Ishmael Obeso
November 7, 2019

Udacity Machine Learning Engineer Capstone Proposal

Proposal

Create a webpage that can take user images and output painting-like renditions of the input images using a brushstroke GAN.

Domain Background

Ever since I watched Ian Goodfellow's interview with Lex Fridman, the idea of Generative Adversarial Networks (GANs) has caught my interest. It's fascinating that a neural network can imitate what is commonly thought of as a uniquely human trait, art. However, I found that most methods such as style transfer and GANs tend to generate images pixel-by-pixel while humans create art through other means such as brushstrokes, stippling, and other techniques that use the constraints of the artists tools to create a certain style. I wanted to work on creating a GAN that closely imitated the way humans create art. To this end, I stumbled across a blog-post called "The Joy of Neural Painting"[1] that introduced the idea of brushstroke GAN's which more closely imitate the way a human generates paintings.

Instead of generating paintings pixel-by-pixel like traditional neural networks, brushstroke GAN's such as the ones used in the paper 'Neural Painters' [2] generate their images by successively painting brushstrokes on a canvas to create a painting-like image. As evidenced in this paper, this technique is able to create more convincing painting-like textures than other models. I believe by leveraging these types of GAN's to more closely imitate human art, the outputs of these models will be more convincing and able to be used in a multitude of ways. For example, leveraging these models to simply create more human-like art will be helpful when it is not feasible to pay an artist for their original work, such as in small video-game studios that want to include some type of artwork in their games. Instead they can simply make use of these GAN's in their games. More generally, being able to generate images that closely resemble human-made paintings will be useful whenever art or fashion is concerned, such as when work-shopping new clothing textures, creating designs for websites, or simply for creating original non-copyrighted work that is aesthetically pleasing without having to pay exorbitant amounts of money for original art.

Problem Statement

With the recent advances in deep learning we are seeing more and more computer-generated images being used in things like advertising, web design, and fashion. However, it is still relatively easy to spot when a computer-generated image is being used, especially when creating painting-like images. These images tend to be created using style transfer and GAN's that build the image pixel-by-pixel, but if we want to create convincing computer-generated paintings we need to start using human-like techniques. In this way, neural painters imitate human artists by using brushstrokes on a canvas to create convincing art.

However, neural painters based on Variational AutoEncoders(VAE's) have not yielded good results, as they are simply not able to recreate the fine details of brushstrokes. In addition, since this breakdown happens even on simple images such as the ones in the MNIST database, a new method will need to be used to create neural painters that can recreate complex images such as the paintings found in the BAM dataset.

Datasets and Inputs

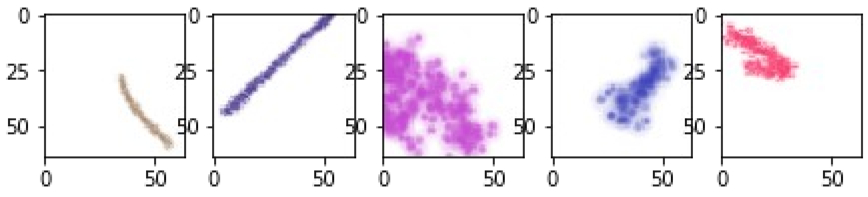
I plan to use the MyPaintBrushstroke dataset provided by researcher Reiichiro Nakano to train the VAE and GAN neural painters to create brushstrokes. This dataset consists of 100k examples of:

actions - these consist of 12-dimensional vectors which characterize the brushstroke. The vectors define start and end pressure, brush size, RGB color values, and brush coordinates.

ex.

```
array([[0.330522, 0.249296, 0.638134, 0.656268, 0.539904, 0.435634, 0.623549, 0.494268, 0.38073 ,
0.896127, 0.924754,
         0.466711]])
```

brushstrokes – images of the brushstrokes that correspond to the action vectors.
ex.



I also plan to use the MNIST and BAM(Behance Artistic Media) datasets to generate painting-like images using the trained neural painters The MNIST database consists of images of integers from 0-9 and the BAM dataset consists of images of paintings. The MNIST dataset will be used as a sanity check for both my networks, to determine if they can recreate images properly before starting training on the full-color painting dataset. The BAM dataset will be used to determine if the GAN neural painter creates painting-like images better than the VAE neural painter.

Sources: https://www.kaggle.com/reiinakano/mypaint_brushstrokes
https://www.kaggle.com/oddrationale/mnist-in-csv
https://bam-dataset.org/

Solution Statement

To create convincing painting-like images, we need a model which paints like a human would, using brushstrokes. I believe that by using a brushstroke GAN, this can be accomplished. The goal of this project is to create a web-page where a user can input an image and receive a painting-like rendition of the input image.

Benchmark Model

For my benchmark model, I will use a Variational Autoencoder (VAE), which are generative models that have been used for image generation and compare the results to the Generative Adversarial Network (GAN).

Source: https://github.com/bhpfelix/Variational-Autoencoder-PyTorch

Evaluation Metrics

Evaluating the performance of generative networks has historically been difficult, and as a result it is mostly done by visual inspection. However, in this instance I believe I can use a more quantifiable evaluation metric. Since the goal is to create painting-like images, I will have both the VAE and GAN take an input painting and see which replicates the painting the best. Since both networks will be using brushstrokes to generate the painting, the results will be determined by which network has the best quality brushstrokes and composition.

In order to measure this, I will use VGG loss as the evaluation metric. VGG loss is based on the VGG19 image classification neural network, which can be used as a feature extractor. This feature extraction can be used to create feature maps of the generated images and the real image. VGG loss uses these feature maps to calculate the Euclidian distance between the feature maps of the generated and real images, and as a result can be used to determine which of the generated images are more similar to real image.

Reference: https://www.oreilly.com/library/view/generative-adversarial-networks/9781789136678/c399f455-2670-4321-92cf-de6cf75cd543.xhtml

Project Design

- Languages: Python 3.7, JS
- Libraries: Pytorch, fastai, numpy, PIL, glob
- External Services: Amazon Sagemaker, S3, Lambda, API Gateway
- Goal: Create a web-page where a user can input an image and receive a painting-like rendition of the input image.
- Steps:
  - Train a VAE Neural Painter
  - Train a GAN Neural Painter
    - Non-Adversarial Training
      - Train Generator and Critic non-adverbially to speed up training.
    - Adverserial Training
      - Connect Generator and Critic to create adverserial GAN training.
  - Brushstroke Painting
    - Use a simple dataset as a sanity check (MNIST)
      - Train the VAE and GAN Neural Painters to recreate MNIST images
      - Use VGG loss to see which type of Neural Painter creates best images
    - Train VAE and GAN on painting dataset
      - Use VGG loss to see which type of Neural Painter creates best paintings
  - Use Amazon services to create an endpoint that a webpage can use to send my model input images from users.

    - Create website for users to input images and receive a painting-like rendition of their image.

References:
1. https://medium.com/libreai/the-joy-of-neural-painting-e4319282d51f
2. *Neural Painters: A Learned Differentiable Constraint for Generating Brushstroke Paintings*. Reiichiro Nakano
arXiv preprint arXiv:1904.08410, 2019.
3. *Teaching Agents to Paint Inside Their Own Dreams*. Reiichiro Nakano.
https://reiinakano.com/2019/01/27/world-painters.html , 2019
4. *Fast.ai MOOC Lesson 7: Resnets from scratch; U-net; Generative (adversarial) networks*.
https://course.fast.ai/videos/?lesson=7 ; Notebook: https://nbviewer.jupyter.org/github/fastai/course-v3/blob/master/nbs/dl1/lesson7-superres.ipynb [Accessed on: 2019–08]