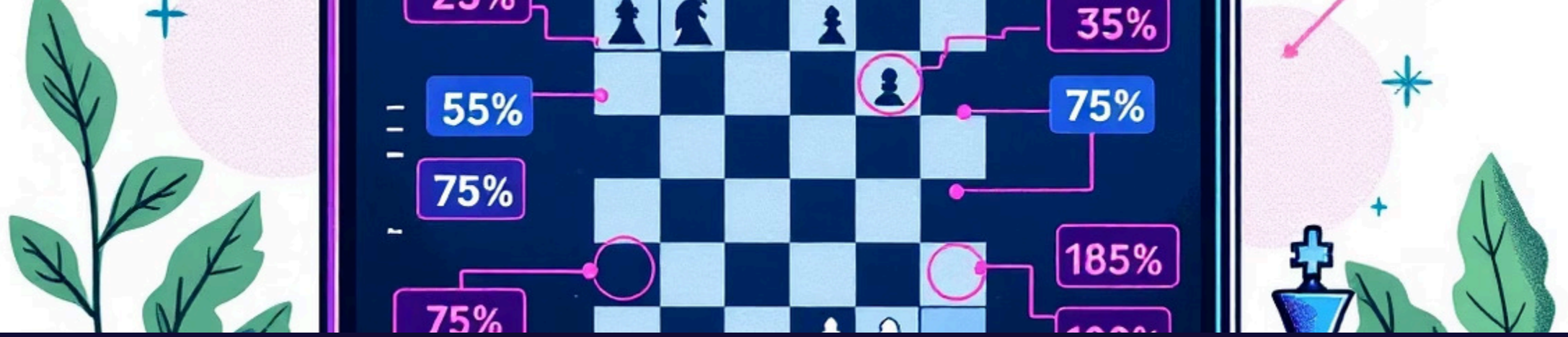
# Monte Carlo Tree Search: Intelligent Game Strategy

A powerful approach for computers to play chess and other strategy games by intelligently exploring possible moves and selecting the best options.




# How MCTS Works: The Basics

### Making a Move Tree

Create a tree where each branch represents a different possible move and response.

### Running Simulations

Play thousands of random games from each position to test outcomes.

### Checking Results

Count how often each move leads to wins, losses, or draws.

### Choosing the Best Move

Select the move with the highest probability of success based on simulations.

# The Challenge of Chess

A chess game tree is **incredibly vast**, with too many possible moves to explore fully:

- ~50 legal moves in a typical position
- After just two moves: 2,500 different positions
- Even powerful computers cannot analyze every possible game path

Instead of attempting an exhaustive search, MCTS focuses on exploring the most promising moves through continuous exploration and learning.

# Balancing Exploration vs. Exploitation

### Initial Random Selection

At the beginning, MCTS selects moves randomly from all legal options, treating them equally.

### Learning from Success

As simulations continue, the search begins to favor moves that have led to successful outcomes more often.

### Maintaining Diversity

Occasionally selects less-explored options to ensure no potentially strong move is overlooked.

This balance between focusing on known strong moves (exploitation) and testing new possibilities (exploration) is what makes MCTS powerful.

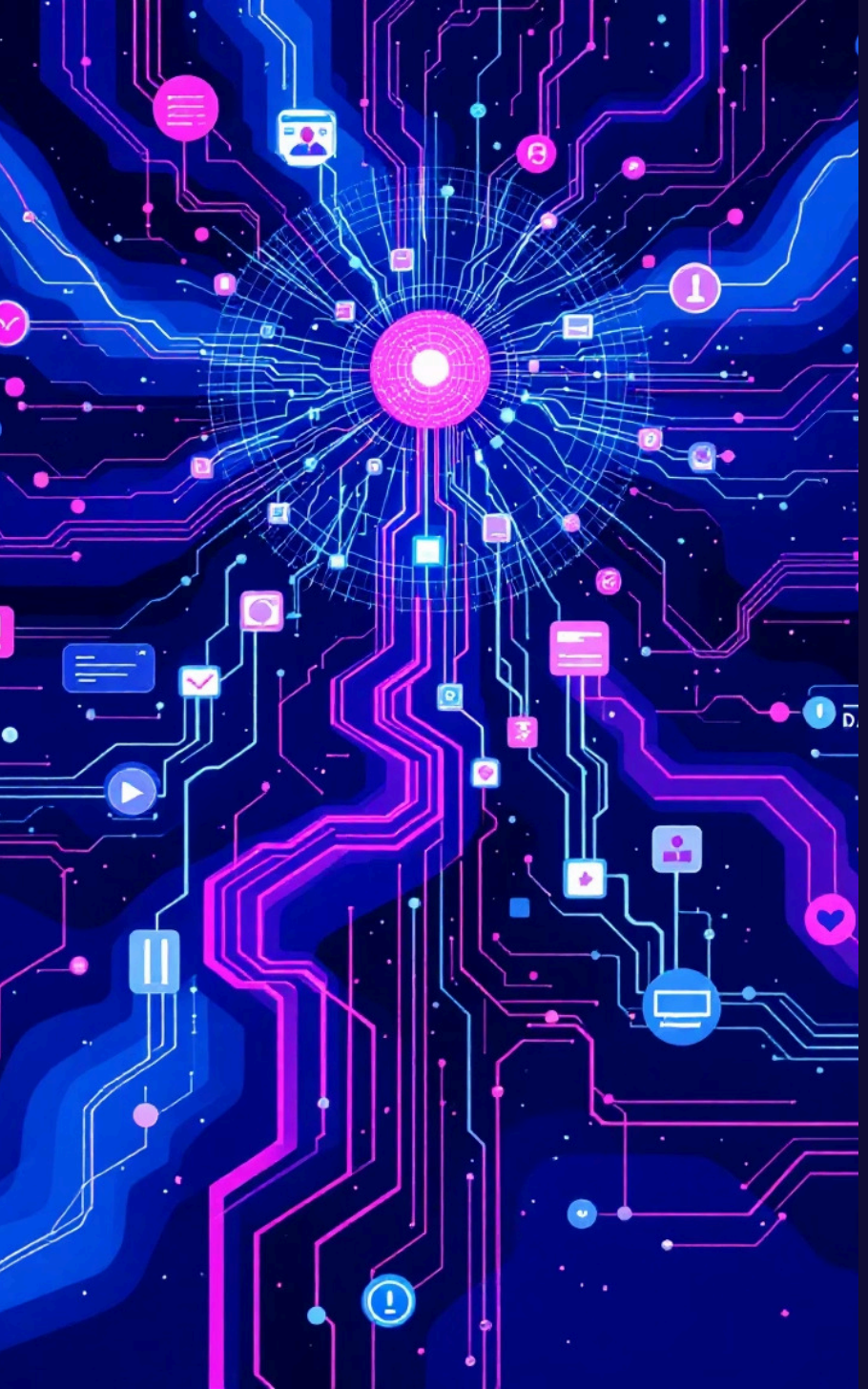
# Mathematical Decision Making

To manage the exploration-exploitation balance, MCTS uses mathematical formulas like the Upper Confidence Bound (UCB1) to decide whether to:

- Explore a new, less-tested move
- Reinforce an already successful strategy

This helps MCTS refine its search intelligently, improving decision quality over time without requiring exhaustive calculations.

By repeatedly simulating games and adjusting its choices, MCTS efficiently finds strong moves, even in complex games where brute-force search would be impractical.



# Our Enhanced Approach: Neural MCTS

## Combining MCTS with Neural Networks

**The Problem with Standard MCTS:**

- Early moves chosen randomly
- Requires many simulations to reach strong conclusions
- Inefficient exploration of weak moves

**Our Solution:**

- Neural network guides MCTS toward better moves from the start
- Reduces random choices
- Makes each simulation more meaningful

# Smart Move Filtering

**Standard Position**

~50 legal moves available

**AI Filtering**

Selects only 3-10 best moves

**Focused MCTS**

Deeper analysis of quality moves

This approach significantly reduces the number of branches MCTS needs to explore, making simulations faster and more efficient.

Risk: If the AI fails to include the best move in its selection, MCTS will never consider it.

# Implementation Details

### Our MCTS Implementation

We provide the code to deploy our enhanced MCTS system. You can adjust input variables to optimize performance, but be cautious:

- Some changes may slow down the system
- Others might make it impossible to run on your machine

The implementation balances computational efficiency with strategic depth, creating a powerful chess-playing system.

# Deployment Process

### Web Service Creation

Gradio creates a web service accessible via browser and as an API for other applications.

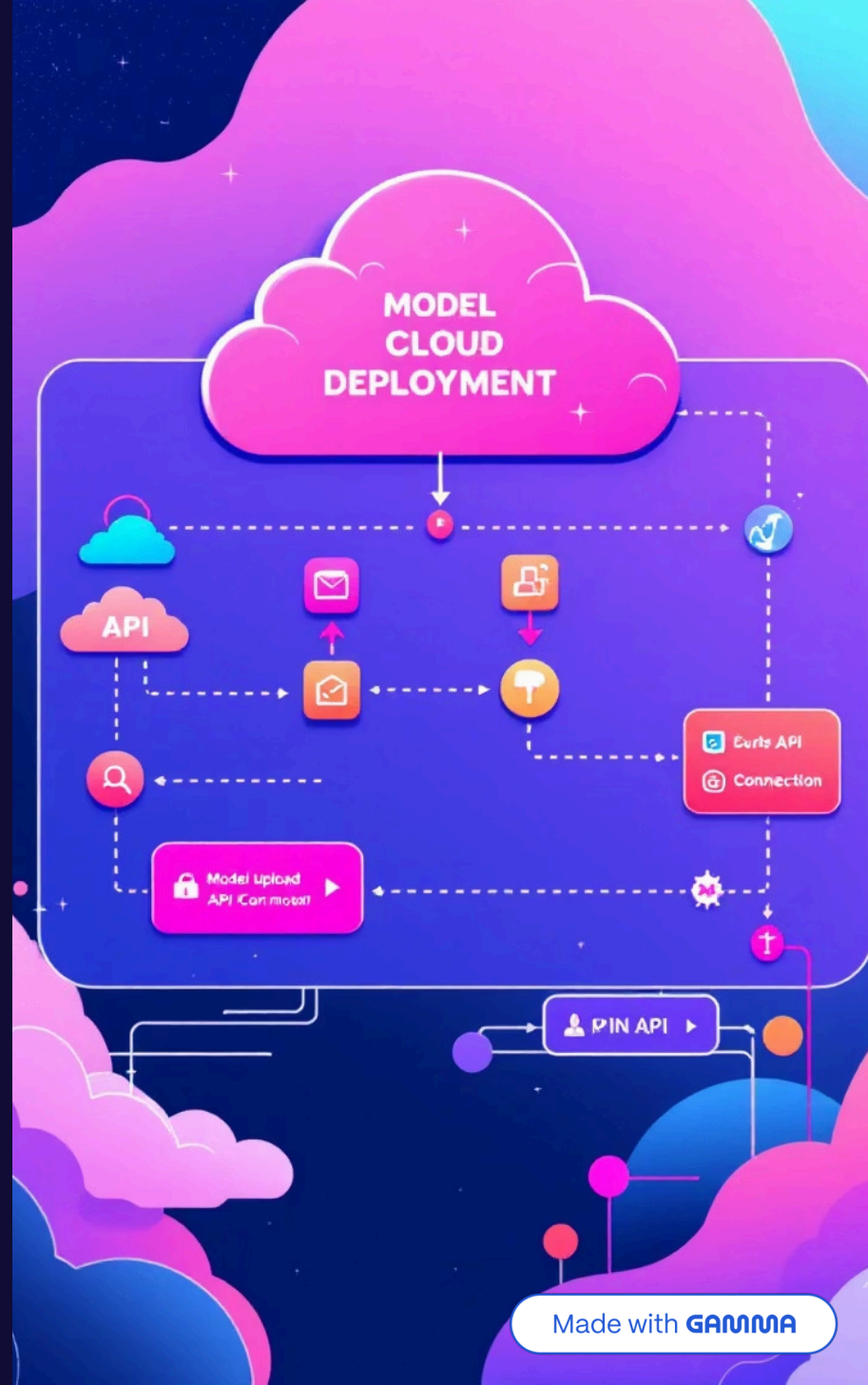
### Model Uploading

We use Huggingface to upload the trained model, which will be downloaded by a Google Colab Notebook.

### API Integration

The deployed model serves as an API for your AI player in competition.

We provide the deployment code and don't recommend modifying it. Copy the URL and paste it into our web platform to test it.



# Competition Setup

**How the Competition Works:**

- Choose your AI player and opponent
- Players compete with limited time per move
- Multiple games determine the best AI

Your trained AI will be downloaded from Huggingface to compete, showcasing the effectiveness of your implementation of Neural MCTS.


This competition format tests both the strategic depth of your AI and its ability to make decisions efficiently under time constraints.