Personalized Avatar Generation through Text Description with Adversarial Conditional Generative Networks (ACGAN)

Ziyang Xu
Northeastern University, Boston, MA

Abstract

In this paper, we propose a method for generating personalized avatar images from textual descriptions utilizing the Adversarial Conditional Generative Adversarial Network (ACGAN) architecture. The ability to create custom avatars from textual descriptions has numerous applications in virtual environments, gaming, social media, and personalized digital content creation. Our approach leverages the power of deep learning to translate textual descriptions into visually realistic avatar images, capturing intricate details and nuances specified in the input text. We describe the architecture of our ACGAN model, detailing the training process and the incorporation of conditional information from text descriptions. We also discuss the dataset used for training and evaluate the performance of our model through qualitative and quantitative analyses. Our results demonstrate the efficacy of the proposed approach in generating personalized avatar images that closely align with the input textual descriptions, showcasing the potential for enhancing user experiences in various digital domains.

Introduction

The automatic generation of avatar images based on textual descriptions represents a significant advancement in the realm of personalized digital content creation. With the ability to synthesize custom avatar images without the need for specialized artistic skills, individuals can bring their envisioned characters to life in virtual environments, gaming, social media platforms, and various other digital contexts. Moreover, professionals in fields such as animation and game design can leverage automatic generation techniques to derive inspiration and streamline character creation processes.

Despite the potential benefits, existing models for generating avatar images often fall short in producing high-quality, industry-standard results. Many of these models yield images that are blurred, distorted, or lack fidelity to the intended characteristics outlined in the textual descriptions. As a result, the challenge of generating personalized avatar images with the level of quality and realism required for widespread adoption persists.

In this paper, our approach encompasses three key contributions: first, the utilization of a clean dataset sourced from Getchu, providing high-quality reference images for training; second, the adoption of a suitable Generative Adversarial Network (GAN) model architecture inspired by techniques such as DRAGAN and SRResNet, tailored specifically for generating avatar images; and third, a novel training methodology that incorporates conditional information derived from textual descriptions, enabling the model to generate personalized avatar images that accurately reflect the specified attributes.

Through comprehensive experimentation and evaluation, our work demonstrates the effectiveness of the proposed model in producing high-quality avatar images that closely align with the input textual descriptions. By addressing the limitations of existing approaches and offering a robust solution for personalized avatar generation, our work contributes to advancing the state-of-the-art in this domain and opens avenues for enhanced user experiences in digital environments.

Code

code

Background

In this section, I will briefly examine some fundamental building blocks employed in constructing our models discussed in this paper.

- **GAN**
  Generative Adversarial Networks (GANs) are a class of artificial intelligence algorithms used in unsupervised machine learning, invented by Ian Goodfellow and his colleagues in 2014. GANs consist of two neural networks: the generator and the discriminator.

  1. Generator: This network generates new data instances. It takes random noise as input and transforms it into data that ideally resembles samples from the training data distribution.
  2. Discriminator: This network evaluates the generated samples, attempting to distinguish between real data instances from the training set and fake data produced by

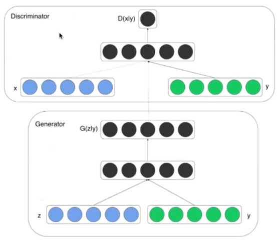

Figure 1: ACGAN Model

the generator. It essentially acts as a binary classifier, assigning a probability that a given sample is real or fake.

The training process involves a competitive interplay between these two networks. The generator tries to produce data that is increasingly indistinguishable from real data, while the discriminator aims to become more accurate in its classification task. This adversarial dynamic leads to the improvement of both networks over time.

- **ACGAN**

In my project, I used an extension of the traditional Generative Adversarial Network (GAN) architecture called Auxiliary Classifier Generative Adversarial Network (ACGAN). In comparison to traditional Generative Adversarial Networks (GANs), Adversarial Conditional Generative Adversarial Networks (ACGANs) introduce the concept of conditioning during both the training and generation phases. The structure is shown in Figure 1.

The Input z for the Generator(note as G) is a random input vector (or latent variables) provided to the generator network. This noise is sampled from a simple distribution, like Gaussian or uniform distribution, The generator in a GAN takes this noise as input and transforms it into synthetic data samples that ideally resemble the true data distribution. The condition y for both the Generator and the Discriminator in my project is the attributes of the avatar(gender, hair color, eye color, etc.) The out put $G(x|y)$ is a 128*128*3 vector in my project, which is a 128*128 RGB image.

The input x for the Discriminator(note as D) in my case is the training image or the image generated from the Generator. The output is the probability that the input data is real or fake.

- **Target Function(Min Max Loss)**

The target functions for both D and G are shown below:

$$V(G, D) := E_{x \sim p_{data}}[\log(x|y)] + E_{z \sim p_Z}[\log(1 - D(G(z|y)))]$$

In this function:

- `D(x)` is the discriminator's estimate of the probability that real data instance x is real.
- `G(z)` is the generator's output when given noise z.
- `D(G(z))` is the discriminator's estimate of the probability that a fake instance is real.

1. $E_{x \sim p_{data}}$
   - This term represents the expected value (denoted by E) of the log probability assigned by the discriminator (D) to real samples (x) drawn from the real data distribution ($p$data).
   - The discriminator's goal is to maximize this term, as it wants to correctly classify real samples as real. The log probability $[\log(D(x|y))]$ should be close to 1 for real samples (x).
2. $E_{z \sim p_Z}[\log(1 - D(G(z|y)))]$
   - This term represents the expected value of the log probability that the discriminator assigns to fake samples ($G(z|y)$) generated by the generator (G), where z is sampled from some prior distribution *pZ* (often a simple distribution like Gaussian).
   - The generator's goal is to minimize this term, as it wants to generate samples that the discriminator classifies as real. The log probability $[\log(1 - D(G(x|y)))]$ should be close to 0 for fake samples($G(z|y)$).

The altered minimax loss, as discussed in the original GAN paper, addresses a common issue where the GAN can become stuck during initial training stages, particularly when the discriminator's task is straightforward. Consequently, the paper proposes adjusting the generator's loss function to encourage the generator to maximize the logarithm of D(G(z)).
Overall, G and D are playing a min-max game. The discriminator aims to maximize V(*G*,*D*) by improving its ability to distinguish between real and fake samples. The generator aims to minimize *V*(*G*,*D*) by generating samples that are indistinguishable from real ones, effectively fooling the discriminator.

Sub-Pixel CNN

Sub-pixel convolutional neural networks (CNNs) are a specific type of CNN architecture designed to address image super-resolution tasks(shown in figure 8). In traditional CNNs, the output of convolutional layers typically has a smaller spatial resolution compared to the input image due to pooling and downsampling operations. However, in tasks like super-resolution, where the goal is to generate a high-resolution image from a low-resolution input, this downsampling can be problematic.

Sub-pixel CNNs aim to overcome this limitation by incorporating sub-pixel convolutional layers. These layers increase the spatial resolution of feature maps by rearranging the feature maps into a higher-resolution space. Instead

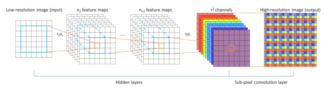

Figure 2: Sub-Pixel CNN

of simply upsampling the feature maps, sub-pixel convolutional layers reshape the feature maps such that each channel corresponds to a specific sub-pixel position in the output image.

During training, the network learns to generate high-resolution images by optimizing the parameters of the sub-pixel convolutional layers to reconstruct detailed features from low-resolution inputs. This approach allows sub-pixel CNNs to effectively upsample images while preserving important details and textures.

Sub-pixel CNNs have been successfully applied to various image enhancement tasks, including single-image super-resolution, image interpolation, and image denoising. Their ability to generate high-quality, high-resolution images from low-resolution inputs makes them a valuable tool in computer vision applications where enhancing image quality is crucial.

Method

I formulate the task of Automatic Genre Classification as follows :

Image Collection

I used an open-sourced images-web-crawler available at Git Hub[1], originates from the website Getchu. Renowned as Getchu[2], this platform specializes in offering information and facilitating the sale of Japanese games. The images obtained from this source are notably diverse, as they are crafted by various illustrators, each imbuing their distinct style into games spanning a wide range of themes. Despite this diversity, the collection remains cohesive, united by its focus on character images, which are integral to the gaming domain. Furthermore, these images boast decent quality and exhibit precise clipping and alignment, attributes essential for their intended purpose within the realm of illustration. With a total of 33,000 images in the collection, their inherent variety, consistent quality, and suitability for our task make them invaluable resources for our project.

Tag Estimation

To address the issue of images collected from Getchu lacking tags, we utilize Illustration2Vec[3], a pre-trained CNN-based tool designed to estimate tags for anime illustrations, albeit with some noise. This tool predicts the probabilities of images belonging to 512 general attributes or tags, such as ”smile” and ”weapon.” From this set, we carefully select 34 relevant tags suitable for our task. The chosen tags and the

[1]https://github.com/amineHorseman/images-web-crawler

[2]https://www.getchu.com/

[3]Pre-trained model available on http://illustration2vec.net/

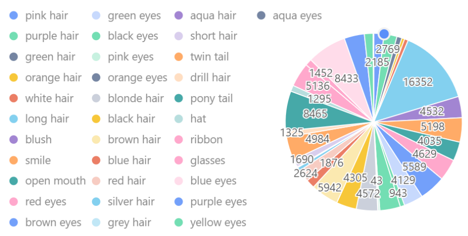

Figure 3: Attributes

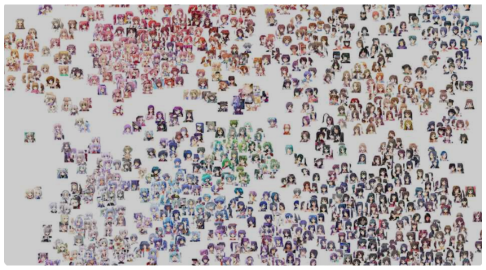

Figure 4: t-SNE 900

corresponding number of images associated with each estimated tag are presented in Figure 2. For tags that exhibit mutual exclusivity, such as hair color and eye color, we select the one with the highest probability from the network as the estimated tag. Conversely, for orthogonal tags like ”smile,” ”open mouth,” and ”blush,” we independently estimate each attribute’s presence using a threshold of 0.25. We aim to illustrate the image preparation process and evaluate the tag estimation performance visually. As an approximation, we apply the Illustration2Vec feature extractor—sharing architecture and weights with the tag estimator—to generate a 4096-dimensional feature vector for each image. These feature vectors are then projected onto a 2D space using t-SNE. Figure 3 displays the t-SNE result for 900 randomly sampled images from the dataset, revealing that character images with similar visual attributes are positioned closely together. This clustering suggests strong performance in tag estimation, facilitated by the shared weights within the network.

Picture Cropping

In the process of image cropping, I employ OpenCV alongside the open-sourced model lbpcascade_animeface[4] for detecting and isolating the desired portion of the image. Given that the majority of the images primarily consist of full-body portraits, the objective is to focus on the head section. The provided code snippet demonstrates this approach. Initially, the lbpcascade_animeface model is utilized to detect faces within the image. Following detection, the script calculates the bounding box for each detected face and subsequently crops the image to extract the head region. To ensure uniformity, the cropped head images are resized to a standard size

[4]https://github.com/nagadomi/lbpcascade_animeface

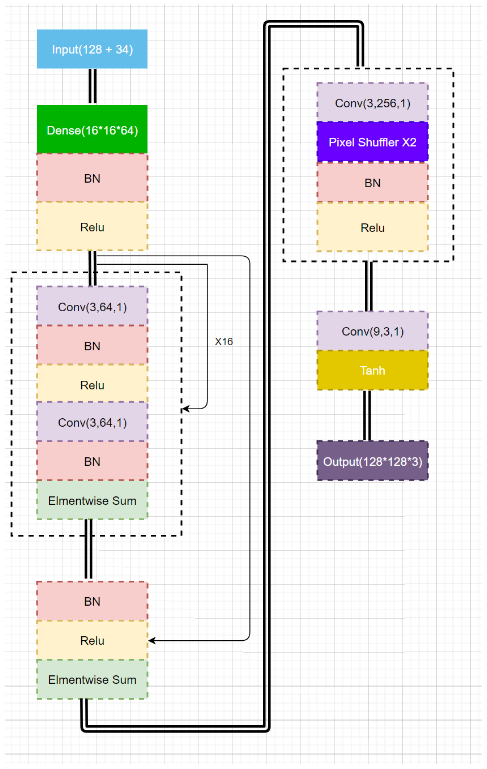

Figure 5: Generator Structure

of 128x128 pixels. Finally, the cropped and resized head images are saved with an appropriate filename for further analysis or utilization. This process ensures that the extracted images specifically highlight the head portion, facilitating subsequent tasks such as facial recognition or attribute analysis.

Network Architecture

The architecture of the generator is depicted in Figure 4, representing a modification derived from SRResNet. Comprising 16 ResBlocks, this model incorporates 3 sub-pixel CNN layers for the purpose of feature map upscaling. Meanwhile, Figure 5 illustrates the discriminator's architecture, featuring a total of 10 ResBlocks. Notably, all batch normalization layers are excluded from the discriminator to prevent the introduction of correlations within the mini-batch, a phenomenon deemed undesirable for computing gradient norms. Additionally, we introduce an extra fully-connected layer to the last convolutional layer, serving as the attribute classifier. Initialization of all weights follows a Gaussian distribution with a mean of 0 and a standard deviation of 0.02.

Result

I will delve into the outcomes of the experiments conducted in this section.

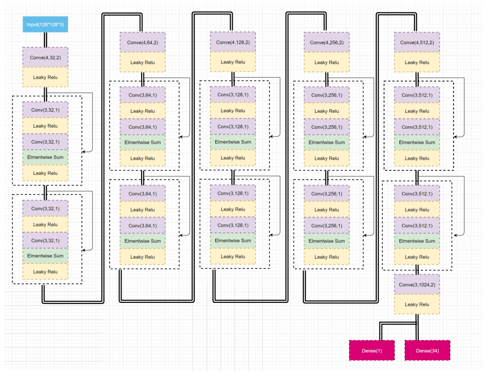

Figure 6: Discriminator Structure

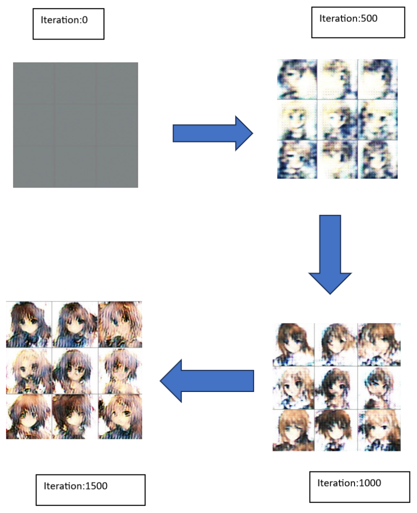

Figure 7: Iteration

Generated Images

Here are the images(figure 6) generated at intervals of 0, 500, 1000, and 1500 iterations.

Figure 6 illustrates an instance of controlling the random noise component while sampling random attributes, enabling the model to generate images with similar major visual characteristics. When sampling random attributes, we adopt a straightforward approach: for categorical attributes like hair and eye color, we uniformly select one attribute at random. For other attributes, each label is independently set with a probability of 0.25. Despite facing hardware limitations (complete 60000 epochs needs more than 400 hours)and spending time fine-tuning hyperparameters such as the parameters for the leaking ReLU activation function, the outcome is not flawless but remains meaningful. As depicted in Figure 6, it's evident that the generator is progressing along the right path. Based on our current evaluation, it's clear that distinguishing between similar colors, such as "white hair," "silver hair," and "gray hair,"

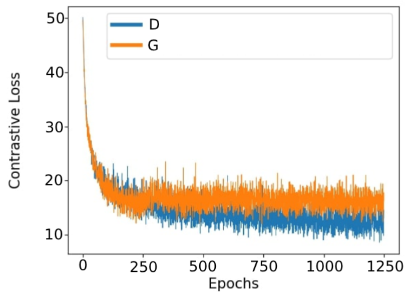

Figure 8: Contractive Loss

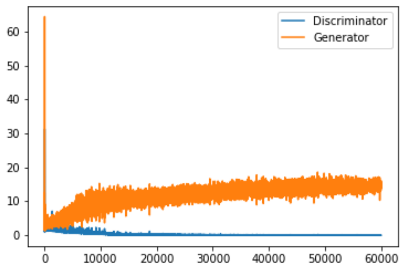

Figure 9: Enter Caption

presents a notable challenge. The lack of distinct boundaries between these shades can lead to classification confusion, resulting in lower precision scores for these attributes in our testing. Conversely, certain rare color attributes like ”orange eyes,” ”aqua hair,” and ”aqua eyes” demonstrate surprisingly high precision, despite their limited representation in the training dataset. This suggests that the visual concepts associated with colors may be relatively simple for the generator to grasp, even with minimal training samples. However, more intricate attributes like ”hat,” ”glasses,” and ”drill hair” exhibit poor performance in our experiments. Despite constituting approximately 5% of the training samples, the complex visual concepts they entail prove to be more challenging for the generator to accurately learn and replicate. Consequently, images generated when conditioned on these labels often appear distorted and are difficult to identify.

Analysis

Below is the Contractive Loss plot of D and G(figure 7). With only 1500 epochs completed, the loss diagram does not appear promising. According to data from other authors, it is recommended that this model undergo training for a significantly longer duration, ideally spanning between 40,000 to 60,000 epochs(figure 8), to achieve high-quality image outputs.

Conclusion

In conclusion, despite encountering significant hardware limitations that restricted training to only 3000 epochs and caused difficulties due to GPU and network collapses, the journey of developing the image generator has been illuminating. It’s evident that when compared with well-trained image generators, there remains a substantial gap in performance. However, there is reason for optimism as the generator demonstrates continuous improvement even within the constrained training timeframe. Moreover, insights gained from the experiments shed light on the challenges posed by attribute complexity within the dataset. While color attributes are relatively easier to generate, more intricate attributes like ”hat,” ”glasses,” and ”drill hair” present significant hurdles. These attributes often lead to distorted and ambiguous generated images, underscoring the need for further exploration and refinement. Despite these challenges, the progress made thus far reaffirms the belief that the project is on the right trajectory towards achieving more robust and sophisticated image-generation capabilities.

Despite encountering challenges and setbacks, my experience in building the model has been invaluable. While it may not have performed as expected, I’ve gained a wealth of knowledge about constructing neural networks and have become more proficient in utilizing TensorFlow. Looking ahead, I’m excited to apply what I’ve learned to one or two more projects this summer, confident that each endeavor will further enhance my skills and understanding in the field of machine learning.

Acknowledgments

I would like to express my sincere gratitude to the 5100 teaching staff whose dedication and expertise have played an instrumental role in shaping my journey into the field of artificial intelligence. Their guidance and support have been invaluable throughout this endeavor.

I am also deeply thankful to my faithful companion, Tomato, a Ragdoll cat whose comforting presence during late-night study sessions provided solace and companionship. Her unwavering loyalty has been a source of strength and inspiration.

Additionally, I extend my appreciation to all those who have contributed to this work in various capacities, whether through insightful discussions, constructive feedback, or moral support. Your contributions have enriched this project and made it possible.

Reference

1. Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adver- sarial networks. arXiv preprint arXiv:1701.04862, 2017.
2. Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. arXiv preprint arXiv:1701.07875, 2017.
3. Yanghua Jin, Jiakai Zhang, Minjun Li, Yingtao Tian, Huachun Zhu, Zhihao Fang. Towards the Automatic Anime Characters Creation with Generative Adversarial Networks.arXiv preprint arXiv:1708.05509