A Missing Data Imputation GAN for Character Sprite Generation

Flávio Coutinho[1][2], **Luiz Chaimowicz**[1]

[1]Departamento de Ciência da Computação
Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte – MG – Brazil

[2]Departamento de Computação
Centro Federal de Educação Tecnológica de Minas Gerais (CEFET-MG)
Belo Horizonte – MG – Brazil

fegemo@cefetmg.br, chaimo@dcc.ufmg.br

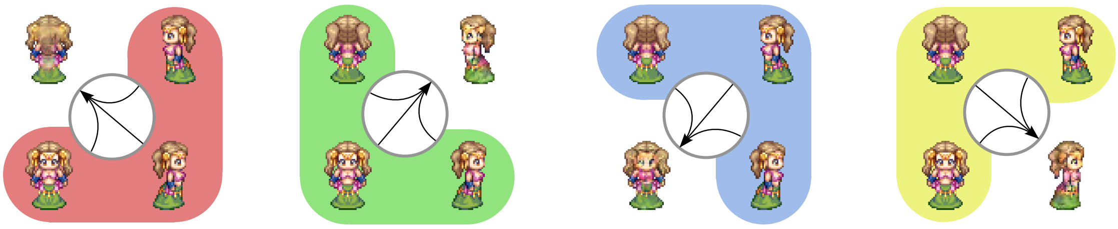

Figure 1. Our model imputes a character's missing pose collaboratively using all the available information from other domains.

Abstract. *Creating and updating pixel art character sprites with many frames spanning different animations and poses takes time and can quickly become repetitive. However, that can be partially automated to allow artists to focus on more creative tasks. In this work, we concentrate on creating pixel art character sprites in a target pose from images of them facing other three directions. We present a novel approach to character generation by framing the problem as a missing data imputation task. Our proposed generative adversarial networks model receives the images of a character in all available domains and produces the image of the missing pose. We evaluated our approach in the scenarios with one, two, and three missing images, achieving similar or better results to the state-of-the-art when more images are available. We also evaluate the impact of the proposed changes to the base architecture.*

Keywords *Generative Adversarial Networks, Procedural Content Generation, Image-to-Image Translation, Missing Data Imputation, Character Sprites.*

1. Introduction

Asset creation is a vital part of the game development process, and it usually takes up a large portion of the project schedule. In particular, the task of character design is seldom executed in a forward-only way, typically involving a lot of going back and forth [Schreier 2017]. In pixel art games, in which the color of each pixel is thoughtfully picked, even small changes to a character might require updating many sprites, especially if characters can face multiple directions and contain different animation sequences spanning many frames [Silber 2015].

Despite the character creation process requiring high creativity and being an established and well-suited responsibility for artists, some involved tasks can become repetitive. For instance, creating normal maps [Moreira et al. 2022] from colored sprites, designing every animation frame [Coutinho and Chaimowicz 2022a], or propagating changes to the many sprites of a character. In that context, recent Procedural Content Generation techniques can help streamline the pipeline, particularly those involving Machine Learning (PCGML). Different works approached character generation through PCGML techniques using Variational Autoencoders (VAEs) [Loftsdottir and Guzdial 2022, Saravanan and Guzdial 2022], Generative Adversarial Networks (GANS) [Hong et al. 2019, Coutinho and Chaimowicz 2022a, Coutinho and Chaimowicz 2024, Serpa and Rodrigues 2019, Choi et al. 2022], and Convolutional Neural Networks (CNNs) [Serpa and Rodrigues 2022], and all of them posed their problems as an image-to-image translation task that generates an image given another (e.g., a normal map from a shaded character). However, if more information is available to the model, it can be leveraged to potentially generate better images.

In this work, we tackle the problem of generating a character sprite in a target pose as a missing data imputation task, using all the images of the character available in other poses. In particular, we propose a model that uses images of pixel art characters in source poses (e.g., facing left, right, back) to impute a missing target direction (e.g., facing front). Figure 1 illustrates our approach.

We propose a generative adversarial network model[1] based on the CollaGAN [Lee et al. 2019] architecture, with changes to the generator topology and the training procedure. Compared to the baselines using the metrics Fréchet Inception Distance (FID) [Heusel et al. 2017] and L_1 distance, the images produced by our model are similar or better than the state-of-the-art. When fewer images are available, the model still produces feasible images, but with less quality. In an ablation study, we show how each of the proposed changes to the original CollaGAN influenced the improved results we achieved.

Thus, our main contributions in this work are:

- a GAN with a single generator/discriminator that can target multiple character poses;
- empirical demonstration that using more of the available information improves the produced sprites; and
- changes to the CollaGAN architecture that enhance the quality of the generated images.

2. Background

In this section we describe some concepts related to generative adversarial networks and then present how such models can approach the image-to-image translation problem.

[1] Source code: `https://github.com/fegemo/mdigan-characters`.

2.1. Generative Adversarial Networks

Goodfellow et al. [2014] introduced the concept of generative adversarial networks (GANs) as a framework for generating content through an adversarial training process. It consists of two models playing different roles in a minimax game: a generator G that evolves to create new content similar to the examples seen during training and a discriminator D that learns to distinguish between real and generated (fake) examples. If an optimal state is reached, G captures the distribution of the training data and can produce new samples that are indistinguishable from the real ones. At the same time, D cannot tell whether an observation is real or fake.

The training algorithm traverses the set of examples for a number of epochs. At each step, G receives a noise prior z and produces new examples, while D is called once to discriminate a minibatch of generated samples and a second time with real ones. The discriminator's loss function $\mathcal{L}_D$ is the mean value (of all examples) of (the log of) the probability of incorrectly labeling an input example x as fake and (the log of) the probability of incorrectly labeling a fake example $G(z)$ as real. D updates its weights *ascending* the stochastic gradient of the following value function, while G updates *descending* it:

$$\min_G \max_D \mathcal{L}(G, D) = \mathbb{E}_x[\log D(x)] + \mathbb{E}_z[\log(1 - D(G(z)))]$$

In this work, we propose a GAN model to generate pixel art character sprites. In particular, it generates a character in a target pose given input images of it facing other source directions. Because it uses multiple images as input to generate a missing one, we approach the problem as a missing data imputation task. However, it can also be regarded as an image-to-image translation problem with multiple images as input and the target as the missing domain. Next, we define the image-to-image translation task and present some of the proposed deep generative architectures using GANs.

2.2. Image-to-Image Translation

Pang et al. [2022] define image-to-image translation as the process of converting an input image x_a in a source domain a to a target b while keeping some intrinsic content from a and transferring it to the extrinsic style of b. The meaning of a domain, style and content differ according to the task. To illustrate, if we want to create a cartoon version (domain b) from pictures of faces (domain a), we are translating faces x_a to b, keeping the person's identity (intrinsic content) but using cartoonish techniques (extrinsic style). Different problems have been approached as image-to-image translation using deep generative models (e.g., GANs, VAEs), such as image colorization [Jiang and Sweetser 2021, Gonzalez et al. 2020], semantic image synthesis [Serpa and Rodrigues 2019, Isola et al. 2017], style transfer [Zhu et al. 2017a], attribute manipulation [Choi et al. 2018, Choi et al. 2020, Lee et al. 2019], and pose transfer [Hong et al. 2019, Coutinho and Chaimowicz 2022a, Coutinho and Chaimowicz 2024].

The diversity of the presented problems involves different characteristics of the task and of the proposed solution. A first important property is the use of **supervision** (label/annotated examples) for training, which largely depends on the availability of such data. For instance, in a translation from grayscale to colored pictures, it is easy to have pixel-wise aligned examples, but that is not the case if we want to transform horses into zebras, as the cost of acquiring completely registered pairs of photos of horses and zebras in the same position in the same environment is impractical. Hence, when paired data is available for some task, we can use **supervised** training [Isola et al. 2017, Lee et al. 2019], whereas when it is not, the algorithm needs to train in an **unsupervised** fashion [Zhu et al. 2017a, Choi et al. 2018, Afifi et al. 2021].

A second characteristic of the tasks is the **number of domains** involved in the translation and how the proposed architecture can deal with them. For instance, many problems consist of only two domains (e.g., grayscale to color, photo to painting, semantic labels to photographs). In contrast, others involve multiple (e.g., translating a neutral face to one smiling, angry, or crying). Hence, the proposed architectures can be **two-domain** [Isola et al. 2017, Zhu et al. 2017a] or **multi-domain**, supporting the translation among all directions [Choi et al. 2018, Choi et al. 2020, Lee et al. 2019]. Additionally, the architectures for two-domain translation can generate images in a single direction [Isola et al. 2017] or in both [Zhu et al. 2017a, Zhu et al. 2017b].

Authors have proposed architectures for tasks with different sets of characteristics. Here, we introduce a model based on the Collaborative GAN (CollaGAN) to generate missing poses of pixel art characters. In our experiments, we compare the proposed architecture to baselines consisting of models based on Pix2Pix [Coutinho and Chaimowicz 2024] and StarGAN [Choi et al. 2018].

Pix2Pix trains with supervision (paired images). It can translate images from one domain into another in a single direction. In contrast, StarGAN trains without supervision but supports multiple domains with a single generator and discriminator pair. CollaGAN, in turn, requires supervision and is multi-domain, with the additional difference that it uses images from multiple domains as input.

3. Related Work

As we investigate the generation of character sprites, we first describe some recent works that tackle the automatic creation of characters. Most also deal with pixel art imagery and use deep generative models. In sequence, we present some works related to the missing data imputation problem, which is how we frame the generation of missing character poses.

3.1. Sprite Generation

Some works propose generating characters in a target pose using a bone graph to indicate the desired positions of each body part. Hong et al. [2019] approached that task with a multiple discriminator GAN (MDGAN). It translates the image of a character (representing its shape and color) and of a target bone-graph sprite into an image of that same character in the new pose. The model consists of a generator and two discriminators, one to determine if two images share the same color and shape, while the other tells whether a character's pose is correct according to some bone-graph sprite.

Similarly, Choi et al. [2022] created a database of character sprites in walking and running animations by feeding video frames of real people into body segmentation networks. Then, they trained a model to generate characters from the body-segmented sprites in arbitrary poses created by users. Albeit successful in their proposed experiments, both systems require tailored datasets that match the positions of characters, making it challenging for the models to generalize, especially for games with different character shapes and movements. In addition, both works use real images in their training sets, which do not conform to characters in typical 2D games, especially those in the pixel art style.

Targeting the generation of in-between frames of animated sketches, Loftsdóttir and Guzdial [2022] propose SketchBetween: a model that takes the initial and final frames of a character animation and sketches of the internal frames, and generates colored versions of the frames in the middle. It takes five images of an incomplete animation as input and provides five images with the rendered sprite animation. Trained on a dataset of cartoon animal animations, it had promising results on shapes and poses similar to the ones from the training set. However, even the higher-quality examples presented the blurriness typical of how VAEs optimize to reduce the average reconstruction error.

Regarding the particularities involved in the generation of pixel art imagery, researchers approached differently: adding specific layers to the generator [Saravanan and Guzdial 2022], framing the problem as a semantic segmentation task [Serpa and Rodrigues 2022, Coutinho and Chaimowicz 2022b], doing post processing steps [Coutinho and Chaimowicz 2024], or adding a histogram loss [Coutinho and Chaimowicz 2022b] while training the generator.

Serpa and Rodrigues [2019] proposed a model based on Pix2pix to generate a grayscale-shaded sprite and another one that segments characters' body parts from rough line art sketches of animation frames from a pixel art fighting game. The generated grayscale sprites were close to the ground truth, but the colored ones diverged, especially for characters in less common poses. In a later iteration of the work [Serpa and Rodrigues 2022], the authors got improved results by framing the problem as a semantic segmentation task and changing the architecture accordingly. The proposed model dropped the adversarial training and used dense connections to increase the network's depth, deep supervision to provide gradients at every step, and a class-weighted focal loss to overcome the class imbalance in the training data.

Saravanan and Guzdial [2022] adapted the Vector-Quantized VAE [van den Oord et al. 2017] to improve the quality of the generated pixel art characters by adding a 1×1 convolution layer pair at the beginning and the end of the encoder and decoder networks. Trained with Pokémon sprites, the model generated embeddings that allowed a PixelCNN [van den Oord et al. 2016] technique to create new images of static characters that tried to follow the training distribution. Using the additional layers helped reduce the blurriness of the generated images.

Investigating the challenges involved in generating pixel art specifically, Coutinho and Chaimowicz [2022] evaluated two hypotheses to improve the quality of the generated images: representing them as indices in a color palette and adding a histogram loss term when training the generator. While the palette representation led to much worse results due to overfitting, penalizing the generator for using colors with a different histogram than the one from the input image yielded slightly improved images.

In [Coutinho and Chaimowicz 2022a], the same authors propose an architecture based on Pix2Pix to translate pixel art characters in a source pose (e.g., looking front) into a target one (e.g., facing right). They trained models in different datasets with under

1k examples. The generated images had varying degrees of quality, with good results for characters more similar to the ones seen during training (e.g., similar shapes or color variations) but bad results for more unique characters. In a later iteration of the work, Coutinho and Chaimowicz [2024] investigated different data augmentation techniques. They proposed a post-processing step to quantize the images to the color palette of the input image. They also assembled a diverse dataset with 14k paired images of characters in four directions and observed that training with much more data yielded better results when validating with the more artistically cohesive and smaller individual datasets.

In this paper, we also tackle the generation of pixel art characters (like [Serpa and Rodrigues 2019, Serpa and Rodrigues 2022, Saravanan and Guzdial 2022]) by translating among different poses (like [Coutinho and Chaimowicz 2022b, Coutinho and Chaimowicz 2022a, Coutinho and Chaimowicz 2024]). However, unlike the other works, we frame the problem as a missing image data imputation task. Hence, instead of a trained model being capable of handling between only two poses (two-domain) and a single direction, our generator receives the images of a character in every available pose and generates an image of it in the one that is missing (multi-domain with multiple inputs).

3.2. Missing Data Imputation

Data analysis can be drastically hindered when relevant parts of information are missing. That can happen for various reasons: data can be absent because it was never collected or produced, it might have been lost, or it might contain errors [Yoon et al. 2018]. Researchers have proposed different missing data imputation techniques to replace absent data with plausible substitutions. The choice of such techniques depends on the data type, among other characteristics. It can be one or a mix of categorical [Yoon et al. 2018, Shang et al. 2017], sequential [Liu et al. 2023], and image [Shang et al. 2017, Lee et al. 2019, Sharma et al. 2019, Shen et al. 2021].

Inaugurating the use of deep learning-based techniques for missing data imputation, Yoon et al. [2018] proposed a generalization of the original GAN to deal with imputing missing values, which they called Generative Adversarial Imputation Nets (GAIN). The generator receives three inputs: the sample with missing values, a mask indicating which values are present, and a random vector of the same dimension that introduces noise. As output, it produces a version of the sample with replaced values for those missing. The discriminator, in turn, tries to distinguish which of the categorical variables are imputed and which are from the original sample.

The imputation task becomes more challenging when the missing data are images due to the higher dimensionality. Some works approach the problem using GANs [Shang et al. 2017, Lee et al. 2019, Sharma et al. 2019, Shen et al. 2021]. An example is the View Imputation GAN (VIGAN) [Shang et al. 2017], that can generate missing values in a target domain by combining a modified CycleGAN [Zhu et al. 2017a] with a Denoising Autoencoder in a three-step training process. A shortcoming of VIGAN is that it performs bi-directional imputation between only two domains. When the task involves more domains, other architectures are better suited. The Multi-Modal GAN (MM-GAN) [Sharma et al. 2019], CollaGAN [Lee et al. 2019], and ReMIC [Shen et al. 2021] can impute missing images among multiple domains and use the information of all available sources as input to the generator.

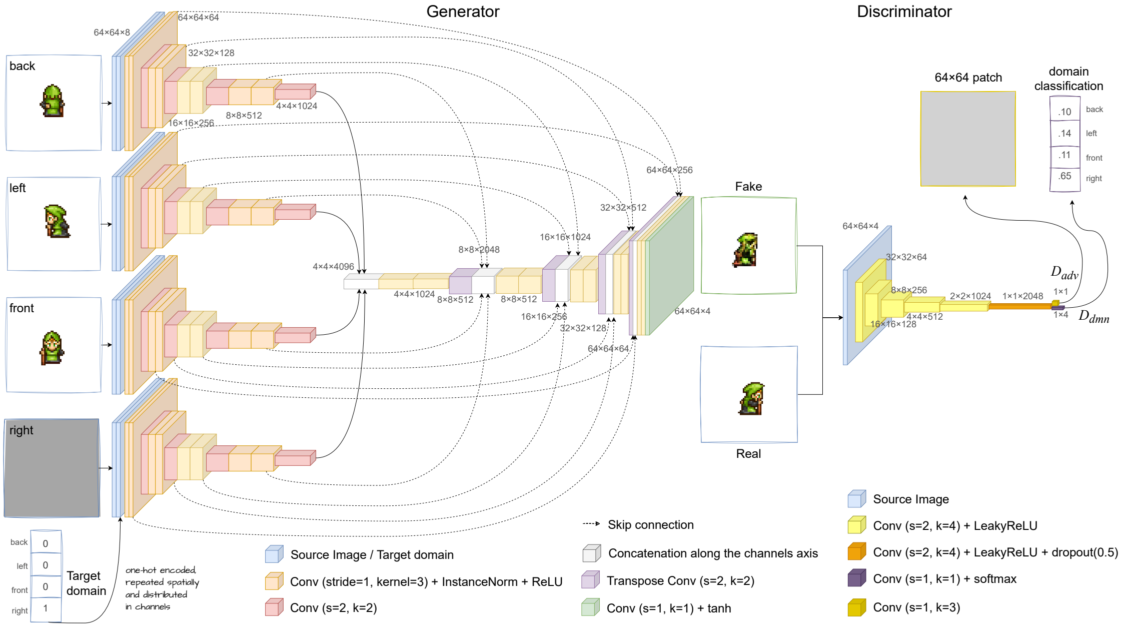

Figure 2. Architecture of the proposed model. LEFT: The generator receives a character in the source domains and a label indicating the target, which is one-hot encoded, spatially spread, and concatenated with each input image. The inputs follow the encoder branches and are concatenated at the bottleneck layer, flowing into the unified decoder. Skip connections provide early outputs to the decoder. RIGHT: The discriminator receives the image (real or fake) that must be distinguished and outputs D_{adv} with the real/fake logit and D_{dmn} with the probabilities of the image being part of each domain.

MM-GAN's generator [Sharma et al. 2019] has an equal number of inputs and outputs, receives samples with images missing in random domains, and outputs imputed values. The discriminator distinguishes between real and imputed same-size patches of a full sample comprising all domains. CollaGAN [Lee et al. 2019] works similarly and was proposed in the same year as MM-GAN. However, it produces an image of a single target domain. Its generator varies depending on the task, but it also receives the images in all available domains, concatenated with the index of the target domain spread spatially and through the channels dimension. Both architectures presented good results in their respective experiments. ReMIC [Shen et al. 2021] also takes the inputs from all available domains and generates the missing ones, like MM-GAN. However, unlike the other two, it disentangles the images and extracts a shared content encoding and a separate style encoding for each domain.

All multi-domain architectures that deal with missing image data imputation [Sharma et al. 2019, Lee et al. 2019, Shen et al. 2021] were tested with medical or natural images, but not with pixel art or other styles. In the next section, we introduce a modified architecture based on CollaGAN to generate missing pixel art characters.

4. Architecture

We propose an architecture based on CollaGAN [Lee et al. 2019] to impute images of pixel art characters in a missing pose (target domain). To facilitate understanding, let us consider that there are domains $N = \{a, b, c, d\}$, one representing each pose. The architecture consists of a single generator and discriminator pair, of which the former creates an image $\hat{x}_t$ of a character in the missing pose t using the available images from all of the other source S poses:

$$\hat{x}_t = G(x_S, t), \text{with } t \in N, S = N - \{t\}$$

Our generator has one encoder branch to process the input from each domain, a single decoder branch with concatenated skip connections, and outputs an image in the missing domain. The discriminator distinguishes images as real or fake, as well as determines their domain through an auxiliary classifier output. Figure 2 shows both network topologies.

4.1. Objective Function

As usually done with GANs, we train both networks adversarially, but also with additional objectives. The generator's loss function has five terms: regressive, cycle consistency, structural similarity, adversarial, and domain classification. In turn, the discriminator trains with adversarial and domain classification objectives.

Training requires a forward and a backward pass. In the first step, a minibatch of paired images with a random missing domain t is fed to the generator G, which synthesizes an image corresponding to the missing t domain. For example, if $S = \{a, b, c\}$ and $t = d$, the images x_a, x_b, x_c are available and we want the model to generate $\hat{x}_d$ as close as possible to the real x_d:

$$\hat{x}_d = G(\{x_a, x_b, x_c, x_{\text{zero}}\}, d),$$

in which x_{zero} is a tensor filled with zeros with the same dimension of an input image.

Subsequently, to ensure cycle consistency, the backward step comprises synthesizing $|N| - 1$ images with each domain in $S = \{a, b, c\}$ as a target, using the generated $\hat{x}_d$ instead of the real x_d. The outputs of this pass, in our example, would be:

$$\begin{aligned}\tilde{x}_{a|d} =& G(\{x_{\text{zero}}, x_b, x_c, \hat{x}_d\}, a)\\ \tilde{x}_{b|d} =& G(\{x_a, x_{\text{zero}}, x_c, \hat{x}_d\}, b)\\ \tilde{x}_{c|d} =& G(\{x_a, x_b, x_{\text{zero}}, \hat{x}_d\}, c),\end{aligned}$$

and should reconstruct the original images x_a, x_b, and x_c.

A regressive loss term $\mathcal{L}_{reg}$ steers the generator towards using the information from the source domains to translate an image to the target, whereas a multiple cycle consistency loss $\mathcal{L}_{mcyc}$ leads it into encoding in $\hat{x}_t$ enough information to allow cyclical reconstruction of the original inputs. Both losses are pixel-wise L_1 distances between the generated and the real images:

$$\mathcal{L}_{reg} = \mathbb{E}_{x_t, x_S}[||x_t - \hat{x}_t||_1]$$

$$\mathcal{L}_{mcyc} = \mathbb{E}_{x_t, x_S}[\sum_{s \in S} ||x_s - \tilde{x}_{s|t}||_1]$$

Besides $\mathcal{L}_{mcyc}$, an additional objective $\mathcal{L}_{ssim}$ is used to improve the quality of the images generated in the backward pass. It uses the structural similarity index measure

(SSIM) [Wang et al. 2004] to compose a loss term between the cyclically generated $\tilde{x}_S$ and the real source images x_S. Its formulation is the same as in the CollaGAN paper and is omitted here for brevity.

The discriminator also uses the other two objectives for the generator: adversarial and domain classification. The adversarial loss uses the one from Least Squares GAN [Mao et al. 2017], which optimizes the square of the errors of the discriminator classification of real and fake images. The discriminator $\mathcal{L}_{adv}^{D}$ and generator $\mathcal{L}_{adv}^{G}$ adversarial losses are:

$$\begin{aligned}\mathcal{L}_{adv}^{D} =& \mathbb{E}_{x_t}[(D_{adv}(x_t)-1)^2] + \mathbb{E}_{\tilde{x}_{s|t}}[(D_{adv}(\tilde{x}_{s|t}))^2]\\ \mathcal{L}_{adv}^{G} =& \mathbb{E}_{\tilde{x}_{s|t}}[(D_{adv}(\tilde{x}_{s|t})-1)^2]\end{aligned}$$

The domain classification objective leads the generator to synthesize images classified as having the intended target domain. For the generator, $\mathcal{L}_{dmn}^{fake}$ considers only generated images, whereas for the discriminator, $\mathcal{L}_{dmn}^{real}$ uses only real images. As a classification, they are calculated using cross entropy, given as:

$$\begin{aligned}\mathcal{L}_{dmn}^{real} =& \mathbb{E}_{x_t}[-\log(D_{dmn}(x_t))]\\ \mathcal{L}_{dmn}^{fake} =& \mathbb{E}_{\hat{x}_t}[-\log(D_{dmn}(\hat{x}_t))]\end{aligned}$$

To summarize, the full objectives of the generator $\mathcal{L}_G$ and the discriminator $\mathcal{L}_D$ are sums weighted by λ scalars given as:

$$\begin{aligned}\mathcal{L}_G =& \mathcal{L}_{adv}^{G} + \lambda_{reg}\mathcal{L}_{reg} + \lambda_{mcyc}\mathcal{L}_{mcyc} + \lambda_{ssim}\mathcal{L}_{ssim} + \lambda_{dmn}\mathcal{L}_{dmn}^{fake}\\ \mathcal{L}_D =& \mathcal{L}_{adv}^{D} + \lambda_{dmn}\mathcal{L}_{dmn}^{real}\end{aligned}$$

4.2. Generator

The generator has four encoder branches, each receiving a source image from a particular domain and a channelized and spatially spread one-hot encoded label of the target domain. There are four downsampling blocks for each branch and a bottleneck layer that concatenates the activation maps from all encoder branches and further processes it. The data is then passed onto a single decoder composed of four upsampling blocks. There are skip connections from the concatenated activation maps (across the encoder branches) from downsampling to the respective upsampling blocks. Compared to the original architecture, our generator contains four encoder branches, while theirs has eight.

The image size and number of channels we use is $64 \times 64 \times 4$, contrasted to the $128 \times 128 \times 3$ configuration of the original architecture. We increased the number of channels for each layer to improve the network capacity: they are four times the original, becoming 64, 128, 256, 512, and 1024 for the blocks in each encoder branch, and 1024, 512, 256, 128, and 64 for the decoder blocks.

4.3. Discriminator

The discriminator receives a batch of images and outputs values D_{adv} that should be one for real images and zero for the generated ones. In addition, it classifies the domain of the image, yielding probabilities D_{dmn} of images having each domain.

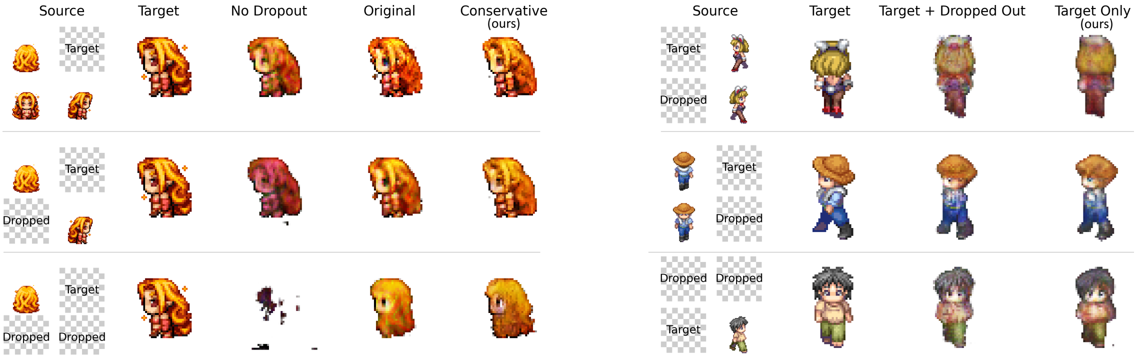

Figure 3. Comparison of input dropout (left) and replacement procedures (right) during training in the proposed model.

The network topology is the same as in the original, with 6 downsampling blocks, each consisting of a convolution that halves the resolution while increasing the number of channels, with a leaky ReLU activation. The last block also contains a dropout layer. Following it, two parallel convolutions represent the D_{adv} and the D_{dmn} outputs, with linear and softmax activations, respectively.

4.4. Training Procedure

At each training step, we select a batch of paired images x_S with random target domains t. The generator receives the batch of $\langle x_S, t\rangle$ and creates the missing $\hat{x}_t$, in the forward pass. Next, $\hat{x}_t$ is used in place of x_t to create a number of new batches equal to $|N|-1$, in which each domain in S becomes the target, in the backward (or cyclical) pass. The generator then creates $\tilde{x}_{s|t}, \forall s \in S$ that must be as close as possible to the original $x_s, \forall s \in S$.

The CollaGAN architecture authors observed that images are much worse as the number of available sources decreases. However, it is common to have use cases in which more than one domain is missing. Hence, they proposed a batch selection strategy called input dropout, in which the model trains with one or more missing domains. For instance, for $|N| = 4$ and $t = d$, when a batch $\langle x_S, t\rangle$ is selected using the input dropout strategy, x_S can have zero, one or two withdrawn images and be one of the following:

$$\begin{array}{lll} x_S = \{x_a, x_b, x_c\} & x_S = \{x_{zero}, x_b, x_c\} & x_S = \{x_{zero}, x_{zero}, x_c\} \\ & x_S = \{x_a, x_{zero}, x_c\} & x_S = \{x_{zero}, x_b, x_{zero}\} \\ & x_S = \{x_a, x_b, x_{zero}\} & x_S = \{x_a, x_{zero}, x_{zero}\} \end{array}$$

In the original CollaGAN, the number of images to be dropped out is chosen uniformly, leaving a 33% chance of having the full source domain set. That strategy improved the results in our task too. However, we observed that a more conservative approach in which the model trains more frequently dropping out few images yields even better results in the scenario of having fewer available images. We adopted chances of 10%, 30%, and 60% to have two, one, and zero images dropped out. Figure 3 (left) compares the three strategies (no dropout, original dropout, conservative dropout) with different numbers of missing images.

Another change we made to the training procedure relates to the backward generation pass. When the cycled images $\tilde{x}_{s|t}, \forall s \in S$ (e.g., $\tilde{x}_{a|d}$, $\tilde{x}_{b|d}$, and $\tilde{x}_{c|d}$) are generated in the original implementation, the image $\hat{x}_t$ produced in the forward pass replaces not only the original target image x_t, but also all images that have been dropped out due to the batch selection strategy. We experimented with having $\hat{x}_t$ replace only x_t and observed better results.

To illustrate the difference, considering a batch with $t = d$ and the domain c dropped out, the backward generated images $\tilde{x}_{s|t}, \forall s \in S$ for the original (left) and our implementation (right) would be as shown next. We highlighted the differences in color:

$$\begin{aligned}
\tilde{x}_{a|d} &= G(\{x_{\text{zero}}, x_b, \hat{x}_d, \hat{x}_d\}, a) & \qquad \tilde{x}_{a|d} &= G(\{x_{\text{zero}}, x_b, x_{\text{zero}}, \hat{x}_d\}, a)\\
\tilde{x}_{b|d} &= G(\{x_a, x_{\text{zero}}, \hat{x}_d, \hat{x}_d\}, b) & \qquad \tilde{x}_{b|d} &= G(\{x_a, x_{\text{zero}}, x_{\text{zero}}, \hat{x}_d\}, b)\\
\tilde{x}_{c|d} &= G(\{x_a, x_b, x_{\text{zero}}, \hat{x}_d\}, c) & \qquad \tilde{x}_{c|d} &= G(\{x_a, x_b, x_{\text{zero}}, \hat{x}_d\}, c)
\end{aligned}$$

Figure 3 (right) compares the generated images when the model trains using the original replacement procedure for the dropped-out images versus our version where only the forward target image is replaced by the one generated in the forward pass. We can note that with the original procedure, the generator produces images with artifacts from domains other than the target one, mostly noticeable through the wrong number of eyes in the examples.

Regarding the number of trainable parameters, the generator contains 104,887,616 values, and the discriminator has 44,726,272. After training, the generator takes ≈110 ms to produce an image using a GeForce GTX 1050 GPU.

5. Methodology

We start by presenting the datasets used in the experiments to propose and evaluate models for translating pixel art characters in different poses. Next, we describe the metrics L_1 and FID used to analyze the quality of the generated images using each model. Finally, we conclude the section by presenting the baseline models used in the experiments.

5.1. Dataset

Unlike tasks that are more commonly tackled in Computer Vision research, we found only one character sprite dataset readily available: TINY HERO[2], which contains 912 paired images of characters facing the back, left, front, and right directions. To increase the number of training examples, we scraped character sprite sheets from different sources from the web, splitting them into individual character sprites, and generated characters modularly by assembling various parts. The dataset contains 14,202 paired images of characters in four directions spanning different art styles. They primarily comprise humanoid characters of different sizes and art styles, but also a few sprites of animals, vehicles, and monsters. Figure 4 shows examples depicting the high variability of the samples.

Images from each source had different character sizes, so the smaller ones were transparency-padded to the largest size, 64×64. We also created an alpha channel with the character shape for the images that lacked one. The training set contains 12,074 examples, and the test set contains 2,128 examples (85% split). During training, we applied hue rotation to each character as data augmentation.

[2] Dual license of GNU GPL 3.0 and CC-BY-SA 3.0. Source: https://lpc.opengameart.org/

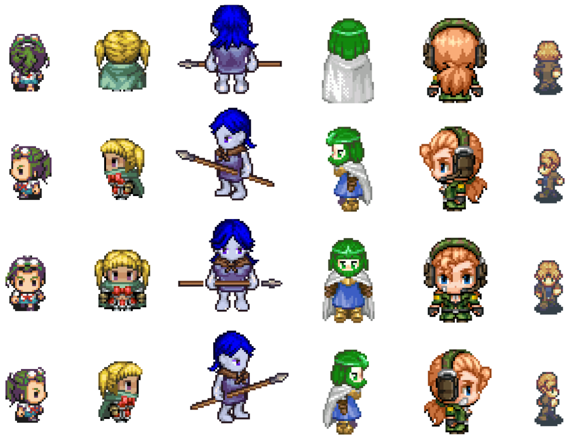

Figure 4. Sample images from the dataset showing different sizes/art styles (columns) facing four directions (rows).

5.2. Evaluation Metrics

The evaluation of generative models is an active research problem with different metrics proposed over the recent years [Buzuti and Thomaz 2023]. We evaluate the quality of a model by how close the generated images are to their ground truth. However, a qualitative analysis is important as the metrics do not always converge.

Hence, we analyze the results qualitatively through visual inspection and quantitatively using the L_1 distance to the target images and the Fréchet Inception Distance (FID) [Heusel et al. 2017]. The L_1 distance measures the absolute difference between the colors of pixels of two image sets (the generated and target). In turn, FID uses the Inception v3 network (proposed for image classification) to get the distance between the feature vectors of the two image sets [Szegedy et al. 2015]. As both metrics are distances, they are zero for identical generated and target images, so lower numbers are better.

5.3. Baseline Models

We compare our model with two other architectures proposed for the image-to-image translation task: Pix2Pix [Isola et al. 2017] and StarGAN [Choi et al. 2018].

Pix2Pix. We trained a modified version of the architecture proposed in [Coutinho and Chaimowicz 2024] for generating pixel art characters in a target pose given an image of it in a source one. Differently from the referenced work, we use 12 such models to support translation from and to all four poses: back, left, right, and front, excluding models from and to the same direction. Each generator has 29,307,844 trainable variables, so the model collection contains 351,694,128 parameters.

StarGAN. We trained a StarGAN-based model to perform multi-domain translation using a single generator and discriminator pair. The generator typically receives the source image and a label indicating the target direction. Still, we found that providing a label of the *source* domain increases the quality of the generated characters. In turn, the original critic receives only the image to be evaluated, but we got better results by sending the source image too (before translation), which makes it perform a *conditional* discrimination. In that case, the network indicates whether the provided image is real/fake *considering that* it is a translation of the source image. For a fair comparison, we train

Table 1. FID and L_1 of our CollaGAN-based model receiving three images and a single for Pix2Pix and StarGAN

Target	Average FID			Average L_1		
	Pix2Pix	StarGAN	CollaGAN-3	Pix2Pix	StarGAN	CollaGAN-3
Back	5.788	3.378	**2.054**	0.05402	0.06429	**0.04530**
Left	2.380	1.250	**1.037**	0.04934	0.06344	**0.03439**
Front	5.392	3.156	**1.955**	0.05875	0.07263	**0.04985**
Right	2.806	1.368	**0.987**	0.04880	0.06273	**0.03360**
Average	4.091	2.288	**1.508**	0.05273	0.06577	**0.04078**

the model using supervision (the original trains without paired images). The generator contains 134,448,128 parameters.

6. Experiments

The model trained with the pixel art characters dataset for 240,000 generator update steps in minibatches of 4 examples, which is equivalent to ≈80 epochs. It took 01:20h to train using a GeForce GTX 1050 GPU. We used early stopping to select the model that had the best metrics on its test set instead of getting the one in the end to prevent overfitting. At every 1,000 update steps, we evaluate the model and select the one with the lowest (best) L_1 value throughout the training procedure. After training, it takes 110.03ms for the model to generate a batch of images.

The generator and discriminator optimize their weights using Adam, with $\beta_1 = 0.5$ and $\beta_2 = 0.999$, and a learning rate that starts as 0.0001 and linearly decays to zero during the second half of the training. The parameters of the objective function were $\lambda_{reg} = 100$, $\lambda_{dmn} = 10$, $\lambda_{ssim} = 10$, and $\lambda_{mcyc} = 10$.

In the following experiments, we start by evaluating the model's performance using three input images (dubbed CollaGAN-3) against the baselines. Next, we evaluate the same model (trained with three source domains) in the scenario of it receiving only two (CollaGAN-2) and one (CollaGAN-1) input images. We then follow up with an experiment to assess different input dropout strategies and an ablation study of the changes proposed atop the original CollaGAN.

6.1. Missing Image Imputation

We trained our proposed model using conservative input dropout and the forward-only replacer strategy. Table 1 shows the values of FID and L_1 for our proposed model and the baselines, with the rows representing the target pose and the columns displaying the metrics for the baselines Pix2Pix and StarGAN, averaged considering the translation from the other source domains and CollaGAN with the three other domains as input.

Regarding FID, CollaGAN-3 had the lowest (best) values of the evaluated models in all target poses and, hence, on average too: 1.508 (CollaGAN-3) versus 2.288 (StarGAN) and 4.091 (Pix2Pix). Also, the L_1 distance for CollaGAN was the lowest (best), with the averages: 0.04078 (CollaGAN-3), 0.05273 (Pix2Pix), and 0.06577 (StarGAN). Next, we visually analyze the generated images.

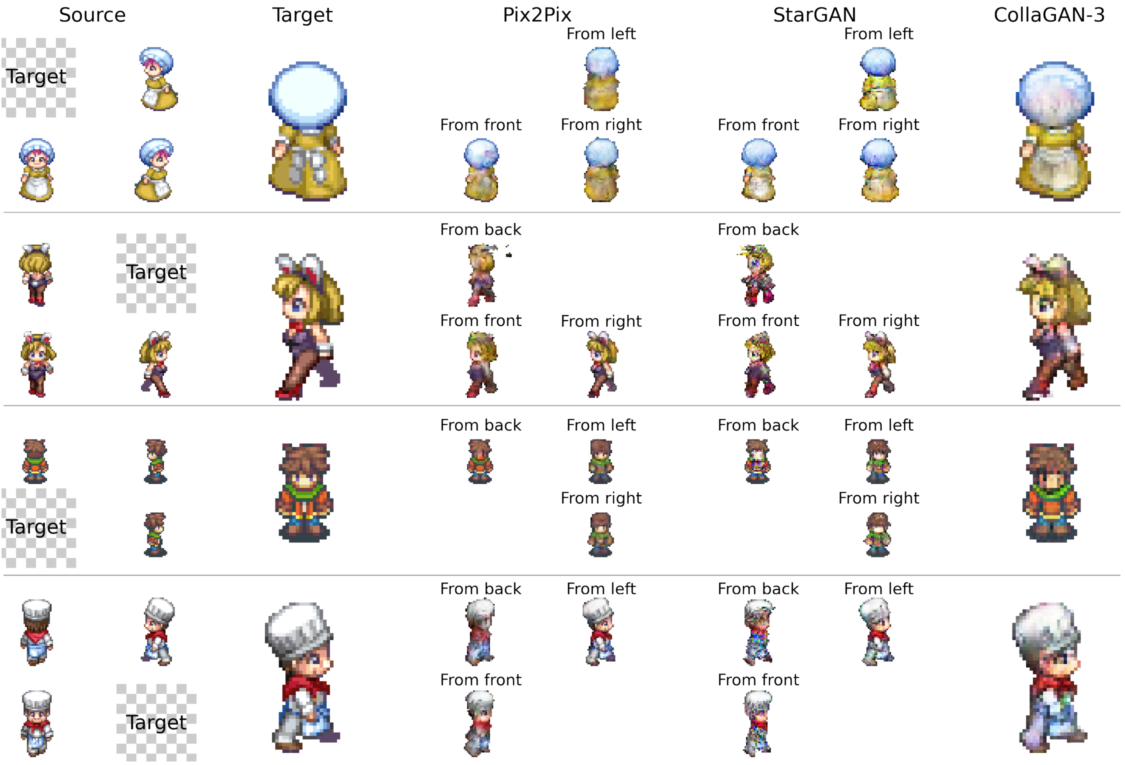

Figure 5. Example images generated in different target domains. The columns show the source images, the target, the generation with the baselines using different source domains, and the generation using all sources with CollaGAN.

Figure 5 shows examples of generated images using the different models, with each row having a different domain as the target. The columns for Pix2Pix and StarGAN show three images per row. As they are models that take a single image as input, we depict the image generated for the target pose from each of the other domains. In contrast, CollaGAN uses all the other domains as input and, hence, has a single generated image for each row.

The quality of the generated images varies with the model and the target pose. We analyze the results qualitatively according to the use of colors and the generated shape. Regarding the former, all models generate images with colors in meaningful positions but employ many variations of the same tones instead of restricting to a small palette. Such undesired behavior can be attenuated by quantizing the colors to the palette of the input images, such as done in [Coutinho and Chaimowicz 2024] in a post-processing step.

Regarding the shape, the poses imputed by CollaGAN are very close to the intended, and so are the ones generated by the baseline models that translate images from left to right and vice-versa. Generating images in that scenario usually consists of learning a horizontal flip transformation, which is an easier task endorsed by the lower FID and L_1 values when the target is left or right. When the target is back, a noticeable artifact is the faint presence of details from the character's face, especially prominent in the images generated with CollaGAN.

Visually inspecting the results shows that the quality of the images generated by

Table 2. FID and L_1 metrics in the scenarios of receiving three, two, and one images and the baselines

Model/Sources	Average FID	Average L_1
Pix2Pix	4.091	0.05273
StarGAN	2.288	0.06577
CollaGAN-1	8.393	0.06449
CollaGAN-2	4.277	0.05035
CollaGAN-3	1.508	0.04078

our model is either on par or better than the baselines. We highlight that the CollaGAN-based architecture contains 104,887,616 trainable parameters, which is 22% smaller than StarGAN and 70% than the collective Pix2Pix. Next, we assess how the model performs when less images are available.

6.2. Generating from Fewer Domains

Even though we propose a model to impute a single missing domain, we also evaluate it in scenarios where it receives two (CollaGAN-2) or only one image (CollaGAN-1). The metrics' values are averaged among all targets and all available sources for each model and scenario (i.e., CollaGAN-3, 2, and 1).

Table 2 compares the proposed model in those situations. We can observe that both FID and L_1 metrics progressively improve as the number of available domains increases, with CollaGAN-2 still having better L_1 than Pix2Pix and StarGAN.

6.3. Input Dropout

We evaluated the impact of different batch selection strategies on presenting examples to the proposed model: Should it always see the three available domains, or should they sometimes be omitted?

We investigated always showing all available domains (none), the original input dropout strategy proposed in [Lee et al. 2019], a curriculum learning approach suggested by Sharma et al. [2019], and our proposed conservative tactic. The original approach has an equal chance of presenting three, two, or a single image in a training step. The curriculum learning approach starts training with easier tasks (using three images) and progressively makes it harder (using a single input) until half of the training, then it randomly chooses between the number of domains to drop out for the second part. Lastly, the conservative approach randomly selects the number of images to drop, but with higher probabilities to keep more images: 60% with 3 images, 30% with 2, and 10% with a single image.

Table 3 presents the results from the models trained with the different input dropout strategies (columns) in the scenarios of having three, two, or one available image as input (rows). We can observe that using any input dropout yields better results than always showing all domains (none). Compared to the original and curriculum learning strategies, our proposed conservative tactic has better FID and L_1 metrics on the average of the three scenarios. In particular, regarding FID, the model trained with the conservative input dropout worsens its performance less drastically with the decrease of input

Table 3. FID and L_1 of different input dropout strategies when the model receives 3, 2 or 1 images as input

	Sources	None	Original	Curric.	Conserv.
FID	CollaGAN-3	4.816	1.911	2.160	**1.508**
	CollaGAN-2	19.050	6.835	9.233	**4.277**
	CollaGAN-1	32.676	11.162	20.303	**8.393**
	Average	18.847	6.636	10.566	**4.726**
L_1	CollaGAN-3	0.04523	0.04277	0.04222	**0.04078**
	CollaGAN-2	0.08003	0.05053	0.07389	**0.05035**
	CollaGAN-1	0.12820	**0.06243**	0.12232	0.06449
	Average	0.08449	0.05191	0.07948	**0.05187**

Table 4. Performance of the modifications made to the original CollaGAN architecture

Modification (cumulative)	Average FID		Average L_1	
	Value	Improv.	Value	Improv.
Original	8.866	—	0.06069	—
+ Increased capacity	11.078	-24.95%	0.05666	6.64%
+ Forward Replacer	6.636	25.15%	0.05191	14.47%
+ Conservative Inp. Drop.	4.726	46.70%	0.05187	14.53%

domains. Regarding L_1, its metrics are better than the other models when two and three images are available.

6.4. Ablation Study

To understand the impact of our changes to the original CollaGAN architecture, we trained and evaluated models that progressively added each modification. Table 4 shows the FID and L_1 values of the generated images averaged over all domains and among the scenarios of the model receiving three, two, and one input domains. The rows show the results of each modification cumulatively: the first one is the original CollaGAN model without any of our proposed changes, the second introduces the first modification, the third uses two changes, and the last includes all three (our final model).

The original model had 6,565,712 trainable variables, but with the increased capacity, there are 104,887,616 parameters. That change alone improved L_1 but worsened FID. The replacement strategy of substituting only the original target with the image generated in the forward step improves both metrics' results. Lastly, training with the proposed conservative input dropout further enhances the results, with FID and L_1 values that are 46.7% and 14.53% better than the original architecture.

7. Final Remarks

We posed the task of generating pixel art characters as a missing data imputation problem and approached it using a deep generative model. It is based on the CollaGAN architecture, from which we proposed changes involving a capacity increase, a conservative input dropout strategy, and a different replacement tactic during the backward step of the training procedure. The experiments showed that all of the changes contributed to achieving better results.

Compared to the baseline models, our approach produces images with similar or better quality when using three domains as input. The model can still produce feasible images in scenarios with fewer available images but with increasingly lower quality.

In future work, we propose the study of other missing image imputation architectures to the same task tackled here, such as ReMIC [Shen et al. 2021] and MM-GAN [Sharma et al. 2019]. Differently from CollaGAN, both methods can receive and generate images in any number of domains. Another line of investigation is to approach the task with architectures that disentangle the source images into content and style codes [Huang et al. 2018] and also latent diffusion models [Rombach et al. 2021]. An interesting outcome of such architectures is their multi-modality nature, in that they can generate different suggestions for the same input.

8. Acknowledgments

This work was partially supported by CAPES, CNPq and Fapemig.

References

Afifi, M., Brubaker, M. A., and Brown, M. S. (2021). HistoGAN: Controlling Colors of GAN-Generated and Real Images via Color Histograms. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7941–7950, Virtual. IEEE Computer Society.

Buzuti, L. F. and Thomaz, C. E. (2023). Fréchet AutoEncoder Distance: A new approach for evaluation of Generative Adversarial Networks. *Computer Vision and Image Understanding*, 235:1–11.

Choi, J.-I., Kim, S.-K., and Kang, S.-J. (2022). Image Translation Method for Game Character Sprite Drawing. *Computer Modeling in Engineering & Sciences*, 131(2):747–762.

Choi, Y., Choi, M., Kim, M., Ha, J.-W., Kim, S., and Choo, J. (2018). StarGAN: Unified Generative Adversarial Networks for Multi-domain Image-to-Image Translation. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8789–8797, Salt Lake City. IEEE.

Choi, Y., Uh, Y., Yoo, J., and Ha, J.-W. (2020). StarGAN v2: Diverse Image Synthesis for Multiple Domains. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8185–8194, Seattle. IEEE.

Coutinho, F. and Chaimowicz, L. (2022a). Generating Pixel Art Character Sprites using GANs. In *2022 21st Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*, pages 1–6, Natal, Brazil. IEEE.

Coutinho, F. and Chaimowicz, L. (2022b). On the Challenges of Generating Pixel Art Character Sprites Using GANs. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 18(1):87–94.

Coutinho, F. and Chaimowicz, L. (2024). Pixel art character generation as an image-to-image translation problem using GANs. *Graphical Models*, 132:101213.

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*, pages 1–7, Worcester. ArXiV.

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, pages 2672–2680, Cambridge. MIT Press.

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.

Huang, X., Liu, M. Y., Belongie, S., and Kautz, J. (2018). Multimodal Unsupervised Image-to-Image Translation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11207 LNCS:179–196.

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, pages 5967–5976, Honolulu. IEEE.

Jiang, Z. and Sweetser, P. (2021). GAN-Assisted YUV Pixel Art Generation. In *Australasian Joint Conference on Artificial Intelligence*, pages 1–12, Sydney. Springer International Publishing.

Lee, D., Kim, J., Moon, W.-J., and Ye, J. C. (2019). CollaGAN: Collaborative GAN for Missing Image Data Imputation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2019-June, pages 2482–2491, Long Beach. IEEE.

Liu, J., Pasumarthi, S., Duffy, B., Gong, E., Datta, K., and Zaharchuk, G. (2023). One Model to Synthesize Them All: Multi-Contrast Multi-Scale Transformer for Missing Data Imputation. *IEEE Transactions on Medical Imaging*, 42(9):2577–2591.

Loftsdottir, D. and Guzdial, M. (2022). SketchBetween: Video-to-Video Synthesis for Sprite Animation via Sketches. In *Proceedings of the 17th International Conference on the Foundations of Digital Games*, pages 1–7, New York. ACM.

Mao, X., Li, Q., Xie, H., Lau, R. Y., Wang, Z., and Smolley, S. P. (2017). Least Squares Generative Adversarial Networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2813–2821, Venice. IEEE.

Moreira, R. D., Coutinho, F., and Chaimowicz, L. (2022). Analysis and Compilation of Normal Map Generation Techniques for Pixel Art. In *2022 21st Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*, pages 1–6, Natal. IEEE.

Pang, Y., Lin, J., Qin, T., and Chen, Z. (2022). Image-to-Image Translation: Methods and Applications. *IEEE Transactions on Multimedia*, 24:3859–3881.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. (2021). High-Resolution Image Synthesis with Latent Diffusion Models. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2022-June:10674–10685.

Saravanan, A. and Guzdial, M. (2022). Pixel VQ-VAEs for Improved Pixel Art Representation. In *Experimental AI in Games Workshop (EXAG) 2022*, pages 1–9, Pomona. ArXiv.

Schreier, J. (2017). *Blood, Sweat, and Pixels: The Triumphant, Turbulent Stories Behind How Video Games Are Made*. HarperCollins, New York.

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, pages 182–191, Rio de Janeiro. IEEE.

Serpa, Y. R. and Rodrigues, M. A. F. (2022). Human and machine collaboration for painting game assets with deep learning. *Entertainment Computing*, 43:100497.

Shang, C., Palmer, A., Sun, J., Chen, K.-S., Lu, J., and Bi, J. (2017). VIGAN: Missing view imputation with generative adversarial networks. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 766–775, Boston. IEEE.

Sharma, A., Member, S., Hamarneh, G., and Member, S. (2019). Missing MRI Pulse Sequence Synthesis using Multi-Modal Generative Adversarial Network. *IEEE Transactions on Medical Imaging*, 39(4):1170–1183.

Shen, L., Zhu, W., Wang, X., Xing, L., Pauly, J. M., Turkbey, B., Harmon, S. A., Sanford, T. H., Mehralivand, S., Choyke, P. L., Wood, B. J., and Xu, D. (2021). Multi-Domain Image Completion for Random Missing Input Data. *IEEE Transactions on Medical Imaging*, 40(4):1113–1122.

Silber, D. (2015). *Pixel Art for Game Developers*. CRC Press, Boca Raton.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2015). Rethinking the Inception Architecture for Computer Vision. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016-December:2818–2826.

van den Oord, A., Kalchbrenner, N., and Kavukcuoglu, K. (2016). Pixel Recurrent Neural Networks Koray Kavukcuoglu. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, pages 1747–1756, New York, NY, USA. JMLR.org.

van den Oord, A., Vinyals, O., and Kavukcuoglu, K. (2017). Neural Discrete Representation Learning. In *Advances in Neural Information Processing Systems*, volume 30, pages 1–10, Long Beach. Curran Associates, Inc.

Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. (2004). Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612.

Yoon, J., Jordon, J., and Van Der Schaar, M. (2018). GAIN: Missing Data Imputation using Generative Adversarial Nets. *35th International Conference on Machine Learning, ICML 2018*, 13:9042–9051.

Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. (2017a). Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2242–2251, Venice. IEEE.

Zhu, J. Y., Zhang, R., Pathak, D., Darrell, T., Efros, A. A., Wang, O., and Shechtman, E. (2017b). Toward Multimodal Image-to-Image Translation. *Advances in Neural Information Processing Systems*, 2017-December:466–477.