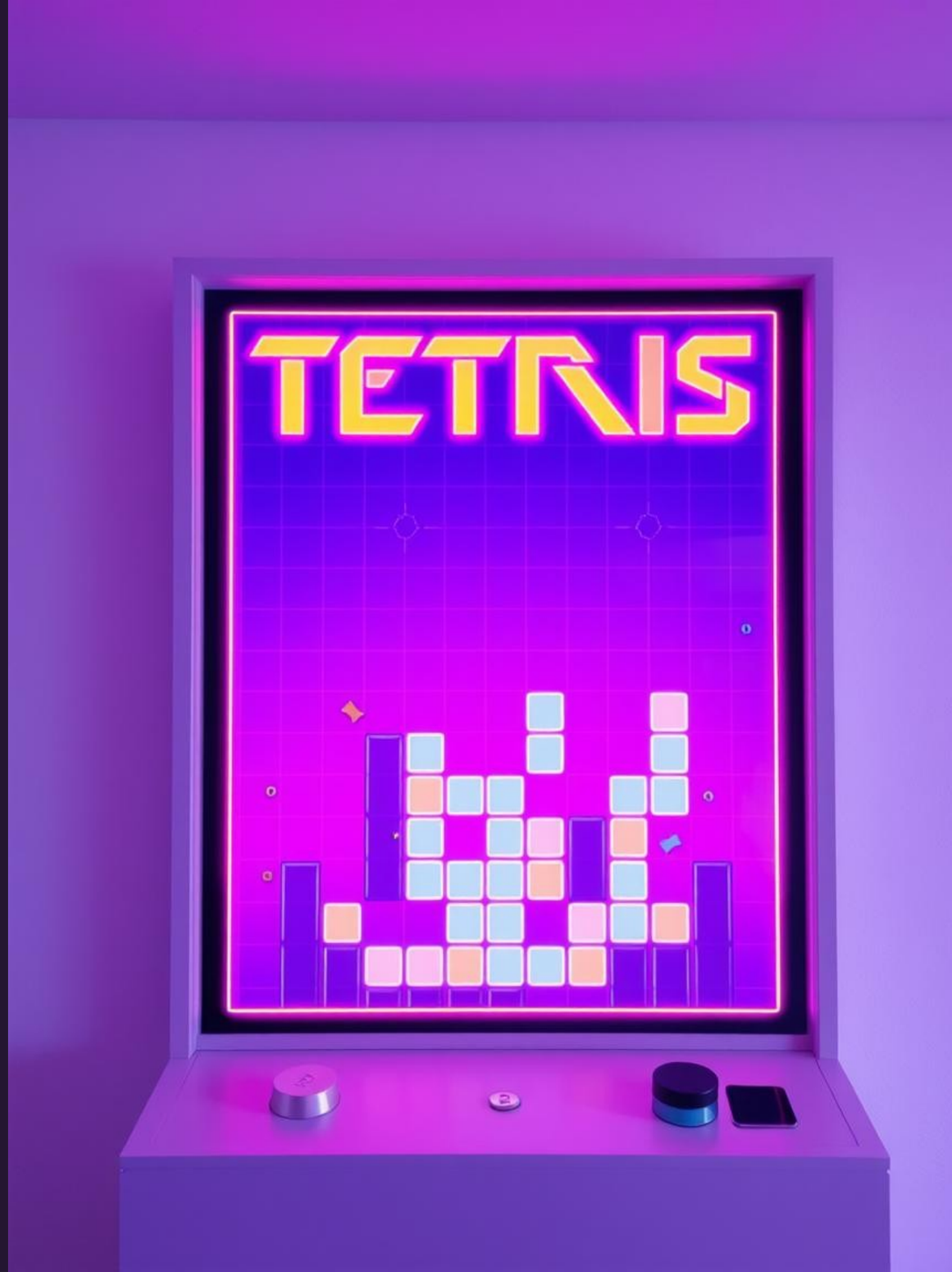
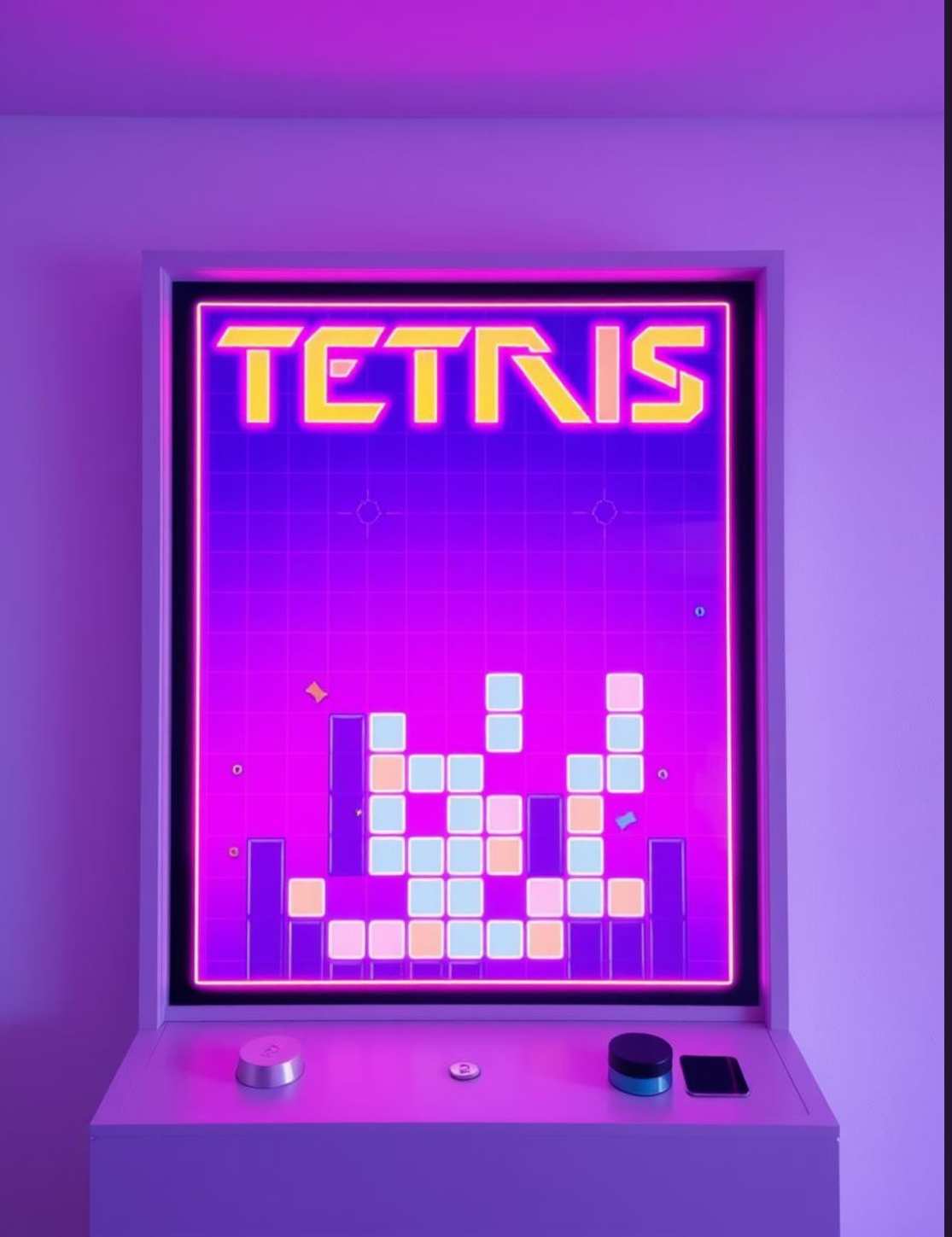
Task 1: Tetris Game with Genetic Algorithm





Introduction

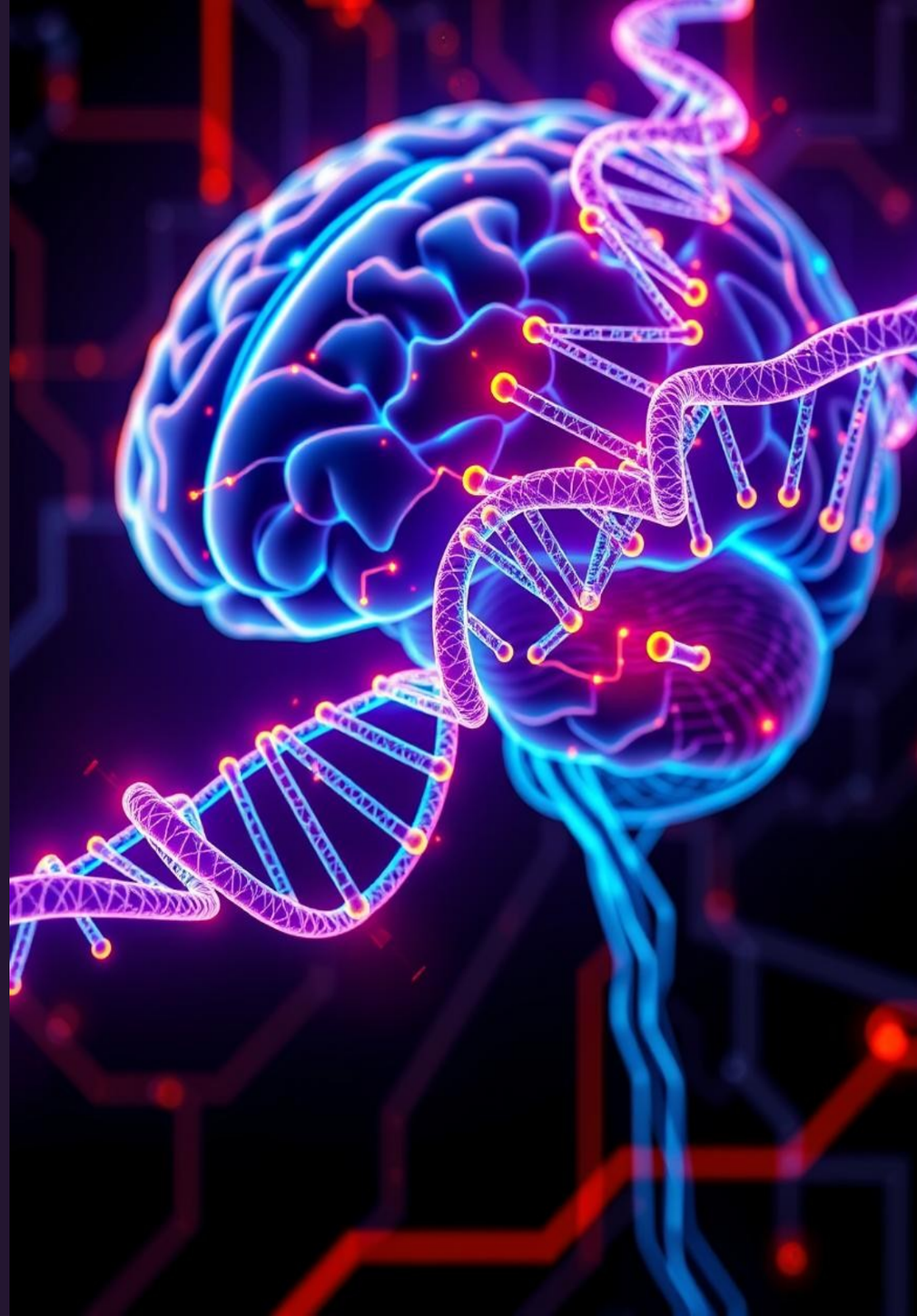
Our implementation utilizes Python with Pygame for visualization and employs a genetic algorithm with a population size of 12 individuals evolving over 10 generations. Each AI player is represented by a set of weights that determines how it evaluates different aspects of the game state.

Through the process of selection, crossover, and mutation, the algorithm gradually discovers more effective playing strategies

The results demonstrate how genetic algorithms can be effectively used to develop game-playing strategies without relying on predetermined rules or patterns, showing the power of evolutionary computation in game AI development.

I. Genetic Algorithm

- **Genes**: Each agent has 6 weights to evaluate game moves.
- **Population**: 12 agents initialized with random weights [-1, 1].
- **Fitness**: Based on Tetris game score after 400 moves.
- **Selection**: Tournament selection (best of 3).
- **Crossover**: One-point, mixing genes from two parents.
- **Mutation**: 20% chance to slightly alter genes using Gaussian noise.
- **Elitism**: Best agent is preserved in each generation.
- **Generations**: Process repeats for 10 iterations to evolve performance.



II. Optimal Tetris

In this part we used the genetic algorithm we made in the genetic_Tetris python file and the implementation of the game in the tetris_game file and simply the we take the best weights achieved y the genetic algorithm in the best_weights file and run the game using it for 600 Iteration and then document the score it achieves.

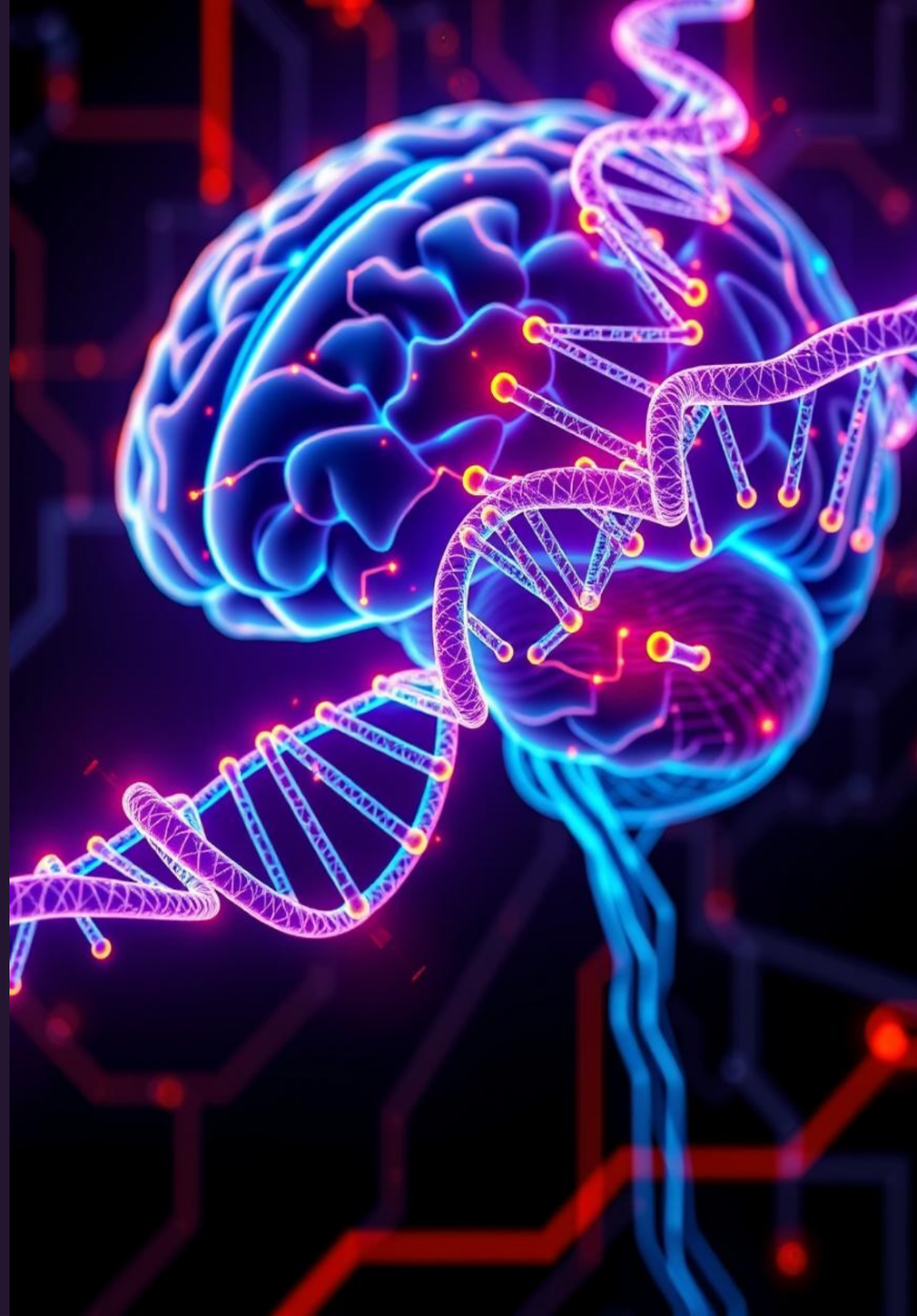
Weight Value	Feature	Effect
-0.250919762305275	num_removed_lines	Reward
0.8571282571167473	max_height	Penalty
0.7601189741906214	new_holes	Penalty
0.24343450017773158	new_blocking_blocks	Penalty
0.9265408472791845	piece_sides	Reward
0.7137977027016299	floor_sides + wall_sides	Reward

III. Results

After using the hyperparameters in the table below:

Hyperparameter	Value	Description
seed	42	Random seed for reproducibility
population_size	12	Number of individuals (chromosomes) in each generation
num_generations	10	Number of generations for evolution
num_iterations	400	Number of Tetris pieces (moves) played per individual evaluation
tournament_size	3	Number of individuals competing in tournament selection
mutation_rate	0.2	Probability that a mutation occurs for each gene/weight
mutation_scale	0.3	Degree to which a gene/weight is mutated

The optimal weights was saved and then used in the optimal_ tetris file to test the game based on them for 600 iterations and the final score for these iterations was **9980**



IV. Genetic Algorithm Progress