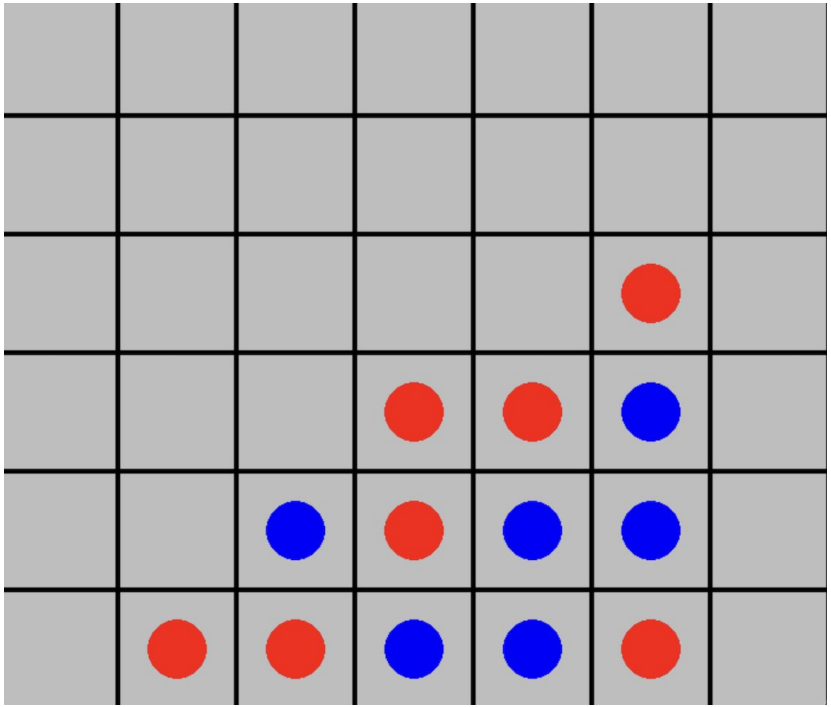
Overview

- Connect-4 is a classic two-player strategy game played on a grid board where players alternate turns, dropping one of their discs into any of the columns.
- The disc will fall to the lowest unoccupied space in that column.
- The goal is to be the first to create a line of four of your colored discs either horizontally, vertically, or diagonally.
- Our game plays on a 6x7 grid



Initial Goals and Objectives

- Our objective was to create a adversarial AI model to play a realistic game of Connect-4
    - Model needs to be able to take optimal offensive actions to connect 4 of their own pieces in a row
    - Model needs to be able to take optimal defensive actions to block the human from connecting 4 in a row
- This is a valuable problem to solve since it creates a model that can be used for players to improve by playing, while still having a realistic enjoyable experience
- This model also is valuable as it shows the ability to use neural networks to solve real problems that require decision-making in uncertainty such as logistics, resource allocation, and scheduling

Methodologies (Neural Network):

- Using Pygame for the user interface, our solution works through a neural network we defined using the PyTorch library that inherits from “nn.Module”
- We defined our neural network to have four fully connected linear layers that, when used consecutively in a forward pass with the Rectified Linear Unit (ReLU) activation functions in between, reduce an input of size 42 into 3
- 3 states: -1 (human wins), +1 (AI wins), 0 (draw).

```
class NN(nn.Module):
    def __init__(self, input_size, num_classes):
        super(NN, self).__init__()
        self.fc1 = nn.Linear(input_size, 34) # How many nodes in layer 1
        self.fc2 = nn.Linear(34, 26) # How many nodes in layer 1
        self.fc3 = nn.Linear(26, 18) # How many in layer 2 and so on
        self.fc4 = nn.Linear(18, num_classes)

    def forward(self, x):
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = F.relu(self.fc3(x))
        x = self.fc4(x)
        return x
```

Methodologies (Minimax):

- 3 levels of difficulty by the AI: easy (depth=1), medium (depth=5), hard (depth=10)
- Utilizing minimax, the AI chooses the most optimal action after using Depth-limited search on the adversarial game tree at the depth respective to the chosen difficulty.
- Terminal utilities are replaced with an evaluation function for non-terminal positions
- This allows our AI to make long term strategic decisions

```
def minimax(self, depth, player):
    if depth == 0 or GameState.is_finished():
        return self.evaluate_board(GameState.board)

    # Maximizing
    if player == 1:
        score = float('-inf')
        # Going down each valid move
        for col in range(7):
            if GameState.is_valid_move(col):
                row = GameState.make_move(col, player)
                score = max(score, self.minimax(depth - 1, player * -1))
                GameState.undo_move(row, col)
        return score
    # Minimizing
    else:
        score = float('inf')
        for col in range(7):
            if GameState.is_valid_move(col):
                row = GameState.make_move(col, player)
                score = min(score, self.minimax(depth - 1, player * -1))
                GameState.undo_move(row, col)
        return score
```

Outcomes (Results & Evaluation):

- We played 50 games on each level of difficulty to compare the AI's performance
- The expert AI had a win rate of 46% which is 32% percent greater than the dumb AI's win rate
- During evaluation, we noticed that the hardest difficulty AI was making more long-term strategic plays whereas the lowest difficulty AI was only making the moves that would have the highest immediate reward

Wins out of 50			
Difficulty	Human Wins	AI Wins	Win Percentage
Dumb	43	7	14%
Mid	39	11	22%
Expert	27	23	46%

Outcomes (Observations & Challenges):

- An unexpected outcome where if our strategy consisted on placing consecutive discs along the bottom row, the AI would not be able to counter it
- AI seemed to prioritize offensive plays over defensive plays.
- The AI plays significantly better towards the end of the game rather than the beginning