Development of a Generative Adversarial Network for Generating Pixel Art Character Sets for Games

Arthur Manoel da Maia
Universidade do Vale do Itajaí
Curso de Ciência da Computação,
São José - SC
maiaarthur744@gmail.com

Anita Fernandes
Universidade do Vale do Itajaí
Curso de Ciência da Computação,
São José - SC
anita.fernandes@univali.br

Dennis Kerr Coelho
Universidade do Vale do Itajaí
Curso de Ciência da Computação,
São José - SC
dennis@univali.br

ABSTRACT

The Brazilian game industry is mostly composed of small studios operating with limited financial resources and small teams. In this scenario, tasks that demand repetitive manual effort become potential bottlenecks in the development flow. Among the pixel art processes in 2D games, the creation of charsets stands out, which are structures formed by multiple animation frames of the same character. This is a particularly costly process, as any alteration to a sprite requires rework on several associated frames to maintain visual consistency. Faced with this challenge, this work proposes the development of a generative adversarial network (GAN) to assist in the automatic generation of charsets, reducing work time and consequently lowering production costs. The objective is to develop a GAN model capable of generating a complete charset in the RPG Maker engine standard, from a single input sprite. The proposed solution consists of implementing an adapted version of the Pix2Pix architecture, configured to generate the character set from a single sprite. After training, the model is evaluated using FID and L1 Loss metrics, complemented by a visual analysis of the results. It is expected to demonstrate that the "one-to-many" approach is viable and can act as a support tool for independent developers, aiming to promote productivity gains.

KEYWORDS

Generative Artificial Intelligence, Generative Adversarial Network, Pixel Art.

1 Introduction

Pixel art has become one of the most widely used techniques by independent developers. Requiring fewer computational resources, pixel art games are more accessible for both development and execution on different platforms. Furthermore, the simple yet nostalgic visual style and lower production cost compared to 3D graphics focused on realism make this approach quite attractive [1]. However, despite being a visually simpler technique, pixel art production still demands manual and detailed work, especially in the creation of animations and character sprites.

To represent character movement, it is necessary to create several variations of the same sprite in different poses. The complete set of animation frames for all directions of the same character is then compiled into a single image, called a sprite sheet or character sheet. In the context of this work, the term character set or charset is used to refer specifically to this set of sprites belonging to the same character, following the convention of the RPG Maker engine. The creation of these artifacts stands out as a costly process, as tasks that demand repetitive manual effort become potential bottlenecks in the development flow. In this case, any change to a sprite implies reworking several other sprites of the same character to maintain visual consistency [2].

Given this intense manual demand and the search for solutions that optimize development time, Generative Adversarial Networks (GANs) emerge as an alternative. These networks work through competitive learning between two models: the Generator, which tries to create convincing images, and the Discriminator, which tries to distinguish real images from fake ones. The process continues iteratively until the Generator can create images so convincing that the Discriminator can no longer differentiate them from real ones [3].

In addition to this, we have the growing trend towards adopting generative AI-based approaches in game development [4], and recent technical advances related to the application of GANs in pixel art [2, 3, 5, 6, 7]. Therefore, this work proposes a new approach. While the literature presents works focused on generating a sprite from another ("one-to-one") or generating a sprite from several ("many-to-one"), this work proposes a "one-to-many" task.

The objective of this research is to develop a GAN model capable of generating a complete character set in the RPG Maker engine standard from a single input sprite. In this specific standard, the character set consists of twelve sprites arranged in a 4x3 grid. Each row represents a direction of movement (forward, left, right, and back), and each column represents a direction of movement, with standardized dimensions of 48 x 48 pixels per cell. It is expected to demonstrate that this approach is viable and can act as a support tool for independent developers, promoting productivity gains and reduced development costs.

2 Proposed Architecture

The architecture chosen for the research is based on Pix2Pix, as it has already proven effective in the literature [5 - 7]. However, an adaptation was made to the original architecture, since the objective

of this work is not only to generate a translated image [2, 3] of the original, but rather an entire charset.

The architecture follows the conditional GAN model [3], where the generator G and the discriminator D can observe the informed conditions, in this case, the images used as original input. Conditional Adversarial Networks (cGANs) are a type of GAN that allows controlling, or "conditioning", the model output. Instead of simply generating random data that resembles the training set, cGANs generate data based on specific user-provided inputs, such as class labels, textual descriptions, or even other images [8].

With the cGAN already trained, the goal is that, given an input containing a character set with 12 repeated frames of the same character, plus a random noise value, the generator should be able to generate a complete character set in the standard used by the RPG Maker engine, that is, an image containing 12 48 x 48 sprites in 4 positions (front, left, right, back), each position having 3 different animation frames. The cGAN should be able to modify the 11 missing positions of the character set.

The loss function adopted is composed of adversarial loss and L1 loss. Adversarial loss measures how well the generator can deceive the discriminator [9]. L1 loss evaluates the absolute difference between the pixels of the generated image and the real image, encouraging the generator to produce results structurally like the original image, reducing noise and distortions. Figure 1 presents an overview of the process adopted in this work.

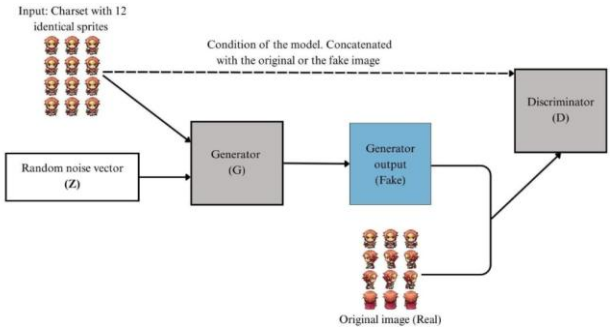

Figure 1: Process adopted in this work.

The Generator implemented in this work (Figure 2) follows the U-Net architecture, inspired by the models proposed in [2] and [9], for generating sprites in pixel art. This architecture adopts an encoder-decoder structure composed of seven down-sampling layers and seven up-sampling layers by skipping connections between the symmetrical layers, which allows preserving important spatial information from the original image during the reconstruction process.

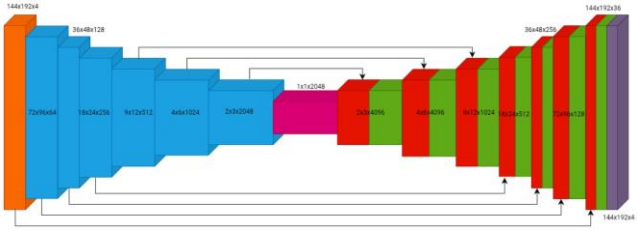

Figure 2: Generator Architecture.

The input image has dimensions of 144 x 192 x 4, corresponding to a character set composed of twelve identical sprites of the same character, each sprite with four channels in the RGBA color space. In the encoder, the network performs a progressive feature extraction, gradually reducing the spatial dimensions and increasing the depth of the feature maps until it reaches a latent vector of 1 x 1 x 2048. In the decoder, the reverse process occurs: the network expands the spatial dimensions again and reduces the depth of the maps, reconstructing the image until it reaches the original resolution of 144 x 192 x 4.

Skip connections directly link to the corresponding layers of the encoder and decoder, allowing low-level information, such as edges, contours, and fine textures, to be preserved throughout the reconstruction process. This feature is especially important in pixel art images, where small visual details greatly influence the final perception of the result.

The discriminator follows the PatchGAN model [2, 9] and aims to evaluate the authenticity of local regions (patches) of the image rather than classifying them. This approach is especially effective for pixel art images because they have low resolutions, meaning that any small variations in color and texture have a large percentage impact [2].

Continuing, the discriminator receives as input pairs of images: the original charset (model condition) concatenated with the actual image from the dataset (real pair) or the image generated by the generator (false pair). Regarding its architecture, it consists of a sequence of convolutional layers that progressively reduce the spatial resolution, producing patches of 2 x 2 pixels. Each of these patches will be evaluated by the discriminator, calculating the probability of a real or false patch set. Thus, the discriminator's output is not a single binary value, but rather a local authenticity map based on the patch results.

3 Dataset

The dataset used in this work was created from 118 charsets, each containing eight distinct characters. During the data preparation process, each charset was segmented into eight parts, each containing twelve-character sprites, resulting in nine hundred and forty-four derived charsets, corresponding to the isolated characters extracted from the original files (Figure 3).

Each of these charsets contains pixel art sprites standardized in the RPG Maker engine format, that is, twelve sprites of the same humanoid character, organized in four directions (front, left, right, and back), with three animation frames for each direction.

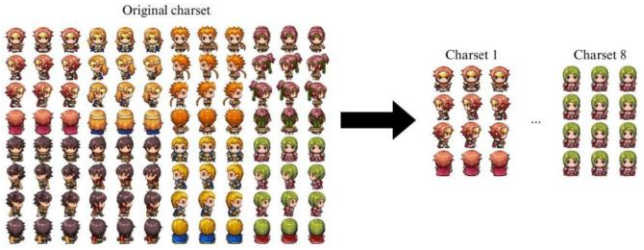

Figure 3: Obtaining the individual character sets from the original character set.

Each sprite in the charset is 48 x 48 pixels in size. This standard was adopted because it is widely used in modern versions of the RPG Maker engine and because it offers greater availability and free examples on the internet compared to other, less commonly used formats.

During dataset curation, to increase the total number of entries, character sets in a different standard, 32 x 32 pixels, were also considered. In these cases, the upscaling technique was applied, expanding the image size to 48 x 48 pixels while maintaining the positions of each sprite within the character set. This information will be extremely important during the image generation process.

4 Preliminary Considerations

This research is currently in the implementation and testing phase of the model using the resources of the Scikit-Learn library.

To test and validate the implemented model, several experiments were planned.

The experiments will be performed on the created dataset, using a GeForce RTX 3060 graphics card with 12 GB of video memory. The development environment will be VSCode, installed on a computer with the Windows 10 operating system. The test/train ratio used will initially be 70/30%.

For network parameter optimization, the ADAM optimizer [10] will be used. This technique is chosen because of its efficiency and adaptive learning rate adjustment capability, and because it has already been successfully used in other studies [5 - 7].

The training will be configured with an initial learning rate of 0.0002 and exponential decay moments (β1 and β2) set to 0.5 and 0.999, respectively. The batch size will be determined experimentally, aiming for the largest possible value that does not exceed the GPU memory, thus ensuring training stability.

The experiment will consist of providing a complete charset containing the same sprite of a pixel art character in all its positions. The cGAN will need to modify eleven of these sprites so that the image maintains the pattern defined by the charset, with four positions, one per line, each line containing three animation frames. The final image format should be 192 x 144, and the sprites should be in 48 x 48 format, as in the original image (Figure 4). The sprites were repeated in the charset positions to provide the cGAN with spatial information of the chosen pattern.

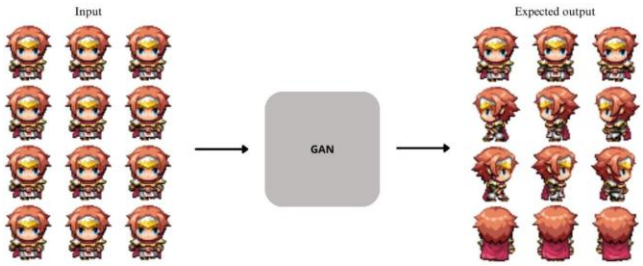

Figure 4: Experiment Example.

After training the model and obtaining the results, an analysis will be performed using the FID and L1 distance evaluation metrics, in addition to a visual inspection process.

REFERENCES

[1] Clarisse Carneiro e Silva, 2023. A utilização da estética da Pixel Art em jogos em épocas de equipamentos de alto desempenho. *TCCS – Cinema de Animação e Artes Digitais*. Acessado em agosto de 2025. Disponível em https://eba.ufmg.br/tccs/index.php/caad/issue/view/12.

[2] Flavio Coutinho and Luiz Chaimowicz, "Generating Pixel Art Character Sprites using GANs,"*2022 21st Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*, Natal, Brazil, 2022, pp. 1-6, DOI: 10.1109/SBGAMES56371.2022.9961120.

[3] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xy, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio, 2020. Generative adversarial networks, v. 27, 2020. *Communications of the ACM*, Volume 63, Issue 11. DOI: https://doi.org/10.1145/342262.

[4] Marcos V. Cardoso, Claudio Gusmão, Jonatha, J. Harris, (Org), 2023. *Pesquisa da Indústria Brasileira de Games*. ABRAGAMES: São Paulo. Acessado em setembro de 202. Disponível em: https://www.abragames.org/uploads/5/6/8/0/56805537/2023_relat%C3%B3rio_final_v4.3.2_-_ptbr.pdf

[5] Flávio Coutinho, Luiz Chaimowicz. 2024. Pixel art character generation as an image-to-image translation problem using GANs, *Graphical Models*, Volume 132, 2024, 101213. DOI: https://doi.org/10.1016/j.gmod.2024.101213.

[6] Flávio Coutinho and Luiz Chaimowicz. 2024. A Missing Data Imputation GAN for Character Sprite Generation. In *Anais do XXIII Simpósio Brasileiro de Jogos e Entretenimento Digital*, setembro 30, 2024, Manaus/AM, Brasil. SBC, Porto Alegre, Brasil, 436-455. DOI: https://doi.org/10.5753/sbgames.2024.241116.

[7] Zhouyang Jiang and Penny Sweetser. 2022. GAN-Assisted YUV Pixel Art Generation. In *Proceedings of AI 2021: Advances in Artificial Intelligence: 34th Australasian Joint Conference, AI 2021,* Sydney, NSW, Australia, February 2–4, 2022. DOI: https://doi.org/10.1007/978-3-030-97546-3_48

[8] Si-An Chen, Chun-Liang Li, Hsuan-Tien Lin. 2021. A Unified View of cGANs with and without Classifiers. *arXiv:2111.01035*, Cornell University. DOI: https://doi.org/10.48550/arXiv.2111.01035.

[9] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou and Alexei A. Efros, 2017. Image-to-Image Translation with Conditional Adversarial Networks,"*2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA, 2017, pp. 5967-5976, DOI: https://doi.org /10.1109/CVPR.2017.632.

[10] Diederik P. Kingma, Jimmy Ba, 2014. Adam: a method for stochastic optimization. *arXiv*. DOI: https://doi.org/10.48550/arXiv.1412.6980