Bagging-based Ensemble Methods for Generative Adversarial Imitation Learning

Brian Ho
University of Toronto
manchon.ho@mail.utoronto.ca

Sophia Wan
University of Toronto
sophia.wan@mail.utoronto.ca

Anthony DiMaggio
University of Toronto
anthony.dimaggio@mail.utoronto.ca

Jasper Chen
University of Toronto
jasperjs.chen@mail.utoronto.ca

Abstract

Inverse reinforcement learning (IRL) and imitation learning (IL) have attracted considerable research due to the prevalence of real-world applications and impact of reinforcement learning (RL) agents. IRL learns a reward function based on an expert's observable actions and imitation learning attempts to directly mimic the actions of expert agents. However, these models often generalize poorly, since any finite set of trajectories can be explained by multiple reward functions; and a similar problem applies to policies where the training set of trajectories does not exhaust the state space. Ensemble learning, particularly bootstrap aggregation (bagging)-based ensemble learning have historically been shown to enhance model robustness and performance. This study proposes a bagging-based ensemble method with weak learners of varying architectures for Generative Adversarial Imitation Learning (GAIL) agents, and examines whether it can push agents closer to expert performance. Our results demonstrate that ensemble learning, utilizing GAIL weak learners, substantially improve overall model performance. We further discuss potential extensions to our approach and additional areas of research.

1 Introduction

In many real-world applications of reinforcement learning (RL), such as self-driving cars, explicitly defining the reward function is prohibitively difficult–a challenge which motivates research in inverse reinforcement learning (IRL) and imitation learning (IM). IRL learns a reward function based on an expert's observable actions and IM attempts to directly mimic the actions of expert agents [1].

One inherent problem with IRL is that any finite set of expert trajectories (paths through an environment) is explainable by multiple reward functions [1]. A similar problem applies to imitation learning policies as well. However, a common improvement to models, often utilized for its ease of implementation, exists in the form of ensemble learning. In particular, Breiman have demonstrated the practical benefits of ensembling in the perspective of bagging [2].

This study employs bagging-based ensemble methods with Generative Adversarial Imitation Learning (GAIL) weak learners to examine whether bagging-based ensemble methods push GAIL agents closer to expert performance. We will use one main metric: policy performance within the RL environment.

Finally, we present an experimental analysis of the prospects of bagging-based ensembles using GAIL in addition to a basic ensemble algorithm for GAIL. We show that ensemble learning improves agent performance over a single model.

2 Background

Our project seeks to build upon and combine aspects of two existing works. First, Ho and Ermon provide an IM framework called Generative Adversarial Imitation Learning (GAIL) [5]. GAIL trains on expert demonstrations for some task and produces an agent which can match the expert's abilities. This is accomplished through generative adversarial training with two neural networks. One of the networks is a policy optimization network trained to mimic the expert trajectories. The other network is a discriminator network trained to tell whether a trajectory was generated by the expert or the policy optimization network. In particular, GAIL aims to find a saddle point for

$$\mathbb{E}_{\pi}[\log(D(s,a))] + \mathbb{E}_{\pi_E}[\log(1-D(s,a))] - \lambda H(\pi)$$

where π represents the trained policy, π_E represents the expert policy, D represents the discriminator network, and H denotes causal entropy. GAIL does so by alternating between an Adam gradient step for the discriminator network and a TRPO step for the policy.

Second, we draw upon Liu et al's research on bagging methods for deep IRL [6]. They examine learning a taxi-routing policy using a simpler maximum-entropy formulation, augmented with ensemble methods. They follow a typical bagging algorithm: sampling subsets from the expert dataset, training several weak learners on each subset, and then averaging the predictions of the weak learners. Lui et al. report a 7% increase in training accuracy over standard MaxEnt deep IRL.

3 Architecture

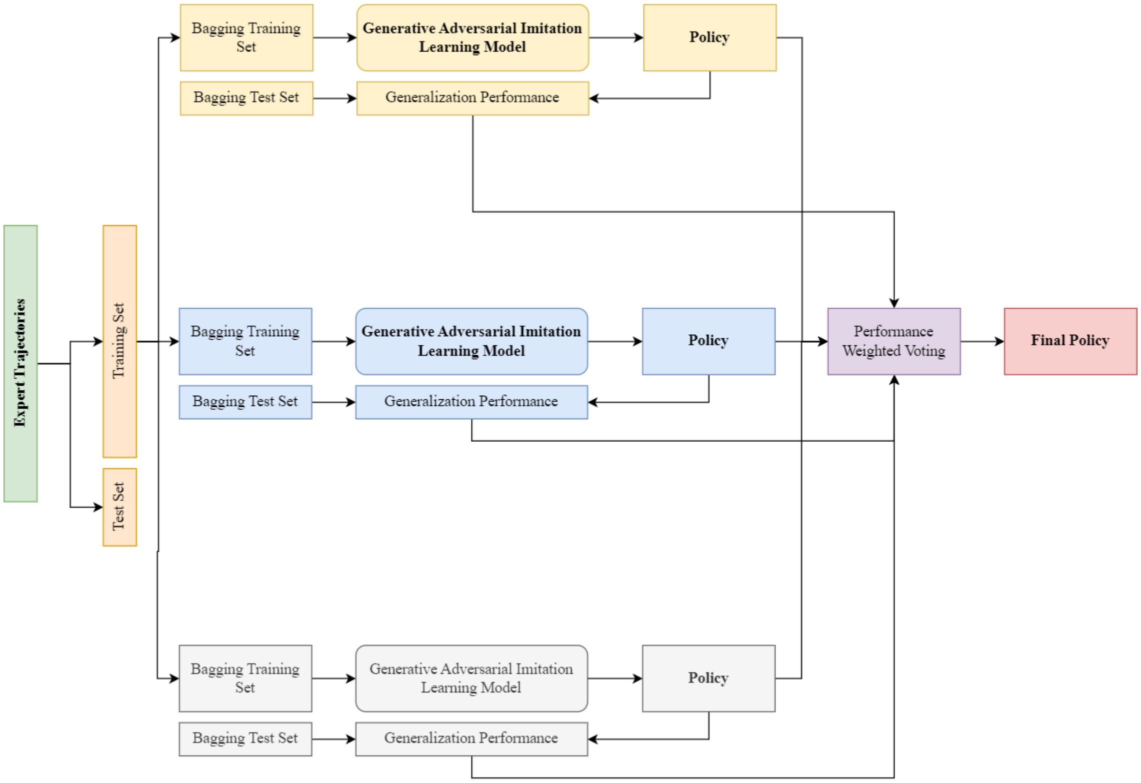
Bagging Training Set
Generative Adversarial Imitation Learning Model
Policy
Bagging Test Set
Generalization Performance
Expert Trajectories
Training Set
Test Set
Bagging Training Set
Generative Adversarial Imitation Learning Model
Policy
Performance Weighted Voting
Final Policy
Bagging Test Set
Generalization Performance
Bagging Training Set
Generative Adversarial Imitation Learning Model
Policy
Bagging Test Set
Generalization Performance

Figure 1: Architecture of Bagging Ensemble Generative Adversarial Imitation Learning Algorithm

Our ensemble adopts the bootstrap aggregation (bagging) approach [2] to ensemble algorithms which can be seen in Figure 1–first with architectural homogeneity, then with varied weak learner architectures. Each weak learner is an independently-trained GAIL model following Ho and Ermon's implementation [5].

We begin with our set of expert trajectories, which is split into a training set, a validation set, and a test set. Our bagging variant with n bags is then performed in three steps. First, we take our training set partition it into n new training sets of equal size–subject to small size differences due to divisibility of the original training set size by n. This is done in a data processing pipeline described further in the next section.

In the second step, we train each weak learner with a different subset of the original training set. In some experiments, we vary the hidden sizes of each weak learner. After training, each weak learner produces a learned policy. In the final step, the three weak learners are aggregated into an ensemble by combining the resulting policies in a majority voting system (for discrete action spaces) or averaging (for continuous action spaces). The final policy is the policy of the ensemble after the voting algorithm.

Lastly, we evaluate the performance of the algorithm against a single Generative Adversarial Imitation Learning (GAIL) model from OpenAI's baseline library trained on the full environment dataset across multiple environments. Performance is then compared using the final test reward value. We utilize an evaluation pipeline to conduct experiments on the proposed architecture in parallel across different sweeps and environments.

4 Data Processing

We use MuJoCo (Multi-Joint dynamics with Contact)–which is a physics engine for facilitating research and development in robotics, bio-mechanics, graphics, and animation–implemented in OpenAI gym [3] as our target environment. The training data is trajectories from expert demonstrations of the desired task, which are represented by a state-action pairs retrieved from the environment and the expert agent.

For our expert demonstration data, we use the datasets provided in the gail module from OpenAI baselines repo [4]. The dataset files can be downloaded from Google Drive. Since the datasets are in .npz format which are serialized numpy array files, it can easily loaded using numpy.

As an example, we'll discuss "stochastic.trpo.Walker2d.0.00.npz". Figure 2 is a sample code snippet for loading the file. By inspecting the shapes of the arrays, this file contains 1500 expert demonstration trajectories, where $obs[i]$, $actions[i]$ and $rewards[i]$ represents the i-th trajectory in the dataset.

```
    obs = np.load(‘stochastic.trpo.Walker2d.0.00.npz’, allow_pickle=True)[‘obs’]
actions = np.load(‘stochastic.trpo.Walker2d.0.00.npz’, allow_pickle=True)[‘acs’]
rewards = np.load(‘stochastic.trpo.Walker2d.0.00.npz’, allow_pickle=True)[‘rews’]
```

Figure 2: Sample code for loading "stochastic.trpo.Walker2d.0.00.npz" dataset file

The array shapes should follow the documentation of the task in OpenAI Gym MuJoCo environment. For instance, assume that the i-th trajectory has n states, then

- $obs[i]$ has the shape $(n, 17)$ since the dimension of the observation space of Walker2d is 17
- $actions[i]$ has the shape $(n, 6)$ since the dimension of the action space of walker2d is 6
- $rewards[i]$ has the shape $(n,)$

All data are used for training. The trained policy is evaluated in the OpenAI gym and the reward is reported as the performance metric. We have computed the total number of state-action pair of each tasks in the dataset in Table 1. This illustrates the size of the dataset.

MuJuCo Task	Number of state-action pair
Walker2d	1,949,076
Half Cheetah	3,000,000
Hopper	2,488,291
Humanoid	311,194

Table 1: Number of state-action pair sample for each MuJoCo Task

5 Experiments and Results

We run three rounds of experiments. 1) To establish a baseline over the four MuJoCo environments we evaluate on. 2) To find the optimal bag count for our implementation. 3) To investigate the effect of network depth on our implementation.

5.1 Implementation Details

GitHub repository

Our implementation began as a fork of Hyungcheol Noh's "gail-pytorch" repository, which is available under a permissive license on GitHub. We made key modifications to the original codebase. First, we upgraded the dependencies to support the "gymnasium" package, which includes the newest versions (v4) of the MuJoCo environments, and then added the proper data processing utilities for those environments. Second, we extended the code to support hyperparameter settings for the underlying policy, value, and discriminator networks.

Finally (and most importantly), we implemented a bagging procedure which splits the expert data and trains a distinct model for each subset. Once all models are finished training, our procedure evaluates their policies by combining the output actions through voting (for discrete action spaces) or averaging (for continuous action spaces). Throughout the rest of this section, test reward will mean the reward generated by this ensembled policy in the actual environment, unless otherwise specified.

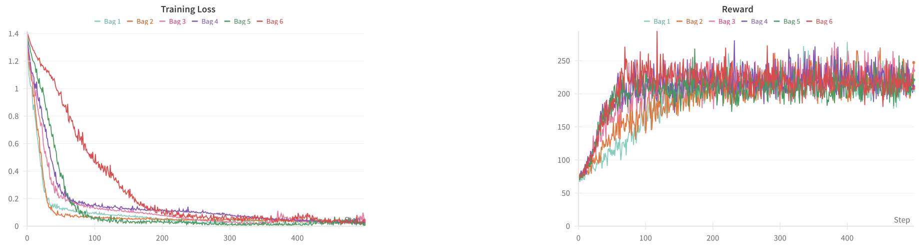

Figure 3: Training loss and reward with the Humanoid-v4 environment

Our loss for training is different for each network in the GAIL algorithm, as each network has a different associated discriminator. The discriminator uses binary cross entropy to push the network weights to correctly identify expert vs. non-expert actions. The policy network is trained with log probability loss over actions. During training, we log the training loss for the policy network and the reward at each iteration. All runs include 500 iterations, with 2000 environment steps per iterations. Figure 3 shows a visualization of the training process for a 6-bag run on the Humanoid-v4 environment.

5.2 Baselines

In order to draw a direct comparison between standard GAIL and bagged GAIL, we began by running baseline tests for each of our four studied environments. GAIL was trained with standard settings, which is to say a hidden size of 50 and three hidden layers, on the full dataset without bagging. We repeated the runs three times for each environment, so as to mitigate variance which might be present from the training process. Figure 4 demonstrates the results, with error bars indicating the range of variance among runs. All reward figures here are normalized to the expert reward.

5.3 Bag-count

We sweep over the bag count for the ensemble. We test 3-bag, 6-bag, and 9-bag configurations. We first perform the bagging-variant on learners with consistent architectures–finding poor overall performance, as seen in tables 2 and 3. Then to introduce architectural variation to the ensemble, we set the i^{th} model's hidden size to be $25i$. Tables 4 and 5 report the results. The 6-bag configuration is the best-performing overall and will be used from here on.

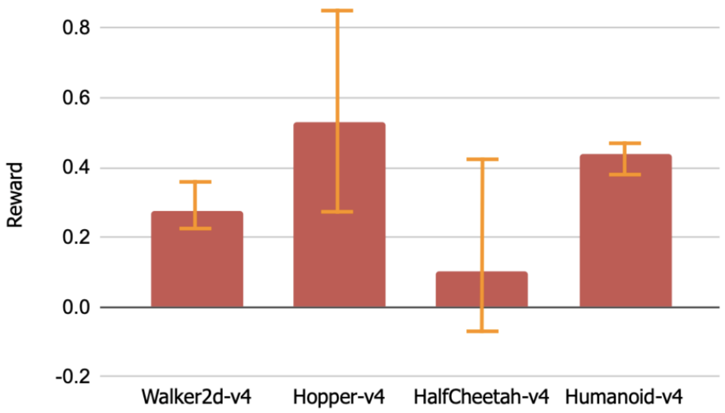

Figure 4: Unbagged GAIL baseline

Environment	3-bags	6-bags	9-bags	Baseline	Expert
Walker2d-v4	305.7	317.25	299.43	461.43	1666.84
Hopper-v4	1588.44	932.68	1958.23	1309.2	2478.7
HalfCheetah-v4	-1553.99	-923.42	-1784.52	87.39	871.04
Humanoid-v4	295.79	264.52	278.82	216.64	492.52

Table 2: Bag-count experiment rewards with constant hidden size and number of layers

Environment	3-bags	6-bags	9-bags	Baseline
Walker2d-v4	18.3%	19.0%	18.0%	27.7%
Hopper-v4	64.1%	37.6%	79.0%	52.8%
HalfCheetah-v4	-178.4%	-106.0%	-204.9%	10.0%
Humanoid-v4	60.1%	53.7%	56.6%	44.0%

Table 3: Bag-count and portion of expert rewards recovered with constant hidden size and number of layers

Environment	3-bags	6-bags	9-bags	Baseline	Expert
Walker2d-v4	276.41	**288.66**	272.52	461.43	1666.84
Hopper-v4	2003.94	**2190.73**	2105.12	1309.2	2478.7
HalfCheetah-v4	53.2	**103.4**	-228.11	87.39	871.04
Humanoid-v4	**303.22**	279.57	294.96	216.64	492.52

Table 4: Bag-count experiment rewards with varied hidden sizes

Environment	3-bags	6-bags	9-bags	Baseline
Walker2d-v4	16.6%	**17.3%**	16.3%	27.7%
Hopper-v4	80.8%	**88.4%**	84.9%	52.8%
HalfCheetah-v4	6.1%	**11.9%**	-26.2%	10.0%
Humanoid-v4	**61.6%**	56.8%	59.9%	44.0%

Table 5: Bag-count and portion of expert rewards recovered with varied hidden sizes

5.4 Effect of Network Depth

Lastly, we investigate the effect of network depth on the test performance of our technique in case bagging changes the optimal number of layers. We test configurations with 1 hidden layer, 3 hidden layers (as before), and 5 hidden layers. All layers have $tanh$ activation. Hidden sizes and other configurations are all as before. Figure 5 reports the results.

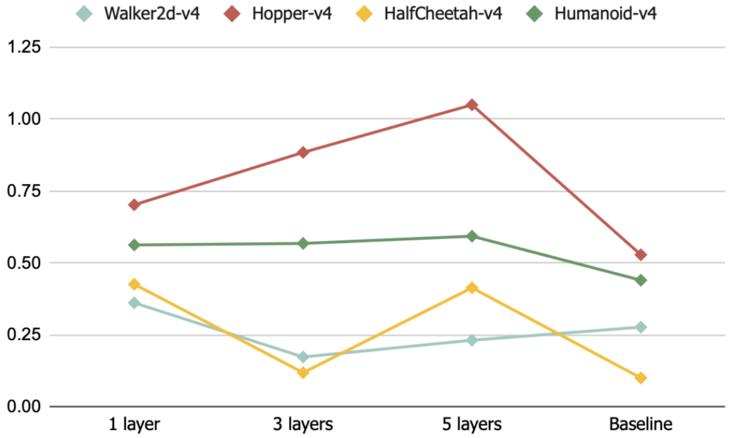

Figure 5: Effect of network depth on bagging performance.

5.5 Summary

Overall, 5-layers and 6-bags is the best combination of those tested. With those settings, we observe an average recovery of 107.5% increase over the baseline performance. The worst-performing model is on Walker2d-v4 (16% decrease compared to the baseline) while the best-performing model is on HalfCheetah-v4 (313.6% increase vs. baseline). Our method recovers, on average, 57.1% of the expert's reward. Hopper-v4 shows the best recovery, scoring 5% above the experts score. Walker2d-v4 again performs the worst, gathering only 23% of the expert's original reward.

6 Discussion

In our experiments, we were able to demonstrate that bagging-based ensemble methods using Generative Adversarial Imitation Learning (GAIL) models as weak learners show considerable improvements in final reward value over a single baseline GAIL model. In particular, the configuration using 5-layers and 6-bags was able to achieve significant improvements on average across our different MuJoCo environments. Our results seem to suggest that a bagging-based ensemble approach is effective in enhancing the overall performance of GAIL given the consistent nature of this result across all experiments.

We believe that our proposed model effectively reduced model variance and yielded more stable and generalized performance, pushing the GAIL agents closer to expert performance. Given previous literature on ensemble-based methods, this result aligns with expectation. Moreover, it is believed that the model's ability to combine multiple weak learners in a majority voting system helped in mitigating the potential risks with poor results due to a sub-optimally trained agent.

However, it is interesting to note the variable performance across environments. Given the varying complex nature of imitation learning–and in turn, deep imitation learning–one possible hypothesis is that variance may lead to effects that negatively impact ensemble methods. This effect is still not well understood and may be worth exploring in the future. We also note that bag-numbers and network complexity impacted the performance as well with performance diminishing at a 9-bag set-up. This result is expected as well as increased complexity may help push agents closer to expert performance but makes overfitting more likely. Furthermore, our method of partitioning rather than sampling meant that ensembles with more bags had weak learners that accessed less data.

6.1 Limitations

There are some limitations in our study worth addressing. For example, in the study, we were unable to propose a rigorous ensemble framework and experimental pipeline. Given the computational costs of training the weak learners and the limited resources in our research, some of our hyperparameter choices may have been sub-optimal. Along similar lines, we did not implement early stopping since adversarial network training dynamics are somewhat difficult to interpret. Future research on GAIL-based models should implement some form of early stopping, such as evaluating environment reward after each iteration and stopping once reward begins to plateau or decrease.

Our results also relied on comparing the final reward. Since our objective is to investigate imitation learning models, the loss would ideally be measured as some form of divergence such as KL divergence or Jensen–Shannon divergence.

Lastly, our method of partitioning the dataset rather than using independent samples limited the amount of expert data available to each discriminator. This provided each discriminator with less information about the intended expert behaviour, and may have made it easier for discriminators to memorize the training data.

6.2 Future Work

Additionally, GAIL is inherently a fairly complex model. Our focus in this paper has been a preliminary exploration of the effects of ensemble-based methods on imitation learning starting with GAIL as the weak learner. This then raises the question of whether our findings carry over to different weak learners in the current space and if there may be a better proposed weak learner.

Fundamentally, our experiments were able to show improvements to GAIL using ensemble learning. However, establishing the best ensemble-based RL and IL model should be further explored and examined. In particular, there may be better weak learners that may produce further improvements to performance in the domain of RL and IL.

6.3 Ethical Considerations

All of our datasets are generated using trained RL models in simulated environments, so we do not anticipate ethical problems arising from the datasets themselves.

In real-world applications, it should be noted that optimal policies in one setting or for one agent will not necessarily generalize to other settings and agents. Thus, practitioners should carefully consider the implications of the experts and settings covered by their expert dataset on both the privacy of experts used, and the policy that will be created. It should also be noted that we are working with relatively simple models in relatively simple environments with relatively simple reward functions. The problems for which imitation learning is needed will not have nicely-defined reward functions for which unbiased experts will be easily accessible.

Lastly, it may be worth noting that despite the aforementioned disadvantages of using partitions rather than independently-chosen subsets, weak learners of this form may be useful in contexts where some data points are likely to be retracted from ethical use. If each weak learner is trained using a partition of the overall training set, then at most one learner needs to be retrained whenever a datum is removed.

6.4 Conclusion

Overall, our study finds that the use of a bagging-based ensemble method for GAIL presents a promising and novel approach to pushing a model closer to expert performance. The increase in performance presents a potential in further frameworks utilizing ensemble-based methods in the space of RL and IL. Although it is commonly recommended to utilize simple weak learners in an ensemble model, we have demonstrated that complex models, which are common in the RL and IL space, can prove to be beneficial in increasing performance. While there were limitations in our study, this preliminary study provides a foundation to further research in this space. In particular, we hope to see further studies on the discussed limitations and refinements to our architecture and methodology.

7 Contributions

In this section, we outline the contributions of each author and acknowledge their respective involvement. In particular, Brain Ho contributed to the design and implementation of the bagging-based ensemble GAIL architecture, Sophia Wan contributed to project planning and feedback. Anthony DiMaggio contributed to the data pipeline and experimentation. Jasper Chen contributed to the data pipeline and architecture design.

Additionally, all members contributed to the paper with Brain Ho contributing to the initial draft of the Data Processing and Conclusion; Sophia Wan contributing to the initial draft of the Introduction, Ethical Considerations, and Limitations/Future Work sections along with the final draft; Anthony DiMaggio contributing to the Background, Experiments, Results sections, and Jasper Chen contributing to the Abstract, Architecture, and Discussion sections along with the the final draft of the paper.

8 References

[1] Saurabh Arora and Prashant Doshi. "A survey of inverse reinforcement learning: Challenges, methods and progress". In: *Artificial Intelligence* 297 (2021), p. 103500. ISSN: 0004-3702. DOI: `https://doi.org/10.1016/j.artint.2021.103500`.

[2] Leo Breiman. "Bagging predictors". In: *Machine Learning* 24.2 (Aug. 1996), pp. 123–140. ISSN: 1573-0565. DOI: `10.1007/BF00058655`. URL: `https://doi.org/10.1007/BF00058655`.

[3] Greg Brockman et al. *OpenAI Gym*. 2016. eprint: `arXiv:1606.01540`.

[4] Prafulla Dhariwal et al. *OpenAI Baselines*. `https://github.com/openai/baselines`. 2017.

[5] Jonathan Ho and Stefano Ermon. "Generative Adversarial Imitation Learning". In: *CoRR* abs/1606.03476 (2016). arXiv: `1606.03476`. URL: `http://arxiv.org/abs/1606.03476`.

[6] Shan Liu et al. "AdaBoost-Bagging deep inverse reinforcement learning for autonomous taxi cruising route and speed planning". In: *Transportation Research Part E: Logistics and Transportation Review* 177 (2023), p. 103232. ISSN: 1366-5545. DOI: `https://doi.org/10.1016/j.tre.2023.103232`.