Proceedings of the Eighteenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment
(AIIDE 2022)

On the Challenges of Generating Pixel Art Character Sprites Using GANs

Flávio Coutinho,[1,2] **Luiz Chaimowicz**[1]
[1]Universidade Federal de Minas Gerais, Belo Horizonte, MG, Brazil
[2]Centro Federal de Educação Tecnológica de Minas Gerais, Belo Horizonte, MG, Brazil
fegemo@cefetmg.br, chaimo@dcc.ufmg.br

Abstract

We pose the problem of generating pixel art character sprites facing one side (e.g., right), given their images facing another one (e.g., front), as an image-to-image translation task and investigate the use of the Pix2Pix architecture to solve it. Aiming to improve the results on unseen data, we propose and investigate two architecture modifications: (a) representing images using color palettes, and (b) adding a histogram loss term to the generator. We compare the results qualitatively and quantitatively using FID and L1 distances between the generated and target images. Results indicate that representing images with color palettes encourages overfitting, and the histogram loss leads to slightly improved results.

Introduction

Recent advances in the field of Machine Learning have led to the development of different Procedural Content Generation techniques (Summerville et al. 2018; Khalifa et al. 2020). In the particular case of cosmetic game assets such as images, generative adversarial networks (GANs) (Goodfellow et al. 2014; Radford, Metz, and Chintala 2015) can be leveraged, if enough data is available. Some works investigated the generation of pixel art characters or landscapes using different methods such as DCGANs (Horsley and Perez-Liebana 2017), VAEs (Gonzalez, Guzdial, and Ramos 2020), and conditional DCGANs (Serpa and Rodrigues 2019; Hong, Kim, and Kang 2019; Jiang and Sweetser 2021; Yang et al. 2022; Coutinho and Chaimowicz 2022), but the obtained results are not as good as with the generation of photorealistic images using the same or similar techniques (Isola et al. 2017; Karras, Laine, and Aila 2018). Improving the quality of the generated content can benefit the asset creation pipeline, for instance, through the use of mixed-initiative systems (Liapis, Smith, and Shaker 2016), or empowering non-technical users to create artistic content.

For the special case of generating pixel art images, existing works typically employ the same or slightly adapted techniques that have been proposed for realistic images, but some characteristics of the task make it particularly challenging. For example, there are no large datasets of pixel art openly available; with a smaller spatial resolution, the amount of information each pixel conveys is much greater than in photos; the use of colors is frequently restricted to a small color palette.

In this work, we study the generation of pixel art images through generative adversarial networks by selecting the task of generating pixel art characters in a target pose (e.g., facing right) by using an image of another pose as an input (e.g., facing front). We frame such a problem as an image-to-image translation task (Pang et al. 2021) and analyze the performance of a general single-modal supervised architecture called Pix2Pix (Isola et al. 2017), which we use as baseline. When using diverse but small datasets, we can observe that the generated images have low quality with the test data, either resulting in blurry images or containing high-frequency noise. Hence, we proceed by proposing two other variations to the baseline architecture: (a) representing images as grids of indices in a palette, instead of using the RGB space; and (b) adding a loss term to the generator that steers the generated images towards having the same color histogram of the source image.

The palette-based variation presented worse results for unseen data, but the one with the histogram loss yielded slightly improved sprites. Nevertheless, both experiments provided insights regarding the generation of pixel art images with GANs. Our contributions are the characterization of tasks involving the generation of pixel art, the negative results from representing images as indices in a palette, and the possibility for image improvement through the use of a differentiable histogram loss.

Related Work

We first define the Image-to-Image Translation problem and present how different works approached it.

Image-to-Image Translation

The problem of Image-to-Image Translation can be described as transferring images from a source into a target domain while preserving some information (Pang et al. 2021) and has seen much development with deep generative models, such as VAEs and GANs.

After the introduction of the original GAN (Goodfellow et al. 2014), another work proposed the deep convolutional GAN (DCGAN) as a set of architecture choices that experimentally induced results with better quality, such as the use

Work	Task	Method	Dataset Sizes	Repres.
Horsley and Perez-Liebana (2017)	(Uncond.) Characters, faces, and creatures	Based on DCGAN	1,210; 517; 36	RGB
Serpa and Rodrigues (2019)	Shaded and segmented sprite from sketch	Two-headed Pix2Pix	530; 85	RGB
Gonzalez, Guzdial, and Ramos (2020)	Domain transfer of sprite given target class	Convolutional VAE	7,204	HSV
Jiang and Sweetser (2021)	Color from grayscale image	Adapted from Pix2Pix	870; 36	YUV
Yang et al. (2022)	Photo to/from pixel art landscape	Adapted from CycleGAN	500	RGB
Coutinho and Chaimowicz (2022)	Target side from source side sprite	Adapted from Pix2Pix	912; 408; 294; 208	RGBA

Table 1: Related works that generated pixel art images.

of strided convolutions instead of pooling layers, and the use of batch normalization (Radford, Metz, and Chintala 2015). To allow some level of control of the generation process, another work introduced the conditional GAN (Mirza and Osindero 2014), a method to indicate what type of content the system should create. In Pix2Pix, Isola et al. (2017) introduced a general architecture for different image-to-image translation tasks by proposing to condition the generation process on the image in a source domain while doing supervised training for the generator to create its version in a target domain (e.g., grayscale to colored images). In addition to the adversarial loss of the generator, to enforce the condition, there is an L_1 distance term between the images in the target domain and the ones created by the generator. The total generator loss is then:

$$\mathcal{L}(G) = \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L_1}(G) \tag{1}$$

The architecture had good results in different translation tasks, such as (both ways) semantic labels to photos, grayscale to colored pictures, sketches to photos, day to night, and photos with missing parts to inpainted ones. We observe that none of the experimented tasks involved pixel art images, or any type of shape deformation – which are two characteristics of the problem we are posing in this work.

Pixel Art Generation

Most development in deep generative models focuses on creating photorealistic images. Table 1 gathers some recent works that targeted the creation of pixel art images.

Horsley and Perez-Liebana (2017) trained a model based on DCGAN to generate pixel art sprites of human-like characters, faces, and creatures. The proposed model is unconditional (i.e., generates images from noise) and the generated images resemble the training set but present a lot of noise even in background areas. The system also suffered from mode collapse, with different z values mapping to similar resulting images.

Gonzalez, Guzdial, and Ramos (2020) trained a convolutional VAE to transfer pixel art images to a different domain, using Pokémon sprites with their associated in-game types (e.g., grass, fire, water) as the domain. With a small dataset of 7,204 images, the authors pre-trained the model with the Anime Face Dataset[1] (63,632 images). The resulting images were generally blurry, with the colors sometimes resembling the ones in the target domain, and the shape was not very well defined.

[1]https://www.kaggle.com/datasets/splcher/animefacedataset

Serpa and Rodrigues (2019) used a Pix2Pix-based architecture in which the generator had two heads to output a shaded grayscale image and a body-part segmented sprite from line sketches of characters. The generation of the shaded grayscale sprite yields images close to the ground truths, especially for character poses similar to the ones seen during training – but the ones from radically different poses were still deemed useful as a starting point for artists to refine. In the case of the body-part segmented sprites, the generated images had a lot of intra-segment noise and used a different color palette mapping for the segments. One appointed cause of the noisy images was the small size of the datasets (530 and 85 images), especially as there were many different poses a character could take.

Jiang and Sweetser (2021) tackled the problem of coloring pixel art sprites given a grayscale version. They used the Pix2Pix architecture and represented images in the YUV color space. They used datasets with sizes 870 and 36 images, and the resulting images produced feasible colorization in most cases.

Yang et al. (2022) proposed a system based on CycleGAN (Zhu et al. 2017) to translate pixel art and real photos of landscapes bidirectionally. Using only 500 images in the pixel art domain, the authors observed that the pixel grid was not sufficiently evident as it should be in the pixel art domain. Also, edges were noisy, and a still high number of colors was used, even with the target domain being typically composed of just a tiny palette.

Coutinho and Chaimowicz (2022) used an adapted Pix2Pix architecture to generate images of pixel art characters facing a target pose from an image of it facing a source, the same task we investigate in this work. Trained with four datasets, the models could reproduce the training data with $L1$ distance close to zero but yielded blurry images or high-frequency color noise for the validation data, especially in small but relatively highly diverse datasets.

In all such works, the datasets were small, considering that deep generative models are typically trained with a number of examples in the order of tens or hundreds of thousands to show enough generalization capability. Still, some of the results had sufficient quality to be used or were at least very promising.

Regarding the image representation, most works processed images in RGB. One work used HSV as it was helpful to relate the images with their type domains (Gonzalez, Guzdial, and Ramos 2020), while another used the YUV space as it yielded more colorful images when compared to RGB (Jiang and Sweetser 2021). From their experiences, we note that the image representation can impact the results, and there may be better-suited choices depending on the task. Lastly, one work used the alpha channel in addition to RGB and noted that, for the generation of character images in a transparent background, the additional information, although primarily redundant with the other channels, yielded images with better-defined shapes (Coutinho and Chaimowicz 2022).

In our work, we use an improved version of Coutinho and Chaimowicz (2022) as a baseline, evaluate its generalization capacity in different tasks, and propose two variations: using palette-indexed colors instead of RGB, and adding a histogram loss term to the generator.

Pixel Art Characterization

Seeking photorealism in image generation requires the network to solve challenging problems, such as achieving crisp images with anti-aliased edges, correct shape and texture with feasible lighting, and so on (Karras, Laine, and Aila 2018). At first glance, generating pixel art could be a simpler objective, but here we show some of its characteristics that introduce new and specific challenges.

Because of the typical lower spatial resolution, pixel art images usually encode much more information per pixel than photos (Kopf and Lischinski 2011). To illustrate, Figure 1 depicts a sprite in which a single pixel can change the expression of a character. This characteristic makes noise and shape deviation much more harmful to the perceived quality of the generated images than in the case of photos.

Due to historical reasons of hardware limitations, artists create sprites using a limited number of colors that constitute that sprite's palette. When generating such images using deep generative models, especially in the case of small datasets, the resulting images will likely include many more colors, extrapolating the intended palette. For example, Figure 2 shows how different the color histograms are of a pixel art image and a photo of a similar landscape. The two hypotheses investigated in this work originate from the idea of wanting to guide the generated images towards having the intended palette.

Regarding the form, pixel art is a stylistic, low fidelity representation of whatever it depicts (Silber 2015). That might hinder the performance of reusing learned weights from non-pixel art datasets. In addition, different shading and texturing patterns are typically used by artists, such as color ramps (gradients) and dithering (color mixing – Figure 3), and those will be likely absent from such datasets.

The reduced size also impacts the frequency distribution throughout the sprite. Real pictures typically have large regions with texture bounded by shape edges, but pixel art presents both patterns interchangeably and very close to each other. For example, a person's face in a picture might have a large region with similar color (low frequency) for its cheek, which can be bounded on one side by the face edge (high frequency). In turn, a pixel art character, typically a lot smaller, uses much fewer pixels for its cheek, interleaving low and high frequency much more often.

Another more circumstantial characteristic that presents challenges to the generation of pixel art is the scarcity of open and large datasets, especially in comparison to the ones used in Computer Vision tasks. Table 2 presents the sizes of some popular datasets with which generative models are trained with. In contrast with the ones used in our

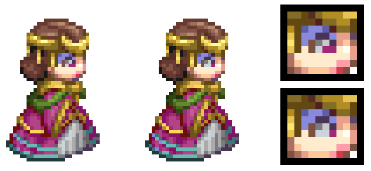
Figure 1: How the value of a single pixel changes the character's facial expression.

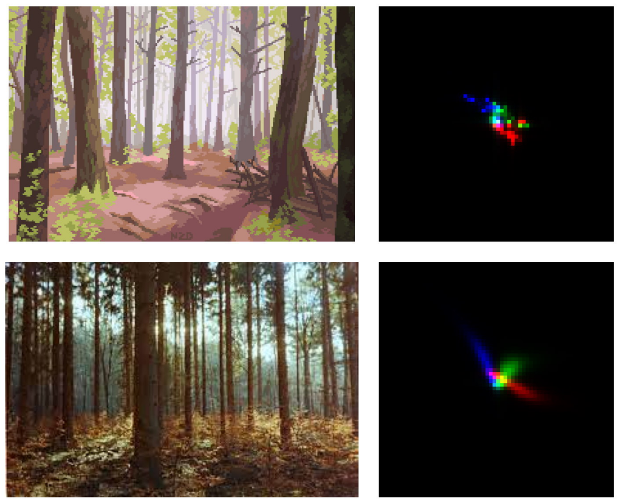
Figure 2: Landscape of a forest in pixel art with its histogram (top), and a photo/histogram of a similar landscape (bottom).

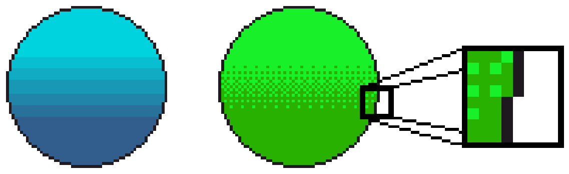
Figure 3: Shading with color ramps (left), with dithering (center), and the intentional crisp edges (right).

Dataset	Size
Pix2Pix's Facades	606
Pix2Pix's Cityscapes	3,475
Pix2Pix's Night2Day	20,120
FFHQ	70,000
CelebA	202,599

Table 2: Sizes of datasets used in Computer Vision tasks.

related works (Table 1), their sizes are 1-4 orders of magnitude higher. Adding to that problem, some data augmentation techniques are inappropriate, such as rotation, since no matter whether it is used with or without interpolation, the operation results in unpleasant artifacts.

Pix2Pix Baseline

We build on top of the work from Coutinho and Chaimowicz (2022), which is based on a Pix2Pix architecture for the task of translating pixel art character sprites from a source into a target side (e.g., front to right-facing). It processes images in 64×64×4 (RGBA) and yielded good results for the training data, but much worse for the validation set. Unlike the reference work, we implemented data augmentation of hue rotation and character offset inside the image.

This model was the base implementation for the other approaches investigated in the following sections, so we detail it here. The generator is a U-net that downscales the input to 1×1 in six Conv-InstanceNorm-LeakyReLU blocks, then upscales the 512-sized latent code back to the original resolution in six ConvTranspose-InstanceNorm-Dropout-ReLU blocks. The input for each upscaling block is concatenated with the output from the corresponding downscaling step. There is a last convolutional layer with a $\tanh$ activation and 4 filters for the RGBA components. In turn, the discriminator follows the PatchGAN model with 2×2 patches, which results in an output with a size of 32×32×1 of real/fake estimations. Those are checked against zeros (fake) or ones (real) using binary cross-entropy, and the resulting values are averaged to get the total cost.

Dataset

The dataset we use in the experiments has 294 pixel art sprites drawn on four sides, and we focused our tests on generating right-facing images from front-facing characters. That is more difficult than translating laterally (usually a horizontal flip) or frontally (generally keeping most of the shape and changing only the texture). The train/validation split was 85%, resulting in 250 sprites used for training the network.

Each character has three slightly different images for each side as we assembled the dataset from sprite sheets of walking animations with three frames. During the split, all frames of each character remained together either in the train or in the validation set.

Approaches

In this section, we first present the architecture modification to represent images as indices in color palettes, and later the use of a differentiable histogram loss term to the generator's cost function.

Palette-Indexed Colors

To enforce the generated images to use only the colors from the palette of the source image, we changed the color representation from RGBA values to indices in an example-specific palette. The palette is extracted when the image is loaded, and we defined a maximum of 256 colors per example, even though the dataset we use has images with at most 54 simultaneous colors.

Regarding the networks, we changed the last layer of the generator to have a number of units equal to the maximum palette size (i.e., 256) and a softmax activation instead of $\tanh$. Both discriminator and generator networks process images using one-hot encoding of the color indices. Furthermore, as the color indices do not represent color distances, we replaced the L_1 term from the generator loss (Equation 1) with a categorical cross-entropy $\mathcal{L}_{cce}$ with the indices of the target image, scaled by some factor λ:

$$\mathcal{L}(G) = \mathcal{L}_{cGAN}(G, D) + \lambda\mathcal{L}_{cce}(G) \qquad (2)$$

The generator could perfectly reproduce data seen during training. At the same time, the quality of the images generated from the validation data started degrading early in the process, indicating the model memorized the data even under dropout regularization on both networks.

We extracted each image's palette by picking the unique colors and filling the remaining color slots with hot pink, absent from the training/validation data. Regarding the order in which indices were assigned to the colors, we tested sorting the colors by their gray value ($0.2989R + 0.5870G + 0.1140B$), by the order of appearance in the image from top to bottom, from bottom to top, and completely random.

The model with shuffled palettes did not converge while all others overfit. We noted that when there was semantics to the color indices in the palettes, the images generated from the validation set presented different results. Figure 4 illustrates this observation in early iterations (1.2k generator updates, before overfitting): when the first indices of all palettes share the semantics of corresponding to either the characters' heads or their feet (4th and 5th columns), the quality of those parts stand out in comparison to the remainder of their bodies.

Taking into account the worse results when training with shuffled palettes, the takeaway is that by moving out of the RGB space into an indexed representation, we deprive the model of the semantics embedded into the RGB space. That led us to hypothesize whether we can define a new space, potentially higher-dimensional, in which colors are represented by embeddings that encode their role inside the dataset. For instance, such a representation could include the relation of colors that appear adjacently or in different patterns, allowing the networks to represent and reproduce textures and shading techniques more easily.

Histogram Loss

Instead of enforcing the strict use of the same palette of the input images, we experimented with a second approach to penalize the generator for producing images with a color histogram different from the one from the input. One challenge here is that for gradient descent to be feasible, all terms of the cost functions must be differentiable, but calculating histograms naively are not (due to thresholding in bins).

We then resorted to a different method to derive an image histogram that replaces thresholding with a kernel weighted contribution to each bin (Afifi, Brubaker, and Brown 2020).

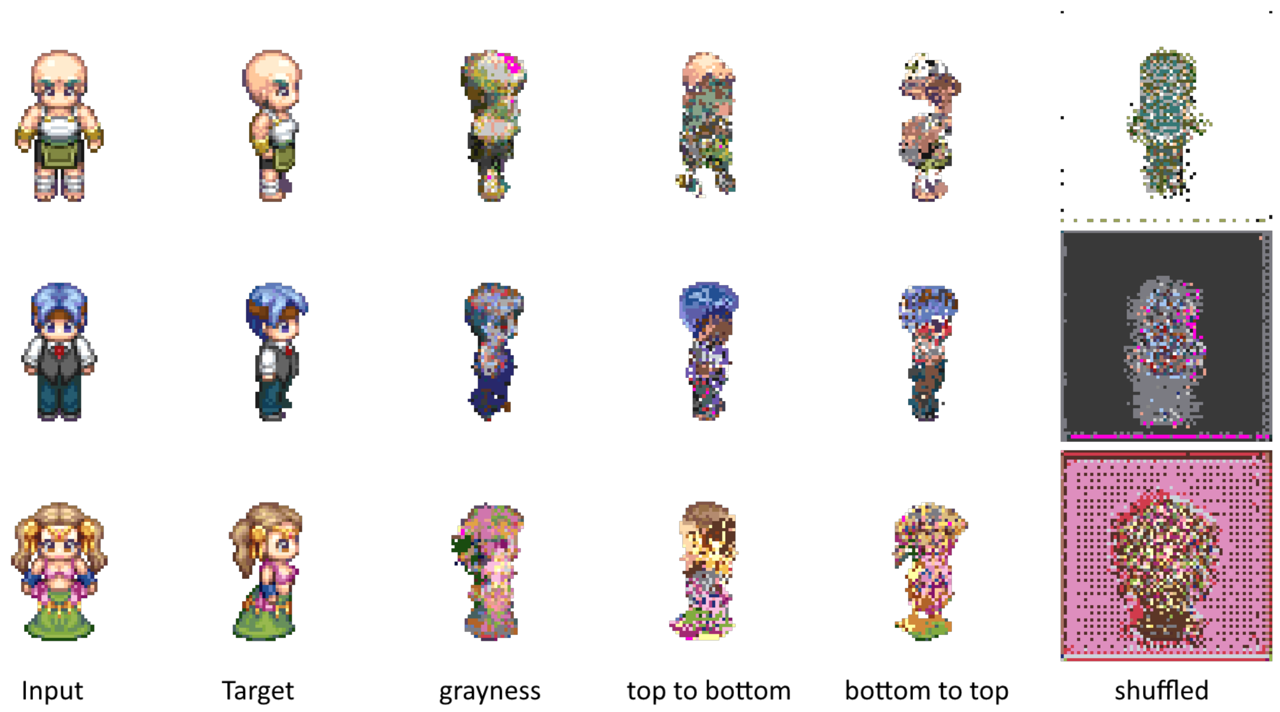

Figure 4: Early output of the generators on validation data from palette-indexed colors varying the mapping of colors to indices.

It uses the log-chroma space, in which two of the RGB channels are used to normalize the third, resulting in two parameters (u, v) for each original color component. By doing so, it is possible to have the color histogram visualized in 2D (Figure 2 shows examples).

To add some control to the colors of generated images, the authors of HistoGAN (Afifi, Brubaker, and Brown 2020) added the target histogram mapped to the latent space of the last two upsampling blocks from StyleGAN (Karras, Laine, and Aila 2018). In addition, the generator tries to minimize the distance from the target histogram to the one from the generated image.

Since we intend to encourage the generator to use the same colors from the source image in our task, we added the histogram loss to the generator cost function. However, we refrain from adding it as input to the last layers of the generator as we do not intend to change the histogram. The generator loss uses a scaling factor λ_{his} and has the form:

$$\mathcal{L}(G) = \mathcal{L}_{cGAN}(G, D) + \lambda_{L_1}\mathcal{L}_{L_1}(G) + \lambda_{his}\mathcal{L}_{his}(G) \quad (3)$$

where the $\mathcal{L}_{his}$ is the Hellinger distance (Simpson 1987) between the histograms of the source and target image:

$$\mathcal{L}_{his}(H_g, H_t) = \frac{1}{\sqrt{2}}||H_g^{1/2} - H_t^{1/2}||_2 \quad (4)$$

We tested different combinations of the λ_{his} hyperparameter, fixing $\lambda_{L_1} = 100$ (Figure 5). When the histogram term is higher than $\lambda_{L_1}\mathcal{L}_{L_1}$, the generator fails to translate the character to the target side but more rapidly reduces the histogram loss. In turn, for too low λ_{his}, the loss acts mainly as a regularizer, preventing the model from memorizing the training data but adding little to no gain over the quality of the images on the validation set.

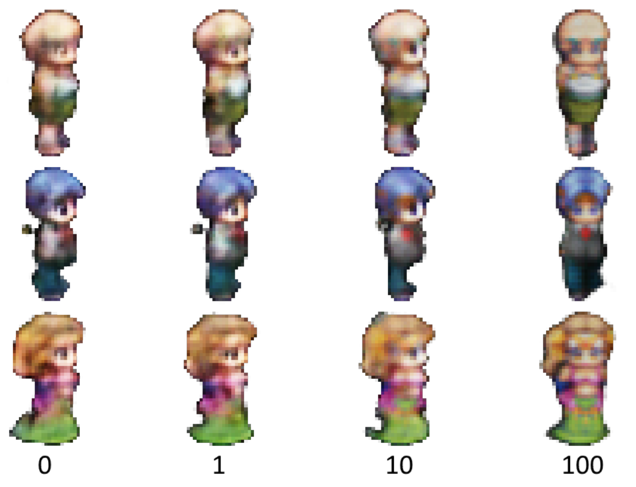

Figure 5: Results from the histogram loss, varying λ_{his}.

Evaluation

We train the baseline with and without augmentation, the model with indexed images, and the one with the histogram loss. The models of both versions of the baseline have $\lambda_{L_1} = 100$. The palette-indexed model used $\lambda_{cce} = 0.01$ and the color indices are sorted by their grayness value. Lastly, the

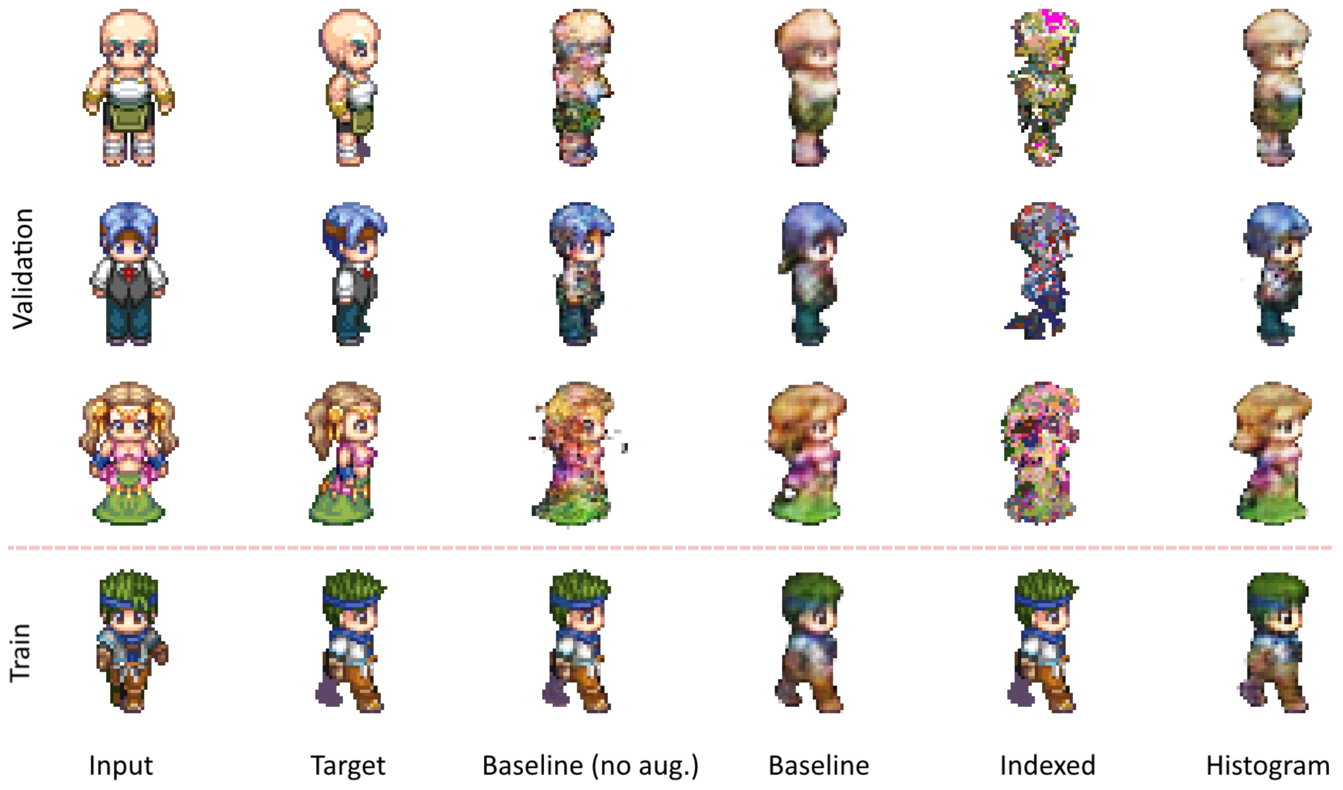

Figure 6: Results from all models after 160 epochs.

Model	FID		L_1	
	train	**val.**	**train**	**val.**
Baseline (no aug.)	0.357	10.830	0.008	0.064
Baseline	6.470	9.901	0.038	**0.058**
Palette-Indexed	0.057	135.100	0.001	10.250
Histogram Loss	6.072	**9.396**	0.040	0.060

Table 3: FID and L_1 distances between the generated images and their targets.

model with the histogram loss has $\lambda_{his} = 1$ and $\lambda_{L_1} = 30$.

We compare the results through visual inspection and quantitatively. For metrics, we used the Frechét Inception Distance (FID) (Heusel et al. 2017) and the L_1 distance between the generated images and their targets. All tests compare how the model performs on training and validation images.

All models are trained for 160 epochs in batches of 4. The discriminators and generators were optimized using Adam ($\beta_1 = 0.5$ and $\beta_2 = 0.999$) with a learning rate of 0.0002.

Figure 6 shows example outputs from the models. First, we can observe that the baseline without augmentation and the model with indexed-palette could memorize the training data and produced degraded images from the validation set. With data augmentation on the baseline, the quality of the images from unseen data improved. Lastly, the images generated from the validation set by the model with histogram loss yielded slightly better translations, which is endorsed by the lower FID and similar L_1 distance (Table 3).

Conclusion

In this work, we frame the problem of creating character images on a target pose from a source as an image-to-image translation problem. First, we investigate the applicability of the Pix2Pix architecture to the problem and observe that it struggles to generate believable images from unseen data. We then raise some characteristics of pixel art that make the task challenging, such as the use of very few colors from palettes and the absence of large and open datasets.

We then investigated two hypotheses: representing images as indices in a palette and adding a histogram loss term to the generator. The model that processes indices of colors yielded worse images for unseen data in a clear overfitting situation. Regarding this approach, instead of using a fixed palette, one might investigate how a higher-dimensional representation that embeds not only information about the channels, but also the colors' roles in the dataset.

The experiment with the histogram loss term produced slightly better sprites, especially considering the dataset used is diverse but very small – 294 images. Compared to the baseline, the FID for the unseen images was reduced by 5.1%. A possible next step is to feed the target palette into the generator as an attempt to steer the usage of colors even more to match the source sprite.

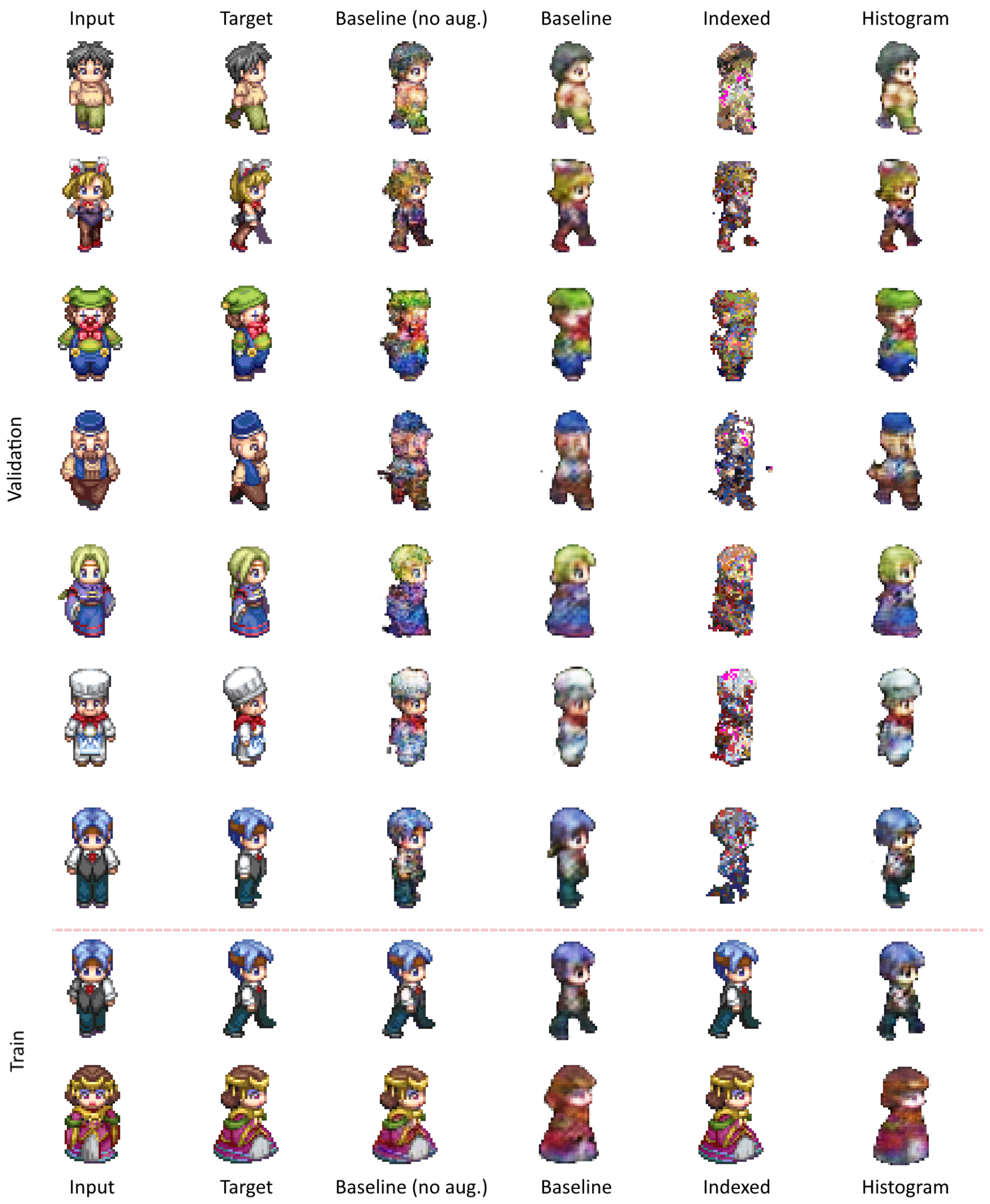

Figure 7: Additional results from the trained models with Histogram images slightly less blurry than those from the baseline.

Appendix: Additional Results

Figure 7 shows additional results. An interesting observation is the better quality of the character in the last validation row since the generator did see one of its three animation frames (because of the train/test split) during training.

Acknowledgments

This work was supported by CAPES, CNPq and Fapemig.

References

Afifi, M.; Brubaker, M. A.; and Brown, M. S. 2020. HistoGAN: Controlling Colors of GAN-Generated and Real Images via Color Histograms. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 7937–7946.

Coutinho, F.; and Chaimowicz, L. 2022. Generating Pixel Art Character Sprites using GANs. *arXiv preprint arXiv:2208.06413*.

Gonzalez, A.; Guzdial, M.; and Ramos, F. 2020. Generating Gameplay-Relevant Art Assets with Transfer Learning. In *Proceedings of the AIIDE Workshop on Experimental AI in Games*, 1–7.

Goodfellow, I. J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative Adversarial Nets. In *NIPS'14: Proceedings of the 27th International Conference on Neural Information Processing Systems*, volume 29, 2672–2680.

Heusel, M.; Ramsauer, H.; Unterthiner, T.; Nessler, B.; and Hochreiter, S. 2017. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. *Advances in Neural Information Processing Systems*, 2017-December: 6627–6638.

Hong, S.; Kim, S.; and Kang, S. 2019. Game sprite generator using a multi discriminator GAN. *KSII Transactions on Internet and Information Systems*, 13(8): 4255–4269.

Horsley, L.; and Perez-Liebana, D. 2017. Building an automatic sprite generator with deep convolutional generative adversarial networks. *2017 IEEE Conference on Computational Intelligence and Games, CIG 2017*, 134–141.

Isola, P.; Zhu, J.-Y.; Zhou, T.; and Efros, A. A. 2017. Image-to-Image Translation with Conditional Adversarial Networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2017-Janua, 5967–5976. IEEE. ISBN 978-1-5386-0457-1.

Jiang, Z.; and Sweetser, P. 2021. GAN-Assisted YUV Pixel Art Generation. In *Australasian Joint Conference on Artificial Intelligence*, 1–12.

Karras, T.; Laine, S.; and Aila, T. 2018. A Style-Based Generator Architecture for Generative Adversarial Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(12): 4217–4228.

Khalifa, A.; Bontrager, P.; Earle, S.; and Togelius, J. 2020. PCGRL: Procedural content generation via reinforcement learning. *Proceedings of the 16th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, AIIDE 2020*, 95–101.

Kopf, J.; and Lischinski, D. 2011. Depixelizing pixel art. In *ACM SIGGRAPH 2011 papers on - SIGGRAPH '11*, volume 30, 1. New York, New York, USA: ACM Press. ISBN 9781450309431.

Liapis, A.; Smith, G.; and Shaker, N. 2016. Mixed-initiative content creation. In *Procedural Content Generation in Games: A Textbook and an Overview of Current Research*, chapter 11. Springer.

Mirza, M.; and Osindero, S. 2014. Conditional Generative Adversarial Nets. *arXiv preprint arXiv:1411.1784*.

Pang, Y.; Lin, J.; Qin, T.; and Chen, Z. 2021. Image-to-Image Translation: Methods and Applications. *IEEE Transactions on Multimedia*, 1–1.

Radford, A.; Metz, L.; and Chintala, S. 2015. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*.

Serpa, Y. R.; and Rodrigues, M. A. F. 2019. Towards Machine-Learning Assisted Asset Generation for Games: A Study on Pixel Art Sprite Sheets. In *2019 18th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*, volume 2019-Octob, 182–191. Rio de Janeiro: IEEE. ISBN 978-1-7281-4637-9.

Silber, D. 2015. *Pixel Art for Game Developers*. CRC Press. ISBN 9781482252316.

Simpson, D. G. 1987. Minimum Hellinger distance estimation for the analysis of count data. *Journal of the American Statistical Association*, 82(399): 802–807.

Summerville, A.; Snodgrass, S.; Guzdial, M.; Holmgård, C.; Hoover, A. K.; Isaksen, A.; Nealen, A.; and Togelius, J. 2018. Procedural content generation via machine learning (PCGML). *IEEE Transactions on Games*, 10(3): 257–270.

Yang, R.; Wang, Y.; Wang, Y.; and Xu, M. 2022. Application of Neural Network in Pixel Art Creation: Bi-directional Conversion between Photo and Pixel Art with GAN Base Model. In *2022 2nd International Conference on Consumer Electronics and Computer Engineering (ICCECE)*, 855–858. Institute of Electrical and Electronics Engineers (IEEE).

Zhu, J.-Y.; Park, T.; Isola, P.; and Efros, A. A. 2017. Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, 2242–2251. IEEE. ISBN 978-1-5386-1032-9.