Synthesizing Programs for Images using Reinforced Adversarial Learning

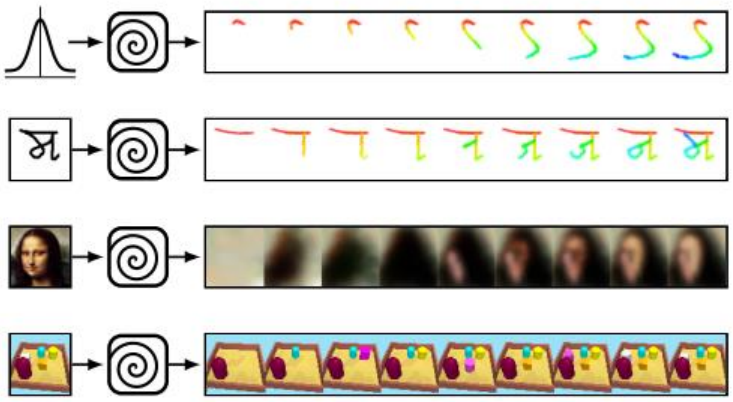
The Problem

Graphics engines have recently been used to represent images as high-level programs, getting rid of the low-level details. The main problem with this is that, by 2018 standards, there weren't any methods to combine deep learning and renderers without requiring a plethora of other things such as a vast amount of supervision and hand-crafted distance functions.

The Spiral Agent

The solution comes in the form of "an adversarially trained agent that generates a program which is executed by a graphics engine to interpret and sample images. The goal of this agent is to fool a discriminator network that distinguishes between real and rendered data, trained with a distributed reinforcement learning setup without any supervision".

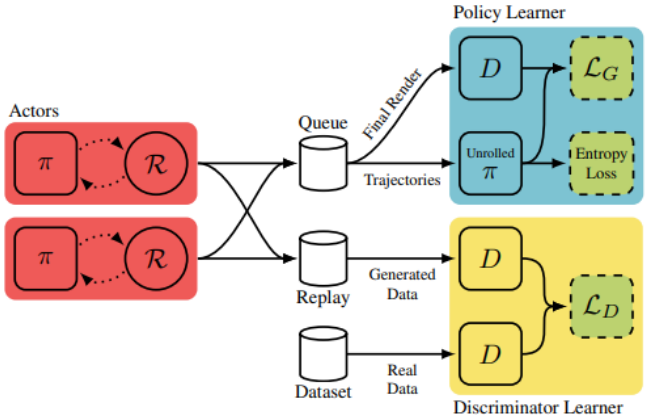
One of the key findings refers to using the discriminator's output as a reward, which in turn result in the agent making more progress towards matching the desired output.



Goal Overview

"Our goal is to construct a generative model G capable of sampling from some target data distribution p_d. To that end, we propose using an external black-box rendering simulator R that accepts a sequence of commands $a = (a_1, a_2, \ldots, a_n)$ and transforms them into the domain of interest, e.g., a bitmap".

The desired output is a distribution (p_a) that, when applied to the external simulator, can be indistinguishable from the input to a discriminator. ($p_d \cong R(p_a)$). p_a is modeled using a recurrent neural network π that, at every time step predicts a distribution over every possible command.



Training Pipeline

Composed of 3 workers:

- Actors tasked with generating training trajectories based on the connection between the rendering simulator and the policy network
- Policy learner that receives the actors' trajectories, combines them into a batch and updates π. L_G also receives an entropy penalty to encourage exploration
- Discriminator learner that uses random examples from p_d and final renders, optimizing L_D

Data Sets

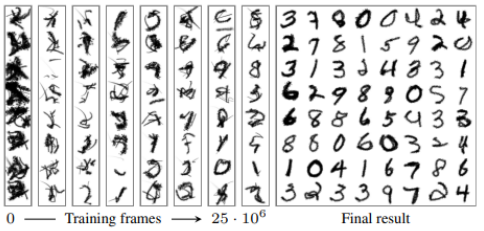
a) MNIST – composed of 70.000 examples of handwritten digits, each of them being a 28x28 grayscale image

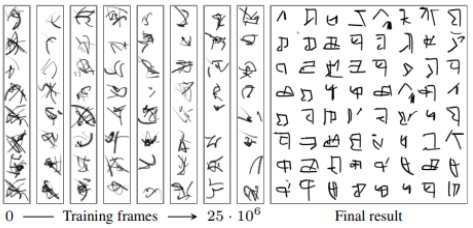
b) OMNIGLOT – made up of 1623 handwritten characters from 50 different alphabets. This adds another layer of complexity over the MNIST dataset

c) CELEBA – a list of 200.000 color headshots of celebrities with a big variance in the setting, lightning and poses.

libmypaint

open-source painting library in which, "the agent controls a brush and produces a sequence of (possibly disjoint) strokes on a canvas C. The state of the environment is comprised of the contents of C as well as the current brush location lt. Each action at is a tuple of 8 discrete decisions"

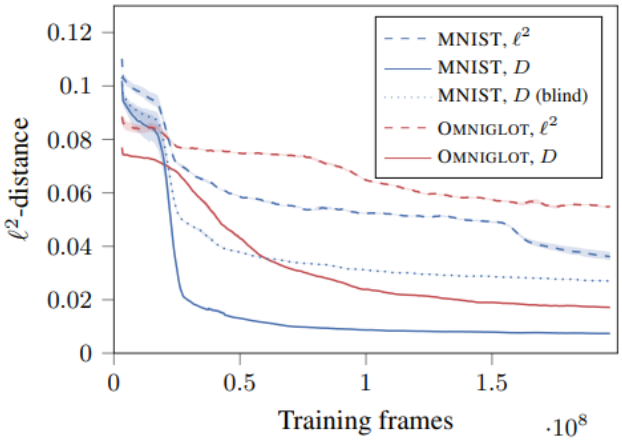


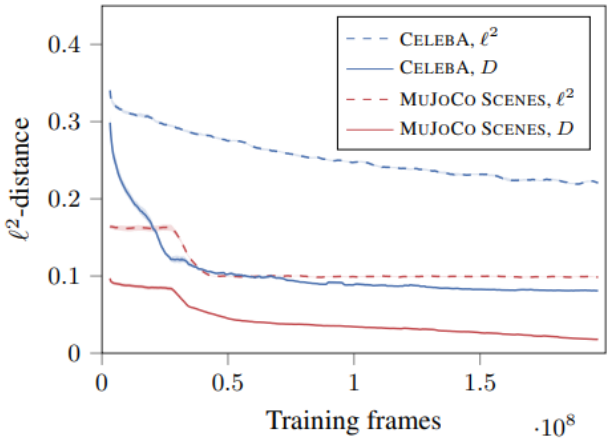


d) MUJOCO SCENES – is a procedural dataset made up of 50.000 color images of a square platform with a maximum of 5 object present on it. In this instance, the agent is set up to construct simple CAD programs that recreate the given input, at any point it has the capability of deciding what object to place, where to place it (in a 16x16 grid), it's size (3 options) and color (3 colors with 4 bins each). Unlike the previous data sets where there was both a conditional and unconditional generation, in the case of the MUJOCO Scenes, we only consider the conditional one. Even though the input platform has at most 5 objects, the agent doesn't have this knowledge beforehand and, since the model is unrolled for 20 time steps, is able to place a maximum of 20 objects.

Rewards

The reward function used on all of the data sets can either be the ℓ^2 score or the discriminator output. Across all of the results, using the discriminator score speeded up the process of training the model and ended up with a lower ℓ^2 error.





Another experiment on these reward functions was ran with the purpose of illustrating the difference between them in practice. The dataset consists of images of a single white circle on a black background in various positions, from which, one is chosen as a target image. “We train a discriminative model D that takes a pair of images and tells whether they match or not”.