CS613 Sprite Generation Using GANs

Tajung Jang
Chris Oriente
Marcus Sullivan
Tuote Huang

What is the problem you are tackling?

- Train a generative model for sprite generation
- Implement a Generative-Adversarial Network (GAN)
- Use for sequential sprite generation

Related Works

1. COT-GAN: GENERATING SEQUENTIAL DATA VIA CAUSAL OPTIMAL TRANSPORT
   a. Used to generate sprite frames
   b. Generates dynamic data that respects causality between generated data
   c. Uses a GAN where the loss function is based on Causal Optimal Transport (COT) and penalization using the Sinkhorn algorithm.
2. Visual Dynamics: Probabilistic Future Frame Synthesis via Cross Convolutional Networks
   a. Uses a CNN autoencoder to synthesize future frames from a single starting frame
   b. Specifically, the CNN autoencoder is used to model the conditional distribution of frame sequences
3. https://www.koreascience.kr/article/JAKO201926358473678.page
   a. Uses a GAN to generate sprite images where there are multiple discriminators to separate shape, color, and other attributes so that the generator learns from multiple domains
   b. Goal is to generate a unique set of sprites that assume a different state off of a set of sprites in their base state
4. CNN Sprite Generator
   a. Utilizes a CNN autoencoder to generate new and unique sprites

What is the basic approach?

Phase 1 (complete):

- Train a type of artificial neural network called a Generative Adversarial Network (GAN) to generate realistic sprites.
- Train our GAN on 3 different datasets of a specific sprite animation. From this we should be able to build a walk animation of a character sprite.

Phase 2 (incomplete):

- Implementation of a DC layer to our GAN.
- Test with more diverse dataset.

Phase 3 (incomplete):

- Implement Causal Optimal Transport in our GAN.

Where are you getting your data from and what does it look like?

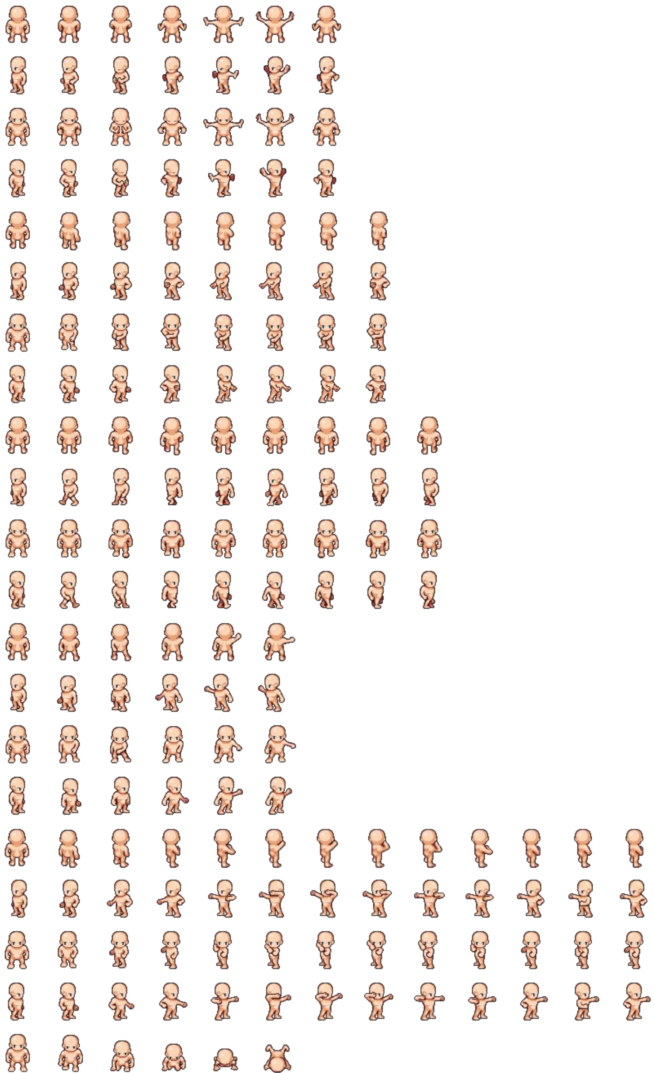
Sprite sheets are sourced from the Liberated Pixel Cup

- An open-source game programming competition held in July 2012

They have a stated goal:
"...make a bunch of awesome free culture licensed artwork, and then program a bunch of free software games that use it."

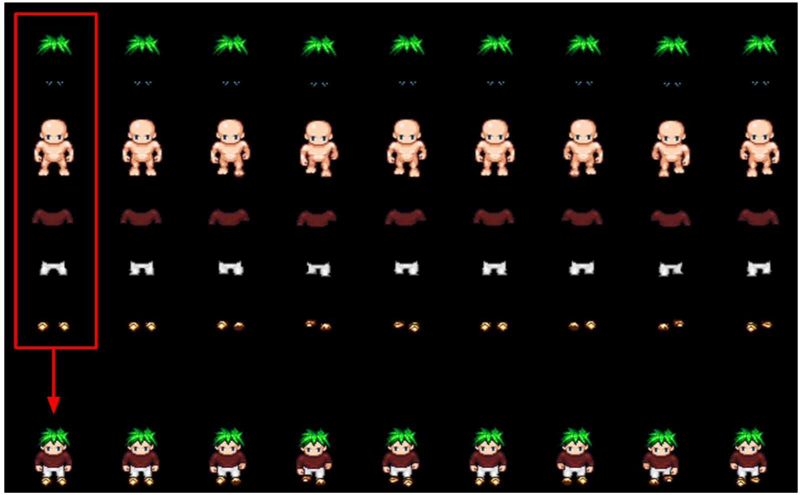
Organized by the Free Software Foundation, Creative Commons, Open Game Art, and the Mozilla foundation.

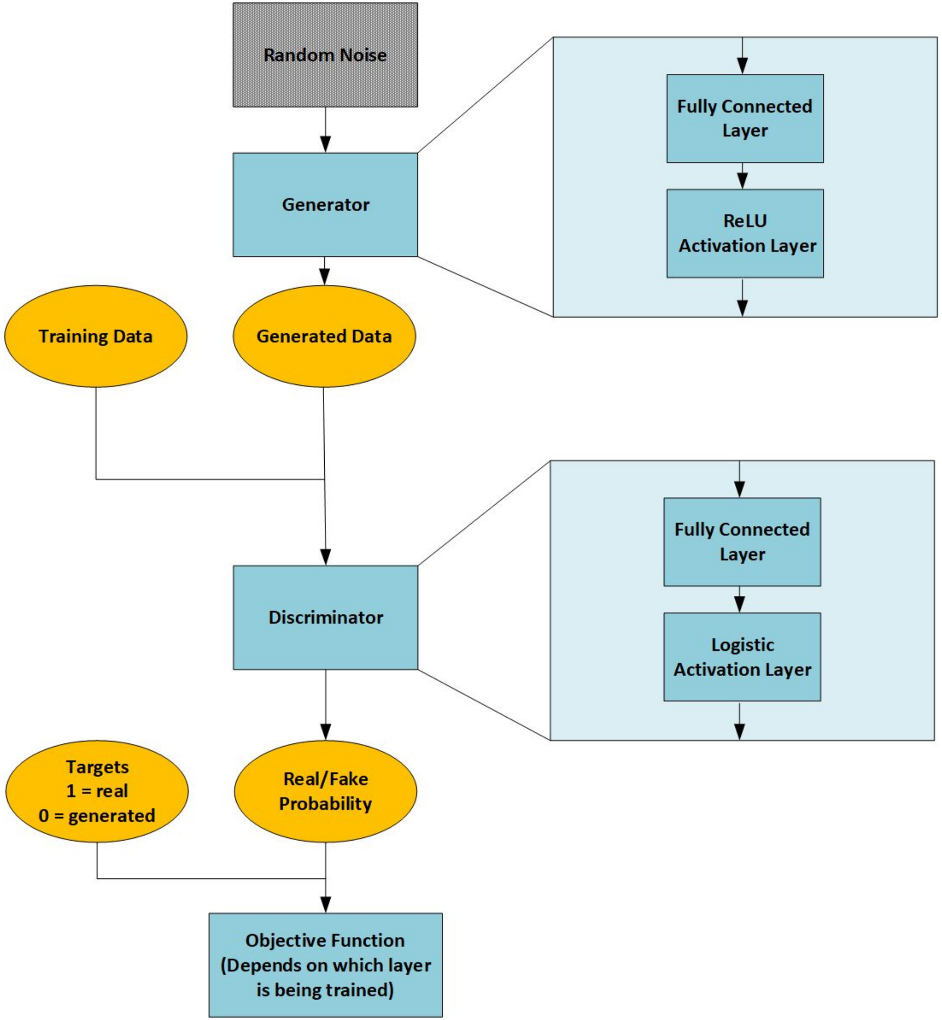
Sprite Sheet Example



Data Example

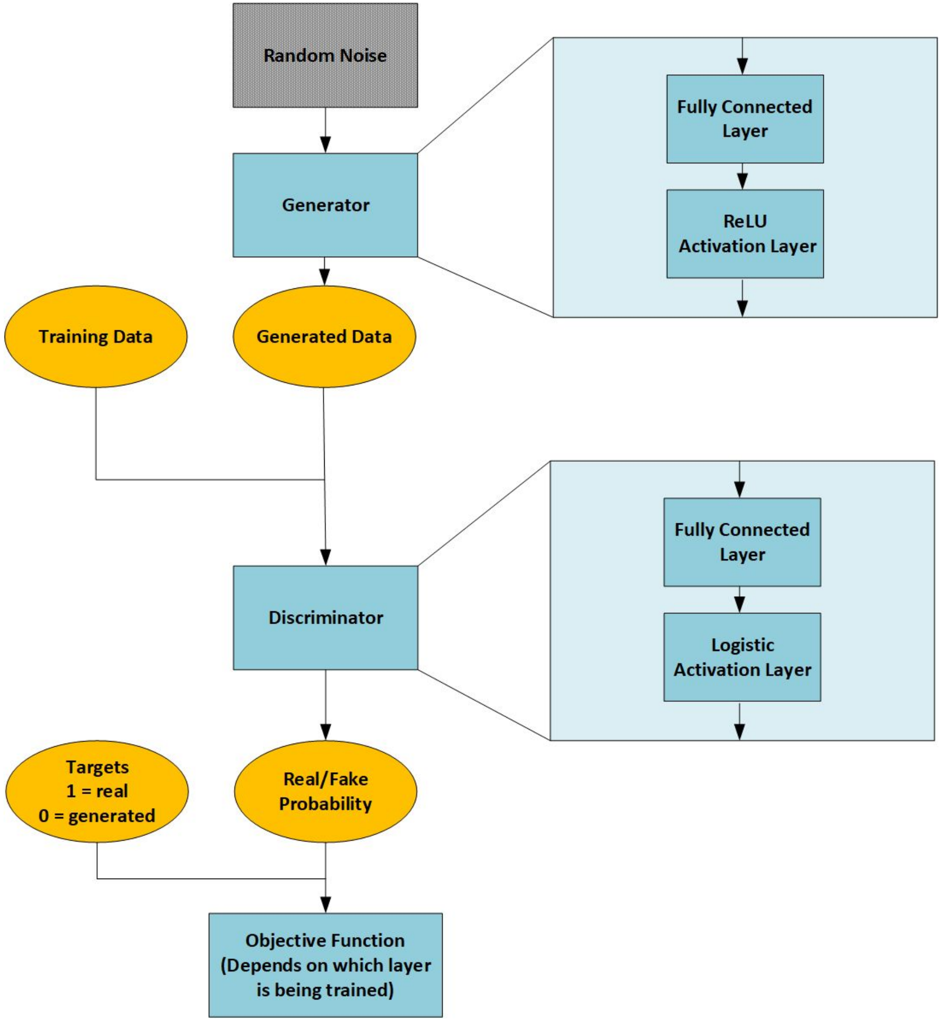
- One character's sprite sheet is a combination of clothes and hair pieces.
- We use the combined character to train our models





General Training Procedure:

- Forward propagate random noise through the Generator
- Take the resulting generated data and combine with the real data.
- Forward propagate the combined data through the Discriminator.
- Calculate the log loss and the gradient of the objective function with respect to its input (Discriminator output).
- Backpropagate the gradient through to the Fully Connected layer of the Discriminator.
- Update weights and bias for the Discriminator.



General Training Procedure:

- Forward propagate random noise through the Generator
- Forward propagate just the generated data through the trained Discriminator.
- Calculate the loss (sum of natural log of the probabilities) and the gradient.
- Backpropagate the gradient through the Discriminator (do not update weights and bias).
- Backpropagate the gradient through to the Fully Connected layer of the Generator
- Update weights and bias for the Generator.
- Repeat

Fréchet Inception Distance (FID)

$$\text{FID} = ||\mu - \mu_w||_2^2 + \text{tr}(\Sigma + \Sigma_w - 2(\Sigma^{1/2}\Sigma_w\Sigma^{1/2})^{1/2})$$

- Compares the distribution of generated images with the distribution of real images
  - where, μ and Σ are the mean and covariance matrix of the generated image.
  - μ_w and Σ_w are the mean and covariance matrix of the world (i.e., real) image.
- A lower value for the FID indicates a lower distance between the two images

CUDA

CuPy

- Replace Numpy matrix operations with CuPy to leverage GPU
- 2.75x Speed up in runtimes
- Didn’t output desired output
- Can be improved with custom C based CUDA kernel or CUDA BLAS
- Expect a linear speedup w.r.t matrix size

Results

- At first it looks like nothing is happening.

Results

50,000 Epochs

50,000 Epochs Negative

- When we looked closely at the image we noticed a faint image

Artificial Enhancement

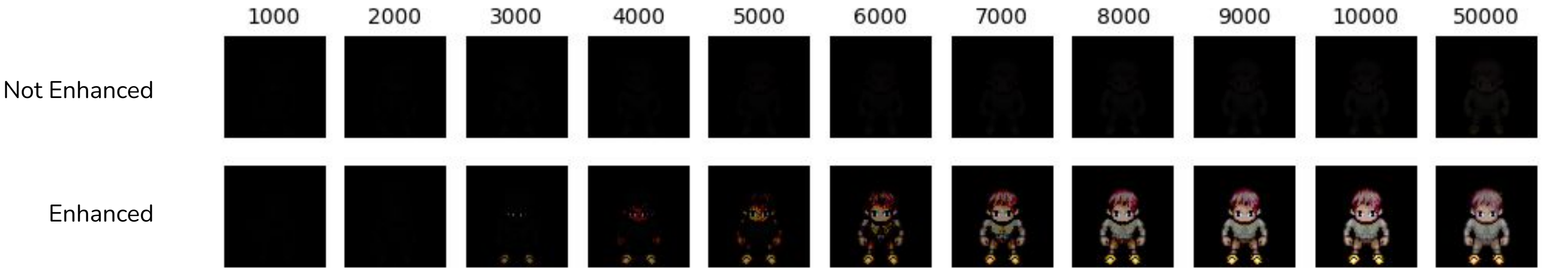
- In order to improve the quality of the generated images, we scaled each channel (RGB) using the following formula

$$enhanced(x) = \begin{cases} x * s, & x > t \\ x, & x \leq t \end{cases}$$

where,

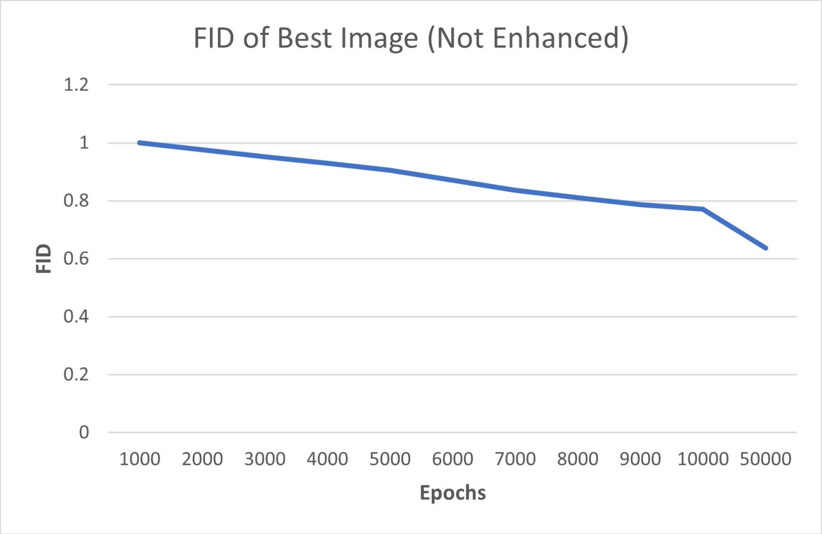
x is the feature value
s is the scaling factor
t is the threshold

Results

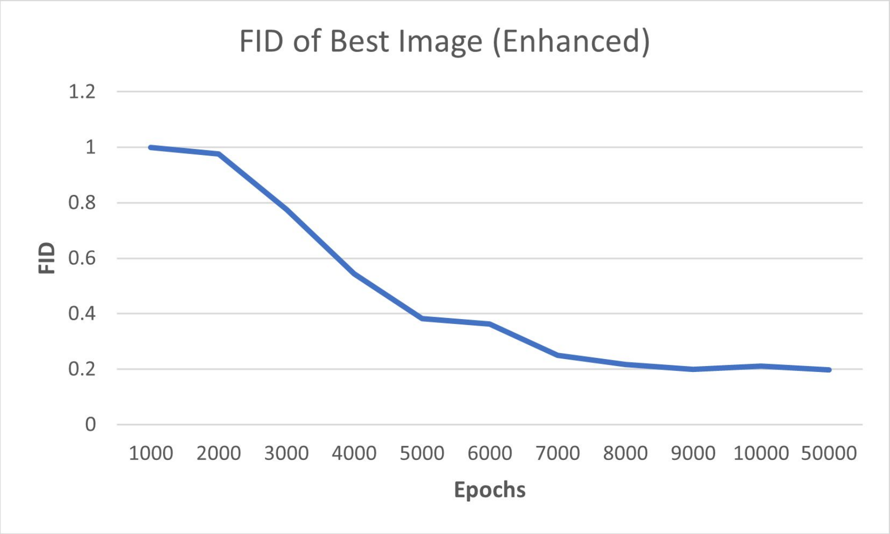


- The resulting images were far more convincing

Results

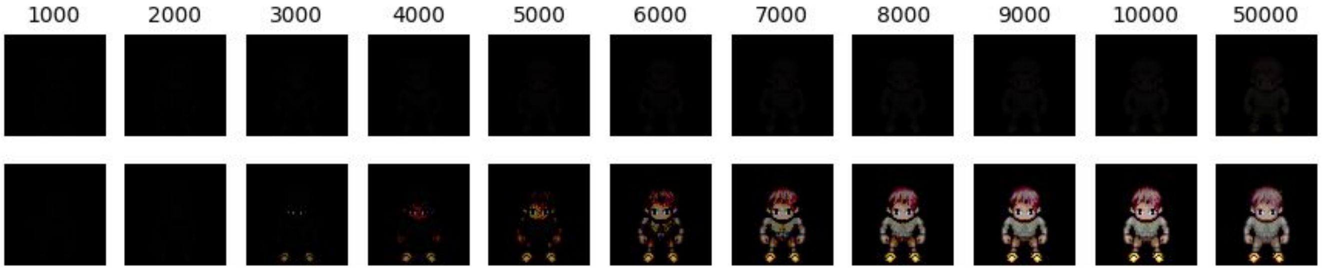


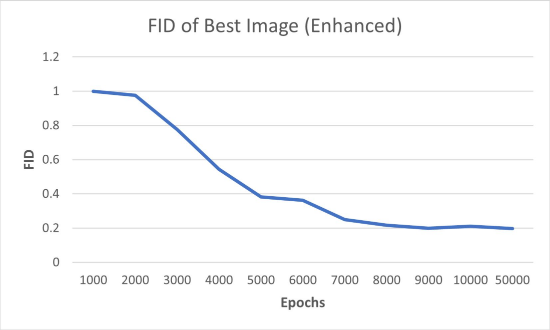
FID of Best Image (Not Enhanced)
1.2
1
0.8
0.6
0.4
0.2
0
FID
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
50000
Epochs



FID of Best Image (Enhanced)
1.2
1
0.8
0.6
0.4
0.2
0
FID
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
50000
Epochs

Results





Number of Epochs (checkpoint)	FID *	Enhanced FID* (s = 15, t = 5)	Change in Enhanced
1000	1.000	1.000	N/A
2000	0.976	0.976	0.023
3000	0.951	0.776	0.200
4000	0.928	0.544	0.231
5000	0.905	0.382	0.161
6000	0.870	0.363	0.019
7000	0.836	0.250	0.112
8000	0.810	0.216	0.034
9000	0.786	0.198	0.017
10000	0.771	0.211	0.013
50000	0.637	0.198**	0.013

*Scaled to be between 1 and 0 by dividing by the maximum FID

** Scaling factor of 9 was used.

- Near optimal after 8,000 epochs

Walk Animation





What could be future extensions of your work?

- Transposed Convolution
  - Deep Convolution GAN (DCGAN)
- Optimal Transport
- GPU implementation
  - Custom kernel
  - Kernel division
  - Asynchronous Stream Scheduling

Bibliography

[1] C. Henrichs, S. Wani, and S. Feroz. CNN Sprite Generator. https://curthenrichs.github.io/CS534-Term-Project-Website, last accessed on June 03, 2022.

[2] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. 2017.

[3] S. Hong, S. Kim, and S. K. and. Game Sprite Generator Using a Multi Discriminator GAN. KSII Transactions on Internet and Information Systems, 13(8):4255–4269, August 2019.

Bibliography

[4] Pennomi. Liberated Pixel Cup. https://opengameart.org/content/liberated-pixel-cup-0, last accessed on June 03, 2022.

[5] T. Xu, L. K. Wenliang, M. Munn, and B. Acciaio. COT-GAN: Generating Sequential Data via Causal Optimal Transport, 2020.

[6] T. Xue, J. Wu, K. L. Bouman, and W. T. Freeman. Visual Dynamics: Probabilistic Future Frame Synthesis via Cross Convolutional Networks, 2016