GAN-Assisted YUV Pixel Art Generation

ABSTRACT Procedural Content Generation (PCG) in games has grown in popularity over the last few years, with Generative Adversarial Networks (GANs) showing promise for game art asset generation. In this paper, we introduce a model that uses GANs and the YUV colour encoding system for automatic colouring of game assets. We use conditional GANs in the Pix2Pix architecture and the YUV colour encoding system for data preprocessing and result visualisation. We experimented with parameters to optimise output. Our experimental results show that the proposed model can generate evenly coloured outputs for both small and larger datasets.

METHOD We chose a conditional GAN as the main structure, as illustrated in Figure 1, specifically a modified Pix2Pix architecture.

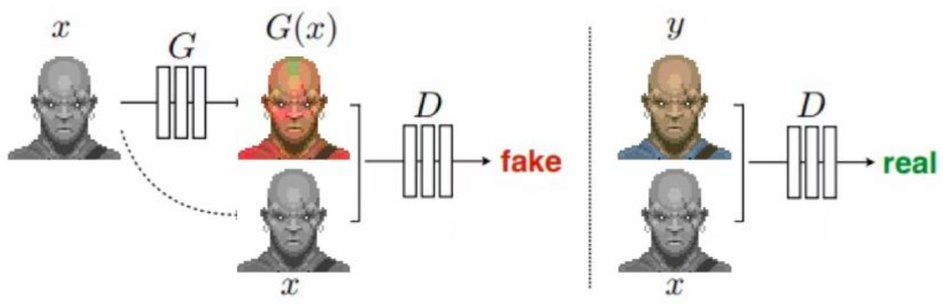

Fig. 1. Structure of conditional GAN (image based on Isola et al.[9]). Both the generator, G, and discriminator, D, are able to receive the grayscale inputs.

DATASETS We used 2 datasets:

- *Ultimate Fantasy Sprites from Oryx Design Labs*. 36 PNG 48x48 pixels in the same theme and suitable for RPG making.
- *Pokemon Images dataset from Kaggle.* 870 official Pokemon characters. Front-facing battle PNG pixel art of 160x160 pixels.

Fig. 2. Samples of original images and Y channel grayscale images after preprocessing.

DATA PREPROCESSING AND VISUALISATION During data preprocessing, the YUV colour encoding system is used to convert all RGB raw image data into YUV images. Y channel is separated as the single-channel greyscale image inputs from YUV images. For results visualisation, the YUV colour encoding system is used to turn the U and V values combined with Y inputs back to RGB values that can be shown as images. Figure 2 shows several samples of original images and Y channel grayscale images after data preprocessing.

Fig. 5. Examples of results with optimised parameters, trained after after 10 (top-left), 25 (lower-left), 50 (top-right) and 200 (lower-right) epochs (left-to-right: input, target, predicted).

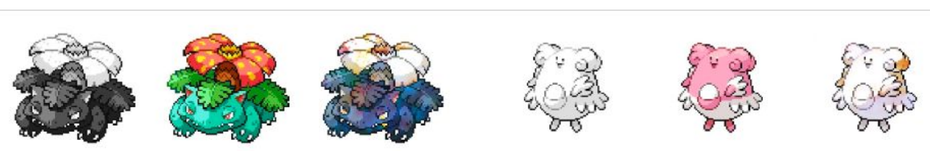

Fig. 7. Sample results of using RGB colour encoding systems to inputs and outputs (left-to-right: input, target, predicted).

Fig. 8. Sample results of using YUV colour encoding systems to inputs and outputs (left-to-right: input, target, predicted).

RESULTS Figure 5 shows several examples of results from our model with optimised parameters. We found that:

- Output gained more detailed colours with increased number of epochs (Fig. 5)
- Compared with RGB colour encoding, the YUV outputs were more detailed and evenly coloured (Fig. 7-8)
- Using LeakyReLU as the activation function for downsampling steps provided better results than ELU (Fig. 9-10)
- Both Adam and SGD optimisers were good options for this model, each with different advantages (Fig. 11-12)

Fig. 9. Sample results of using ELU as activation function in down sampling steps (left-to-right: input, target, predicted).

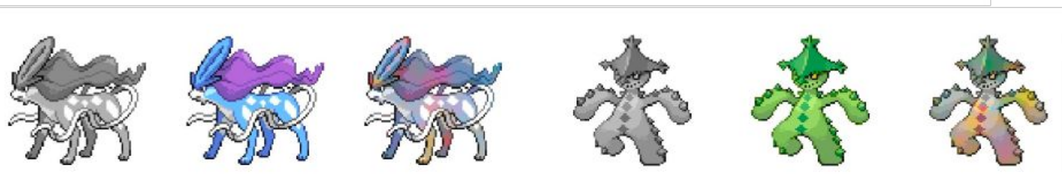

Fig. 10. Sample results of using LeakyReLU as activation function in down sampling steps (left-to-right: input, target, predicted).

Fig. 11. Results for Adam as optimiser, training after 50 (top) and 200 epochs.

Fig. 12. Results for SGD as optimiser, training after 50 (top) and 200 epochs.

Australian National University

Zhouyang Jiang, Penny Kyburz (Sweetser)
School of Computing, penny.kyburz@anu.edu.au

AI 2021: The 34th Australasian Joint Conference on AI