Implementing AI Algorithms to Generate Self-learning Art Styles

CS4991 Capstone Report, 2024

Caleb Lee
Computer Science
The University of Virginia
School of Engineering and Applied Science
Charlottesville, Virginia USA
cjl2pub@virginia.edu

ABSTRACT
AI Art has raised many questions regarding the protection and definition of creators' rights to claim their work as original. Such questions have caused legal and moral debates that have escalated to antagonism from programmers and artists alike. To resolve this conflict, I propose an algorithm that analyzes an artist's style from the artist rather than from unsolicited sources, allowing artists to make art pieces while retaining full copyright protection. We propose taking an existing open source drawing software and adjusting the design to be compatible with AI algorithms to mimic an artist's drawing style. Future work would extend this trend to different media, such as 3D modeling and/or problem-solving in more abstract fields such as mathematics or science, depending on the degree of success.

1. INTRODUCTION

In recent years, we have seen significant advancements in the field of Artificial Intelligence. It has been incorporated into several facets of daily life in both the public and private sectors of society. Recently, many applications have been introduced in the creative and performance sectors, where art and music are replicated through complicated neural networks to produce novel works in a similar fashion to human beings. Such advancements arise in the form of adaptive neural networks which change behavior through dynamically allocated output patterns instead of conventional input-output systems.

However, an issue that has been introduced with the advent of such a technology arises in the discussion of whether such application in computer science and art causes problems in both fields. Copyright conflicts introduced with the data-acquisition process of AI raise legal and socio-technical problems which conflict affect the interests of both parties. For this reason, the field is controversial for programmers to work on the development and execution of said technology.

2. RELATED WORKS

Recent advancements in Convolutional Neural Networks (CNN Networks) have made it possible for algorithms to extract both features and general details from artworks, which gives programmers a much wider and more holistic view of an artist's method of creating individual art. In addition, aesthetic analysis from human perception has made its way into the design of these neural networks, where mathematicians and software developers are paying more attention to the thought procss of art historians and artists in their interpretation of their work and how it

factors into their brushstrokes, color palette choices, and other implementations of art fundamentals. A much more recent development of text-to-image generation of parameterized networks, DALL-E, has also seen great promise in this field (Cetinic, et. al., 2022).

A very recent school of thought emerged from interpretationist AI, instead of the more conventional form of AI. This field asserts that such algorithms should be implemented with a degree of plasticity so this method of data analysis can be molded and adapted by the inputs it is given, so more creativity can be exercised by the AI than are typically allowed to exist under the parameters of a more input/output-based design process. In addition, the more traditional method restricts free thought and tempts programmers to see AI only as product-makers instead of thinking programs, which is contrary to the creative practice (Mateas, 2001).

3. PROPOSAL DESIGN

From here, we will outline the general method of implementation for our proposed product. We draw a sketch in our drawing software, then train a neural network to recognize its features before asking it to generate a complete version of our drawing.

3.1 General Framework

Before we outline the general procession of the algorithm, it is important to preface that there are certain limitations to the final product, along with some preliminary(?) notes as to how data will be acquired. The method of gathering training data would be, by design, a tricky endeavor, due to GIMP saving files into .XCF format, which is the aggregate sum of all layers and effects into one layer. Therefore, the scope of this project will be limited to simple sketches and/or shapes that can be easily recognized by the AI.

Proceeding with the general outline of the process: to begin, we draw various sketches of common objects, such as dogs, flowers, or others, preferably with an asymmetrical component to allow the AI to catch crucial differences in drawings. All of this training data will be fed to a custom-option in GIMP—an open source, Python-compatible drawing tool--which parses the sketch as training data with a prompt, asking which label to assign it to. Afterwards, an external machine learning hosted in Google Colab will use this training data to begin structuring the sketch features, where a neural network will analyze the intricacies of the way the object is drawn. From there, the data is fed back to the neural network, where it, along with the Python code, will be exported in a .ipynb file to allow GIMP to use this software as an extension. From there, we can ask the program to pull up our list of labels and generate a respective image given a list of options.

3.2 Data Acquisition

For our data acquisition, we will draw a series of sketches which will be aggregated as training data, 20 per batch. However, unlike aimlessly drawing sketches with no sense of focus, we will be careful to vary the drawing with angles and proportions in distinct yet similar ways to allow the AI to recognize these similarities and differences as an art style. Because we want the AI to use a brushstroke as a reference, each image will be broken into parts, where we first sketch a part of our desired product to use as our “initial sketch.” For example, if we want a landscape of a forest, we only draw the horizon and one tree, then ask the AI to generate layers of trees which aggregate to a forest. The final forest used for training data, however, will be a complete sketch of the

initial sketch, so that the AI partitions the "before" and "after" sketches as "training" and "performance" data. From there, the data is fed to a Google Colab algorithm.

3.3 Google Colab Implementation

After the training data has been acquired, a Python algorithm will take this data and incorporate it into a pre-generated neural network. Using our "initial" and "final" sketches, we design the network to take in features such as paint color, shading, and other details missing from the "final" sketch, allowing the AI to generate an idea of what the final product should look like, given these trends of data. After the data has been sufficiently trained, it can be used as a GIMP extension.

3.4 GIMP Implementation

After the AI program has been exported onto GIMP, we can establish a tab in GIMP which allows the user to set up a list of images they wish to generate based on what they specified as training labels earlier (See Section 3.1). From there, the user is able to select "begin initial sketch" to draw out a perimeter sketch of the desired background/ object, and after completing, they can select "generate image" to let the AI create an approximation of their desired format.

4. ANTICIPATED RESULTS

Ideally, the program should generate a simple, complete drawing based on an outline sketch provided by the user. The goals for the final product consist of a color palette, detail size, and shading resembling our input data, albeit missing a few details to allow for AI generation fault tolerance. In addition, the program should be able to recognize the general shape of the final product, given the portions we specified as training data.

In terms of applications, we also hope to be able to utilize this software with all the features GIMP provides by default, such as alpha-locking, and multiple drawing layers. We also hope that the initial sketch is not over-ridden so that our original work is preserved while generating the complete rendition of the landscape or image.

5. CONCLUSION

After completing this implementation, we can see that the early stages of this prototype yielded a few primitive but inquisitive results: while the shape of the output remained the same as the intended final product, the colors were slightly misallocated, occasionally stepping beyond the outline of the sketch or retaining the color in the horizon rather than for the object.

We can see here that there is promise in AI in art as a good visualization and supplementary software for artists, allowing them to streamline the process to complete their work more efficiently by automizing the more tedious components. Despite its rather short list of features, we can see that with proper development, we will be able to export this software as a standalone extension to help both beginner and experienced artists by allowing them to pre-render and generate backgrounds, lighting, or even articles of clothing for characters and environments.

6. FUTURE WORK

For now, we have to consider a few things when advancing said technology. We first would like to see if said technology is compatible with other digital art workstations, such as Krita, Photshop, or other third-party vendors. Next, we should optimize the algorithm to take each layer of

drawing into consideration, where paint, line thickness, and hue saturation can be taken as individual and co-dependent variables to be generated without interfering with the others.

REFERENCES

Cetinic, E., & She, J. (2022). Understanding and creating art with AI: Review and outlook. *ACM Transactions on Multimedia Computing, Communications, and Applications*, *18*(2), 1–22. https://doi.org/10.1145/3475799

Mateas, M. (2001). Expressive AI: A Hybrid Art and Science Practice. *Leonardo*, *34*(2), 147–153. http://www.jstor.org/stable/1577018