Presentation outline (if you want to add ur own script lines add em here)

Introduction, framing the problem

* Deep learning has seen huge success in perfect information games
  + Applying these concepts/strategies into hidden information and strategic deception remains an open challenge
* Compare hidden vs complete

**Complete Information Games:**

* All players know everything about the game: players, actions, sequence, and payoffs for all outcomes.
* The rules and consequences are common knowledge.
* Attached image, mention how that with perfect information, a tree/graph can be used to simulate certain actions. Image shows AlphaZero, Google’s super AI for playing chess. While AlphaZero doesn’t create a fully visualized map, it uses tree search algorithms (i.e MCTS) to explore different branches of the tree.
* Examples: Chess, Tic-Tac-Toe, the basic Prisoner's Dilemma.

**Incomplete Information Games:**

* At least one player lacks full knowledge about some relevant aspect of the game (e.g., other players' payoffs or characteristics).
* Involves private information
* Examples: Poker
* Why Traditional Methods don’t work

Approach to the Problem: simplest way possible to explore characteristics of the problem

* Bluffing game similar to poker
* In Bluff Game, each round, every player is assigned a random integer value between 1 and 10 in the form of a card, representing the strength of their hand—where 1 is the weakest and 10 is the strongest.
* On their turn, which occurs multiple times within a round, players can choose from four standard actions: check, call, raise, or fold
  + The game follows traditional Texas Hold’em rules for small and big blinds (required initial bets), round termination, and determining the winner.
* Unlike Texas Hold’em, there are no public cards or shared values; each player only knows their own hand strength and the number of chips each player
* To simplify the game, we assume that each player has an unlimited supply of chips. However, to prevent unrestricted betting, a fixed betting structure is imposed. Raises occur in increments of 5 chips, with a maximum bet cap of 100 per round, ensuring that the game remains focused on decision-making under uncertainty rather than resource hacking.

Baseline, using a DQN

* Since we’ve developed an alternative game to learn hidden information, we decided to use pre-developed algorithms to develop a baseline understanding
* We implemtened a Deep Q-Network (DQN) agent using a multi-layer perceptron (MLP) for Q-learning
  + Note that this is simply used as a baseline and is likely not out final model/approach
* Our current baseline model uses a DQN agent with a replay memory to store past transitions and train on batches of sampled experiences
* Network architecture consists of four MLP layers with outputs of 64, 128, 64, and 32 neurons, followed by a final linear layer to estimate Q-values.
  + Each MLP block applies a Tanh activation function to introduce non-linearity
* To balance exploration and exploitation, the epsilon-greedy strategy [6] is used: the agent chooses a random action with probability ε and the greedy (best-estimated) action with probability 1 − ε. The value of ε decays over time, initially favoring exploration and later shifting toward exploitation.

Baseline is poop, why is it bad, why our model might be good

Our Approach + Results

* Probabilistic Action Selection
  + For imperfect games, the optimal strategy cannot be determined by knowing the values of states
    - This is the inherent problem of imperfect information games in which you cannot know the value of each state
  + For q-learning, this problem is fixed by maintain a specific belief about other players’ strategy and use these beliefs to determine the values at specific states
  + However, the select belief strategy is likely to vary between different players
  + Hence, the model may not be robust against various types of opponents, potentially performing well against certain players while struggling against others
* Computation time
  + Since q-learning involves a tree search, even search algorithms such as Monte-Carlo search algorithms still require huge levels of computation
  + Research has also shown that since Q-learning directly parameters and updates value functions without explicitly modelling the environment, a lot of input data would be needed to develop a sufficient model-algorithm
    - <https://proceedings.neurips.cc/paper_files/paper/2018/file/d3b1fb02964aa64e257f9f26a31f72cf-Paper.pdf>

Our Model

* Local minima easier to find in two smaller functions compared to one big thing
* Mixed strategy

Reflection

* Tehcnically, a NN can learn anything, but being an AI engineer is about finding the best way for the network to learn