PA026 Project Report

Tomáš Repák

485329

1 Project Introduction

The goal of this course's project was to improve my already existing personal project **ArmorSets-GAN** [arm]. This project aims to generate new, not yet seen outfits for a character in the World of Warcraft game. World of Warcraft offers a large amount of "fashion" possibilities, which I enjoyed. Therefore, I decided to try to use a neural network to generate new armor pieces/armor combinations. The first version of the model was in my opinion pretty successful (although requiring about 14 days of training). There were some flaws in the generated pictures, such as unclear boundaries of the character, or double-faced characters as seen in Figure 1. I decided to try to remove these flaws in this course.

1.1 Project Description

1.1.1 Generative Adversarial Networks

The whole project is based on the concept of Generative Adversarial Networks (GANs). GANs were first proposed by Goodfellow et al. [GPAM+14] and became famous since then. The general concept of GANs is to use a system of two neural networks to learn how to synthesize realistically looking outputs from a randomly sampled *latent vector*.

To the best of my knowledge, GANs are not widely used in industry and serve more as an academic playground. There are, however, applications such as Wonder [won] or AI Art Maker [aiA], which can generate images from a given description. I can imagine Wonder having subscriber-only features, so some profit can most likely be made using GANs. I am not sure whether these applications employ GANs directly, but GANs can definitely be applied to such use cases, that is why I am mentioning it. The most prominent applications of GANs are for example NVIDIA's StyleGAN [KLA18], GAN for music generation [DHYY17] or several applications of GANs for DeepFake [She18].

There are several applications of GANs in terms of fashion generation, such as [clo, dee, RS20]. To the best of my knowledge, though, my model is the first to be applied to World of Warcraft characters.

Figure 1: Examples of the flaws of the first model - unclear character boundary on the left and double face on the right.

The underlying concept of Generative Adversarial Networks is the following. Assuming we are dealing with image data, there are two convolutional neural networks[1], so-called **discriminator** D and a **generator** G. As their names suggest, the goal of the discriminator is the following:

Lemma 1. *Given input i, the goal of the discriminator D is to classify the input i as either real or fake. That is, D(I) ∈ {"real", "fake"}*

The goal of the generator is the following:

Lemma 2. *Given a latent vector v of arbitrary size, generate an output in such a way, that makes the discriminator classify the generated output as a real one. That is, D(G(v)) = "real"*

This can be thought of as a competition between these two models. The improvement of the discriminator hinders the generator's performance and vice-versa. In theory, such a competition is called a *zero-sum game* and when the game reaches a stale state, where both players cannot improve their strategy, it is said that they have reached *Nash Equilibira*. In the settings of GANs, though, the convergency (equilibria) is usually never achieved. In addition, it is not clear whether this equilibrium even exists [FO20] or whether GANs can even be considered a zero-sum game [noz]. Although the theoretical research is not yet finished, it is without a doubt that GANs can work in some applications.

1.2 Dataset Description

Since in my subjective opinion, the best-looking characters in World of Warcraft are the Zandalari Trolls, I decided to use them as the character body shape for which I will generate the transmog sets.

I have sat down and manually collected 1046 screenshots of 3D models from WoWHead's Transmog section, filtered to results containing only male Zandalari Trolls [wow]. [2] The images can be found in the attached zip file. The dataset is also available for download on the project website [arm].

In this course, I have extended the previous dataset containing about 800 examples to today's 1046. The images that are being collected are user-made and thus there are new fashion sets being created over time. Therefore, it is optimal to extend the dataset from time to time and the dataset can never be complete.

The caveat of the dataset that negatively influences the model's training process is the WoWHead logo (See Figure 2), which is present in the background, and that the background is of a dark color, which often blends with the character with no clear boundaries. As of this course, I aimed to mitigate these issues.

[1] In general, the underlying models can be any models (Support Vector Machines, Decision Trees, etc.)

[2] In the dataset zip file, there are more images, because I have initially collected outfits also for other races than Zandalari Trolls. Only Zandalari Trolls are used for training, though.

Figure 2: WoWHead logo in the background after background removal

2 Implementation

2.1 Tools used

- Python 3.7.0
- Tensorflow 2.7.0
- tqdm 4.31.1
- opencv-python 4.5.4
- CUDA 11.5.1_496.13
- cuDNN v8.2.2 (July 6th, 2021), for CUDA 11.4
- NVIDIA GeForce RTX 2080

2.2 Model

When starting with a new project from a field I never tried, I have closely followed the following tutorials [tuta, tutb]. These tutorials contain the most important ideas and basic model definition used in my project. After getting it to work somehow, I added tweaks from other sources, such as Label Smoothing, Label Noise, Label Switching, Gaussian Noise layers, Dropout, etc. Overall, these improvements mostly aim to regularize the discriminator. Some GAN models employ an unbalanced training schedule - A generator is trained more often than the discriminator - but that is not employed in my project.

The final model architecture is summarized in Figures 3 4 and 5. As an input to the generator model, a 100-dimensional latent vector is provided. This vector is fed through a Dense layer, which is then reshaped into a quarter of the final image (Thus, the height and width dimensions of the output image have to be divisible by 4). This small image is fed through two regular convolution operations before applying two consecutive Conv2DTranspose operations, which are each equal to x2 upsampling. LeakyReLU is used throughout the whole model combined with the tanh activation function. There are no pooling layers present. In the initial model, there were GaussianNoise layers added before each convolution operation to improve generalization. These are not present in this course's model. The output from the generator is fed into the discriminator part of the GAN.

The discriminator is a simple classification Convolutional Neural Network. The last layer is activated using the Sigmoid function and Binary Cross Entropy is used as a loss function.

The model developed during this course was trained for 1350 epochs using a learning rate of 0.0001, which was lowered to 0.00005 after 1000 epochs.

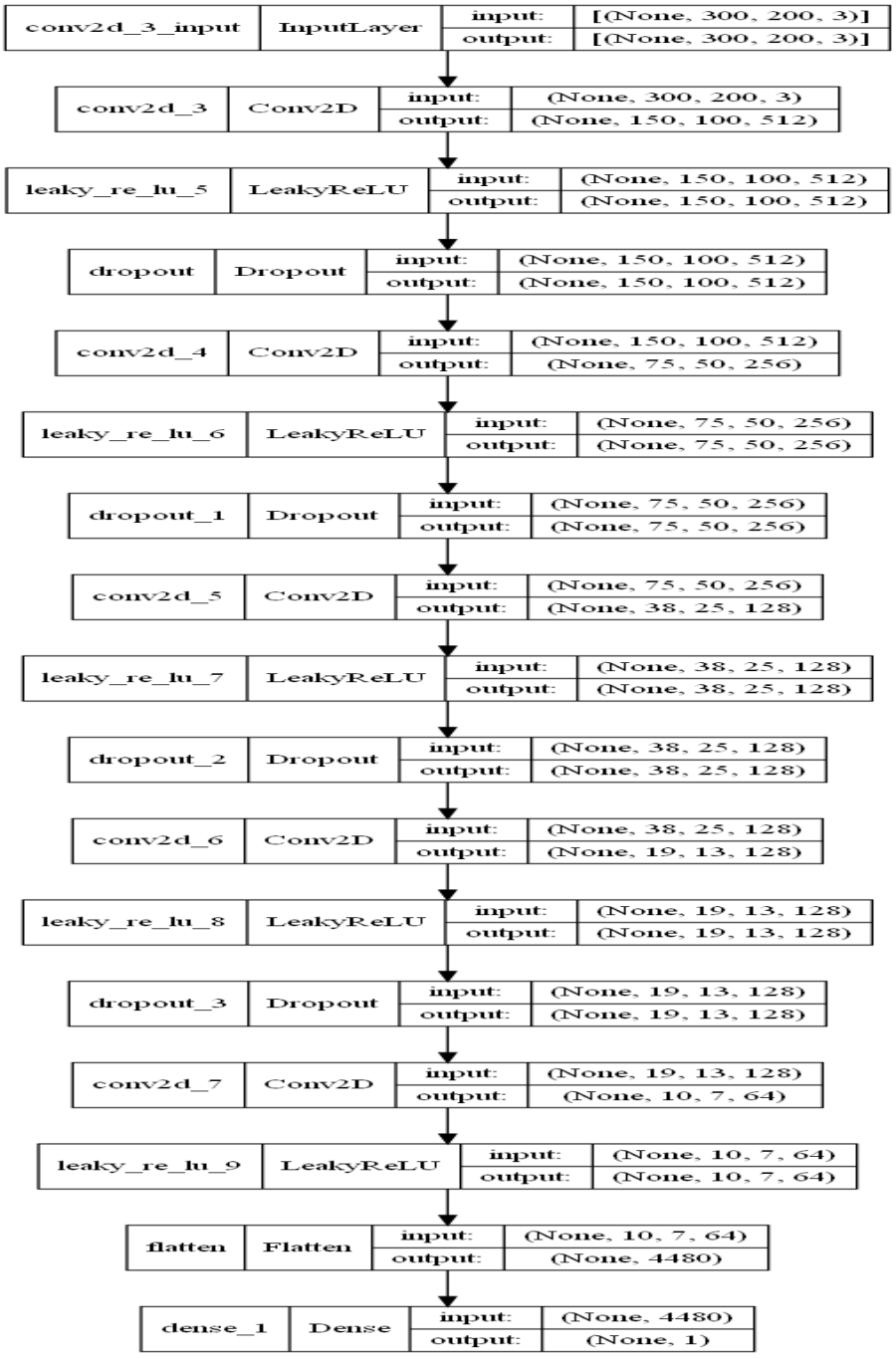

Figure 3: Discriminator model architecture

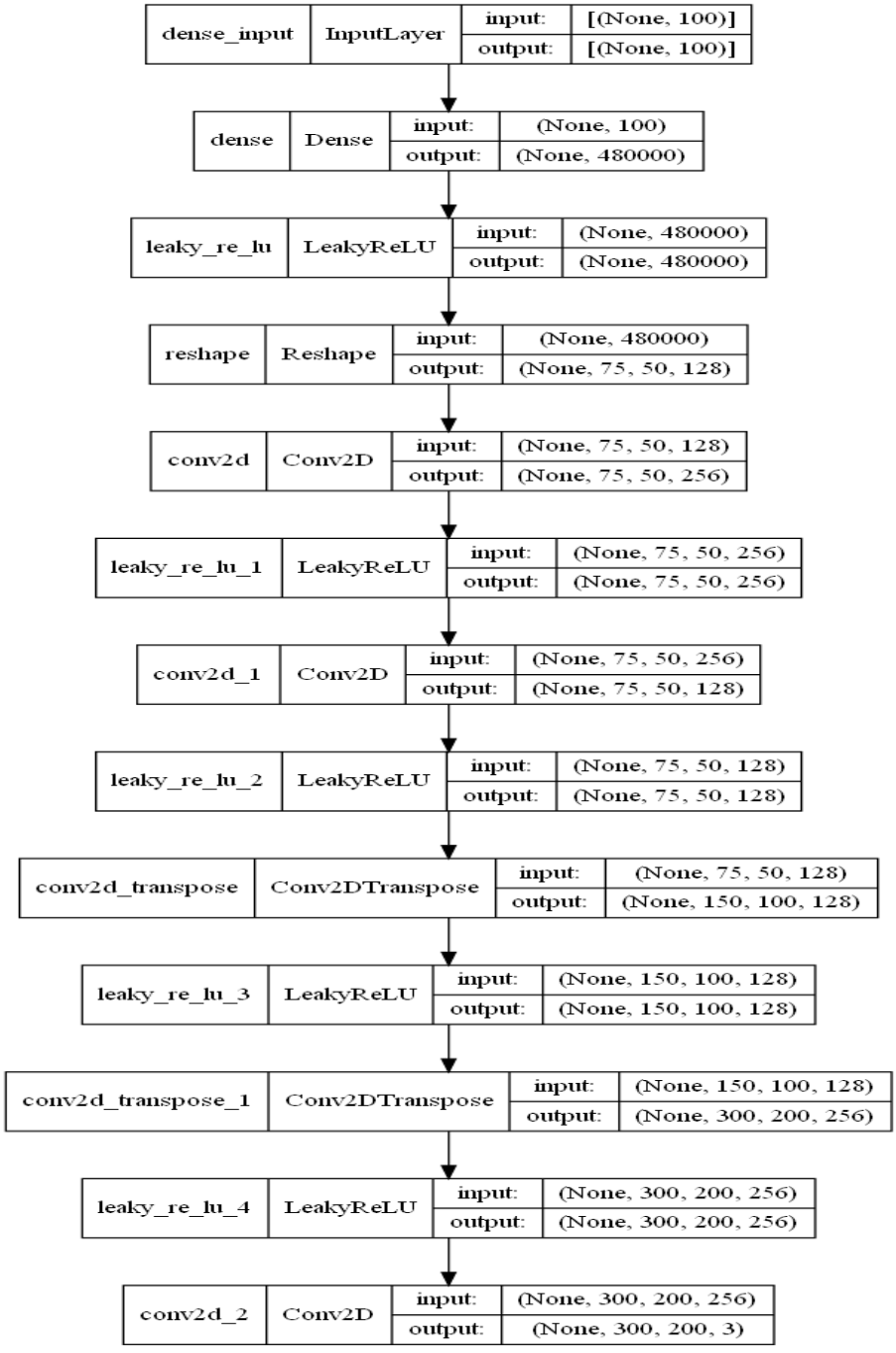

Figure 4: Generator model architecture

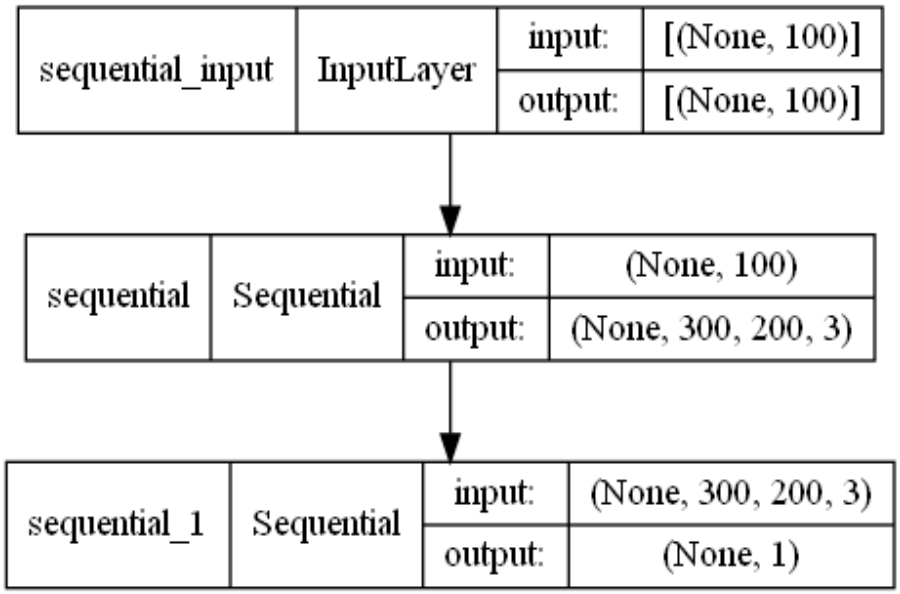

Figure 5: GAN architecture

2.3 Preprocessing

The preprocessing of training examples for my model is fairly standard. During the training phase, only images containing Zandalari Trolls (files ending with "_ZT") are loaded. After loading, each image is resized to the model's input size (300x200) and normalized to the [-1, 1] range. After this course, additional steps introduced in the next subsection are also applied. After all preprocessing is done, recolor augmentation is applied. This is done by swapping the RGB channels of the image. This results in doubling the training dataset size to 2092 examples. All the training examples, including the preprocessed ones, are attached to this project's submission.

2.4 Preprocessing implemented in this course

The aim of this course was to improve the quality of the resulting outputs. While higher quality outputs can most likely be achieved by increasing the model complexity (Number of layers, number of filters per layer, output image resolution, ...), it is not feasible in terms of my computational resources. The current model's settings and image size of 300x200 pixels is close to reaching the maximum memory available in the RTX 2080 GPU RAM (8GB) even with batch size 1.

Since I had to look elsewhere than increase the complexity, I focused on improving the training data. Since one of the issues with the initial model was that the outputs had unclear character boundaries, after discussion with

```python
def merge_original_and_largest_region(orig, largest_reg, use_relative_size_threshold=False):
    # Extracting largest components succeeds at removing the wowhead logo in the background
    # But also removes some connected areas within the character. Mitigate this issue by the
    # Following code. Take the original image with its background subtracted, along with the
    # extracted image. If these two images disagree in the pixel value, keep the original one,
    # if it is not background color (x, x, x)
    for i in range(largest_reg.shape[0]):
        for j in range(largest_reg.shape[1]):
            img_tup = (largest_reg[i][j][0], largest_reg[i][j][1], largest_reg[i][j][2])
            orig_tup = (orig[i][j][0], orig[i][j][1], orig[i][j][2])
            if img_tup == (255, 255, 255) and (orig_tup != (255, 255, 255) and len(set(orig_tup)) != 1):
                largest_reg[i][j] = orig[i][j]

    return largest_reg
```

Figure 6: Merging Algorithm

doc. Horák, I have tried to change the background color to a white one, so that the model has to learn clear boundaries. Removing the background makes sense since all of the training images are acquired in the same way, on the same background [3].

Removing the background color results in a Figure 2. We can see that there are lots of WoWHead logos scattered around the main character. This WoWHead logo is comprised of several RGB colors, ranging from [1, 1, 1] up to [35, 35, 35]. Removing all of these possible colors resulted in a severely degraded main character, so another way was necessary.

Noticing that the character is always the biggest object in the image, I used the OpenCV2 library to extract the largest closely connected component from the image. As expected, the largest closely connected component was always the main character. Refer to Figure 7.

As you can notice, there are many "holes" in the largest closely connected component, so I introduced a way to fill in these holes correctly by merging the largest closely connected component with the image with a subtracted background, where the holes are still filled.

Since the WoWHead logo is comprised of RGB values of repeated elements (ie. [2, 2, 2], [18, 18, 18], etc.), we can use this knowledge to our advantage in the merging algorithm.

The merging algorithm is presented in Figure 6. Before the merging, there are also image morphology operations applied - Image opening and Image closing for overall image improvement.

[3] Except 2 manually added examples of my personal transmog in-game. These images are removed when this advanced preprocessing is applied.

Figure 7: Training image after extracting the largest closely connected component

Figure 8: Preprocessing process with merging

2.5 How to run the generator model and directory description

Please refer to the README_PA026 file attached.

3 Evaluation

Evaluation of GANs is not an easy task. There have been several metrics invented, such as the Frechét Inception Distance, but in my opinion, a numeric result does not convey much information in my use case. It is generally known that humans can perceive and evaluate images better than one number. I preferred to inspect all of the results on my own and decide whether they were good or not. To be honest, I expected the model to perform better and I am not at all satisfied with the achieved results. That being said, I am planning to expand my attempts on improving the project when I have some spare free time.

The training of the model started promising. The model was able to learn to define clear character boundaries faster than the initial model. The euphory faded over time, as I started to notice that the model was not improving. For instance, comparing the outputs from both the previous and the updated model in epoch #775 (See Figure 9), we can see that the previous model was already able to grasp the resemblance of the troll's face. Generating the face became a huge problem for the updated model. Even after the full 1350 epochs, the model was not able to produce even a slight resemblance to the troll's face. In fact, it seems as if the network specifically refused to learn the face, see Figure 10.

Regarding the issue from the initial model with doubled-face, this issue simply becomes irrelevant as the model is unable to learn to generate even a single face.

Overall, I am glad that I could extend my previous work. I am sure that the extended dataset will be of good use when I will expand the project even further. I am disappointed with the results achieved in this course, but I am happy that I was able to (to a great extent, but not completely) solve the background removal problem.

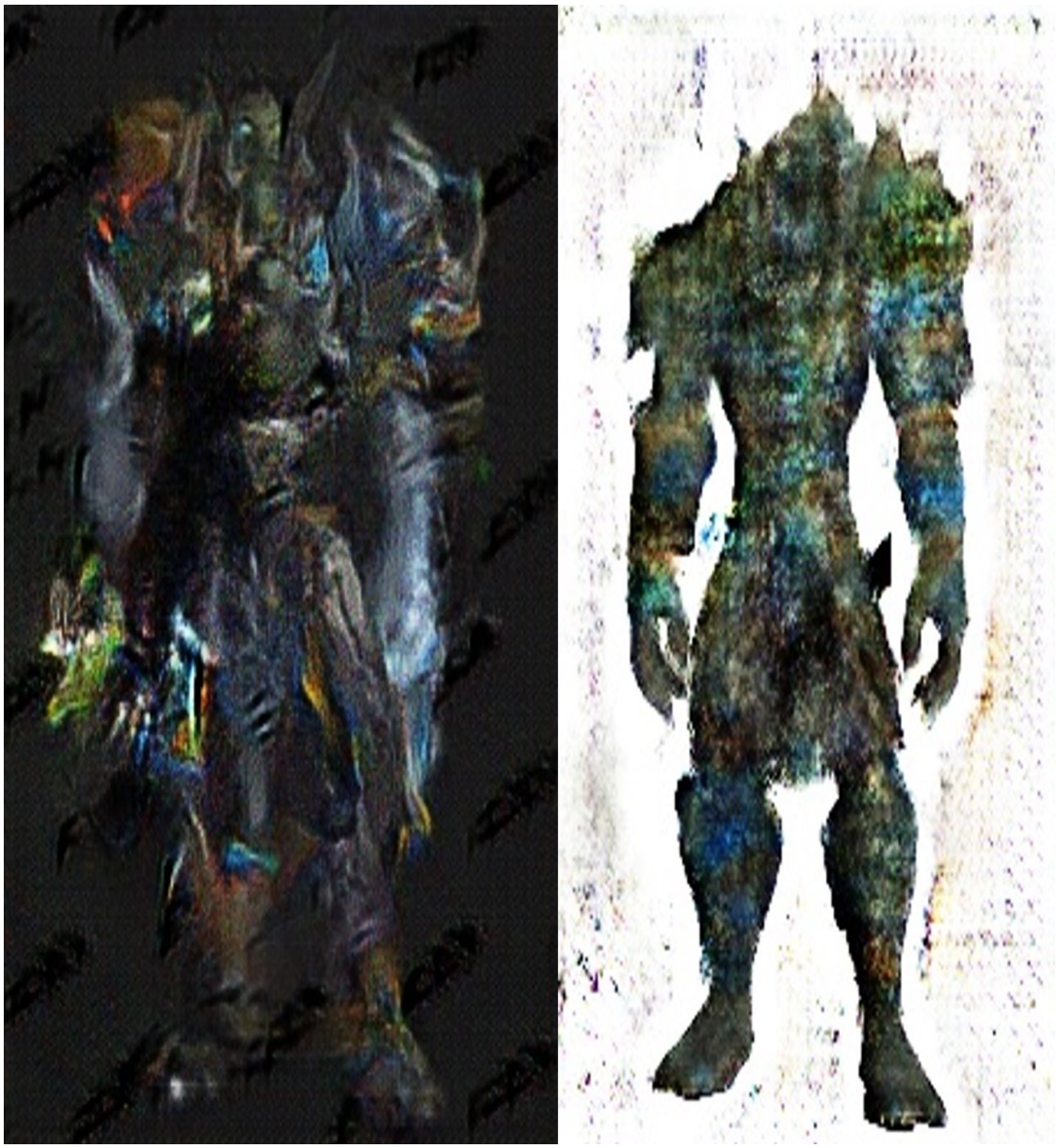

Figure 9: Comparison of the 775-th epoch of both models. Previous model's output is on the left, the updated model's output is on the right.

Figure 10: Output of the 1350-epoch updated model. We can see that especially the face has not been created.

4 Space for improvement

When I will have some spare time, I would like to improve the project even further. Some of my ideas for improvement are listed below.

- As always, extend the training dataset.
- As doc. Horák suggested - Try to tweak the contrast or increase brightness in the training images.
- Try to use visual transformers - encode the training images into a lower-dimensional vector and then reconstruct the full image back.
- Try to use conditional-GAN - or more specifically, annotate the training images with several features, such as the class of the character (Rogue, Warrior, Mage, Priest, Shaman, Monk, Death Knight, Warlock), class of the armor (Leather, Cloth, Plate, Mail), and other features.
- Try to use a more complex model when I get to upgrade my PC with a more powerful GPU.
- Probably the most complex idea was to somehow combine GANs for individual parts, ie. train a GAN for generating only faces, train a GAN for generating only weapons, and so on.
- Another idea is to fixate the appearance of several items and generate only combinations of already existing items (The most similar to the actual game principle), meaning the output would be a vector, ie [weapon_id, helmet_id, chest_id, ...]

References

[aiA] https://hotpot.ai/art-maker.

[arm] https://github.com/fapannen/armorsets-gan.

[clo] https://devpost.com/software/clothinggan.

[dee] https://towardsdatascience.com/deepstyle-part-2-4ca2ae822ba0.

[DHYY17] Hao-Wen Dong, Wen-Yi Hsiao, Li-Chia Yang, and Yi-Hsuan Yang. Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment, 2017.

[FO20] Farzan Farnia and Asuman E. Ozdaglar. Gans may have no nash equilibria. *CoRR*, abs/2002.09124, 2020.

[GPAM+14] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[KLA18] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks, 2018.

[noz] https://app.oxfordabstracts.com/events/1297/submissions/180352/question/22752/programme-builder/download.

[RS20] Amir Hossein Raffiee and Michael Sollami. Garmentgan: Photo-realistic adversarial fashion transfer, 2020.

[She18] Tianxiang Shen. Deep fakes " using generative adversarial networks (gan). 2018.

[tuta] https://machinelearningmastery.com/how-to-develop-a-generative-adversarial-network-for-an-mnist-handwritten-digits-from-scratch-in-keras/.

[tutb] https://towardsdatascience.com/image-generation-in-10-minutes-with-generative-adversarial-networks-c2afc56bfa3b.

[won] https://apps.apple.com/us/app/wonder-ai-art-generator/id1621278575.

[wow] https://www.wowhead.com/outfits/race:31/gender:0.