**Abstract Art Evolution Using Genetic Algorithm**

Submitted in partial fulfillment of the requirements of the degree

## BACHELOR OF ENGINEERING IN ARTIFICIAL INTELLIGENCE MACHINE LEARNING

By

**Atharva Chaudhary 121A9007 Keegan Dsouza 121A9015 Aniket Patil 121A9040**

**Aryan Samant 121A9050**

Supervisor

**Prof.**



**Department of Artificial Intelligence Machine Learning**

**SIES Graduate School of Technology**

## Nerul, Navi Mumbai - 400 706

**University of Mumbai**

**(AY 2023-24)**

# CERTIFICATE

This is to certify that the Mini Project entitled **“Abstract Art Evolution Using Genetic Algorithm”** is a bonafide work of **Atharva Chaudhary(121A9007), Keegan Dsouza(121A9015),Aniket Patil(121A9040),Aryan Samant(121A9050)** submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of **“Bachelor of Engineering”** in **“Artificial Intelligence Machine Learning”.**

#### (Prof. )

Supervisor

#### (Dr. Varsha Patil) (Dr. K. Laxmi Sudha)

Head of Department Principal

# Mini Project Approval

This Mini Project entitled “**Abstract Art Evolution Using Genetic Algorithm”** by **Atharva Chaudhary(121A9007), Keegan Dsouza(121A9015),Aniket Patil(121A9040),Aryan Samant(121A9050)** is approved for the degree of **Bachelor of Engineering** in **Artificial Intelligence Machine Learning.**

**Examiners**

**1………………………………………**

(Internal Examiner Name &Sign)

#### 2…………………………………………

(External Examiner name &Sign)

Date: Place:

# Contents

[Abstract ii](#_TOC_250001)

Acknowledgments iii

List of Abbreviations iv

List of Figures v

List of Tables vi

List of Symbols vii

1. [Introduction 1](#_TOC_250000)
   1. Introduction
   2. Motivation
   3. Problem Statement &Objectives
   4. Organization of the Report
2. **Literature Survey**
   1. Survey of Existing System
   2. Limitation Existing system or research gap
   3. Mini Project Contribution

#### Proposed System

* 1. Introduction
  2. Architecture/ Framework
  3. Algorithm and Process Design
  4. Details of Hardware &Software

#### Design and Methodology

* 1. Design Details
  2. Methodology
  3. Algorithm Implementation

#### Results and Discussions

* 1. Implementation
  2. Result and Discussions

1. **Conclusion and Future Scope References**

**ACKNOWLEDGEMENT**

We would like to express our thanks to the people who have helped us the most throughout our project. We are grateful to our guide ( **Prof. Archana Arudkar**) and coordinator (**Prof Archana Arudkar)** for nonstop support for the project.

A special thanks goes to each other who worked together as a team in completing the project, where we all exchanged our own interesting ideas, thoughts and made it possible to complete our project with all accurate information. We also wish to thank our parents for their personal support and attention who inspired me to go my own way.

We would also like to extend our sincere gratitude to our Principal (**Dr. K. Laxmi Sudha**) and our Head of the Department (**Dr. Varsha Patil**) for their continuous support and encouragement.

We also would like to thank our other faculty members for providing us with all the required resources and references for the project.

## ABSTRACT

## This study explores the evolution of abstract art through the application of Genetic Algorithms (GAs). By employing computational processes inspired by biological evolution, GAs are used to generate and refine abstract artworks. The algorithmic approach allows for the emergence of diverse and innovative visual compositions, leading to the creation of non-representational art that transcends conventional human design. Through a process of selection, recombination, and mutation, the algorithm iteratively refines and evolves abstract art forms. The results showcase the potential of GAs in the field of art, offering a new perspective on the generation of abstract visual expressions and encouraging interdisciplinary collaboration between art and technology.

1. **INTRODUCTION**

**1.1 Introduction**

Art has always been an expression and evolution of human creativity. At the same time, many art forms challenge traditional norms and push the boundaries of art. Abstract art is an avant-garde genre that emphasizes non-representational materials, colors, and patterns as expressions of thoughts and ideas, without direct reference to the physical world.

In today's digital age, the intersection of art and technology has opened new avenues for exploration and artistic experimentation. An interesting new way is to use genetic algorithms (GA) to create abstract art. Inspired by the principles of biological transformation, GA provides a unique computational model for the creation and editing of abstract works of art.

A user-friendly system that uses genetic algorithms to transform input images into diverse, compact and visually striking abstract works of art. We aim to provide accessibility to users of all backgrounds with a beautiful user interface powered by React.

Key properties include diversity and entropy-based fitness. Paint plays an important role in enhancing and beautifying your photos.

**1.2 Motivation**

**Algorithmic creativity**: Genetic algorithms provide a unique algorithmic approach to the creative process. It enables new and unexpected work to be seen, reveals the creative potential of machines in the arts and challenges preconceptions about the role of professionals, theater and technology.

**Effectiveness and Research:** Genetic algorithms can be replicated on a very large scale. This allows for the exploration of a wide range of graphic art styles and genres, providing artists with a powerful tool to experiment and develop their artwork.

Inspiration for artists: These works can serve as a source of inspiration for artists, providing new technologies and methods to expand their creativity. It empowers artists to harness technology as a means of artistic expression.

**Artistic Innovation**: Abstract art has a long history of challenging artistic conventions and encouraging new forms of creative expression. By introducing Genetic Algorithms into the creative process, we aim to push the boundaries of abstract art even further, fostering innovation and novelty in the art world.

**1.3Problem Statement and Objective**

**Compact Image Synthesis**: Develop an efficient, robust system for generating compact images while preserving key characteristics.

**Optimization and Scalability**: Address fine-tuning challenges in genetic algorithm parameters. Optimization of the genetic algorithm for efficiency.

Painterly Filters Integration: Integration of painterly filters to enhance artistic quality.

1. **LITERATURE SURVEY**

**2.1Survey of Existing System :**

**Computational Art and Algorithmic Creativity**: Examine the concept of computational art and the role of algorithmic creativity in the generation of abstract artworks.

Review studies that investigate the potential of algorithms to create abstract art with artistic value.

**Gallery Exhibitions and Art Installation**: Some artists have used Genetic Algorithms to create abstract artworks for gallery exhibitions and art installations. These works often demonstrate the fusion of technology and art, showcasing the capabilities of algorithms in producing aesthetically pleasing abstract compositions.

**Online Platforms for Collaborative Art Evolution**: Online platforms have allowed users to collaborate in evolving abstract art using Genetic Algorithms. These platforms often involve collective decision-making in the evolutionary process, resulting in art generated by the crowd.

**2.2Limitation of Existing System :**

**Artistic Contextual Understanding**: Existing systems may struggle with understanding the broader artistic context, which is critical for producing abstract art that resonates with art history and cultural references.

**Reproducibility Issues**: Some existing systems may have limited reproducibility, making it difficult to recreate specific artworks or replicate results consistently.

**Limited User Control**: Users of existing systems may have limited control over the evolution process. Fine-tuning artistic parameters and guiding the evolution of abstract art can be challenging in some systems.

**Complexity and Computational Resources**: Some existing systems can be computationally intensive, requiring substantial processing power and time to generate abstract art. This may restrict accessibility for artists with limited resources.

**2.3 Mini Project Contribution**

**Algorithm Implementation**: Develop and implement the Genetic Algorithm (GA) for abstract art evolution. This includes coding the GA framework, fitness functions, and the evolution process.

**Parameter Optimization**: Experiment with different parameters within the GA, such as population size, mutation rates, and selection mechanisms, to fine-tune the art generation process for improved results.

**User Interface (UI) Development:** Create a user-friendly interface for the project, allowing users to interact with and control the abstract art generation process. This might involve developing a graphical user interface (GUI) for users to experiment with the algorithm.

**Testing and Validation** : Perform rigorous testing and validation of the algorithm to ensure its functionality and reliability in generating abstract art. This includes conducting experiments and collecting data for analysis.

1. **PROPOSED SYSTEM**

**3.1** **Introduction**Revealing art has always been a driving force of human imagination and has led to the exploration of thought, ideas and beauty through various forms. In painting, abstract art illuminates the vagaries of human thought by using non-representational text and colors to convey meaning and inspire emotions. The intersection of art and technology has led to an era of exciting possibilities, and methods, especially algorithms (GA), can play a significant role in the transformation of artworks.  
  
The concept offers a vision for the evolution of graphic design using genetic algorithms. Leveraging the power of GA, we focus on creative journeys that push the boundaries of artistic expression and explore the intersection of algorithmic and abstract aesthetic processes. The system is designed to combine the skills of human designers with the computational power of GA to create abstract art that is not only aesthetically pleasing, but also significant.  
  
The following sections will describe the design, materials and operation of the proposed system, providing a better understanding of how genetic algorithms can drive the evolution of graphic arts. In an era where technology and art are increasingly intertwined, this system aims to be a trailblazer in pushing the boundaries of abstract art, sparking innovation, and inviting us to reconsider the very essence of artistic creation.  
  
**3.2Architecture/ Framework**  
Data Collection and Preprocessing:Initial abstract art samples or seed images are collected or generated.These images may be preprocessed to ensure they meet specific size, format, or quality requirements.  
**Genetic Algorithm Core**:The heart of the system is the Genetic Algorithm (GA) core, responsible for evolving abstract art.  
• Components within the GA core include:  
• Population: A set of candidate artworks represented as individuals.  
• Chromosomes: Each individual artwork represented by a chromosome.  
• Genes: Elements of the chromosome representing characteristics of the artwork (e.g., colors, shapes, patterns).  
• Fitness Function: Evaluates the quality and aesthetic appeal of each artwork.  
• Selection Mechanism: Determines which artworks are chosen for reproduction based on their fitness scores.  
• Crossover: Combines the genes of two parent artworks to create offspring.  
• Mutation: Introduces random changes to the genes of offspring.  
**Artistic Feedback and User Interaction**:Users or artists may provide feedback on generated artworks.This feedback can be incorporated into the fitness function to guide the evolution process.User interactions may involve adjusting parameters, providing preferences, or selecting promising artworks.  
**Database or Repository**: Stores a collection of evolved artworks at various stages. Enables the comparison and analysis of generated art over time. May include metadata and annotations for

each artwork.  
**Visualization and User Interface**:Provides a graphical user interface (GUI) for users to interact with the system. Displays evolving  abstract art in real-time or through periodic updates.  
Allows users to adjust parameters, explore the art, and select favorites.  
**Artistic Evaluation and Reporting**:Evaluates the aesthetic quality and creativity of generated abstract art.Summarizes key findings, trends, and user preferences. May  include data visualization and reports on the evolution process.  
**Ethical Considerations**: Addresses ethical aspects related to art generation, copyright, and attribution.Ensures the responsible use of technology in creative endeavors.  
Output and Export: Provides options to save, export, or share the final abstract art creations.  
Allows users to showcase their selected artworks or integrate them into other projects.

**3.3Algorithm and Process Design**

A diagram of algorithm flowchart

Description automatically generated

**3.4Details of Hardware &Software**

* Python programming language
* OpenCv
* React
* Javascript
* CSS
* HTML
* Python Flask

**4. DESIGN AND METHODOLOGY**

**4.1Design**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**4.2Methodology**

**4.3Algorithm Implementation**

# Given : Y = w1x1 + w2x2 + w3x3 + w4x4 + w5x5 + w6x6 and inputs values are (x1,x2,x3,x4,x5,x6)=(4,-2,7,5,11,1)

# Using Genetic Approach to find maximise the output

# Steps involved

# --------------------------------------------------------------------------------------------------------------

"""

1. The fitness of each solution in the population is calculated.

2. The best individuals (parents) are selected based on their fitness.

3. The next generation is generated through crossover.

4. Some variations are introduced to the offspring through mutation.

5. The new population is created based on the selected parents and mutated offspring.

"""

# Input Paramters

# ---------------------------------------------------------------------------------------------------------------

equation\_inputs = [4, -2, 3.5, 5, -11, -4.7] # The input values for the linear equation

num\_weights = 6 # The number of weights to be optimized

# Initialise Population

# ----------------------------------------------------------------------------------------------------------------

import numpy

sol\_per\_pop = 8 # Number of solutions (individuals) in the population

num\_parents\_mating = 4

# Defining the population size.

pop\_size = (

sol\_per\_pop,

num\_weights,

) # Size of the population, a 2D array with sol\_per\_pop rows and num\_weights columns

new\_population = numpy.random.uniform(

low=-4.0, high=4.0, size=pop\_size

) # Randomly initialized population within the specified range

# print(new\_population)

# Fitness Function

# ------------------------------------------------------------------------------------------------------------------

def cal\_pop\_fitness(equation\_inputs, pop):

"""Calculating the fitness value of each solution in the current population.

The fitness function calculates the sum of products between each input and its corresponding weight."""

fitness = numpy.sum(pop \* equation\_inputs, axis=1) # sum along the columns (axis=1)

return fitness

# Selection of the Fittest

# -------------------------------------------------------------------------------------------------------------------

def select\_mating\_pool(pop, fitness, num\_parents):

# Selecting the best individuals in the current generation as parents for producing the offspring of the next generation.

parents = numpy.empty((num\_parents, pop.shape[1])) # initializes an empty matrix to store the selected parents

for parent\_num in range(num\_parents):

# The index of the solution with the maximum fitness is found using numpy.where(), and the corresponding row (solution) is added to the parents matrix

max\_fitness\_idx = numpy.where(fitness == numpy.max(fitness))

max\_fitness\_idx = max\_fitness\_idx[0][0]

parents[parent\_num, :] = pop[max\_fitness\_idx, :]

fitness[max\_fitness\_idx] = -99999999999 # The fitness of the selected parent is set to a very low value to avoid selecting it again

return parents

# Crossover for Mating

# ----------------------------------------------------------------------------------------------------------------------

def crossover(parents, offspring\_size):

offspring = numpy.empty(offspring\_size)

# The point at which crossover takes place between two parents. Usually, it is at the center.

crossover\_point = numpy.uint8(offspring\_size[1] / 2) # The crossover point is set at the midpoint of the genes (weights)

for k in range(offspring\_size[0]):

# Index of the first parent to mate.

parent1\_idx = k % parents.shape[0]

# Index of the second parent to mate.

parent2\_idx = (k + 1) % parents.shape[0]

# The new offspring will have its first half of its genes taken from the first parent.

offspring[k, 0:crossover\_point] = parents[parent1\_idx, 0:crossover\_point]

# The new offspring will have its second half of its genes taken from the second parent.

offspring[k, crossover\_point:] = parents[parent2\_idx, crossover\_point:]

return offspring

# Mutation

# -------------------------------------------------------------------------------------------------------------------------

"""For each offspring, a random value is generated from a uniform distribution between -1.0 and 1.0.

This random value is added to the gene at index 4 (fifth gene) of each offspring. This introduces a small random variation in the offspring's genes"""

def mutation(offspring\_crossover):

# Mutation changes a single gene in each offspring randomly.

for idx in range(offspring\_crossover.shape[0]):

# The random value to be added to the gene.

random\_value = numpy.random.uniform(-1.0, 1.0, 1)

offspring\_crossover[idx, 4] = offspring\_crossover[idx, 4] + random\_value

return offspring\_crossover

# Main Loop

# --------------------------------------------------------------------------------------------------------------------------

num\_generations = 10000

for generation in range(num\_generations):

print("Generation : ", generation)

# Measuring the fitness of each chromosome in the population.

fitness = cal\_pop\_fitness(equation\_inputs, new\_population)

# print(fitness)

# Selecting the best parents in the population for mating.

parents = select\_mating\_pool(new\_population, fitness, num\_parents\_mating)

# Generating next generation using crossover.

offspring\_crossover = crossover(

parents, offspring\_size=(pop\_size[0] - parents.shape[0], num\_weights)

)

# Adding some variations to the offsrping using mutation.

offspring\_mutation = mutation(offspring\_crossover)

# Creating the new population based on the parents and offspring.

new\_population[0 : parents.shape[0], :] = parents

new\_population[parents.shape[0] :, :] = offspring\_mutation

# The best result in the current iteration.

print(

"Best result : ", numpy.max(numpy.sum(new\_population \* equation\_inputs, axis=1))

)

# Getting the best solution after iterating finishing all generations.

# At first, the fitness is calculated for each solution in the final generation.

fitness = cal\_pop\_fitness(equation\_inputs, new\_population)

# Then return the index of that solution corresponding to the best fitness.

best\_match\_idx = numpy.where(fitness == numpy.max(fitness))

print("Best solution : ", new\_population[best\_match\_idx, :])

print("Best solution fitness : ", fitness[best\_match\_idx])

**5. RESULTS AND DISCUSSIONS**

**5.1 Implementation**

**5.2 Result and Discussion**

**6. CONCLUSION AND FUTURE SCOPE**

**6.1 Conclusion**

During our research, we see the evolution of abstract art led by genetic algorithms, a process that demonstrates the power of algorithms to create fascinating, thought-provoking works of art. We explore the balance between algorithmic precision and artistic intuition, exploring how collaboration between humans and machines can inspire innovation and create abstract art that resonates with audiences.  
The combination of genetic algorithms and abstract art reveals a land of endless possibilities. The journey doesn't end here, but it calls us to push the boundaries of creativity, find new paths, and celebrate the relationship between art and technology.During our research, we see the evolution of abstract art led by genetic algorithms, a process that demonstrates the power of algorithms to create fascinating, thought-provoking works of art. We explore the balance between algorithmic precision and artistic intuition, exploring how collaboration between humans and machines can inspire innovation and create abstract art that resonates with audiences.  
The combination of genetic algorithms and abstract art reveals a land of endless possibilities. The journey doesn't end here, but it calls us to push the boundaries of creativity, find new paths, and celebrate the relationship between art and technology.

**6.2 Future Scope**

**AI-Generated Art Galleries**: Establishing galleries and exhibitions dedicated to AI-generated abstract art can foster appreciation for algorithmic creativity and offer a platform for showcasing the latest advancements.

**Cross-Disciplinary Collaborations**: Collaborations between artists, computer scientists, psychologists, and other experts can enrich the field. Interdisciplinary research can yield new insights into the aesthetic and psychological aspects of abstract art.

**User-Centric Interfaces**: Enhancing user interaction and control within the system is a significant future scope. Intuitive interfaces that allow users to actively shape the evolutionary process, adjust artistic parameters, and provide feedback will make the technology more accessible to artists and art enthusiasts.

**REFERENCES**

1. Das, S., & Suganthan, P. (2011). Differential evolution: A survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation, 15*, 4–31.
2. Holland, J. (1975). *Adaptation in natural and artificial systems* (p. 100). Ann Arbor: University of Michigan Press.
3. Gao, W., & Liu, S. (2011). Improved artificial bee colony algorithm for global optimization. *Information Processing Letters, 111*, 871–882.
4. Karaboga, D. (2005). *An idea based on honey bee swarm for numerical optimization*. Technical report-tr06, Erciyes university, engineering faculty, computer engineering department, (Vol. 200, pp. 1--10).