problem

Losing information is inevitable when encoding the high dimensional images to low dimension latent vectors. Thus, we will lose considerable information when we perform style encoding. As a result, StyleGAN generator cannot generate images similar to original images.

4.2. Human Pose Estimation

The visualization result of human pose is shown in Figure 12.

Figure 9. left: source image, right: target image

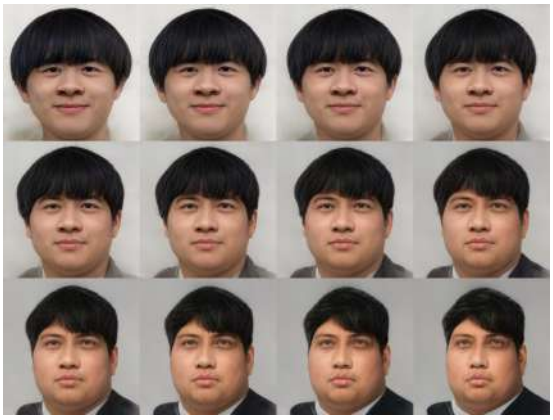

Figure 10. Style fusion between source image and target image, upper left corner is the original reconstructed source image; while lower right corner is the original reconstructed target image

Figure 11. face position problem

Figure 12. pose result

4.2.1 Implementation Detail

We conduct an experiment on the accuracy of real-time human pose. There are left person and right person. Each action we do 10 times.

Action	Correct (L)	Correct (R)	Accuracy (%)
raise left hand	10	10	**100**
raise right hand	10	10	**100**
punch left	10	10	**100**
punch right	10	10	**100**
hands together	8	7	75

Table 1. Evaluation table for pose2action conversion. `correct (L)` and `correct (R)` stand for the number of correctly detected case for left person and right person in the game interface respectively. We test each action for 10 times.

4.2.2 Evaluation Metric

The result is shown in Table 1. We got an excellent performance in hand raising hand and punching, which achieves 100% accuracy. As for the last action "hands together", the accuracy is less than other. The main reason is that when we perform this action, the key point position of our hands will get very close, leading to self-occlusion. Under such condition, we observe that the prediction result of mediapipe becomes more unstable, so the average accuracy drops to 75%.

4.3. Game Interface

After integrating with the human pose estimation and StyleGAN, we can use our body motions to control the game character and show the corresponding animation on the screen. Also, when the game character is being attacked, the face of the character will indeed change to the next style image. Finally, we have a live demo on the oral presentation, and the game is working perfectly from it starts to it ends.

5. Conclusion

We successfully combine StyleGAN and human pose estimation in an interesting fighting game. Our pose classification achieves excellent performance in most poses that we define. However, using the StyleGAN-based model to reconstruct arbitrary real-face images is still challenging. In the future, we can try other face morphing methods to make our results better. Also, defining much more actions will make the game even more interesting.

References

[1] Adrian Adewunmi. Street fighter clone made in python. https://github.com/russs123/brawler_tut. 3

[2] Hongliang Jiang Kaiheng Weng Yifei Geng Liang Li Zaidan Ke Qingyuan Li Meng Cheng Weiqiang Nie Yiduo Li Bo Zhang Yufei Liang Linyuan Zhou Xiaoming Xu Xiangxiang Chu Xiaoming Wei Xiaolin Wei Chuyi Li, Lulu Li. Yolov6:

A single-stage object detection framework for industrial applications. *arXiv:2209.02976*, 2022. 2

[3] Manu Mathew Deepak Poddar Debapriya Maji, Soyeb Nagori. Yolo-pose: Enhancing yolo for multi person pose estimation using object keypoint similarity loss. *arXiv:2204.06806*, 2022. 2

[4] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. *Advances in Neural Information Processing Systems*, 33:12104–12114, 2020. 2

[5] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 4401–4410, 2019. 1, 2, 3

[6] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8110–8119, 2020. 2, 3

[7] Camillo Lugaresi, Jiuqiang Tang, Hadon Nash, Chris McClanahan, Esha Uboweja, Michael Hays, Fan Zhang, Chuo-Ling Chang, Ming Guang Yong, Juhyun Lee, et al. Mediapipe: A framework for building perception pipelines. *arXiv preprint arXiv:1906.08172*, 2019. 1, 2, 3, 4

[8] Elad Richardson, Yuval Alaluf, Or Patashnik, Yotam Nitzan, Yaniv Azar, Stav Shapiro, and Daniel Cohen-Or. Encoding in style: a stylegan encoder for image-to-image translation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 2287–2296, 2021. 2, 3

[9] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv preprint arXiv:2207.02696*, 2022. 2, 3