

Using CGANS to generate a game character in the desired pose

Raghav Moar (CH16B045)

Introduction:

Making a character act in a game is a tedious and time-consuming process. Currently, each frame in a game video is done by professional artists and to achieve realistic pose its a tough job. We want to automate this task. This project will enable the user to transfer whatever pose they want to the game character which otherwise involves a lot of finesse and experience. Our network will detect those poses and will create a game character enacting those poses. In this project, we only model and focus on generating poses for a single game character

Background literature:

In the paper Everybody dance now by Caroline Chan et.al^[1], the group had focused on transferring the pose of a source video usually consisting of a professional dancer to a target video consisting of people with no experience in that field. However, that was limited to just a human target. We will extend it to game figures. We will also use the Openpose model and the Conditional GAN (CGAN) structure from pix2pix mentioned in the paper. However, we will not do multiple screen predictions by Generator but just a single one due to time constraints.

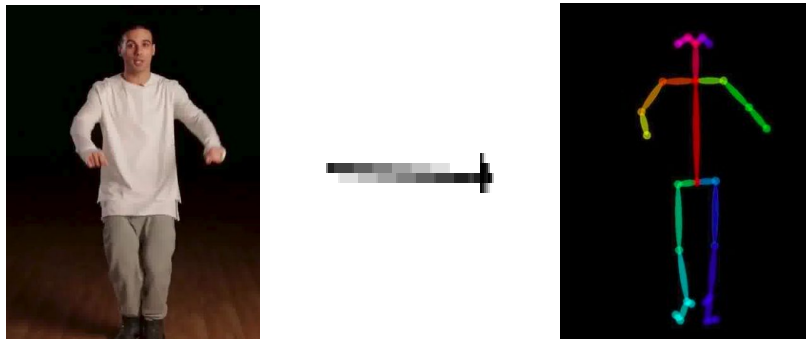
Method used

To solve the problem statement, we needed to convert a pose from domain $f(x)$ to the desired game character in that pose $g(x)$. Where x is the desired pose.

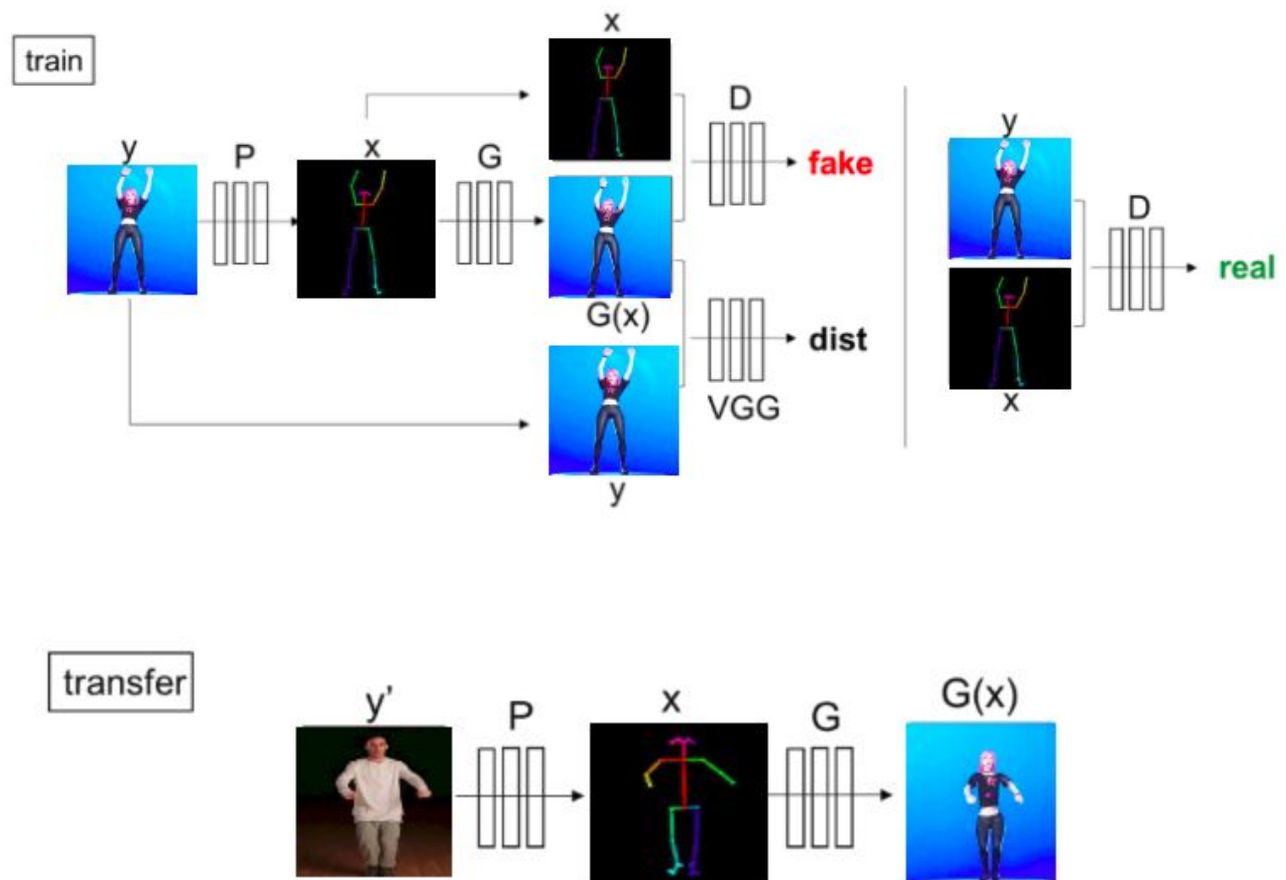
The problem is broken into two parts:

- 1) Pose estimation to procure x
- 2) Using CGAN model to transform x to desired game character pose

The pose estimation was successfully done by using the Openpose algorithm developed by CMU.



We have used the following architecture called pix2pix which is a type of CGAN developed by NVIDIA for the purpose of transforming the image from one domain to another. Using this step we successfully obtain $g(x)$



Implementation Details:

1) Data Collection & Cleaning:

A single video game (Fortnite) character (target subject) dancing for 27 min of frame rate 30 fps was collected from youtube. The part where the subject used various props like guitar etc while dancing was removed out (18 min left). Also, the transition frames were observed and removed. The subject was also centered in the frame. This successfully provided us with 31230 images.

2) Pose Estimation:

The target input video was feed to OpenPose API developed by CMU. The output video skeleton in that pose in a black background was obtained. Images were then extracted and resized to 512*512 (input for pix2pix). After pose estimation, there were still some undetected parts during pose estimation, which then were removed from both the videos. All video processing was done using FFmpeg API in Linux platform.

3) Training using pix2pix model:

We used the pix2pix model developed by NVIDIA to train on google cloud. It took 22 hrs for the model to complete 2 epochs. We couldn't train further due to constraints on GPU availability on google cloud platform. The model was trained using 5 types of training loss:

- i) Discriminator Real Loss (D_{real})
- ii) Discriminator Fake Loss (D_{fake})
- iii) Generator Loss (G_{gan})

iv) Generator Vgg loss (Compare Real vs Generated image) (G_vgg)

v) Generator feature loss (Comparison of feature map generated by Discriminator when the Real and fake image was passed) (G_feat)

Total Discriminator Loss= [D_real + D_fake] * 0.5

Total Generator Loss = G_gan + G_vgg + G_feat

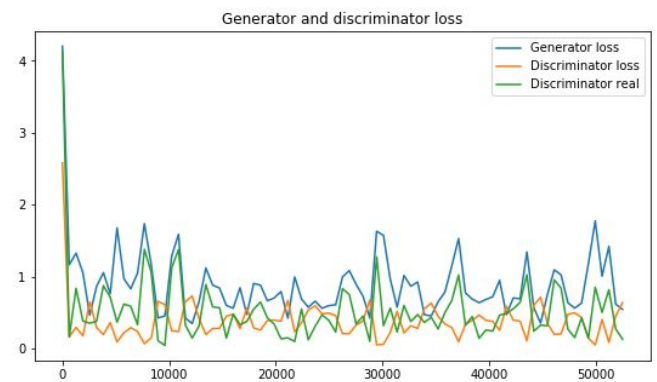
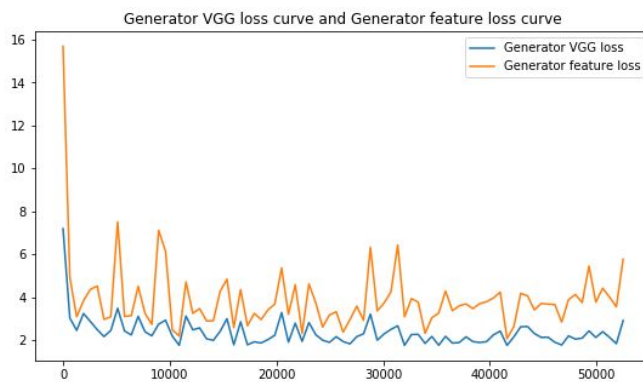
We keep Discriminator loss less otherwise it will overcome not allow the generator to learn.

4) Final Transfer from the given source image to target image

We use Openpose API to convert source image to the pose skeleton. Now using generator given the pose skeleton image we get our target character in that pose

Results

Loss curve



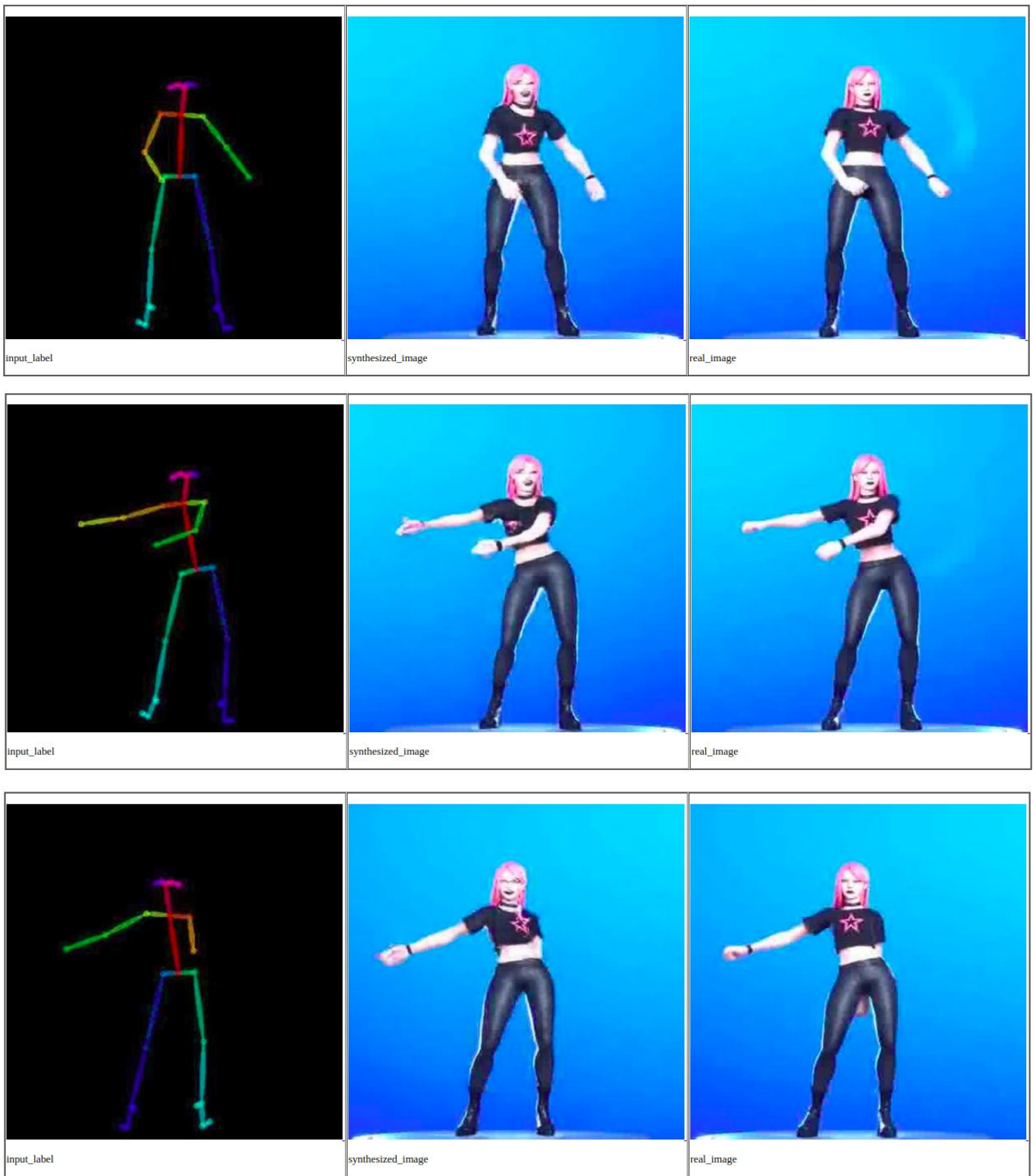
We notice that all the losses decrease as we move forward with iterations. The discriminator losses oscillate around 0.5 indicating a balance between discriminator and generator.

Cross-validation result

The model was tested on cross-validation set of 112 samples. We get the following results

| Type of loss | Mean Loss |
|-------------------------|-----------|
| Generator feature loss | 3.740 |
| Generator Vgg loss | 1.040 |
| Generator Loss | 0.666 |
| Discriminator Fake Loss | 2.048 |
| Discriminator Real Loss | 0.387 |

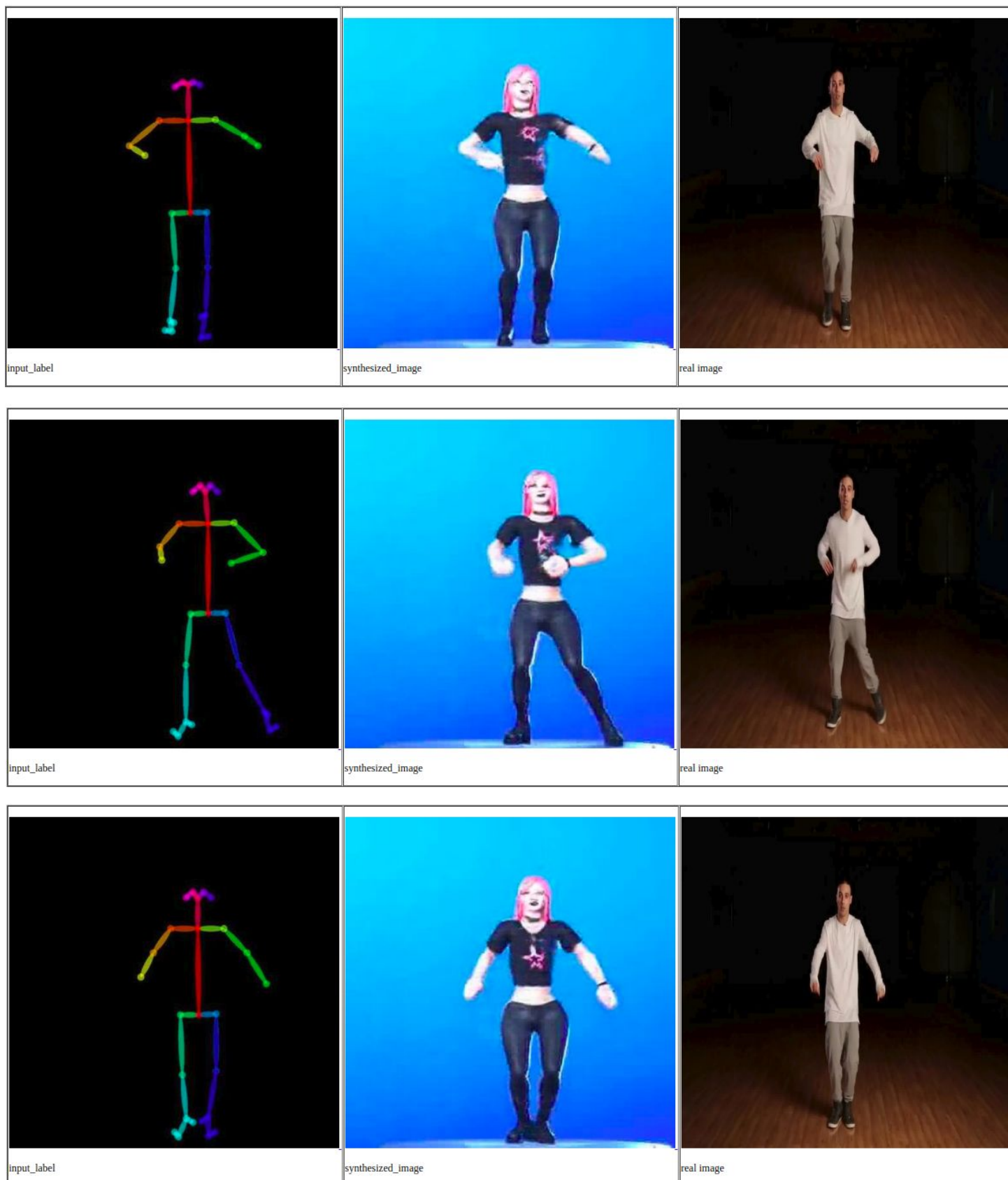
The losses are similar to losses on the training set. Below are some images generated on poses from cross-validation set



We can see that the model was able to capture different details like shirt, background and pose perfectly. However, the face and hands are not perfect and are really noisy. This may be because the pose input is not carrying any information about the facial expression and also there is no information about fingers from the pose causing noisy hand results.

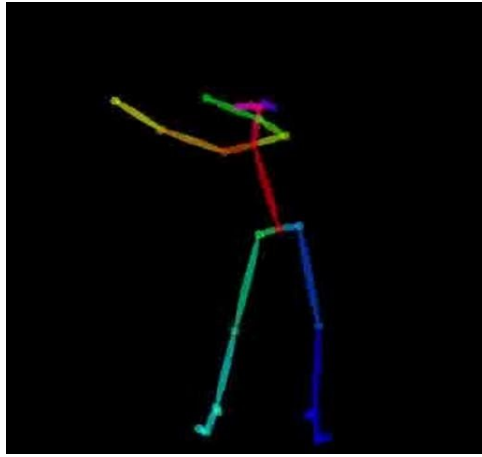
Test result

Our model was tried from starting to end to transfer images of the source image to target pose image. The model was also tried on the source video giving us good target video results.

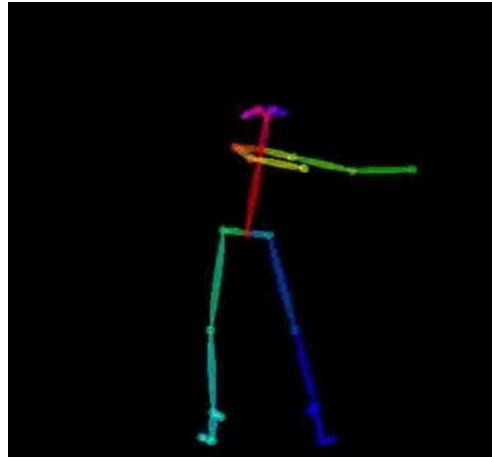


Analysis:

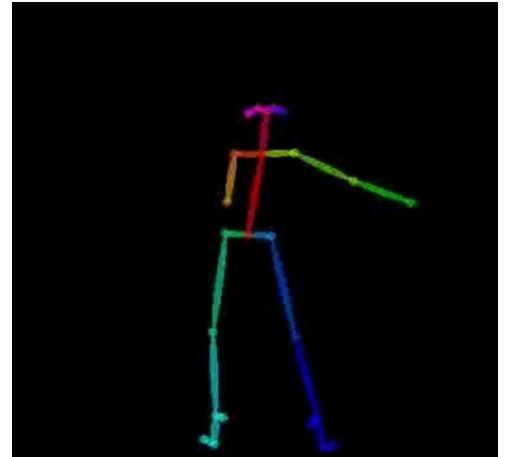
Negative Results



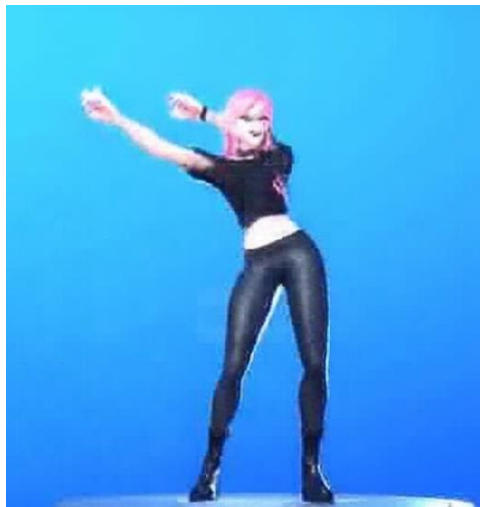
(i)



(ii)

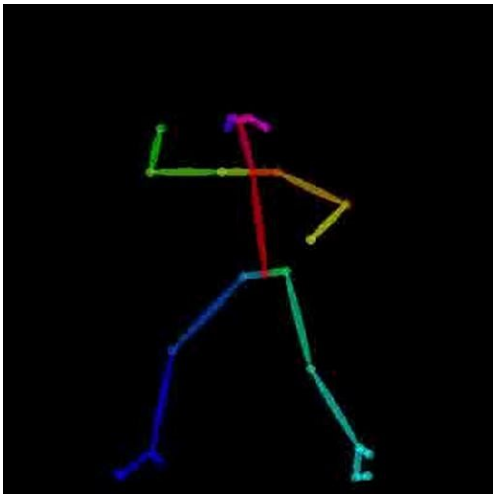

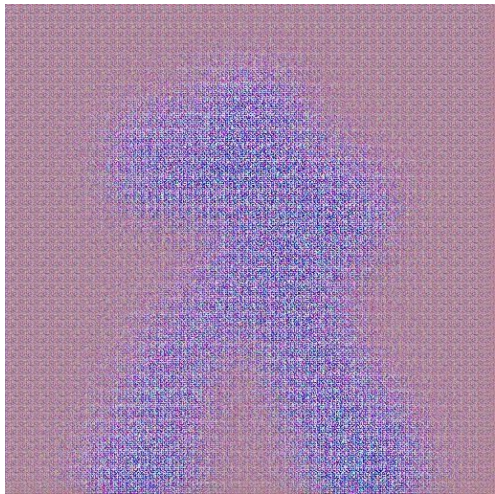
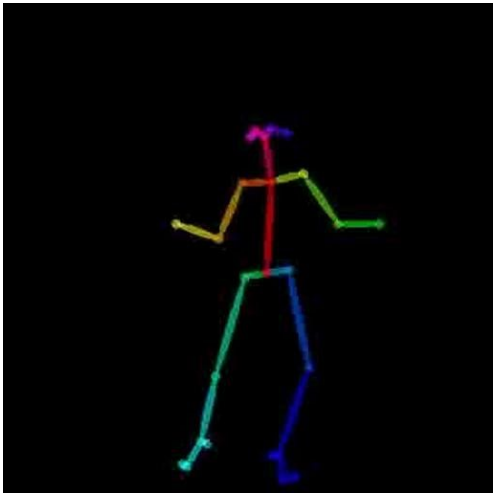

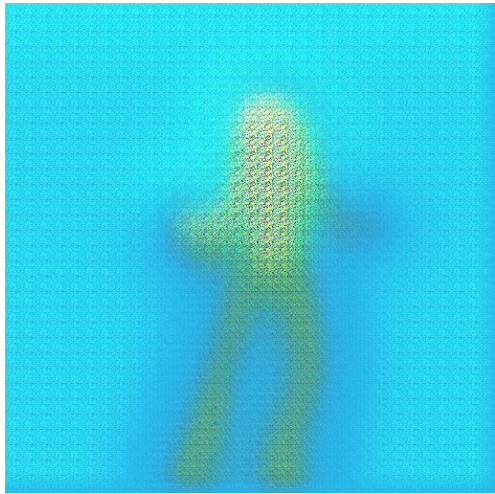
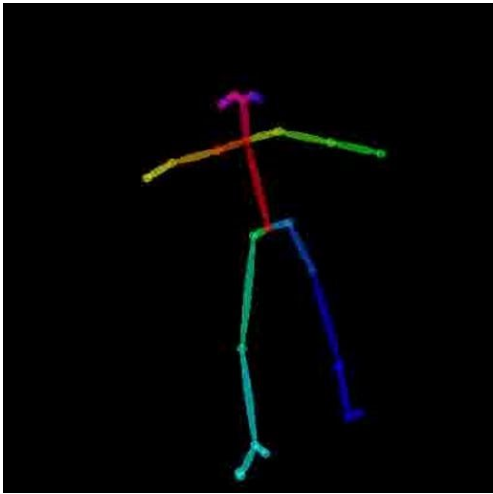




(iii)



- i) The star was not printed on the t-shirt
- ii) Overlap of hand was not registered
- iii) Due to incomplete pose estimation, the generated image has a missing hand

Initial Learning period:

| <u>Input Image</u> | <u>True Target</u> | <u>Generated Target</u> |
|--|--|---|
|  |  <p>Training iteration 1</p> |  |
|  |  <p>Training iteration 50</p> |  |
|  |  <p>Training iteration 100</p> |  |

We wanted to see how the pix2pix model starts to learn:

Observation:

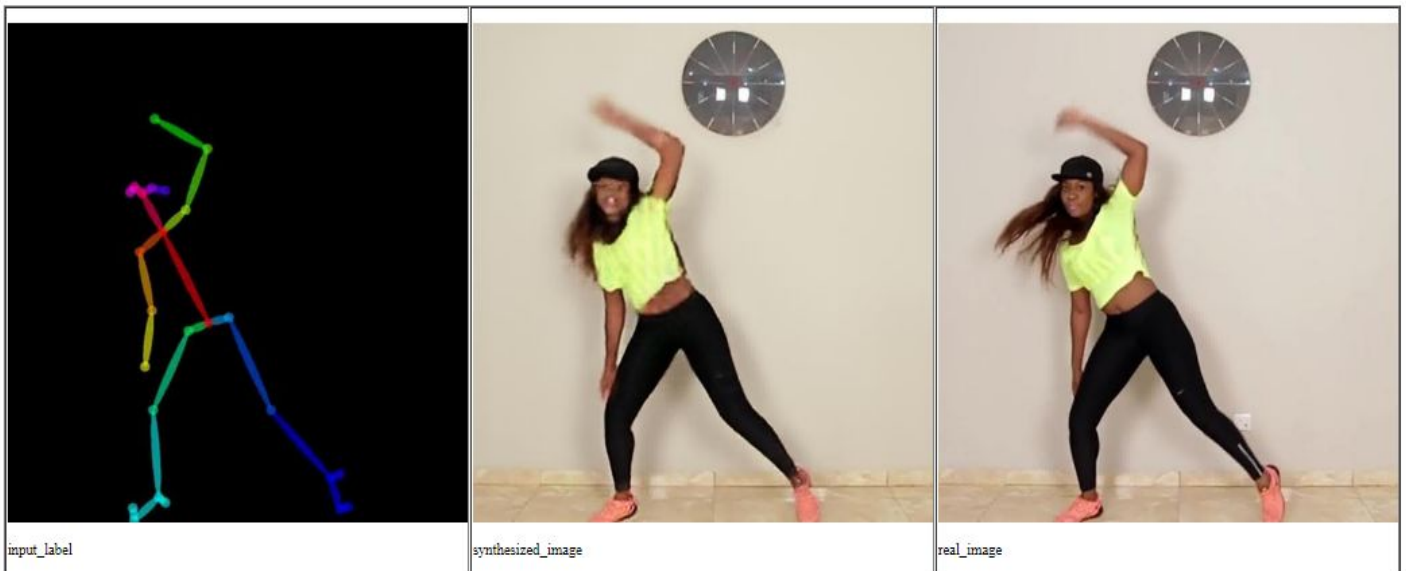
- 1) Initially, during the first learning iteration, the model produces a random image based on the initialized parameters.
- 2) The background is still separated in the first generated image because of input background color being black which is different from the stick figure
- 3) After 50 iterations we see that the model has learned to identify background color as blue and is able to localize the target figure clearly
- 4) After 50 more iterations, the model has learned to generate body parts related to each stick color in the figure.
- 5) It has also learned to generate a graded background.

PART B

HUMAN TO HUMAN TRANSFER

Training:

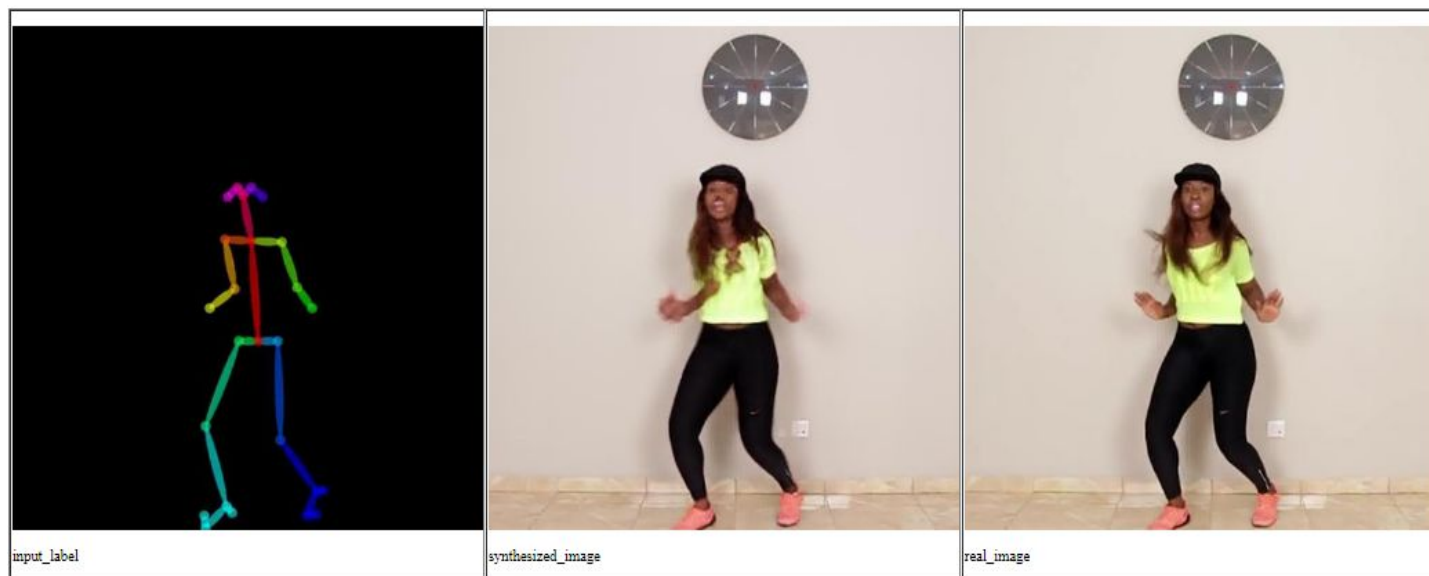
epoch [1]



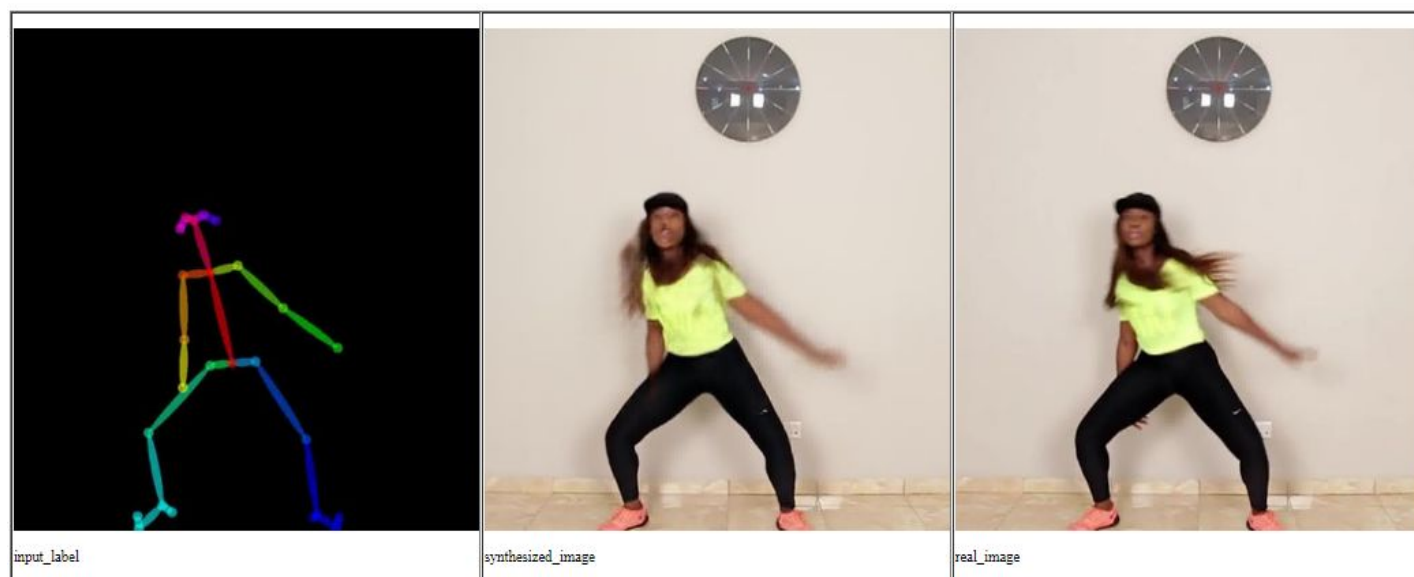
epoch [2]



epoch [4]



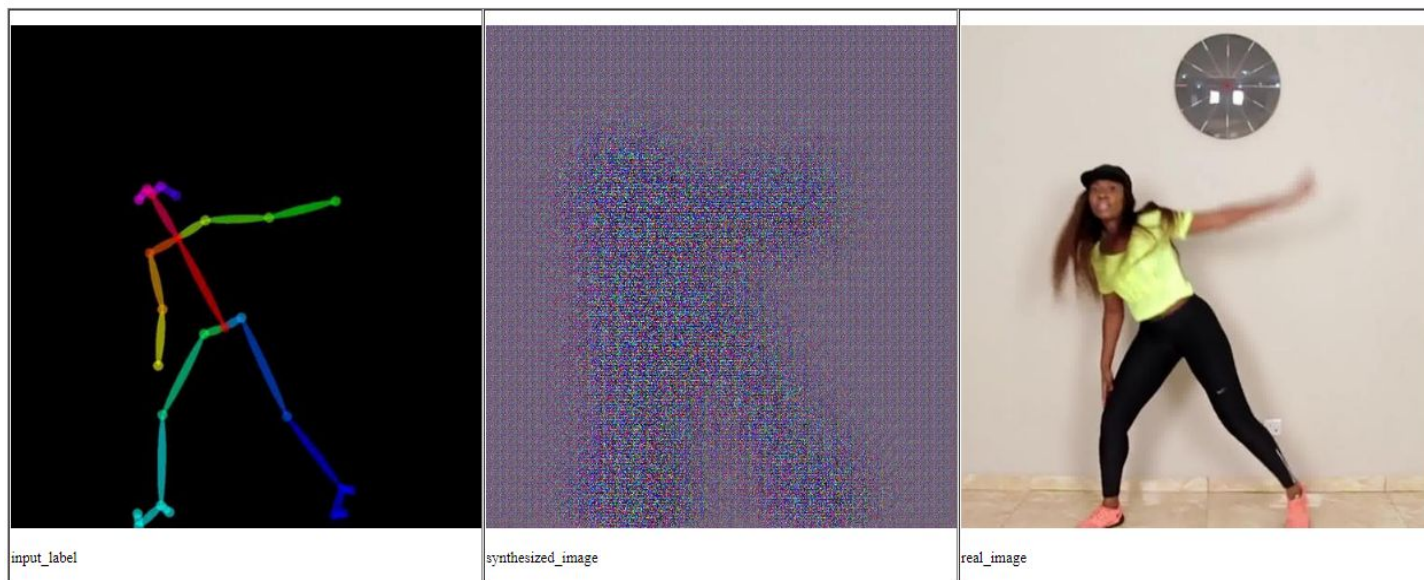
epoch [5]



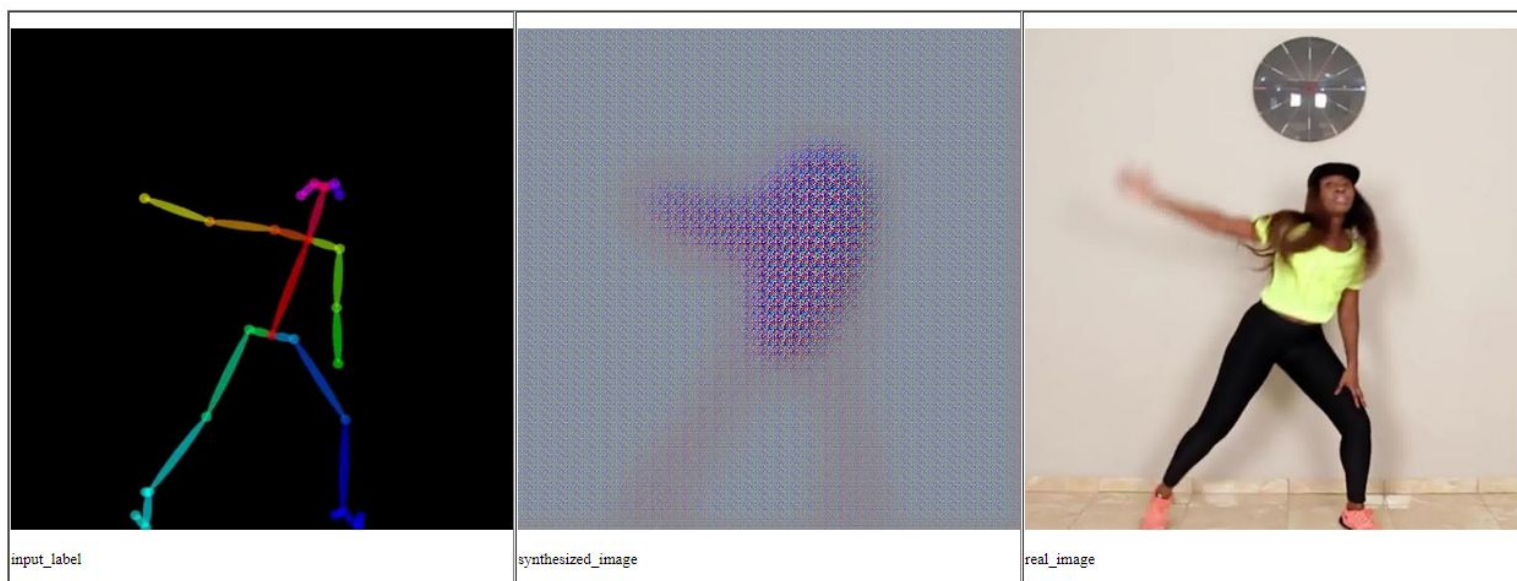
Initial Training

To see how is GAN learning the images:

epoch [1]



epoch [21]



epoch [36]



epoch [56]

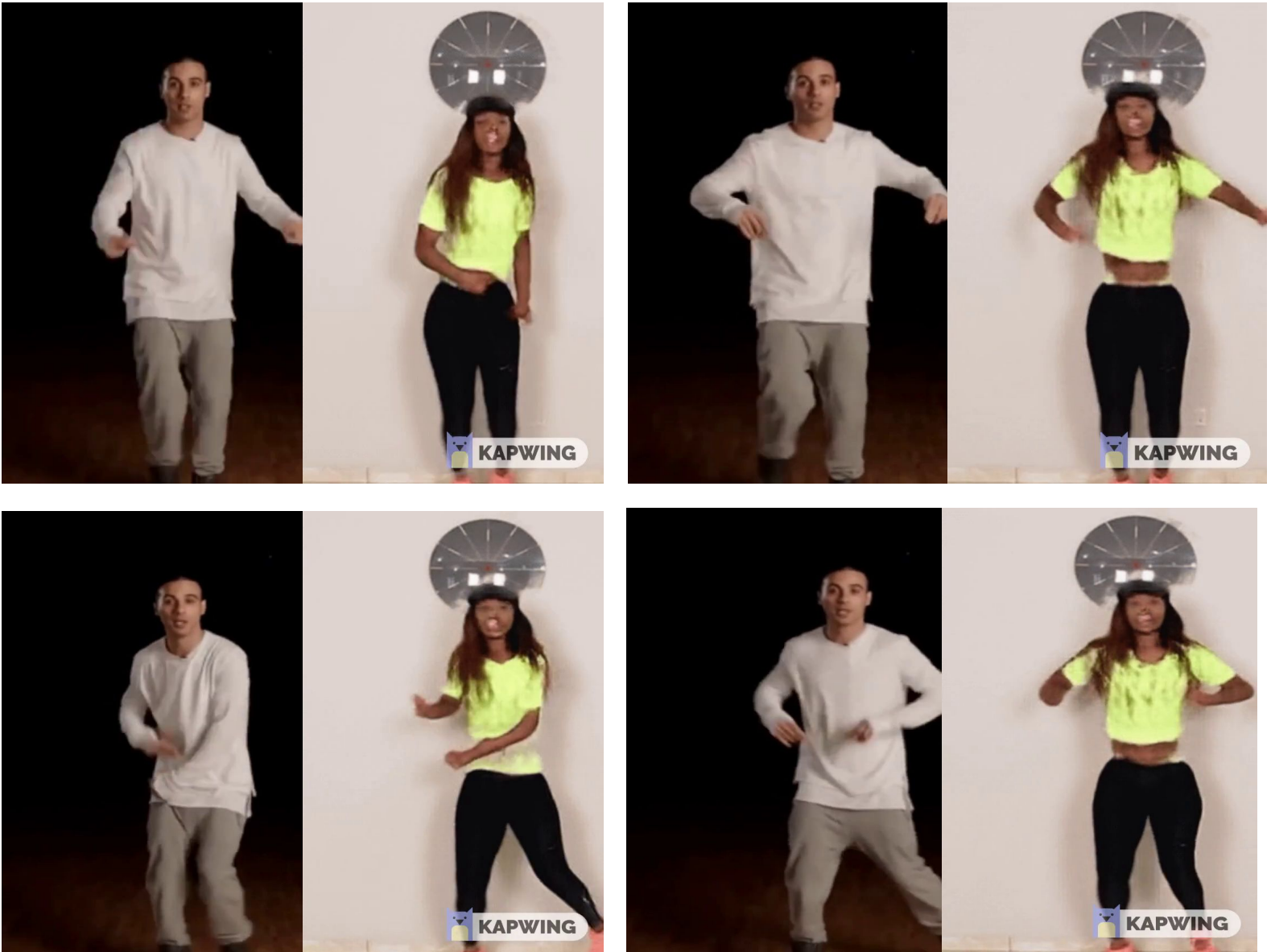


epoch [152]



Transfer Human to Human:

We transfer from a freestyle technique from a freestyle dancer on a Zumba dancer
Sample



Link for the video(to show transfer):

https://drive.google.com/open?id=1F9ijYSzb_F4c47Et5K2-zy6EXJ6Gyetk

References

[1]Everybody dance now: <https://arxiv.org/abs/1808.07371>

[2]Image-to-Image Translation with Conditional Adversarial Networks: <https://arxiv.org/pdf/1611.07004.pdf>