Hidden Information, Teamwork, and Prediction in Trick-Taking Card Games

Hadi Elzayn, Mikhail Hayhoe, Harshat Kumar, Mohammad Fereydounian

hads@sas.upenn.edu, mhayhoe@seas.upenn.edu, harshat@seas.upenn.edu, mferey@seas.upenn.edu

Rules of *Four Hundred*

- Players sit across from team members and are dealt 13 cards each
- Players bet expected number of 'tricks' they plan to take
- In each round after bets are placed
  - Players take turns choosing a card to play in order, starting with the player who took the previous trick
  - Suit of first card played determines 'lead suit'
  - Players must play 'lead suit' if they have it
  - Winner of trick is determined by highest card of 'lead suit' or highest card of 'trump suit' (Hearts) if any were played
- After 13 rounds, score of each player is increased by number of tricks they bet if they met or exceeded it
- Score of player is decreased by bet if they were unable to meet it
- Game concludes when one team member has 41 points or more, and the other player has positive points

Betting

Learning Problem: Find an optimal betting policy $\beta^{i,*}$ to maximize expected reward for player i given each other and the play of the others:

$$\beta^{i,*} = \arg\max_{\beta \in \mathcal{B}} E\left[\sum_{t=0}^{13} R(\beta(\mathcal{H}_0^i), \pi_t^{i,*}(\mathcal{H}_t^i))\right]$$

Supervised Learning Approach: Given some particular card playing strategy and initial hand, the play can expect to win some number of tricks. Therefore, develop a model which predicts tricks taken by the end of the round given some observed initial hand compositions

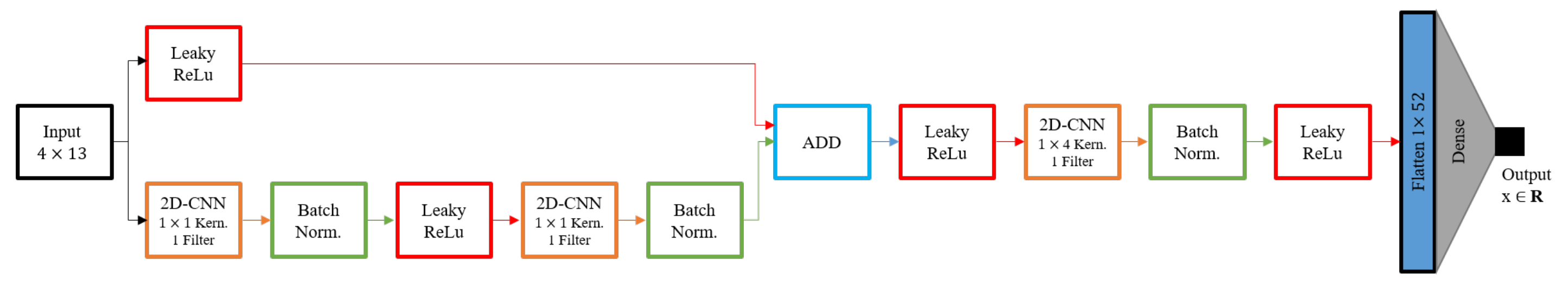
Generate Data: Initial hands serve as input data, number of tricks won functions as label.

Implement Neural Network for regression to minimize loss defined by square difference between score received and the best possible.

$$\ell_{\text{bet}}(y, \hat{y}) = (y - \text{sign}(\hat{y} - y) \cdot \hat{y})^2 = \begin{cases} (y - \hat{y})^2, & \text{if } y \geq \hat{y}, \\ (y + \hat{y})^2, & \text{otherwise} \end{cases}$$

Loss function is asymmetric: Penalizes more for bets which are higher than the tricks obtained

Neural Network Architecture for Betting



Card Play

Learning Problem: Find an optimal playing policy $\pi_t^{i,*}$ to maximize expected reward for player i given each other and the play of the others:

$$\pi_t^{i,*} = \arg\max_{c \in \mathcal{H}_t^i} E[R(\beta^{i,*}, c)|S_t^i]$$

Reinforcement Learning Approach:

We define the reward at the end of each round by

$$\text{Reward} = \begin{cases} \text{bet}, & \text{if tricks} \geq \text{bet}, \\ -\text{bet}, & \text{otherwise.} \end{cases}$$

Approach informed by Neural Fitted Q-iteration [2] in which the reward is provided as a label to the state representation. We define the label as

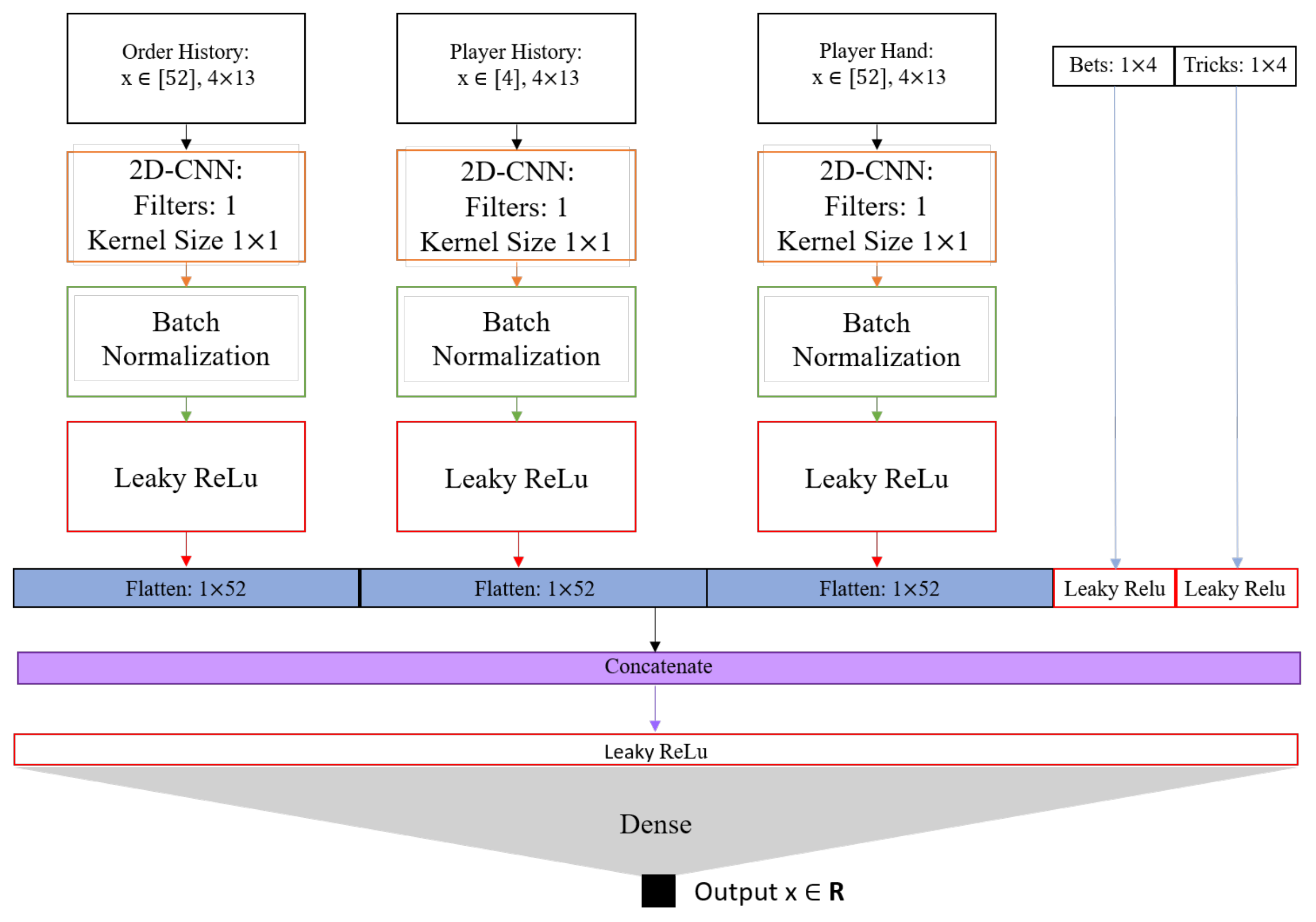
$$\text{Value}(s) = \gamma^{13-t} \cdot (\text{Reward}_{\text{Team member 1}} + \text{Reward}_{\text{Team member 2}}) + \mathbf{1}\{\text{Team won the trick}\}$$

We include the indicator as reward shaping [3] to speed up convergence

The state representation:

- Similar to AlphaGo [4], we encode the state by a matrix representation where the location and value indicate the card, time it was played, and by whom it was played
- Initial bets and number of tricks won are also encoded in the state representation

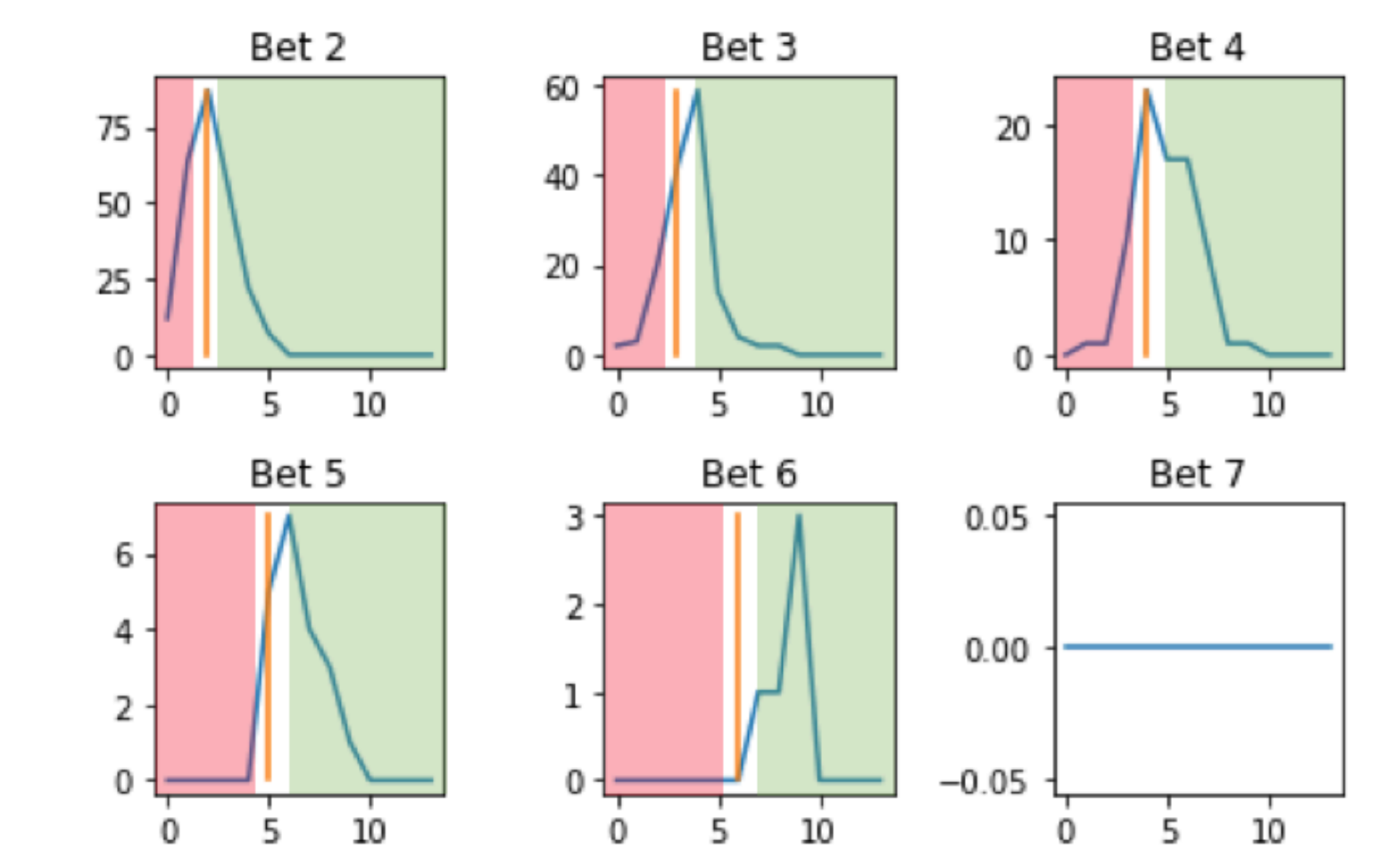
Neural Network Architecture for Card Play


Order History: x ∈ [52], 4×13
Player History: x ∈ [4], 4×13
Player Hand: x ∈ [52], 4×13
Bets: 1×4
Tricks: 1×4
2D-CNN: Filters: 1 Kernel Size 1×1
Batch Normalization
Leaky ReLu
Flatten: 1×52
Leaky Relu
Concatenate
Leaky ReLu
Dense
Output x ∈ **R**

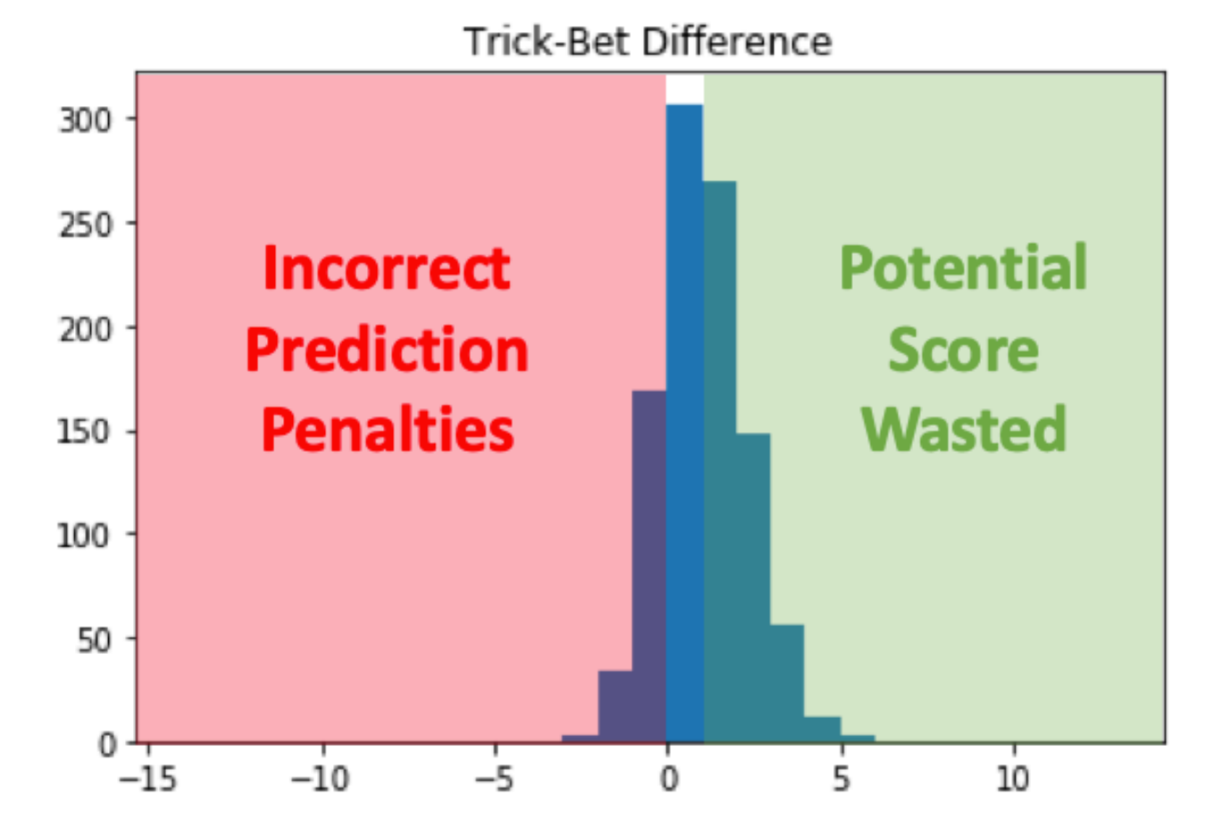
Future Work

- Examine approach on similar games such as *Spades, Hearts, and Tarneeb*
- Consider *invariance discovery* for generalizeation

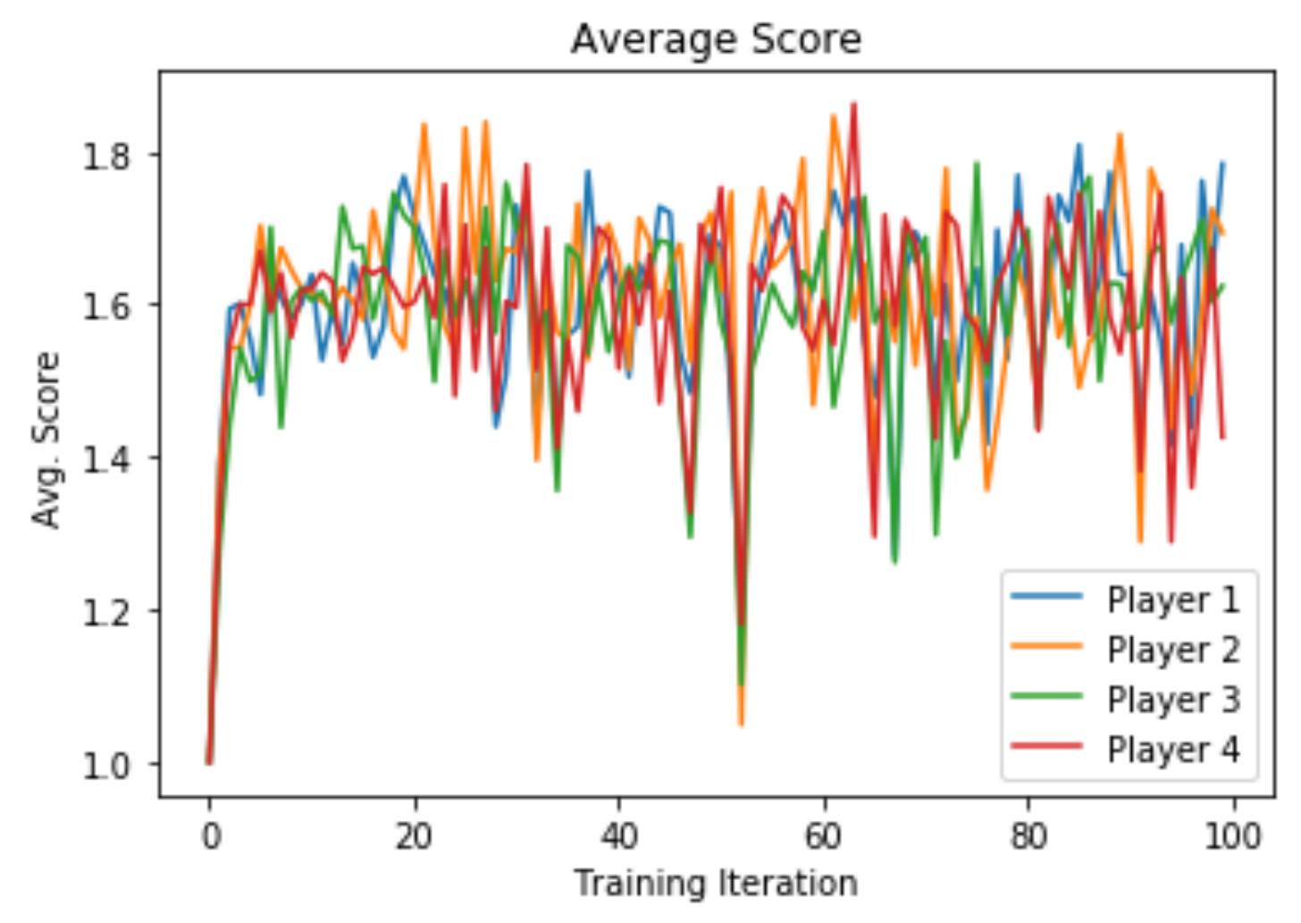
Results



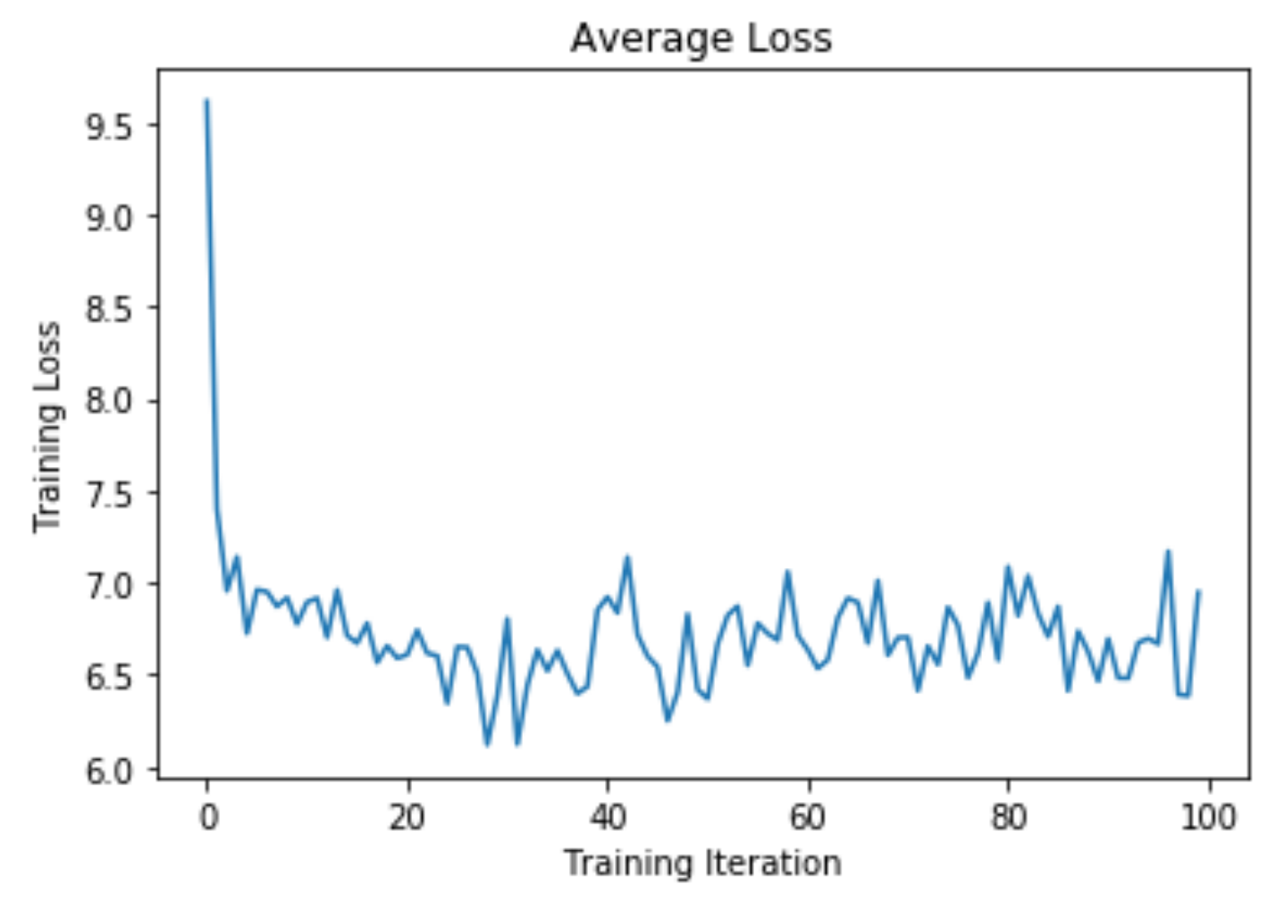
(a) Tricks won over 1,000 games, based on bet made. No bets were higher than 6



(b) Histogram showing difference between tricks won and bet made



(c) Average score showing the performance of the self play NN model



(d) Loss function for the betting NN, trained in tandem with the playing NN

	Random	Greedy	Heuristic	*Over400*
Heuristic Win %	100	100	-	10.5
Over400 Win %	100	100	**89.5**	-

Baselines

Random Play-Random Bet: Random bet selects a random value from {2,3,4,5} during the betting stage, and in each round chooses a random card to play from the set of valid cards.

Greedy Play-Model Bet: During play, greedy simply selects the highest card in its hand from the set of valid cards, without consideration of its partner

Heuristic Play-Model Bet: A heuristic strategy was defined based on human knowledge of the game. This heuristic player takes into account available knowledge of the team member's actions as well as opponents' actions

References

[1] A. J. Hoane Jr. M. Campbell and F. Hsu. Deep blue. *Artificial Intelligence*, 134(1-2):57-83, 2002
[2] M. Riedmiller. Neural Fitted Q iteration – First experiences with a data efficient neural reinforcement learning method. *European conference on Machine Learning,* 05:317-328, 2005
[3] V. Minh et al. Human-level control through deep reinforcement learning. *Nature*, 518:529-533, 2015
[4] D. Silver et al. Mastering the game of Go without human knowledge. *Nature*, 550:354-359, 2017