GOPS Training Results

Jonathan Stokes

July 15, 2024

1 Introduction

This is the results from a series of experiments which attempted to improve the performance of the GOPS Transformer AI versus a uniformly random agent. To date these attempts have been unsuccessful. However I was able to repeat the preliminary results were the GOPS Transformer AI achieved a 65% rate a uniformly random opponent.

2 Data Filtering Experiments

I tried training the GOPS Transformer model on three slightly different datasets. The original dataset $\mathcal{D}_0$ was generated be letting two uniformly random agents play against each other for $n = 10k$ games. This dataset was then filtered to consist of only game state move pairs that resulted in a win $\mathcal{D}_0^w$. I then filtered $\mathcal{D}_0^w$ again to create three data sets to use in training the GOPS Transformer model. This resulted in three different training regimes: Default, Greedy, and Unique. In all these training regimes the stopping condition for training was an increase in loss on the validation set.

2.1 Bootstrap Training

To formalize the bootstrap training method used in each of these regimes. Lets introduce two agents a learner L_i and a teacher T where i is the training round starting at 0 and T is invariant to training round. Initially both agents select moves uniformly at random, $\mathsf{L}_0 \sim U$ and $\mathsf{T} \sim U$. L_0 and T play n games against each other generating a data set $\mathcal{D}_0$ of game states and moves, which is then filters this dataset to consist only of game state move pairs that resulted in a win $\mathcal{D}_0^w$.

To formalize these steps lets introduce the training function $w(x, y)$, the data generation function $g_n(x, y)$, and the filtering function $f_r(x)$ the which varies with training regime r. The function $g_n(x, y)$ generates a data set by letting two agents play n games against one another,

$$g_n(\mathsf{L}_i, \mathsf{T}) = \mathcal{D}_i \tag{1}$$

$f_r(x)$ is the filtering function,

$$f_r(\mathcal{D}_i) = \mathcal{D}_{i,r}^w \tag{2}$$

and $w(x, y)$ is the training function,

$$w(\mathsf{L}_i, \mathcal{D}_{i,r}^w) = \mathsf{L}_{i+1} \tag{3}$$

Now letting m be the number of bootstrap training iterations the training algorithm can be written as,

Algorithm 1 Bootstrap Training

Given L_0, T, n, m
for $i \in [0, \ldots, m]$ **do**
  $\mathcal{D}_i \leftarrow g_n(\mathsf{L}_i, \mathsf{T})$
  $\mathcal{D}^w_{i,r} \leftarrow w_r(\mathcal{D}_i)$
  $\mathsf{L}_{i+1} \leftarrow w(\mathsf{L}_i, \mathcal{D}^w_{i,r})$
end for

2.2 Training Regime: Default

In the default training regime the filtering function $w_{default}(x)$ returns only those state move pairs that eventually result in a win,

$$w_{default}(\mathcal{D}_i) \rightarrow \mathcal{D}^w_{i,default} \tag{4}$$

The performance results L_i of Equation (4) and Algorithm 1 with $n = 10k$ are given in Table 1. Table 1 shows that while initially L_i appears to make some progress bootstrap training ultimately hurts its performance.

Model	i	T Wins	L_i Wins	Ties	Unique States
U.Rand	0	5011	4971	18	**116,637**
V17	1	4314	**5665**	21	111,459
V18	2	4539	5449	12	111,072
V19	3	4969	5010	21	110,334
V20	4	4921	5061	18	110,046

Table 1: T is a uniformly random agent, L_i is a Transformer model.

2.3 Training Regime: Greedy State Move Pairs

In the greedy state move pair training regime the filtering function $w_{greedy}(x)$ returns only those state move pairs that eventually result in a win and of those only the most frequent of those for each state,

$$w_{greedy}(\mathcal{D}_i) \rightarrow \mathcal{D}^w_{i,greedy} \tag{5}$$

The performance results L_i of Equation (5) and Algorithm 1 with $n = 10k$ are given in Table 2. Table 2 shows that while initially L_i appears to make some progress bootstrap training ultimately hurts its performance.

Model	i	T Wins	L_i Wins	Ties	Unique States
U.Rand	0	5011	4971	18	**116,637**
V13	1	4299	**5692**	9	111,709
V14	2	4749	5235	16	109,714
V15	3	4876	5114	10	108,829
V16	4	4942	5043	15	108,607

Table 2: T is a uniformly random agent, L_i is a Transformer model.

2.4 Training Regime: Unique State Move Pairs

In the unique state move pair training regime the filtering function $w_{unique}(x)$ returns only those state move pairs that eventually result in a win and of those only one move state pair,

$$w_{unique}(\mathcal{D}_i) \rightarrow \mathcal{D}^w_{i,unique} \tag{6}$$

The performance results of using Equation (6) and Algorithm 1 with $n = 10k$ to train the learner L_i are given in Table 3, which shows that while initially L_i makes progress using bootstrap training with a dip at $i = 3$ before achieving its best performance at $i = 4$. After which the performance of L_i steadily decreases.

Model	i	T Wins	L_i Wins	Ties	Unique States
U.Rand	0	5011	4971	18	**116,637**
V21	1	4158	5831	11	111,310
V22	2	4046	5939	15	111,222
V23	3	4289	5694	17	111,338
V24	4	3478	**6505**	17	111,520
V25	5	3765	6219	16	111,340
V26	6	3726	6255	19	111,457
V27	7	3915	6066	19	111,354
V28	8	3999	5980	21	111,559
V29	9	4136	5852	12	111,752

Table 3: T Player 1 is a uniformly random agent, L_i is a Transformer model.

2.5 Comparing Results of Default, Greedy, and Unique Training Regimes

Figure 1 compares L_i performance under all three training regimes by plotting the performance results in Tables 1 to 3. Its clear from Figure 1 that the Unique training regime outperforms the other two training regimes. I do not know why this is the case.

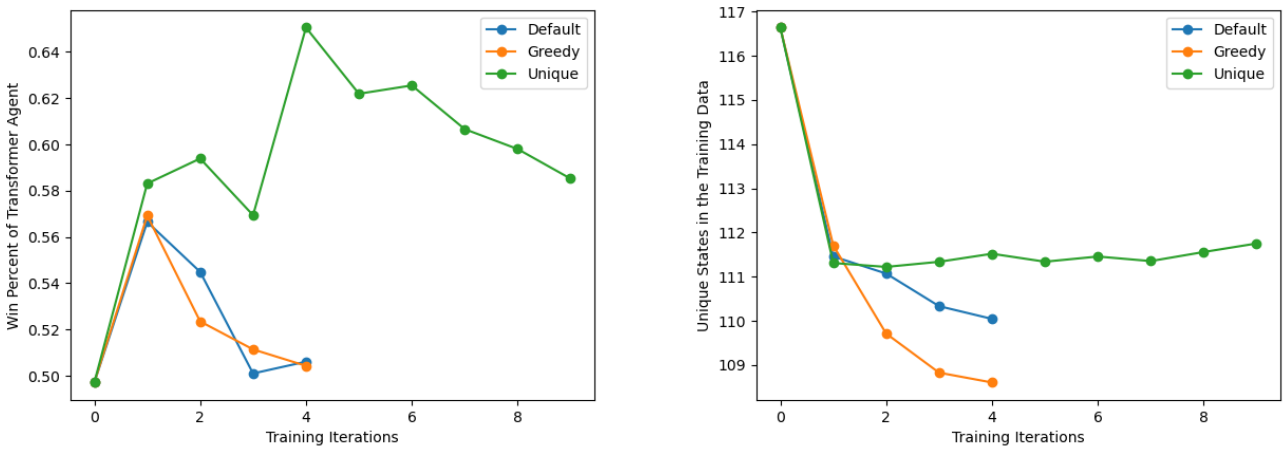

Figure 1: (Left) Transformer model win percentage, (Right) Number of unique states.

It is also not clear to me why performance of L_i collapses under all training regimes with successive bootstrap training or why that collapse takes longer in the Unique training regime than in the Default or Greedy regimes.

However on seeing the results in Figure 1 it is clear that there are more unique games states under the Unique training regime than under the Default or Greedy training regimes. This led me to hypothesize that as the number of unique states in the training data decreased with successive Bootstrap training iterations so did the performance of the learner L_i. The next investigates this hypothesis among others.

3 Attempts to Improve Transformer Model Performance

This section introduces and tests a number of hypothesis on improving the performance of the GOPS Transformer. I would treat the results in this section as preliminary, the results were generated quickly to

test the hypotheses, and suggest that hypothesis are all false.

3.1 Hypothesis: Decreasing Unique States Causes Decreased Performance

To test the hypothesis that the decreasing number of states in the training data results in the decreasing performance of L_i I modified the Transformer model. The new Transformer model instead of playing the card with the highest probability given the game state the new model selects a random card among the top k cards with the highest probability. Lets denote this new Transformer model as $\mathsf{L}_{i,k}$.

The performance results for $\mathsf{L}_{i,1}$ and $\mathsf{L}_{i,2}$ over several Unique regime training iterations where the training data was generated using $g_n(\mathsf{L}_{i,2}, \mathsf{T})$ are given in Table 4. In this table $p_n(\mathsf{L}_{i,k}, \mathsf{T})$ denotes the function generating the performance results and is the same identical to the data generation function except for the learner argument which may be $\mathsf{L}_{i,1}$ or $\mathsf{L}_{i,2}$.

Model	i	k for $g_n(\mathsf{L}_{i,k}, \mathsf{T})$	k for $p_n(\mathsf{L}_{i,k}, \mathsf{T})$	T Wins	$\mathsf{L}_{i,k}$ Wins	Ties	Unique States
U.Rand	0	n.a.	n.a.	5011	4971	18	**116,637**
V30	1	2	1	4136	**5848**	16	111,182
V30	1	2	2	4468	5521	11	113,776
V31	2	2	1	4172	5809	19	111,518
V31	2	2	2	4500	5488	12	113,830
V32	3	2	1	4306	5677	17	111,064
V32	3	2	2	4615	5370	15	113,852
V33	4	2	1	4680	5301	19	111,153
V33	4	2	2	4948	5039	13	113,764

Table 4: Wins for experiment 5 where T is a uniformly random agent and L_i is a Transformer model.

The results in Table 4 show that the number of unique states is higher as a result generating the training data with $k = 2$. See Figure 2 for a comparison of the number of unique game states generated by $k = 1$ and $k = 2$ in $g_n(\mathsf{L}_{i,k}, \mathsf{T})$.

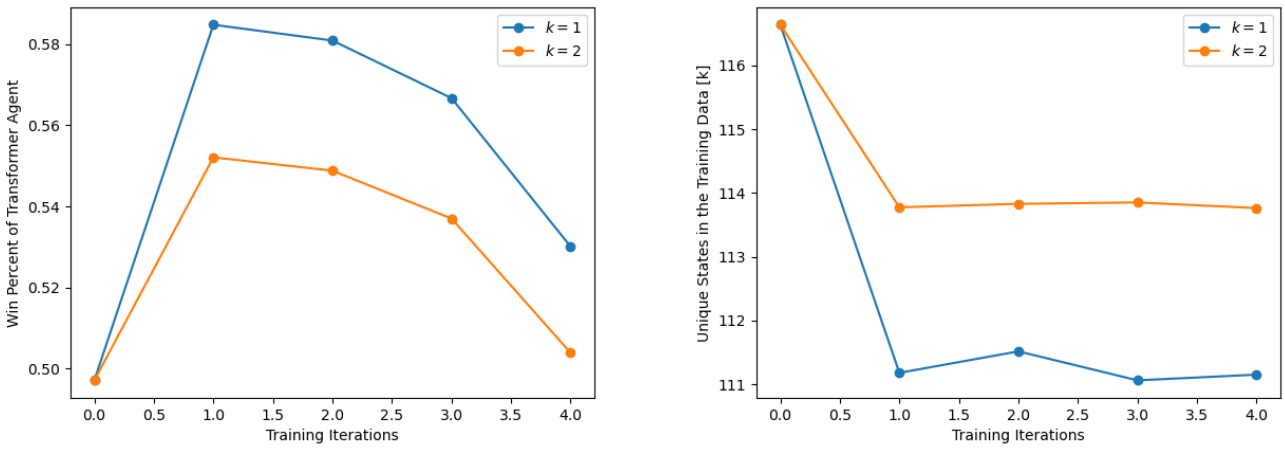

Figure 2: In Unique Regime (Left) Performance of Transformer model, (Right) Unique states

However training on the data with more unique games states does not improve the learners performance, see Figure 2. In fact the Transformer model trained on data with more unique states only reaches a win rate of 58% significantly below the win rate of 65% previously achieved. This suggests that the hypothesis that learner performance is correlated with the number of unique states in the training data appears false. Also as seen in Default and Greedy training regimes the performance of $\mathsf{L}_{i,k}$ appears to decrease with each Bootstrap training iteration after the initial one.

3.2 Hypothesis: All We Need Is More Data

Another hypothesis I had was that more training data would improve performance. To test this I tried using all the data generated in the Unique training regime to train Transformer model V34. The performance of the resulting model given in Table 5 is not impressive.

Trials: n	k	$\mathsf{L}_{i,k}$
100	1	**64%**
100	2	62%
1000	1	56%

Table 5: Wins for experiment 6 where T is a uniformly random agent and $\mathsf{L}_{i,k}$ is a Transformer model.

There is a lot of variance in the results in Table 5 between $n = 100$ and $n = 1000$ but no matter the value of n it appears that adding more data alone does not result in better performance. One explanation for this might be that there is a lot of overlap between the game states in successive training datasets so that combining datasets does not add many state move pairs. However looking at Figure 3 we can see that there is less than a 16% overlap between states in any two successive training datasets.

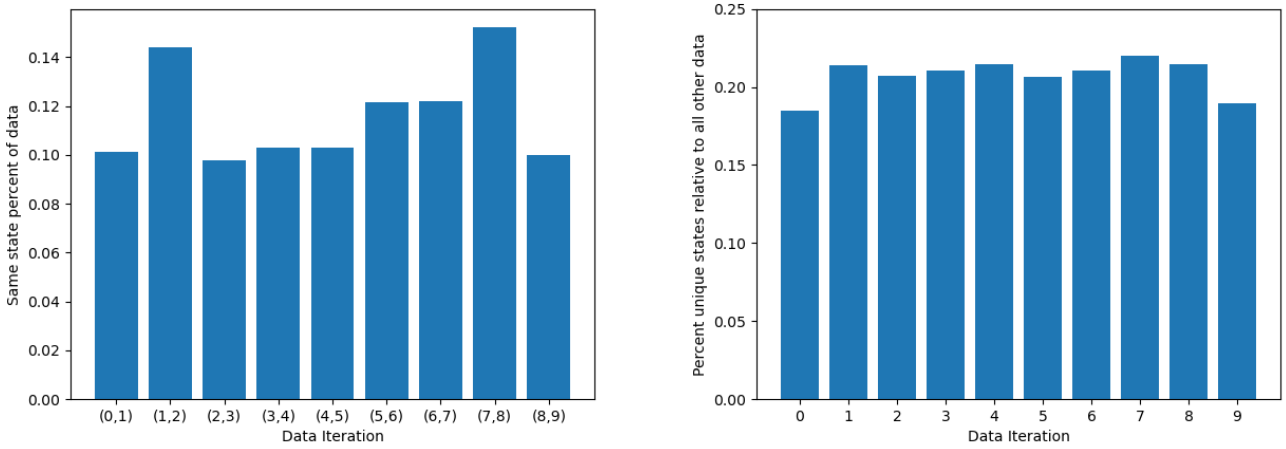

Figure 3: In Unique Regime (Left) states overlapping between sequential training data iterations, (Right) states overlapping any other training data iterations

In fact, as also shown in Figure 3 there is less than a 25% overlap in games states between any dataset and all other datasets generated in the Unique training regime. This leads to the conclusion that adding more unique state move pairs to the training data for the GOPS Transformer model alone does not increase performance.

3.3 Hypothesis: Transformer Model Is Memorizing The Training Data.

Figure 3 showed that the overlap between game states in successive iterations of training data is less than 25%. Therefore if a look up table was used to select moves whenever possible we should not expect the look up table agent to out perform a random agent any more than $50\% + 25\% = 75\%$ of the time. Of course this is likely an upper bound on performance as making a good move 75% of the time does not necessarily result in a 75% win rate, however if it did this lookup model would outperform the Transformer models win rate of 65%.

So I tested a Lookup Table model M trained on all the data from the Unique training regime.

Training Data	Games	T	M	Ties	Unique States
V34	1000	476	523	1	11,837
V34	10000	4946	5042	12	112,347

Table 6: Wins for experiment 7 where M is a uniformly random agent and T is a Transformer model.

The results for the Lookup Table model M are given in Table 6. M performs only slightly better than a random agent suggesting that the Transformer model is doing more than simply memorizing the training data as Transformer model achieves a win rate of 65%.

4 Conclusion

While I was able to recreate the preliminary performance results of the GOPS Transformer model I was not able to improve on them. Additionally I tested several hypothesis on how to improving the models performance and this exploration, especially the fact that more data did not improve model performance, led me to ask if one can learn from randomness; a question I look at in another set of notes. In future work I hope to see if constraining the distribution of the learner and teacher will lead to better learner performance as suggested in my nodes on learning from randomness.