# Learning Game Board Evaluation

**CSE 60171 – AI Semester Project**


Allison Regier

Shawn O'Neil

# The Problem

**In a MiniMax framework, designing evaluation functions for games can be difficult.**

**Can we develop an evaluation function which learns relative utilities over board states by playing games repeatedly?**

# The Solution

## Solution 1: StringEval

Learns utilities over each board state, has no opinion about boards it hasn't seen. Unable to generalize across boards.

## Solution 2: FeaturesEval

Learns utilities over individual features of board states, combines features utilities to get board utility. Able to generalize features across similar boards.

# The Solution

## Solution 3: NeuralEval

Use a neural network to predict the utility of a given board state.

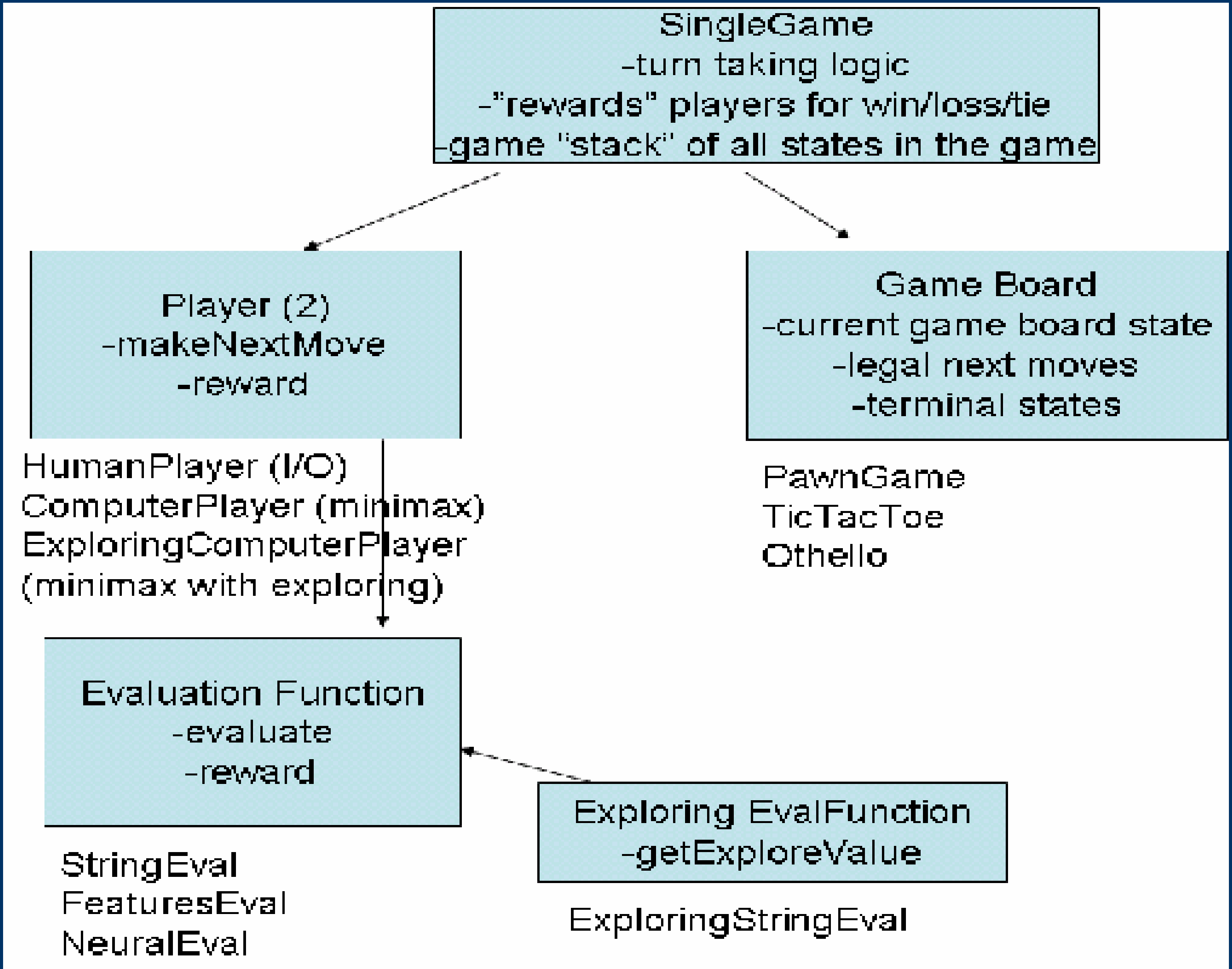
**Inputs:** Each square on the board is an input

**Outputs:** Probability of white winning, probability of a tie

Hidden layer with some number of nodes.

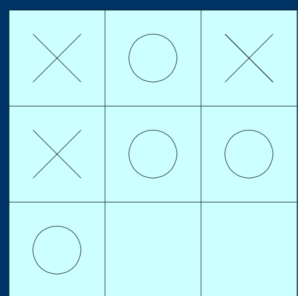
Utility based on some combination of the outputs of the neural network.

# The Solution



# StringEval Overview

## Board States Encoded Uniquely As Strings:



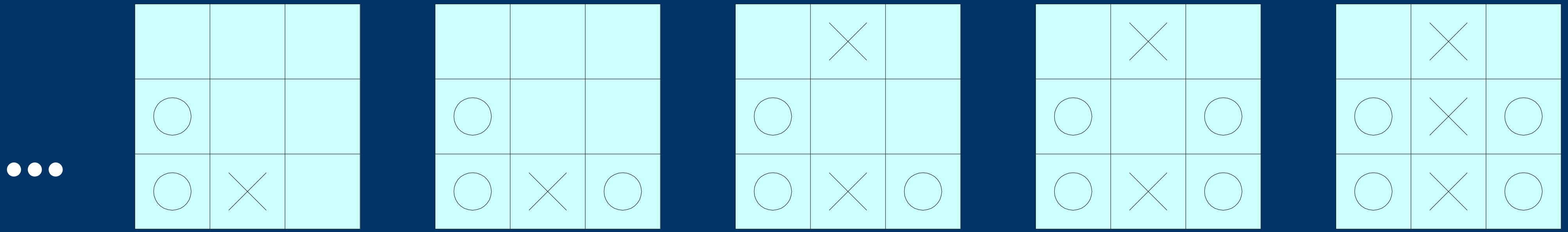
1:-1:1:1:-1:-1:-1:0:0

## And Associated With a Utility:

| | | |
|---|---|---|
| 0:-1:-1:1:1:0:-1:0:-1 | - | 0.75 |
| 1:-1:1:1:-1:-1:-1:0:0 | - | 1.50 |
| -1:-1:0:1:-1:1:-1:0:1 | - | 0.50 |

# StringEval Overview

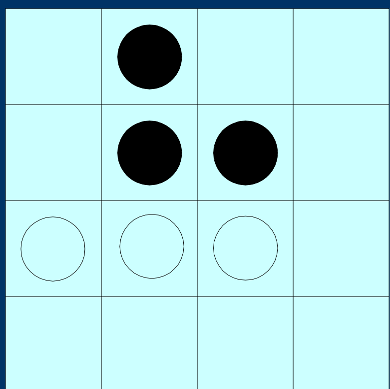
**At End of Game, Adjust Utilities For States Seen Based on Winner and Move Number:**



| | | | | |
|---|---|---|---|---|
| 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| +/- 1/32 | +/- 1/16 | +/- 1/8 | +/- 1/4 | +/- 1/2 |
| 1.03125 | 1.0625 | 1.125 | 1.25 | 1.5 |

# FeaturesEval Overview

## Board States Encoded As Array of Features



Num 1's: 3
Num -1's: 3
Longest Row 1's: 3
Longest Row -1's: 2
...

## Each Value For Each Feature is Given Utility:

Num 1's:

1 : 0.75
2 : -1.25
3 : 0.96
...

Num 1's:

1 : -0.10
2 : 2.35
3 : 5.07
...

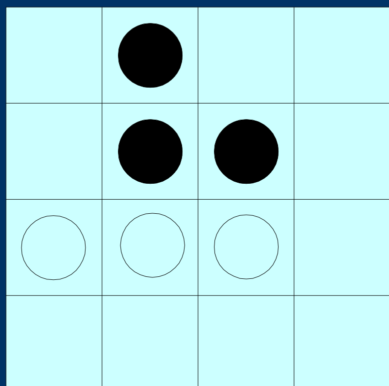
Longest Row 1's:

1 : 1.57
2 : 0.01
3 : 2.53
...

...

# FeaturesEval Overview

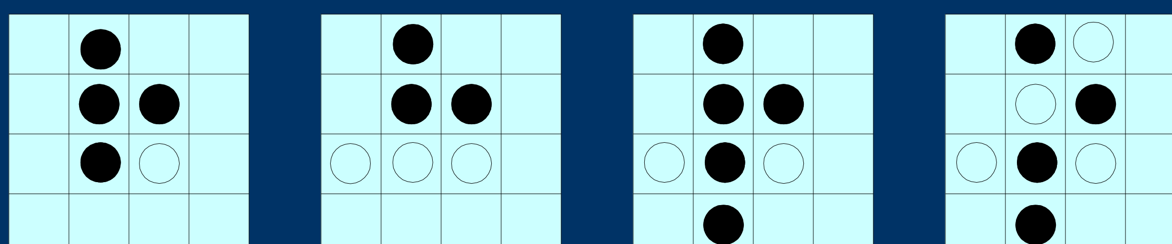
## Compute Board Utility as Sum of Feature Utils:



| | | | |
|---|---|---|---|
| Num 1's: | 3 : 0.96 | | |
| Num -1's: | 3 : 5.07 | | |
| Longest Row 1's: | 2 : 2.53 | → | 5.46 |
| Longest Row -1's: | 2 : -2.10 | | |
| ... | | | |

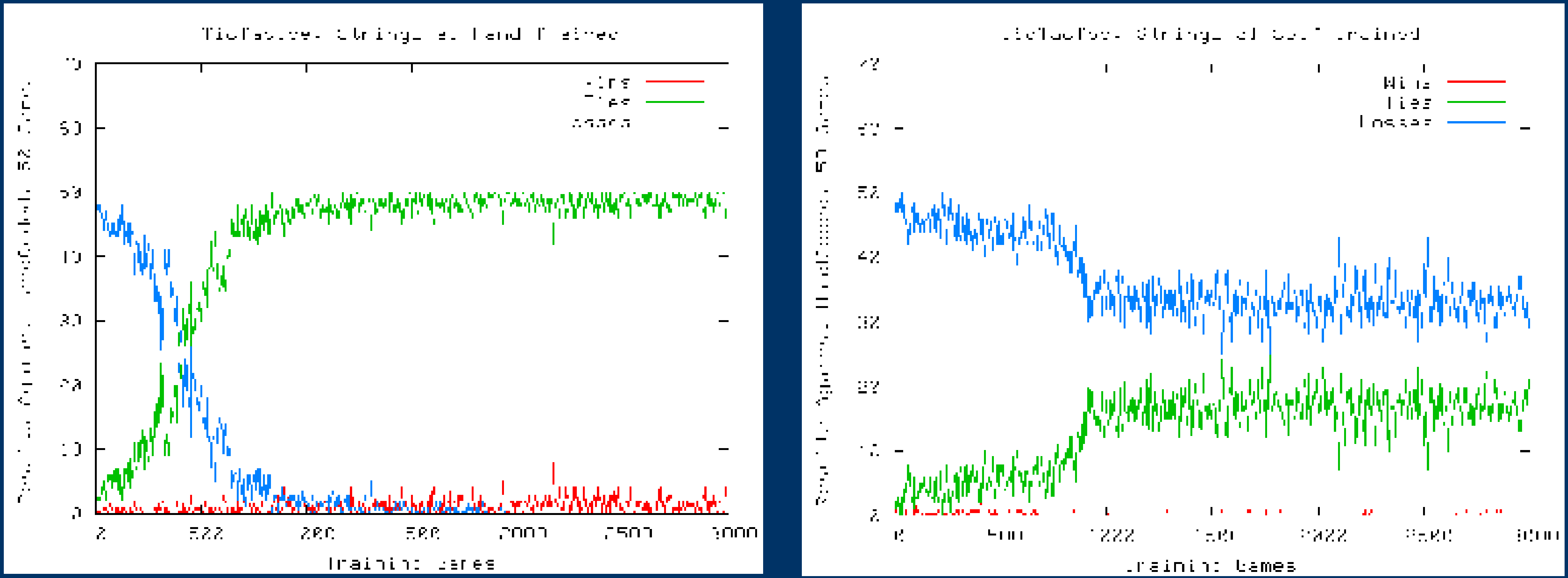
## Update Util's For Feature/Value's by History

Showing Only Num 1's Feature

...  ... White Wins

| | | | | | |
|---|---|---|---|---|---|
| **Old:** | 1 : 1.0 | 3 : 1.0 | 2 : 1.0 | 4 : 1.0 | |
| | +/- 1/4 | +/- 1/3 | +/- 1/2 | +/- 1/1 | ... |
| **New:** | 1 : 1.25 | 3 : 1.33 | 2 : 1.5 | 4 : 2.0 | |

# TicTacToe StringEval

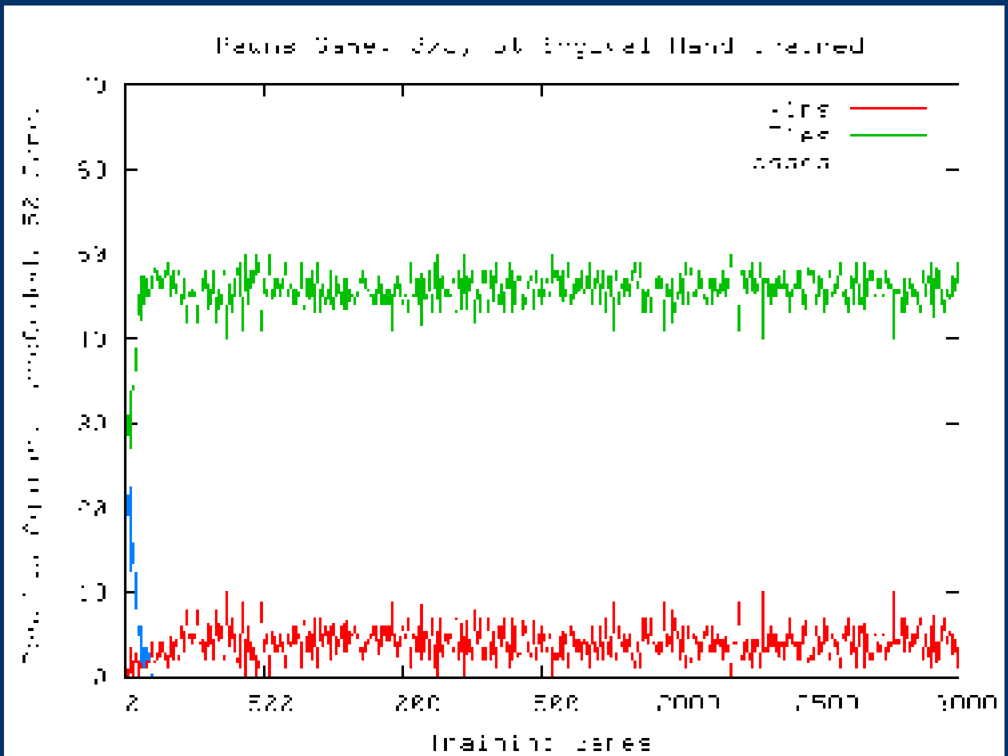


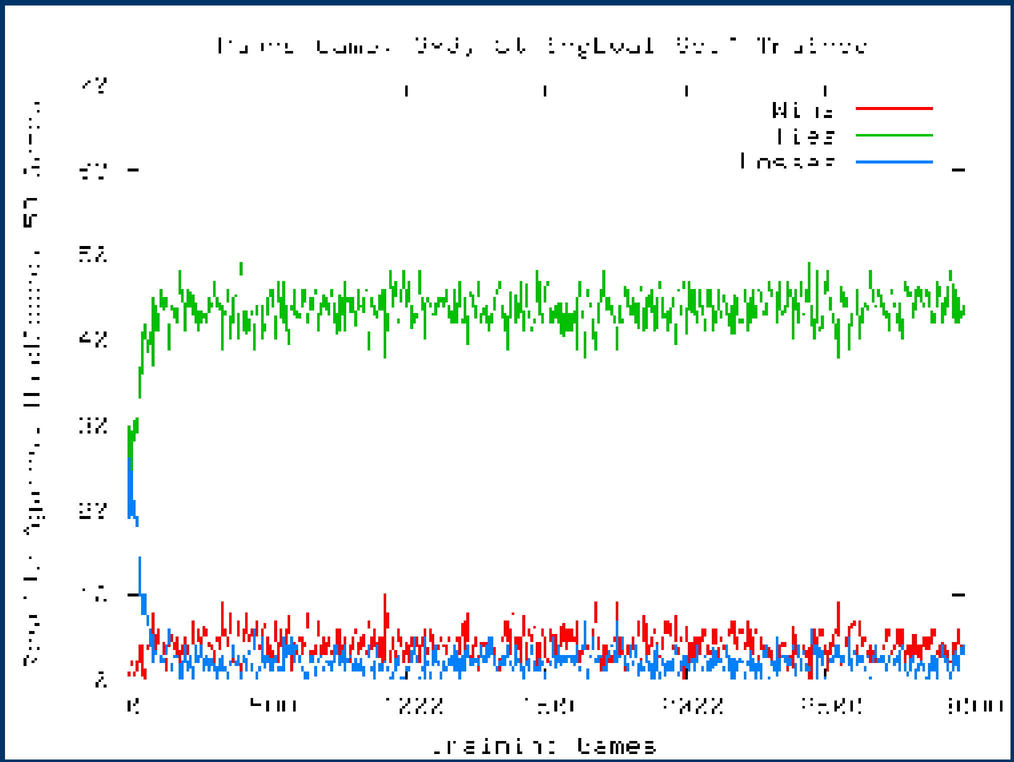

**“Cat's Game” - Ties are Optimal**

**Medium/Large State Space**

**Hand Trained Explores Important States**

# Pawns 3x3 StringEval

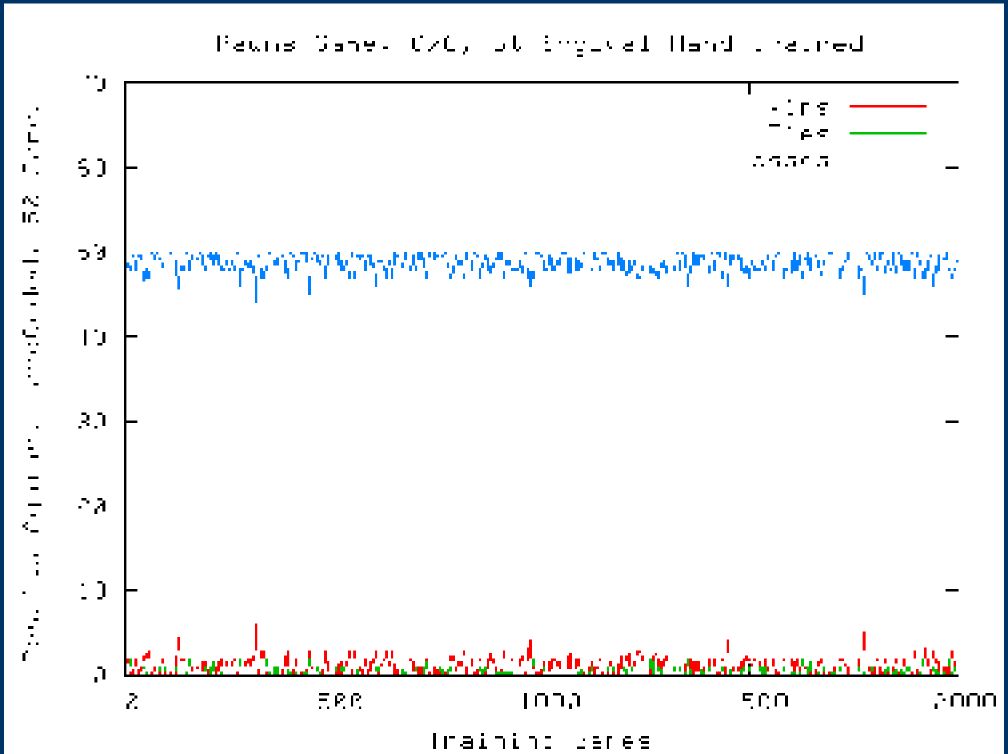


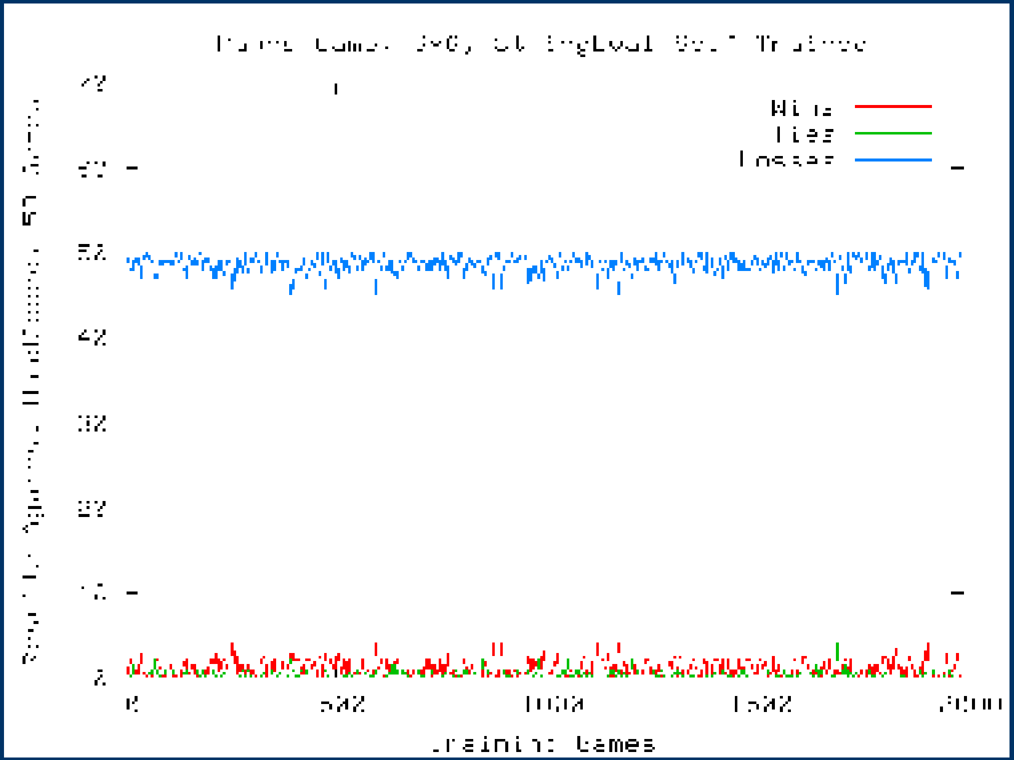


**Also “Cat's Game” - Ties are Optimal**

**Small State Space – Quick Learning**

# Pawns 6x6 StringEval

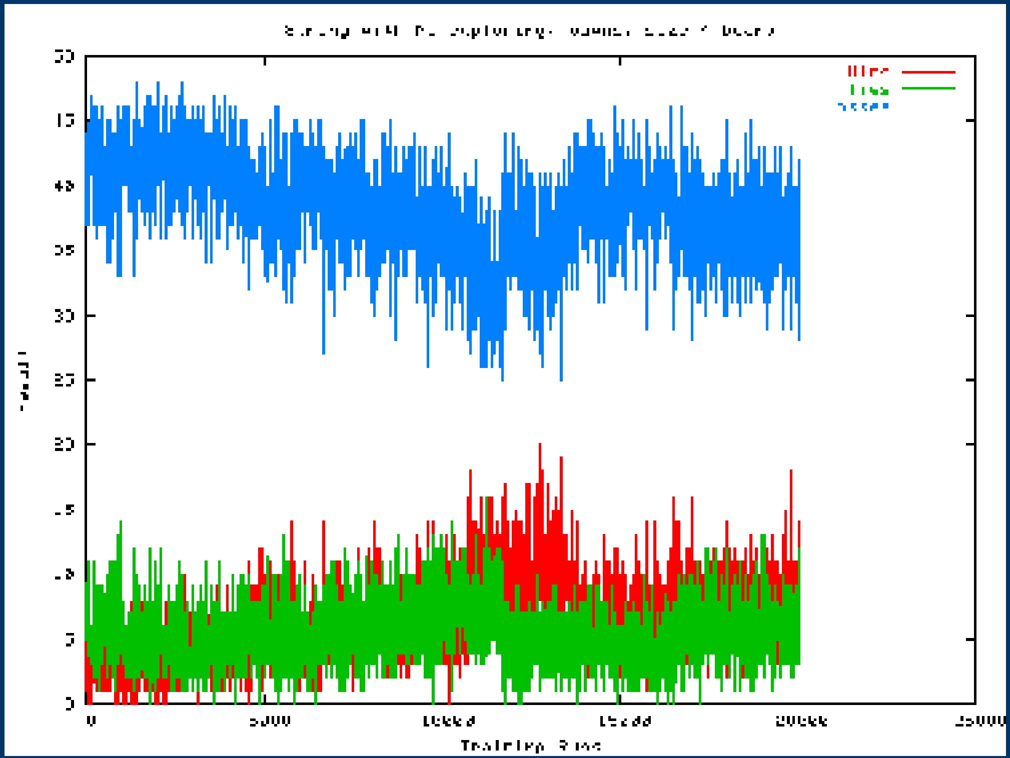


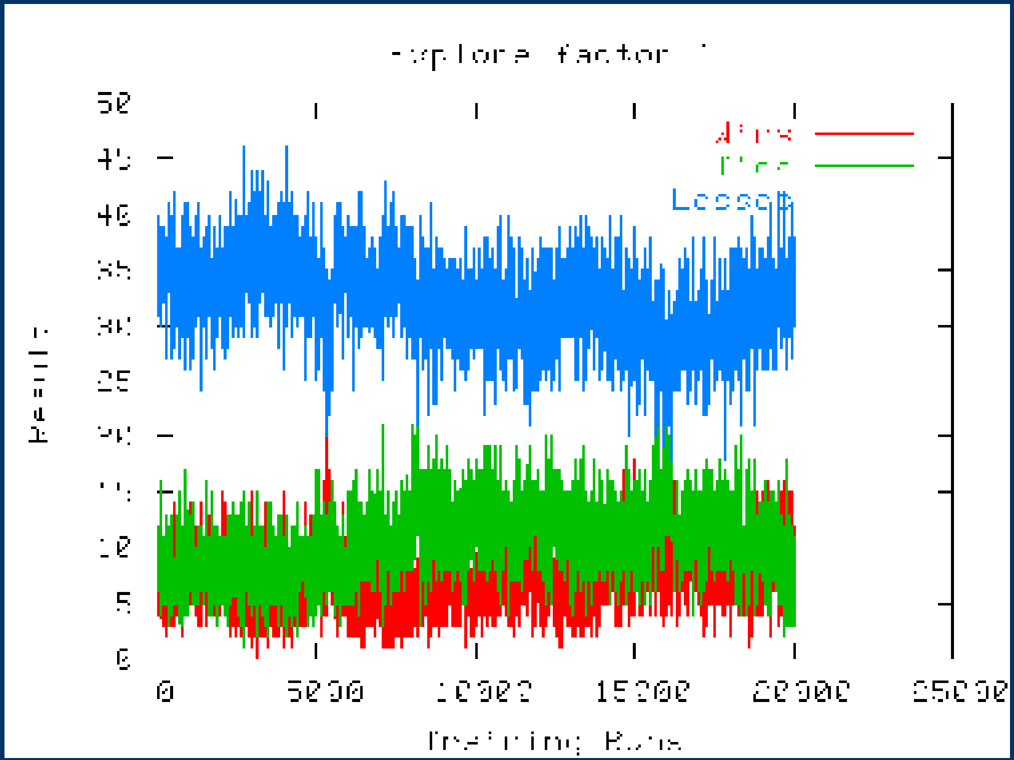


**Very Large State Space**

**No Learning (In 2K Training Games...)**

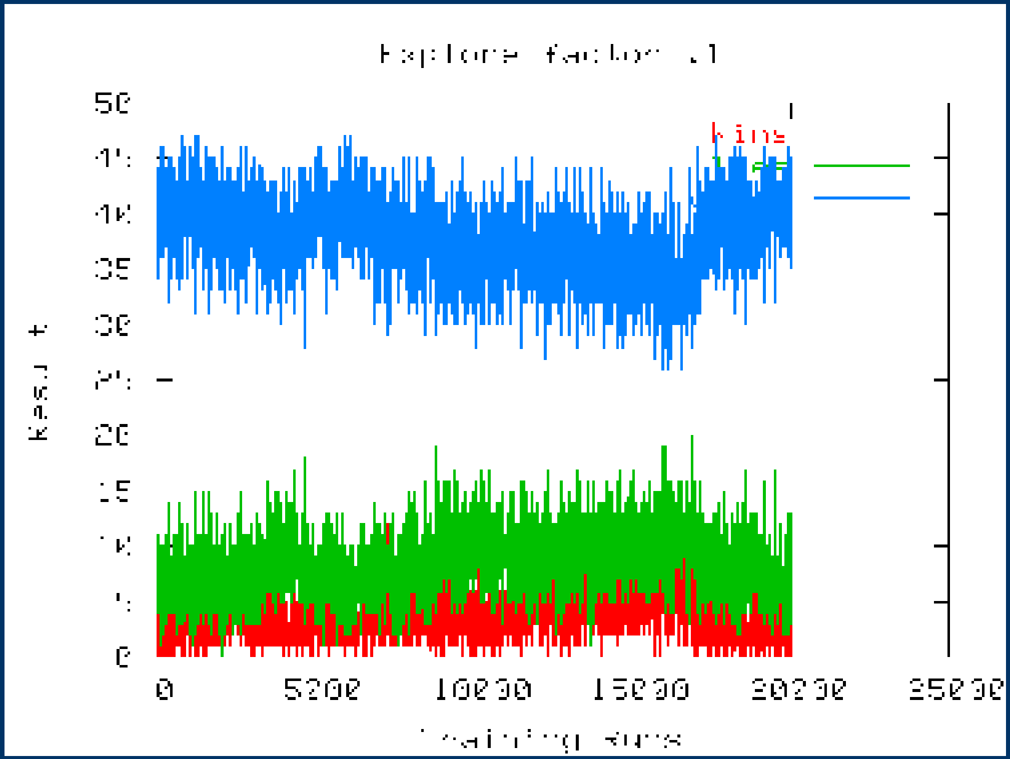
# Pawns 4x4 StringEval Exploring

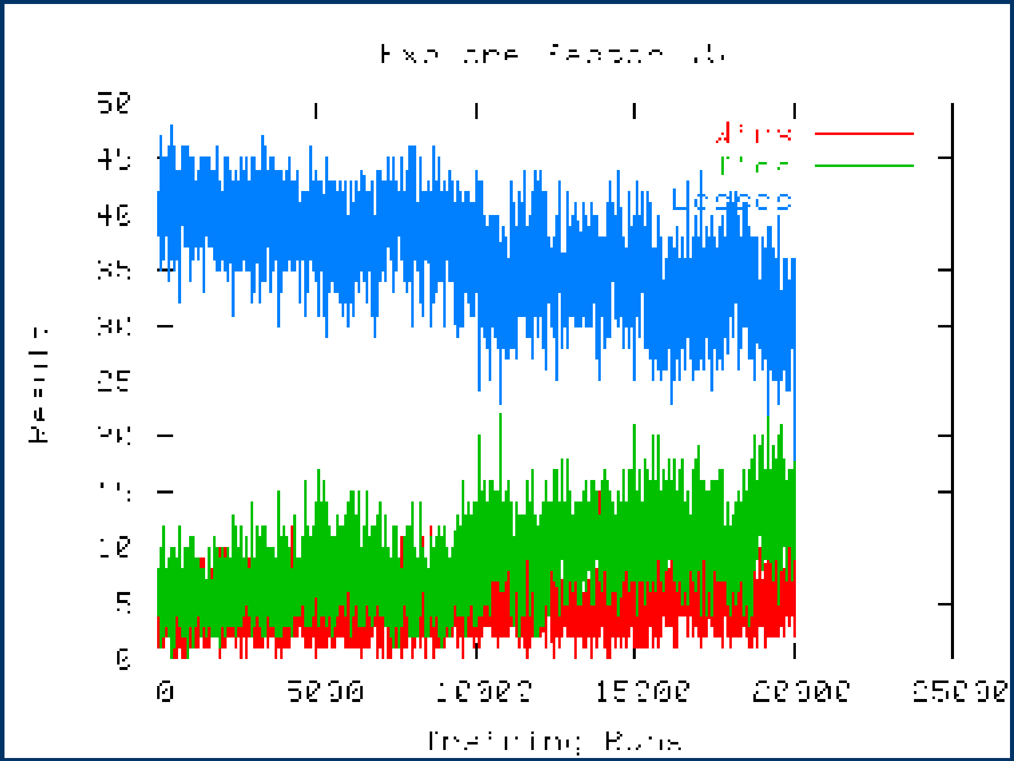





**Learning curve is much slower with exploring.**

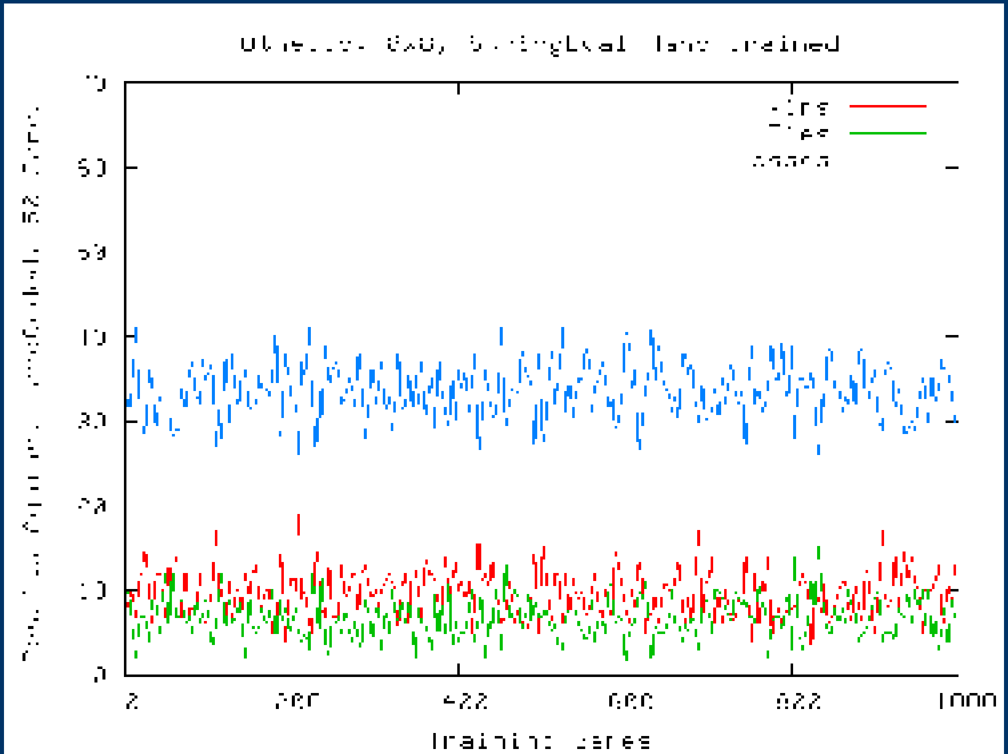
# Pawns 4x4 StringEval Exploring

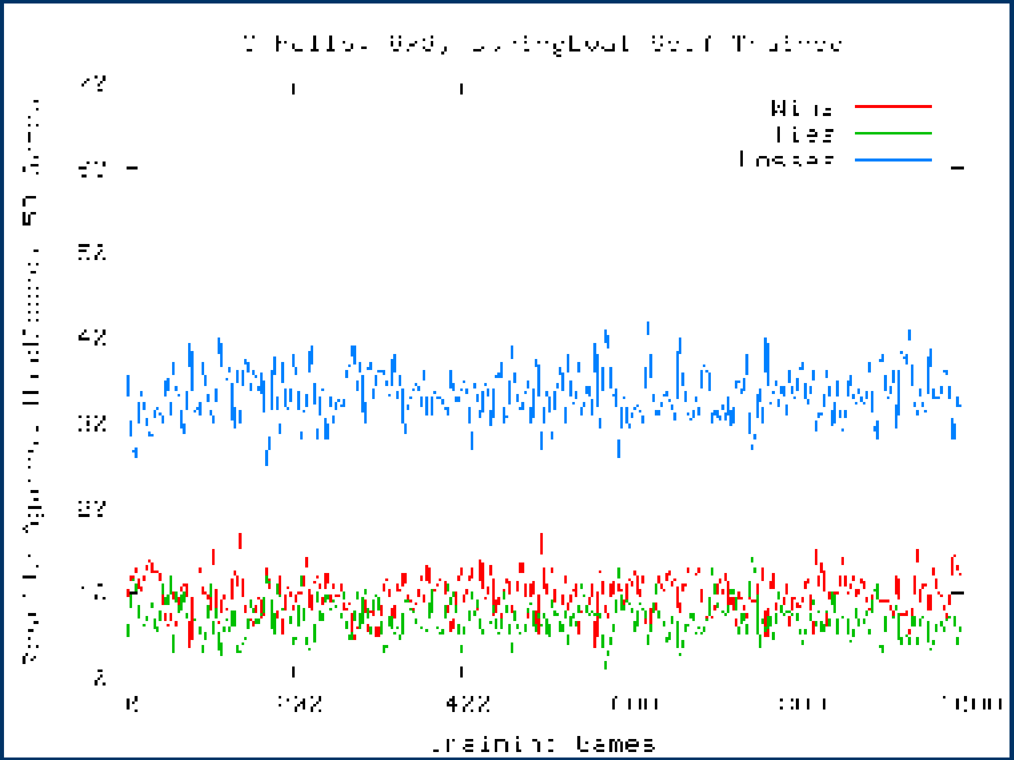



**More compressed learning with smaller explore factor**

# Othello 8x8 StringEval

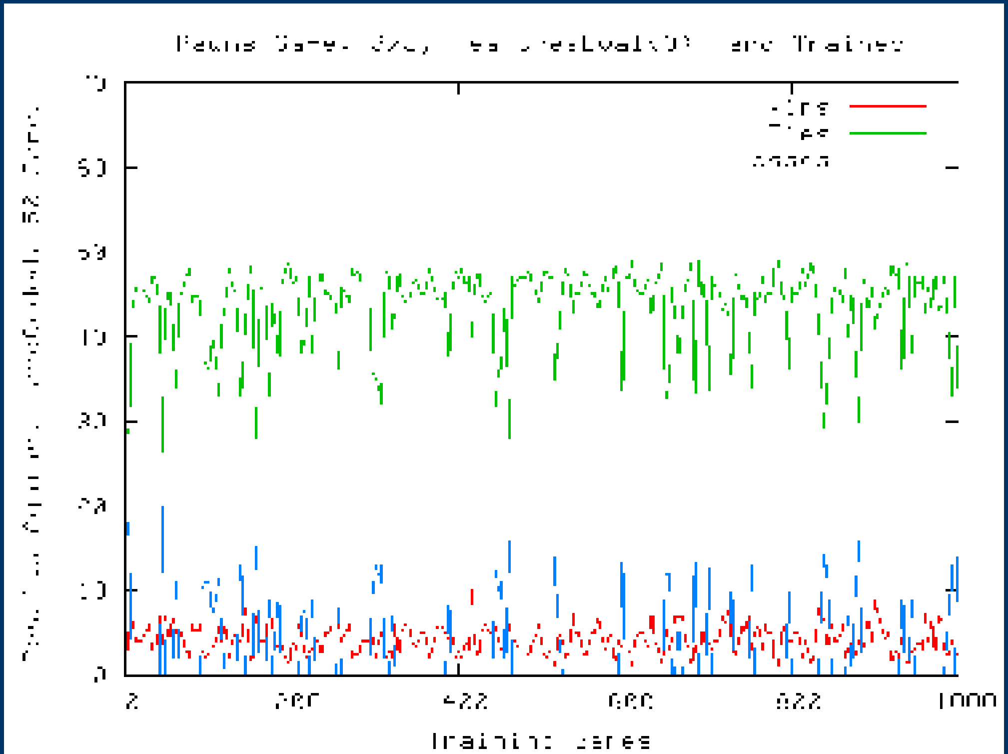


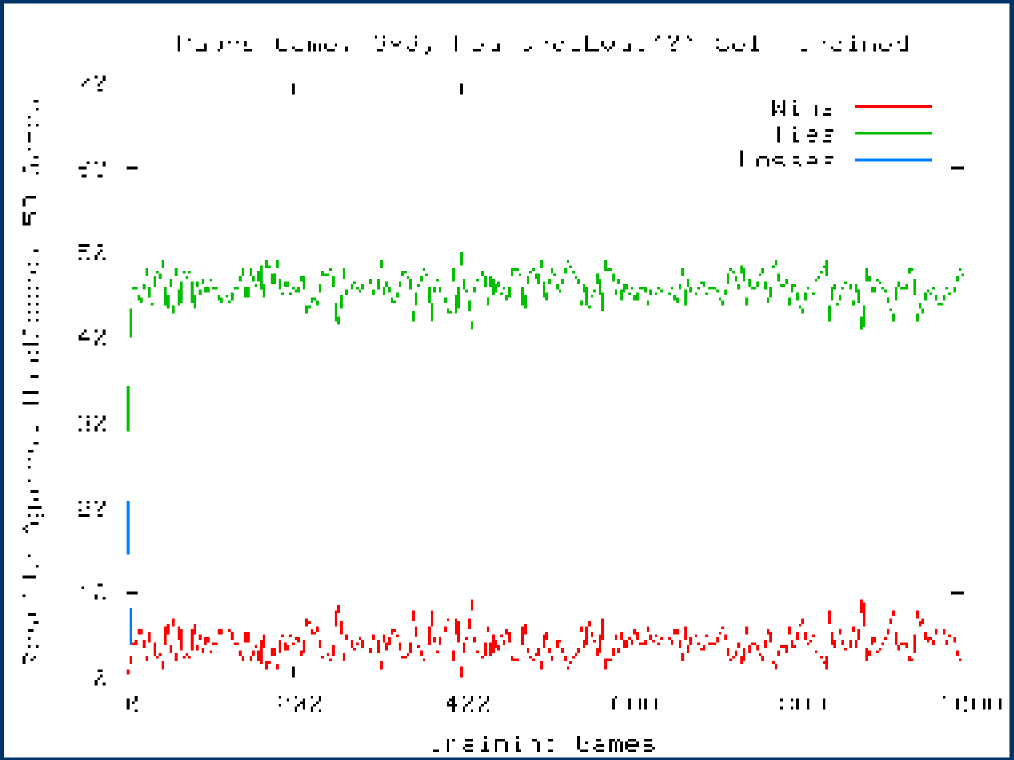



**Another Very Large State Space**

**Some Wins/Ties By Chance (Search depth 4)**

# TicTacToe FeaturesEval(0)



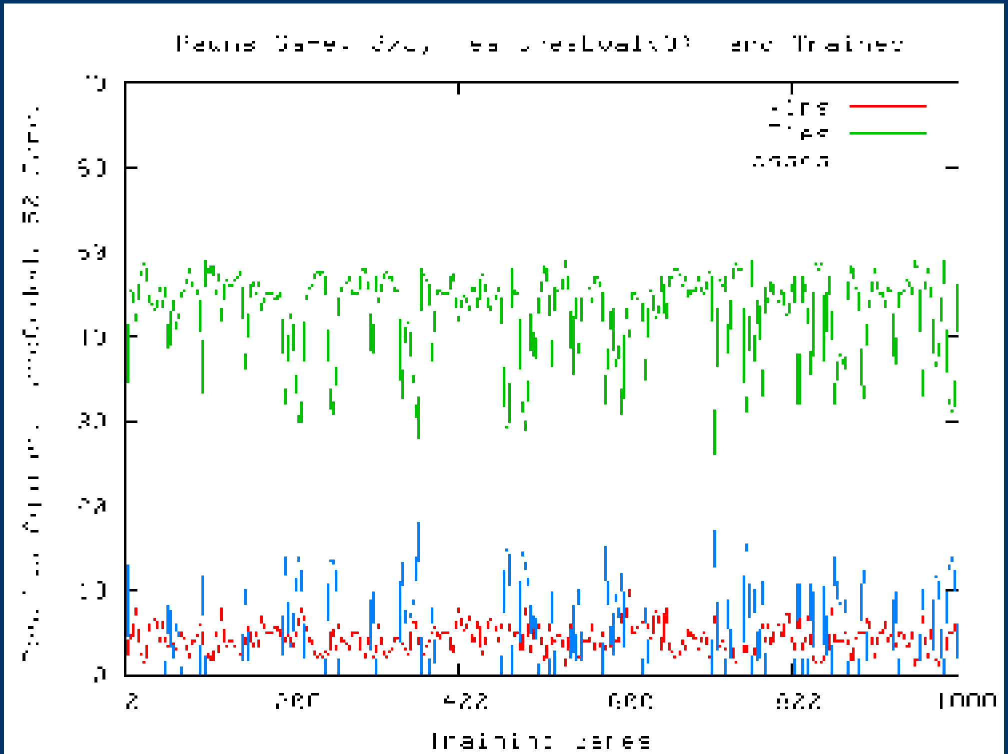


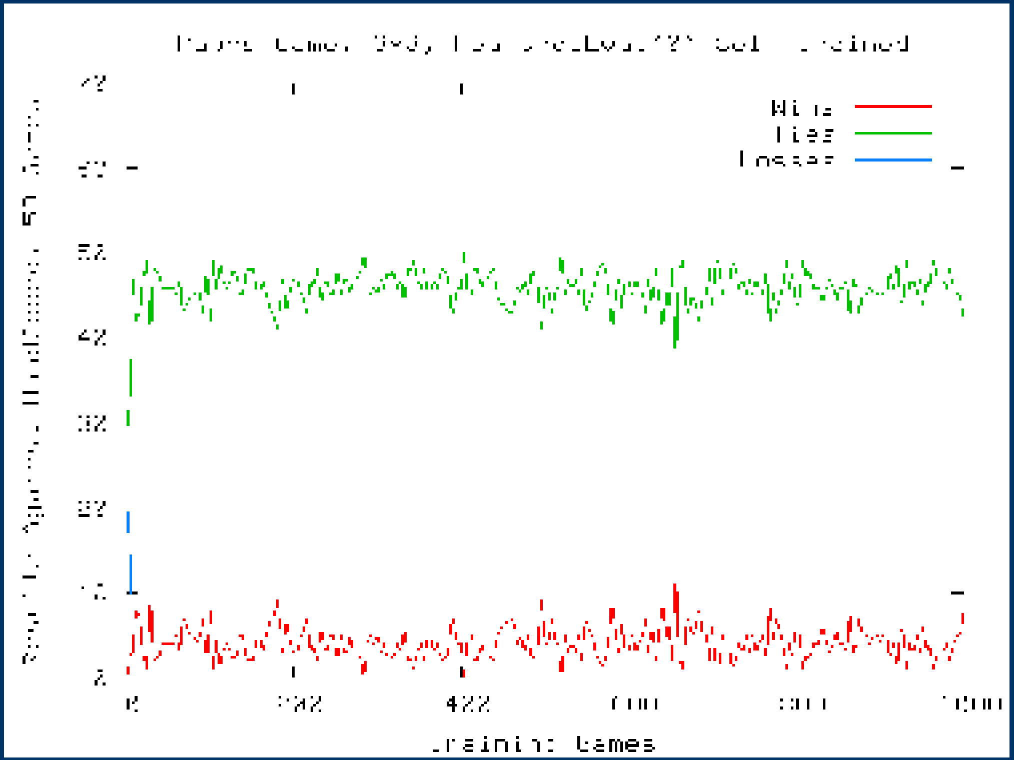

**“Cat's Game” - Ties are Optimal**

**Very Quick Initial Learning – Despite # States**

**Erratic Changes in Hand Trained?**

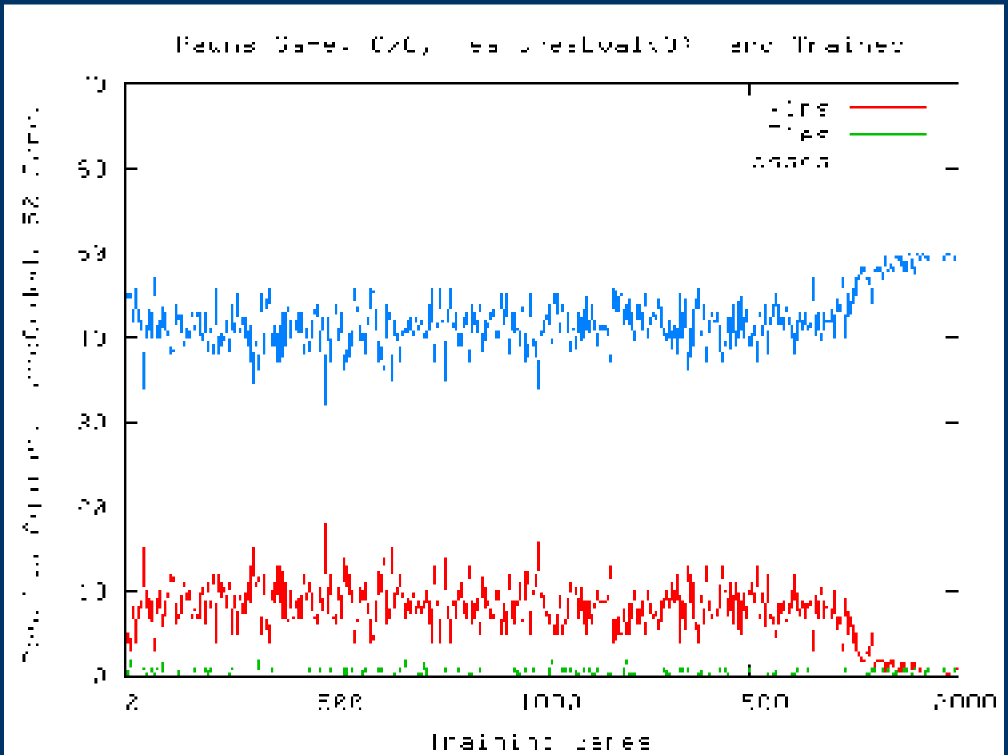
# Pawns 3x3 FeaturesEval(0)

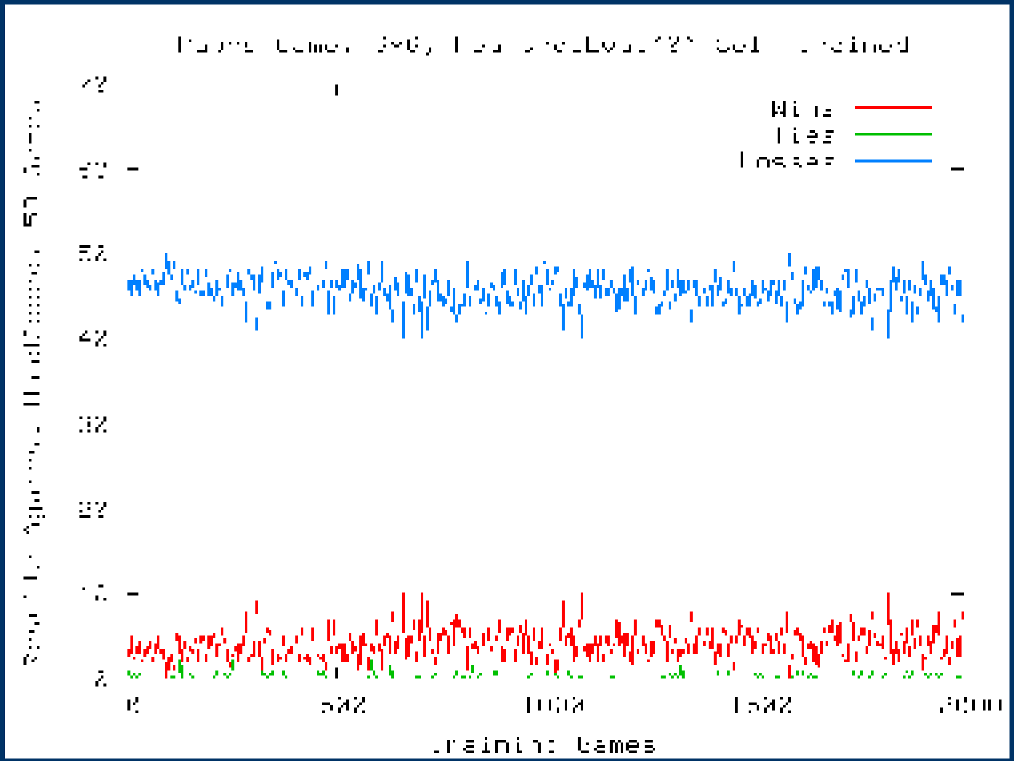



- **Again, “Cat's Game” - Ties are Optimal**
- **Quick Initial Learning**
- **Again, Erratic Changes with Hand Trained**

# Pawns 6x6 FeaturesEval(0)



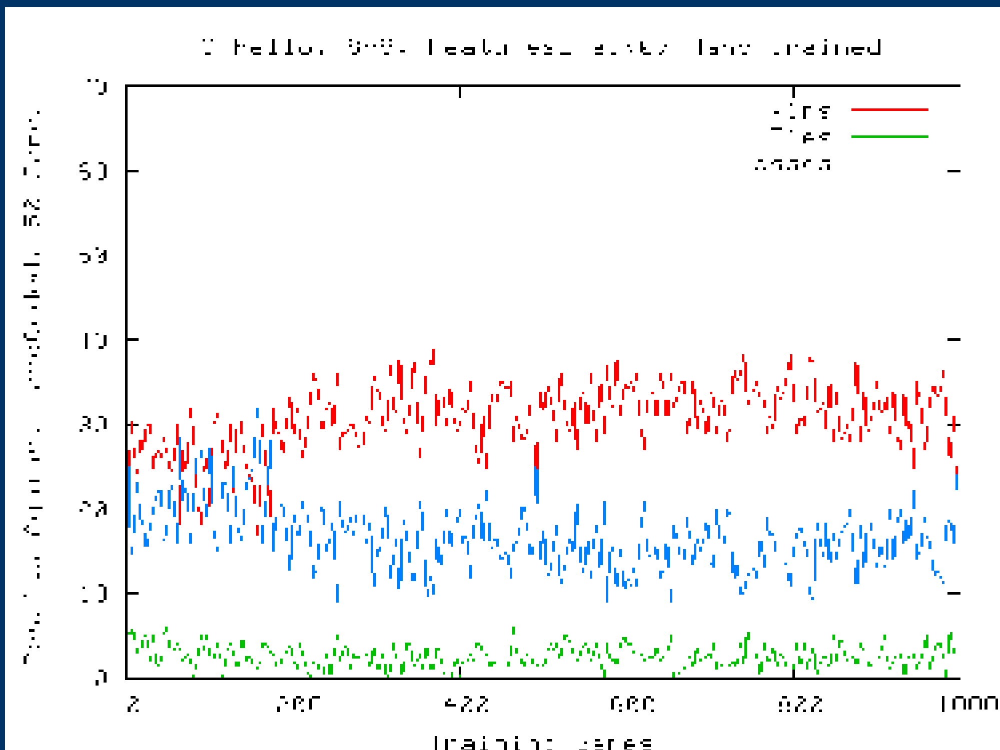



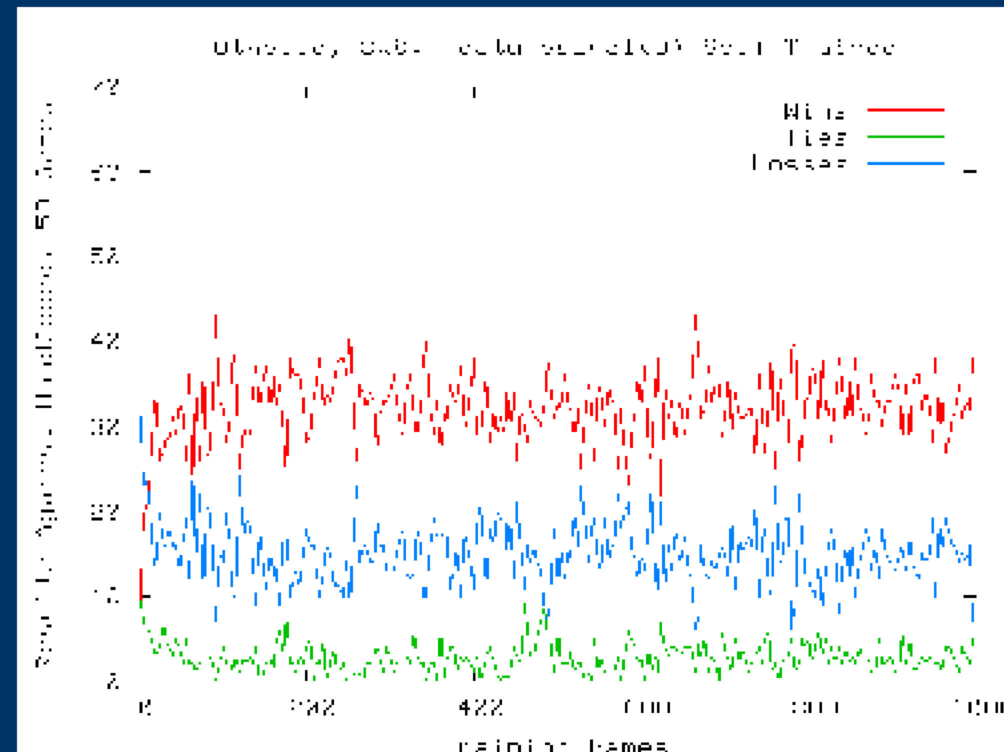

**No Learning For Self Trained**

**Minimal Learning on Hand Trained**

**Hand Trained Dropoff ~ Over Learning?**

# Othello 8x8 FeaturesEval(0)



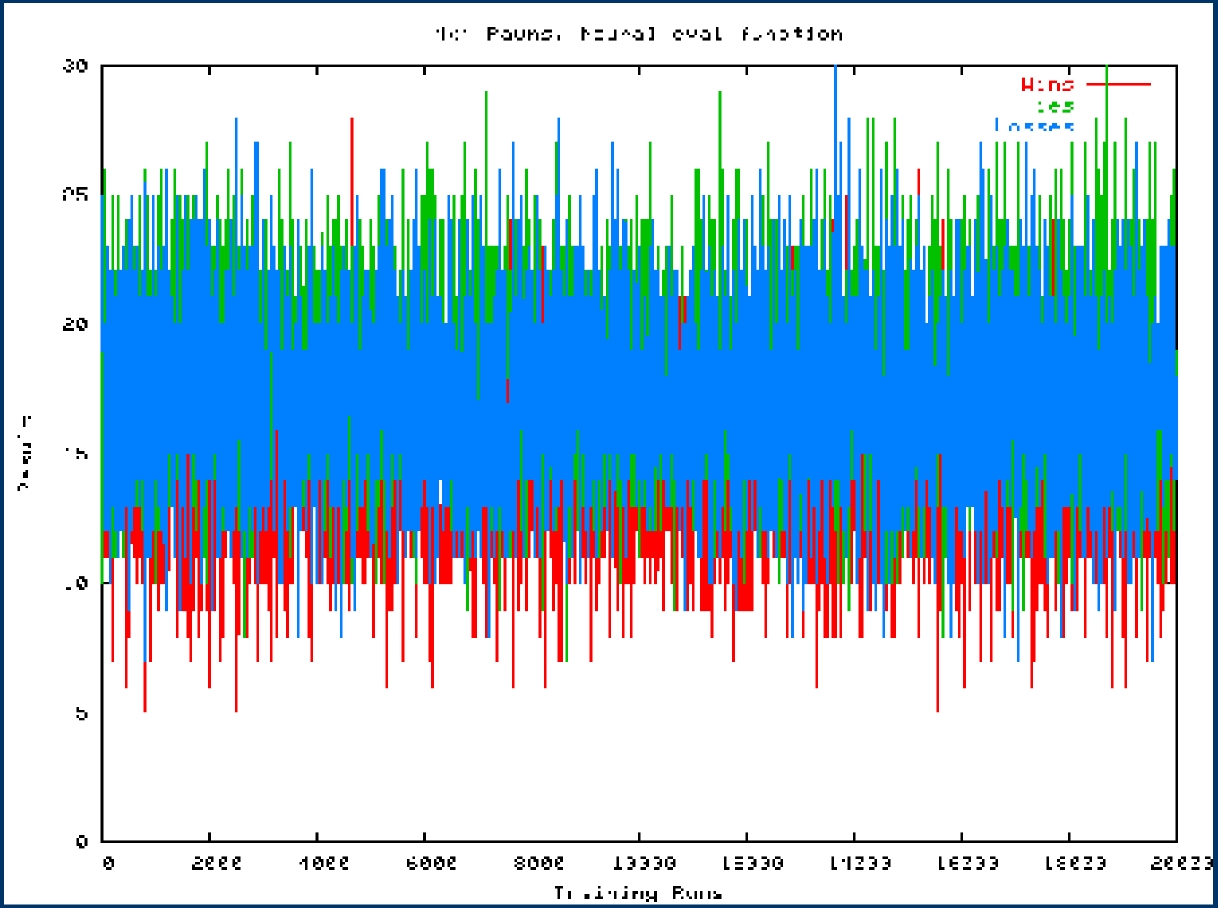




**Decent/Good Performance**

**Very Quick Initial Learning**

**Some Slower Learning Over Next ~200 Games**

# Pawns 4x4 NeuralEval




**Seems pretty random**

**Relatively good performance with no training??**

# Possible Uses

**Because of the speed of learning with the FeaturesEval, it may be possible to use it to learn to play against a human at about their own level. Having an opponent of about equal skill level is usually more fun.**

# Unanswered Questions

**Why didn't our neural net learner work? Given the good results with TDGammon, we expected this to work better.**

**Not enough time to adjust parameters (hidden layers, learning factor, lamda)**

**Unlike backgammon, our games have no stochastic factor (dice roll). Not exploring enough of the state space? Utility function "less" continuous?**

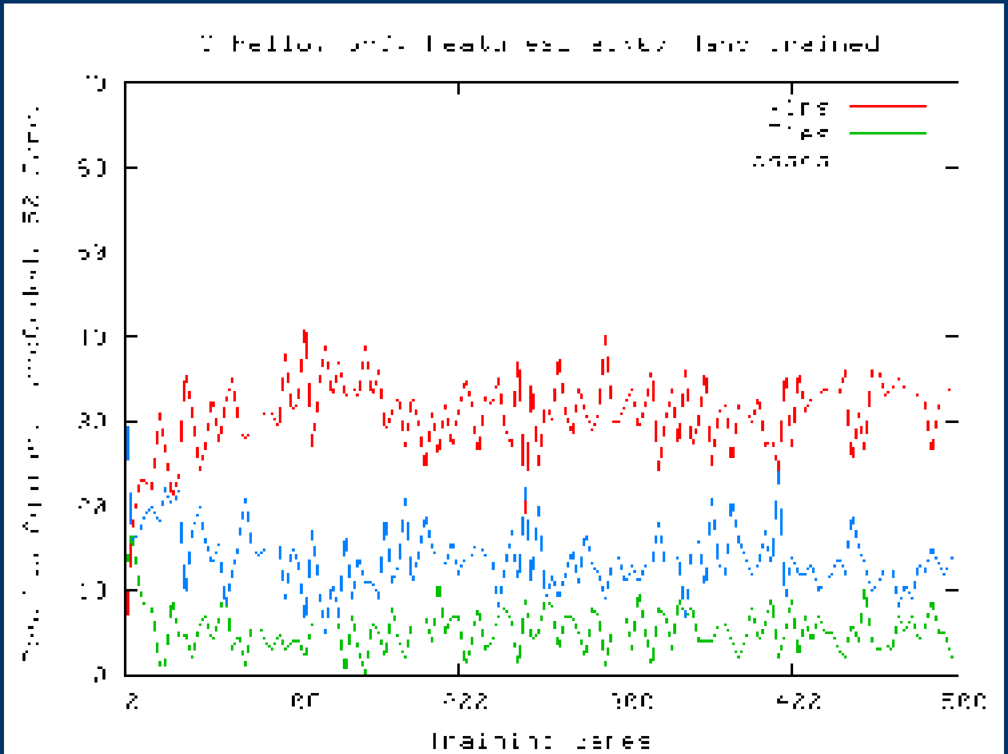
# Unanswered Questions

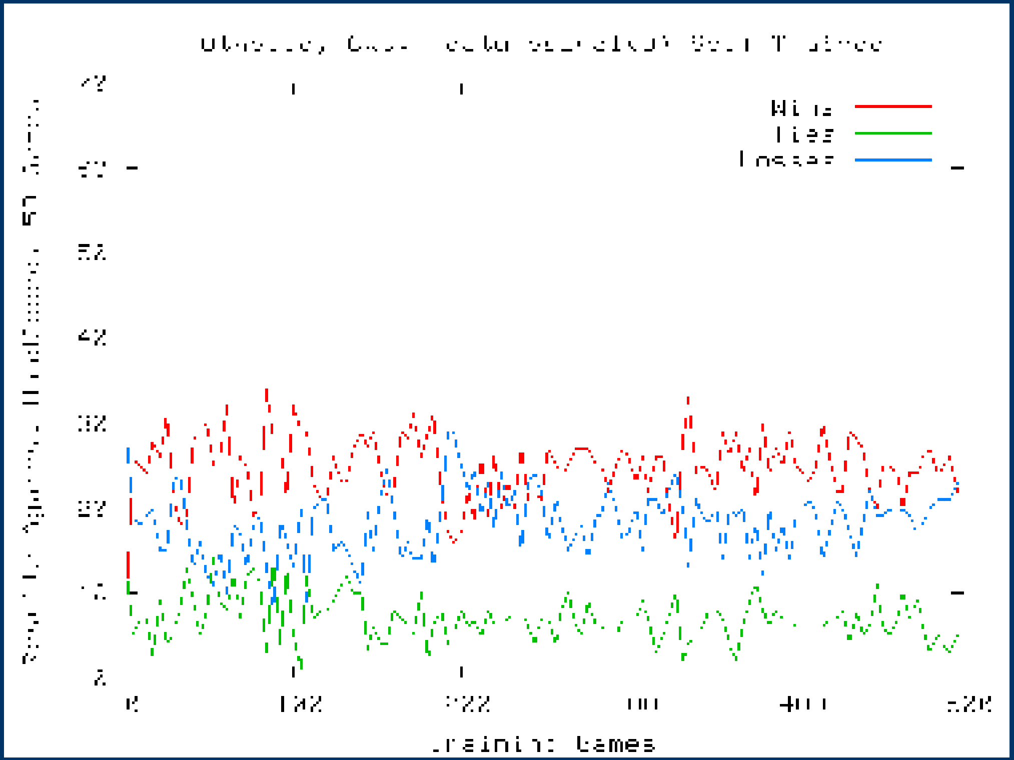
When does overlearning occur? How can we prevent it?

If we wanted to actually play against these players, we would want to stop the training phase before they start overlearning.

Possibly: Keep a periodic snapshot and record the record, choose the best snapshot.

# Demo: Othello 6x6 FeaturesEval(0)






**Demo: Uses Hand Trained FeaturesEval**

**Trained on 500 Games**