A FINE-TUNING APPROACH TO BELIEF STATE MODELING

Anonymous authors
Paper under double-blind review

ABSTRACT

We investigate the challenge of modeling the belief state of a partially observable Markov system, given sample-access to its dynamics model. This problem setting is often approached using parametric sequential generative modeling methods. However, these methods do not leverage any additional computation at inference time to increase their accuracy. Moreover, applying these methods to belief state modeling in multi-agent settings would require passing policies into the belief model—at the time of writing, there have been no successful demonstrations of this. Toward addressing these shortcomings, we propose an inference-time improvement framework for parametric sequential generative modeling methods called belief fine-tuning (BFT). BFT leverages approximate dynamic programming in the form of fine-tuning to determine the model parameters at each time step. It can improve the accuracy of the belief model at test time because it specializes the capacity of the model to the space of local observations. Furthermore, because this specialization occurs after the action or policy has already been decided, BFT does not require the belief model to process it as input. As a result of the latter point, BFT enables, for the first time, approximate public belief state search in imperfect-information games where the number of possible information states is too large to track tabularly. We exhibit these findings on large-scale variants of the benchmark game Hanabi.

1 INTRODUCTION

A Markov system is a sequential process in which future events are independent of past events, conditioned on the current Markov state (Gagniuc, 2017). If the Markov state cannot be directly observed, the system is said to have partial observability. This work considers three kinds of Markov systems with partial observability: hidden Markov models (HMMs), partially observable Markov decision processes (POMDPs), and factored observation stochastic games (FOSGs). An HMM is a Markov system in which, at each time step, a distribution determined by the current (unobservable) Markov state generates an observable and a new (unobservable) Markov state (Rabiner & Juang, 1986). A POMDP is a generalization of an HMM in which an agent influences the trajectory of the system by taking actions (Kaelbling et al., 1998). A FOSG is a generalization of a POMDP in which multiple agents receive observations from the system and take actions that influence the trajectory of the system (Kovarík et al., 2019).

An important inference problem associated with Markov systems with partial observability is belief state modeling. The objective of belief state modeling is to compute the posterior (called the belief state) over the current Markov state, conditioned on all known information. This problem is important because belief states are sufficient information to forecast future event and to anticipate how the system will respond to external actors. In sufficiently small systems, the problem can be solved exactly using the classical forward algorithm (Rabiner, 1989), which is based on tabular dynamic programming. However, in larger systems, the forward algorithm is inapplicable, as it scales quadratically in the number of Markov states.

In such cases, an appealing alternative is to learn an approximate model of the belief state using a parametric sequential generative model. Parametric models are appealing because they can be scaled to very large settings and can be trained from samples. However, we suggest that naively performing inference on parametric models has two significant drawbacks. First, doing so does not

leverage any additional computation at inference time to improve the accuracy of the model. While the idea of performing additional local improvement is widely leveraged in reinforcement learning under the terms decision-time planning and search (Silver et al., 2017; Schrittwieser et al., 2020; Brown et al., 2020a), it has gone overlooked in the context of approximating belief states in large systems. Second, in the context of public belief state modeling in multi-agent systems, parametric models require a representation of the policy as input at each time step. At the time of writing, there has been no successful demonstration of this. As a result, performing belief modeling in large multi-agent systems remains out of reach, rendering celebrated algorithms for imperfect information games inapplicable (Brown & Sandholm, 2017; Moravčík et al., 2017; Brown & Sandholm, 2019).

To address these shortcomings, we propose an inference-time improvement framework for parametric models called belief fine-tuning (BFT). At each time step, BFT uses the belief model for the current step to generate an empirical distribution of current Markov states. Next, it uses this empirical distribution, along with the dynamics model (and player policies), to generate an empirical distribution of next Markov states and observations. Finally, BFT fine-tunes the belief model for the next time step using the latter empirical distribution. BFT can improve the accuracy of a belief model, even if it has been trained to convergence, by specializing its capacity to the space of local observations (Silver et al., 2008). Furthermore, BFT can model belief states even without taking actions or policies as input because, during fine-tuning, the action or policy has already been fixed.

To demonstrate the efficacy of BFT, we proceed with our experimental agenda in two parts, focusing on the cooperative imperfect-information game Hanabi (Bard et al., 2019). First, we verify that, as advertised above, BFT can improve the accuracy of a belief model in various settings. Second, we investigate the performance of decision-time planning running on top of BFT. We find that in cases in which tracking exact belief states is tractable, BFT yields performance competitive with that of an exact belief. Furthermore, in cases in which tracking the exact belief state is intractable, we find that performing search on top of BFT can yield substantial improvements over not performing search. This is the first instance of successful approximate belief state-based search in a multi-agent setting in which computing an exact belief state is intractable.

2 Background and Notation

We describe three formalisms for Markov models with partial observability, and then present a unifying notation for them.

2.1 Hidden Markov Models

An HMM is a tuple $\langle \mathcal{T}, \mathcal{O} \rangle$ where $\mathcal{T} \colon \mathbb{W} \to \Delta\mathbb{W}$ is the transition function and $\mathcal{O} \colon \mathbb{W} \to \Delta\mathbb{O}$ is the observation function, where $\Delta\mathbb{O}$ is the simplex on $\mathbb{O}$. At each time, the current Markov state w^t generates a new Markov state $W^{t+1} \sim \mathcal{T}(w^t)$. The new Markov state generates a new observation $O^{t+1} \sim \mathcal{O}(W^{t+1})$. The belief state is the posterior $\mathcal{P}(W^t \mid o^1, \ldots, o^t)$ over the Markov state, given the history of observations.

2.2 Partially Observable Markov Decision Processes

A POMDP is a tuple $\langle \mathcal{T}, \mathcal{O}, \mathcal{R} \rangle$ where $\mathcal{T} \colon \mathbb{W} \times \mathbb{A} \to \Delta\mathbb{W}$ is the transition function, $\mathcal{O} \colon \mathbb{W} \to \Delta\mathbb{O}$ is the observation function, and $\mathcal{R} \colon \mathbb{W} \times \mathbb{A} \to \mathbb{R}$ is the reward function. At each time step, an agent selects an action $a^t \in \mathbb{A}$ as a function of the history of its actions and observations $(o^1, a^1, \ldots, o^t)$. This action is used in conjunction with the current Markov state w^t to generate a new Markov state $W^{t+1} \sim \mathcal{T}(w^t, a^t)$ and a reward $r^t = \mathcal{R}(w^t, a^t)$. The observation for the next time step $O^{t+1} \sim \mathcal{O}(W^{t+1})$ is determined as a function of the new Markov state. The belief state is the posterior $\mathcal{P}(W^t \mid o^1, a^1, \ldots, o^t)$ over the Markov state, given the history of actions and observations.

2.3 Factored Observation Stochastic Games

A FOSG is a tuple $\langle \mathcal{T}, \mathcal{O}_{pub}, \mathcal{O}_{priv(1)}, \ldots, \mathcal{O}_{priv(n)}, \mathcal{R}_1, \ldots, \mathcal{R}_n \rangle$, where $\mathbb{T} \colon \mathbb{W} \times \mathbb{A} \to \Delta\mathbb{W}$ is the transition function, $\mathcal{O}_{pub} \colon \mathbb{W} \to \Delta\mathbb{O}_{pub}$ is the public observation function, $\mathcal{O}_{priv(i)} \colon \mathbb{W} \to \Delta\mathbb{O}_{priv(i)}$ is player i's private observation function, and $\mathcal{R}_i \colon \mathbb{W} \times \mathbb{A} \to \mathbb{R}$ is player i's reward

	General	HMM	POMDP	FOSG
Markov Variable	X	W	W	$(W, S_1, \ldots, S_n)$
Emission Variable	Y	O	O	O_{pub}
Control Variable	Z		A	$(\pi_1^t, \ldots, \pi_n^t)$

Table 1: The relationship between notations for belief state modeling for different settings.

function. At each time step, each agent selects an action $a_i^t \sim \pi_i^t(s_i^t)$ as a function of the history of its observations and actions $s_i^t = (o_{pub}^1, o_{priv(i)}^1, a_i^1, \ldots, o_{pub}^t, o_{priv(i)}^t)$. The joint action $a^t = (a_1^t, \ldots, a_n^t)$ is used in conjunction with the current Markov state w^t to generate a new Markov state $W^{t+1} \sim \mathcal{T}(w^t, a)$ and a reward $r_i^t = \mathcal{R}(w^t, a)$ for each player. The public observation $O_{pub}^{t+1} \sim \mathcal{O}_{pub}(W^{t+1})$ and each player's private observation $O_{priv(i)}^{t+1} \sim \mathcal{O}_{priv(i)}(W^{t+1})$ are determined as functions of the concurrent Markov state. In FOSGs the relevant belief state is the posterior $\mathcal{P}(W^t, S_1^t, \ldots, S_n^t \mid o_{pub}^1, \pi_1^1, \ldots, \pi_n^1, \ldots, o_{pub}^t)$ over the current Markov state and each player's action-observation history, conditioned on the sequence of public observations and the sequence of policies. This belief state is called the public belief state.

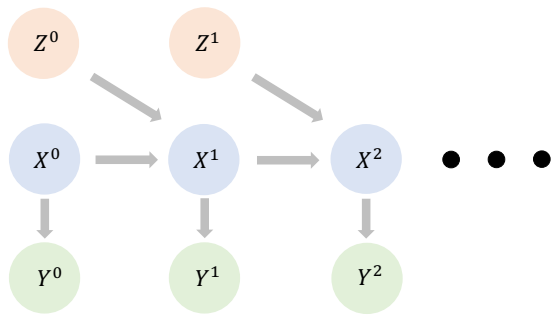

Figure 1: A graphical model depicting the belief state modeling problem. The Markov variable X^t and an externally specified control variable Z^t stochastically determine the next Markov variable X^{t+1}. Each Markov variable X^{t+1} stochastically determines the contemporaneous emission variable Y^{t+1}. The belief state modeling problem is to approximate the posterior $\mathcal{P}(X^t \mid Y^1, Z^1, \ldots, Y^t)$ over the current Markov variable given all past and present emission variables and all past control variables.

2.4 General Notation for Belief State Modeling

Regardless of whether the belief state modeling problem is in the context of HMMs, POMDPs, or FOSGs, belief state modeling can be expressed as modeling a distribution $\mathcal{P}(X^t \mid Y^1, Z^1, \ldots, Y^t)$, where each $Y^j \sim \mathcal{Y}(X^j)$, each Z^j is specified by an external process, and the process moves forward in time by sampling $X^{j+1} \sim \mathcal{X}(X^j, Z^j)$, as depicted in Figure 1. We call X the Markov variable, Y the emission variable, and Z the control variable. In HMMs, the Markov variable is the Markov state $X = W$, the emission variable is the observation $Y = O$ and the control variable is null. In POMDPs, the Markov variable is the Markov state $X = W$, the emission variable is the observation $Y = O$ and the control variable is the action $Z = A$. In FOSGs, the Markov variable is a tuple of the Markov state and each player's action-observation history $X = (W, S_1, \ldots, S_n)$, the emission variable is the public observation $Y = O_{pub}$, and the control variable is the joint decision rule $Z = (\pi_1^t, \ldots, \pi_n^t)$ (i.e., the policy for that time step). Table 1 summarizes the relationships between these notations. To minimize clutter, we will use this more concise notation going forward. In all cases, we assume sampling access to $\mathcal{Y} \colon \mathbb{X} \to \Delta\mathbb{Y}$ and $\mathcal{X} \colon \mathbb{X} \times \mathbb{Z} \to \Delta\mathbb{X}$.

2.5 Sequential Generative Modeling

The sequential generative modeling problem is to model the distribution of one sequence of variables, conditioned on another sequence of variables. Because it is a superset of the belief state mod-

eling problem, its solution methods are immediately applicable to belief state modeling. Among the most popular of these methods are parametric modeling approaches. Parametric modeling approaches use a set of sequences $\{(X_k^0, Y_k^0, Z_k^0, \ldots, X_k^T, Y_k^T)\}_k$ collected from the dynamics model $(\mathcal{X}, \mathcal{Y})$ to minimize the difference between the output of a parameterized model $f_\theta(Y_k^0, Z_k^0, \ldots, Y_k^t)$ and the distribution over $X^t \mid Y_k^0, Z_k^0, \ldots, Y_k^t$ observed in the dataset, for each t and k, over parameters θ. They generally require two design choices. First, whether to maintain a running summary of the conditioning sequence using a recurrent network (Hochreiter & Schmidhuber, 1997) or whether to model the desired distribution as a direct function of the running sequence at each time step (Vaswani et al., 2017). Second, what kind of model to use to capture the desired distribution. Common choices include autoregressive models, adversarial models (Goodfellow et al., 2014), variational models (Kingma & Welling, 2014), or score-based models (Hyvärinen, 2005), among others. Because the experiments in this work focus on the Hanabi benchmark (Bard et al., 2019), which involves long sequences and discrete output variables, we choose to use a recurrent architecture with an autoregressive output (Sutskever et al., 2014). However, BFT is not specific to these design choices and can also be combined with other approaches for sequential generative modeling in the context of belief state modeling.

3 METHODOLOGY

BFT comes from the approximate dynamic programming perspective on belief state modeling. We describe this perspective below, discuss its advantages, and then introduce BFT.

3.1 BELIEF STATE MODELING AS APPROXIMATE DYNAMIC PROGRAMMING

A second way to approach belief state modeling—distinct from standard sequential generative modeling setups—is as an approximate dynamic programming problem. Observe, for some fixed control variable z^{t-1}, previous belief state b^{t-1}, and current emission variable y^t, we have

$$b_{y^t}^t = \mathcal{P}(X^t \mid y^t, z^{t-1}, b^{t-1}) \tag{1}$$

$$\propto \mathcal{P}(y^t \mid X^t, z^{t-1}, b^{t-1})\mathcal{P}(X^t \mid z^{t-1}, b^{t-1}) \tag{2}$$

$$= \mathcal{P}(y^t \mid X^t)\mathbb{E}_{X^{t-1}\sim b^{t-1}}\mathcal{P}(X^t \mid z^{t-1}, X^{t-1}). \tag{3}$$

Line (1) follows by definition, line (2) follows by Bayes rule, and line (3) follows by the Markov property and the law of total probability. As a result of this proportionality, the procedure

1. Sample $X^{t-1} \sim b^{t-1}$
2. Sample $X^t \sim \mathcal{X}(X^{t-1}, z^{t-1})$
3. Sample $Y^t \sim \mathcal{Y}(X^t)$

yields a sample $X^t \sim b_{Y^t}^t$. Thus, given sample-access to $\mathcal{X}, \mathcal{Y}$, and the distribution of X^0, it is entirely possible to train a belief model inductively via approximate dynamic programming, whereby the belief states at time step one bootstrap from the initial belief state, the belief states at time step two bootstrap from the belief states at time step one, and so forth, as shown in Figure 2.

While this procedure is known, it is unusual for it to be used to train belief state models because the training data it produces is biased—unless the approximation to b^{t-1} is exactly correct, the training data for $b_{Y^t}^t$ will only come from an approximately correct distribution. In contrast, sampling full trajectories guarantees that the training data will come from exactly the correct distribution.

3.2 ADVANTAGES OF APPROXIMATE DYNAMIC PROGRAMMING

Despite that approximate dynamic programming produces biased samples, we argue here that the approximate dynamic programming perspective offers advantages that have gone largely overlooked in existing literature. These advantages derive from the fact that approximate dynamic programming allows us to generate local data. Given previous belief state b^{t-1}, generating samples for the current time step is as simple as following the procedure above. In contrast, generating unbiased versions of the same samples would require sampling a collection of initial Markov states $X_1^0, \ldots, X_k^0 \sim \mathcal{X}^0$ and propagating them forward using the same sequence of control variables $z^0, z^1, \ldots$, and rejecting the rollouts that yield emission sequences $Y^0, \ldots, Y^{t-1}$ that are inconsistent with the previously

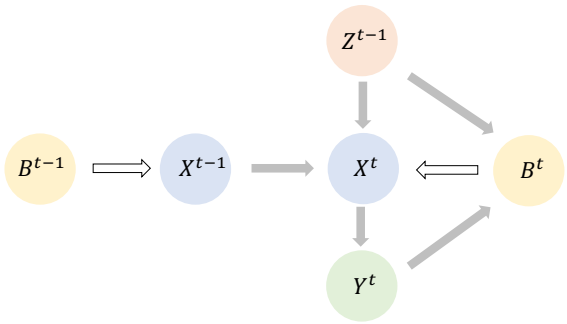

Table 2: A graphical model depiction of the dynamic programming perspective on belief state modeling. Gray arrows denote causal relationships; white arrows denote distributional modeling.

observed sequence of emission variables. For large t, it becomes prohibitively unlikely to sample a trajectory with exactly the same sequence of observations.

The ability to generate local data is important in the context of belief state modeling may be helpful for two reasons. First, it may increase the accuracy of the belief model by allowing it to perform an additional improvement step immediately prior to inference time. Even if the model was trained to convergence offline, it is typically the case that the model does not possess sufficient capacity to capture the exact belief for every possible sequence of emission and control variables. Therefore, an additional improvement step can provide utility by reallocating the belief model's capacity to the space of immediate possible futures, thereby increasing its ability to approximate the belief state for those futures. These ideas have long been understood in reinforcement learning literature (Silver et al., 2008), but have yet to be explored in the context of belief modeling.

A second advantage of inference-time improvement is that the control variable can be omitted from the belief model. This is a result of the fact that the Markov variable only depends on the control variable from the previous time step (not the current time step). Therefore, so long as inference-time improvement takes place after the control variable for the previous time step has been decided, the belief state depends only on the contemporaneous emission variable. In the context of HMMs or POMDPs, where the control variables are nulls and actions, respectively, the ability to omit the control variable carries little value. However, in FOSGs, where the control variables are themselves joint policies, the ability to omit the control variable is an important attribute. At the time of writing, there have been no convincing demonstrations of belief state modeling at scale in FOSGs due to the fact that it is not clear how to train a sequential generative model as a function of policy networks.

3.3 Belief Fine-Tuning

Toward leveraging the advantages of inference-time improvement discussed above, **we propose a framework for belief fine-tuning (BFT).** BFT takes a pretrained belief model f_θ as input and, at each time step during online inference, performs the following procedure:

1. Sample $X_1^{t-1}, \ldots, X_k^{t-1} \sim f_{\theta^{t-1}}(y^0, z^0, \ldots, y^{t-1})$.
2. Sample $X_i^t \sim \mathcal{X}(X_i^{t-1}, z^{t-1})$ for each $i = 1, \ldots, k$.
3. Sample $Y_i^t \sim \mathcal{Y}(X_i^t)$ for each $i = 1, \ldots, k$.
4. Set $\theta^t \leftarrow$ fine-tune$(\theta, \{(Y_i^t, X_i^t) \mid i = 1, \ldots, k\})$

For the final step, exactly how the fine-tuning is performed depends on the structure of the architecture f. If f takes the control variable as input, then z^{t-1} should also be passed into the model during fine tuning; if f is recurrent, then the hidden state $h_\theta^{t-1}(y^0, z^0, \ldots, y^{t-1})$ should also be passed into the model; if f takes the full sequence as input, then the full sequence should also be passed in during fine-tuning. In principle, BFT could produce accurate distributions from an untrained model that only took a single emission variable Y as input. However, we do not recommend this approach, as it essentially amounts to training a new generative model at each time step from scratch.

4 EXPERIMENTS

We divide our experimental investigation into two parts. In the first part, we explore the extent to which BFT can improve the belief quality of a parametric model in HMMs, POMDPs, and FOSGs. In the second part, we explore the idea of performing search on top of beliefs from BFT. We commit to releasing all code if accepted.

We use the cooperative card game Hanabi (Bard et al., 2019) for these experiments. Hanabi resembles something in the realm of a cooperative Solitaire—the players are tasked with collectively playing cards of each suit in rank order into piles on the table. However, a characterizing feature of Hanabi is that no player can observe its own cards. Instead, each must infer information about its own cards from other players actions. A more elaborate description of Hanabi is in the appendix.

In Hanabi, if the participants in the game agree to follow a fixed joint policy, the system can be considered an HMM from the perspective of a third party who does not observe the players' cards. If all but one of the players in the game agree to follow a fixed (joint) policy, the system can be considered a POMDP from the perspective of the player with a varying policy. If none of the players' policies are fixed, then the system can be modeled as a FOSG. Therefore, Hanabi allows us to investigate all three systems within the same infrastructure.

4.1 BELIEF FINE-TUNING FOR IMPROVING BELIEF QUALITY

For our first set of experiments, we verify that BFT can improve the accuracy of a pre-trained parametric belief model in HMMs, POMDPs, and FOSGs. For our experimental setup, we trained policies using independent R2D2 (Kapturowski et al., 2019) that collectively score around 24 out of 25 points, using the same hyperparameters as those found in (Hu & Foerster, 2020). We then trained a Seq2Seq model (Sutskever et al., 2014) to convergence for an HMM belief state, the POMDP belief state, and the public belief state (i.e., the FOSG belief state). For the Seq2Seq model, we used the same hyperparameters as those found in (Hu et al., 2021). For the FOSG case, we did not feed the players' policy networks into the Seq2Seq model.

At evaluation time, we used RLSearch (Fickinger et al., 2021) on top of the R2D2 policies to generate the actions for the non-fixed players in each setting. In short, RLSearch performs additional fine-tuning of the policy or value network at decision time, using samples from the belief state and the same learning algorithm that was used during training time. We used the same hyperparameters as those found in the original paper, with the exception that we used the search policy at every time step, rather than sometimes using the blueprint policy (i.e., the policy computed using R2D2). This choice was made so as to highlight the amount of improvement achievable from BFT. For BFT, we fine-tuned the encoder of the belief network for 10,000 gradient steps at each decision-point using the same hyperparameters that were used for offline training.

	V0 Belief	Seq2Seq	Seq2Seq + BFT (Ours)
HMM	2.08 ± 0.01	1.67 ± 0.01	1.58 ± 0.01
POMDP	1.73 ± 0.01	1.52 ± 0.01	1.39 ± 0.01
FOSG	2.12 ± 0.01	1.81 ± 0.01	1.62 ± 0.01

Table 3: Cross entropy per card with standard error for different settings, aggregated over 300 games.

Table 3 shows results for the Seq2Seq model, BFT on top of the Seq2Seq model, and a baseline called the V0 belief, which is naively from independence assumptions and the rules of the game (in ignorance of the actual player policies). The V0 belief serves as a sanity check that the belief model is working and provides a sense of scale about cross entropy values for each setting. The scale for the POMDP setting differs because it conditions on the acting player's private information and the belief is over only one player's hand; in contrast, the belief in the HMM and FOSG settings are over both players' hands and condition on only public information.

For the HMM setting, we observe BFT yields a statistically significant improvement over the belief model. This improvement comes despite the facts that 1) the belief model was trained using a stream of freshly generated data (so there is no overfitting problem), 2) the belief model was trained

to convergence, 3) there is no distribution shift at test time. We suggest that this result is evidence of the benefit of reallocating the capacity of the neural network at inference time.

For the POMDP setting, we observe that BFT yields an even larger improvement over the Seq2Seq model. While facts 1) and 2) of the above paragraph still hold true here, fact 3) does not. In particular, in the POMDP setting, there is covariate distribution shift—the acting player uses a search policy at evaluation time, rather than the blueprint policy used at training time. In principle, if the model had a sufficiently large amount of capacity and was trained to convergence, covariate distribution should not impact model performance. However, as a matter of practice, as is illustrated by the results in Table 3, covariate distribution shift can have a large effect on model performance.

Finally, in the FOSG setting, we observe that BFT induces the largest improvement. In this setting, the belief model faces not only covariate distribution shift, as it did in the POMDP setting, but also concept distribution shift, as a result of the facts that the belief model is ignorant to the control variables (the player policies) and that the distribution of control variables changes at test time because the agents are using search.

4.2 Belief Fine-Tuning For Decision-Time Planning

For our second set of experiments, we investigate whether the improvement in belief quality yielded by BFT leads to downstream improvements in search performance. For these experiments, we used the same hyperparameters as those found in (Fickinger et al., 2021) for the search algorithms and (Hu et al., 2021) for the Seq2Seq model. Again, we used 10,000 gradient steps on the encoder of the belief model with the same hyperparameters as offline training.

5-card 8-hint Hanabi

Blueprint	SPARTA Single Exact	RL Search Single Exact	SPARTA Multi Exact	RL Search Multi Exact	RL Search Multi Seq2Seq	RL Search Multi BFT (Ours)
24.23 ±0.04	24.57 ±0.03	24.59 ±0.02	24.61 ±0.02	24.62 ±0.03	24.35 ±0.03	24.58 ±0.02

5-card 2-hint Hanabi

Blueprint	SPARTA Single Exact	RL Search Single Exact	SPARTA Multi Exact	RL Search Multi Exact	SPARTA Multi Seq2Seq	RL Search Multi BFT (Ours)
22.99 ±0.04	23.60 ±0.03	23.61 ±0.03	23.67 ±0.03	23.76 ±0.04	23.35 ±0.03	23.69 ±0.03

Table 4: Expected returns with standard error, aggregated over 2000 games. RL Search Multi Exact is an upper bound on the performance we would hope for from BFT. RL Search Multi Seq2Seq is our baseline. BFT significantly outperforms the Seq2Seq model, and nearly matches the performance of exact beliefs.

We show performance results for (5-card, 8-hint) and (5-card, 2-hint) Hanabi in Table 4. We compare single and multi-agent SPARTA (Lerer et al., 2020) with exact beliefs, and single and multi-agent RLSearch (Fickinger et al., 2021) with exact beliefs, to two versions of multi-agent RLSearch with approximate beliefs. In one approximate-belief version, we used the Seq2Seq model trained offline. In the other version, we performed BFT on top of the Seq2Seq model after the search policy had been computed. In both the 2-hint and 8-hint variants, exact-belief multi-agent search substantially improves upon the blueprint policy. Seq2Seq beliefs fall far short of this improvement, while Seq2Seq + BFT nearly matches the performance of exact beliefs. These results suggest that the improvement in cross entropy from BFT observed in the previous section has an important effect on downstream search.

We can further investigate the added value of BFT by looking at the self-assessed improvement of multi-agent RLSearch with each of the three belief types (Seq2Seq, Seq2Seq + BFT, Exact), shown in Figure 2. We compute the increase in expected return by performing rollouts from the public belief state to the end of the game, using both the policy determined by RLSearch and the blueprint policy. Using the exact belief (blue) corresponds to computing the true improvement, whereas any error in the approximate beliefs will spill over into the improvement computation. We observe

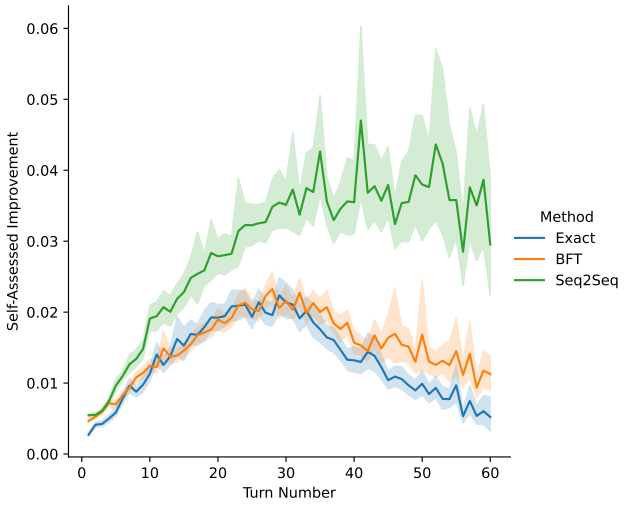

Figure 2: Without BFT, multi-agent RLSearch with Seq2Seq overestimates its performance.

that BFT (orange) mostly yields comparable self-assessed improvements to an exact belief, with a small amount of overestimation toward the end of the game. In contrast, the Seq2Seq belief (green) significantly overestimates the amount of improvement over the blueprint throughout the course of the game. These results help explain those found in Table 4—the Seq2Seq model is not able to properly assess policy values without BFT, leading it to believe its search policy is making large improvements when in fact its improvements may be small or non-existent.

7-card 4-hint Hanabi				
Blueprint	RLSearch Single Seq2Seq	RL Search Single BFT (Ours)	RL Search Multi Seq2Seq	RL Search Multi BFT (Ours)
23.67 ± 0.02	24.14 ± 0.04	24.18 ± 0.03	23.71 ± 0.07	24.18 ± 0.03

Table 5: Expected returns with standard error, aggregated over 1000 games. Multi-agent search with BFT significantly outperforms the multi-agent search with a Seq2Seq belief. Computing exact beliefs is intractable in this setting.

For our final experiment, we examine 7-card Hanabi—a setting in which there is so much private information that it is not possible to compute an exact belief. For this experiment, we include single and multi-agent variants of RLSearch using approximate beliefs from Seq2Seq and BFT. Similarly to the results in Table 4, we observe that multi-agent RLSearch without BFT outperformed the blueprint (though not with significance) but was significantly outperformed by every other setup. In contrast, multi-agent RLSearch with BFT significantly outperforms the blueprint. This is the first instance of public belief state-based search in a setting in which computing exact beliefs is intractable. For single-agent RLSearch, BFT resulted in better performance than Seq2Seq. We computed a one-sided t-test (computed by pairing deck orderings) and found that this improvement occurred with a p-value of 0.052.

5 Related Work

There has been much recent focus among the machine learning community on large, relatively general purpose, machine models that are fine-tuned for more specific tasks, such as BERT (Devlin et al., 2019), GPT-3 (Brown et al., 2020b), or DALL-E (Ramesh et al., 2021). Our work relates to this literature in the way it handles belief states for FOSGs—first by training a belief model ignorant

of the control variable offline, then later specializing the belief model to a specific control variable during online inference, though on a much smaller scale.

More generally, the idea of improving a model from a small amount of data at inference time also places our work close to the meta-learning literature (Hospedales et al., 2021). While there are various ways to perform meta-learning, such as recurrent neural networks (Ravi & Larochelle, 2017) or hypernetworks (Qiao et al., 2018), among the most popular is to fine-tune a network initialized using second order gradient descent (Finn et al., 2017).

BFT is also closely related to the idea of performing additional improvement at decision-time, which has a rich history in reinforcement learning literature—including many of the most widely acclaimed game-playing AI systems (Tesauro et al., 1995; Silver et al., 2017; Moravčík et al., 2017; Brown & Sandholm, 2017; 2019). However, the literature that specifically uses fine-tuning for decision-time planning is much smaller. The first work to do so with large non-linear networks is that of Anthony et al. (2019), who show that a policy gradient fine-tuning-based search can perform competitively with MCTS in Hex. More recently, Fickinger et al. (2021) investigate the idea of fine-tuning-based search using independent Q-learning in Hanabi and policy gradients in Ms. Pacman (Bellemare et al., 2012). However, this fine-tuning-based search was applied only to the policy, while beliefs were either tracked tabularly or were not fine-tuned.

Our work also relates to literature that attempts to work with public belief states at scale in FOSGs. The Bayesian action decoder scales approximate public belief states to Hanabi using Hanabi-specific independence heuristics and an expectation propagation-like procedure (Foerster et al., 2019). While we also examine the performance of approximate belief states on Hanabi, our method differs in that it uses no independence heuristic, is focused on inference-time improvement, and is not tied specifically to Hanabi. Another paper investigating public belief states in large settings is that of Šustr et al. (2021), who propose a scalable method for approximating the value of a public belief state from samples. More precisely, they investigate a value function architecture that takes samples from public belief states as input, rather than a closed form public belief state. Our contribution is complementary to theirs, as our method aims to produce high quality public belief state samples.

Finally, our work also relates to particle filtering literature (Doucet & Johansen, 2009). Particle filtering maintains works by maintaining a set of Markov states $\{x_i\}_i$ (called particles) and corresponding weights $\{w_i\}_i$ that collectively approximate the belief state. At each time step, similarly to BFT, the collection of particles is propagated forward in time using the dynamics model. The weights are updated based on the proportional posterior probability of each particle. One disadvantage of particle filtering compared to BFT is that all of the particles may become infeasible; in other words, every particle may have probability zero given the sequence of emission and control variables. Moreover, whereas particle filtering can only resample particles from its finitely supported empirical distribution, BFT can produce arbitrary resamples.

6 Conclusion and Future Work

In this work we introduced BFT, a framework for improving belief state models during online inference. We sought to demonstrate two claims: i) BFT can improve the performance of a belief model ii) this gain in belief accuracy yields downstream gains in search performance. We provided evidence for both claims using large scale variants of the Hanabi benchmark.

We see three lines of work worth pursuing as follow ups to this one. The first is to investigate whether incorporating meta-learning into BFT can improve its performance. While the inference-time improvement framework can be cast as a meta-learning problem, we did not actually incorporate this into the BFT objective. The second is to investigate the design of belief models that can quickly model public belief states. While BFT allows us to effectively model public belief states, it requires performing fine-tuning each time a public belief state is required. This is fine for algorithms which only use public belief states to construct the root of their search trees (Brown et al., 2018; Lerer et al., 2020), but may be too slow to use for algorithms that use public belief state-based value functions (Moravčík et al., 2017; Brown et al., 2020a; Sokota et al., 2021). The third line is to investigate eliminating the need for sampling access to the dynamics model. While MuZero (Schrittwieser et al., 2020) accomplished this for deterministic fully observable settings, determining the best way to do this for partially observable settings remains an open problem.

REFERENCES

Thomas Anthony, Robert Nishihara, Philipp Moritz, Tim Salimans, and John Schulman. Policy gradient search: Online planning and expert iteration without search trees. *CoRR*, abs/1904.03646, 2019. URL http://arxiv.org/abs/1904.03646.

Nolan Bard, Jakob N. Foerster, Sarath Chandar, Neil Burch, Marc Lanctot, H. Francis Song, Emilio Parisotto, Vincent Dumoulin, Subhodeep Moitra, Edward Hughes, Iain Dunning, Shibl Mourad, Hugo Larochelle, Marc G. Bellemare, and Michael Bowling. The hanabi challenge: A new frontier for AI research. *CoRR*, abs/1902.00506, 2019. URL http://arxiv.org/abs/1902.00506.

Marc Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47, 07 2012. doi: 10.1613/jair.3912.

Noam Brown and Tuomas Sandholm. Superhuman ai for heads-up no-limit poker: Libratus beats top professionals. *Science*, pp. eaao1733, 2017.

Noam Brown and Tuomas Sandholm. Superhuman ai for multiplayer poker. *Science*, 365(6456): 885–890, 2019. ISSN 0036-8075. doi: 10.1126/science.aay2400. URL https://science.sciencemag.org/content/365/6456/885.

Noam Brown, Tuomas Sandholm, and Brandon Amos. Depth-limited solving for imperfect-information games. In *NeurIPS*, 2018.

Noam Brown, Anton Bakhtin, Adam Lerer, and Qucheng Gong. Combining deep reinforcement learning and search for imperfect-information games. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 17057–17069. Curran Associates, Inc., 2020a. URL https://proceedings.neurips.cc/paper/2020/file/c61f571dbd2fb949d3fe5ae1608dd48b-Paper.pdf.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020b. URL https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.

Arnaud Doucet and Adam Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of Nonlinear Filtering*, 12, 01 2009.

Arnaud Fickinger, Hengyuan Hu, Brandon Amos, Stuart Russell, and Noam Brown. Scalable online planning via reinforcement learning fine-tuning. In *International Conference on Learning Representations*, 2021.

Chelsea Finn, P. Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.

Jakob Foerster, Francis Song, Edward Hughes, Neil Burch, Iain Dunning, Shimon Whiteson, Matthew Botvinick, and Michael Bowling. Bayesian action decoder for deep multi-agent reinforcement learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 1942–1951. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/foerster19a.html.

Paul Gagniuc. *Markov Chains: From Theory to Implementation and Experimentation*. 05 2017. ISBN 978-1-119-38755-8. doi: 10.1002/9781119387596.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, 1997.

Timothy M. Hospedales, Antreas Antoniou, Paul Micaelli, and Amos J. Storkey. Meta-learning in neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, PP, 2021.

Hengyuan Hu and Jakob N Foerster. Simplified action decoder for deep multi-agent reinforcement learning. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=B1xm3RVtwB.

Hengyuan Hu, Adam Lerer, Noam Brown, and Jakob N. Foerster. Learned belief search: Efficiently improving policies in partially observable settings. *CoRR*, abs/2106.09086, 2021. URL https://arxiv.org/abs/2106.09086.

Aapo Hyvärinen. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(24):695–709, 2005. URL http://jmlr.org/papers/v6/hyvarinen05a.html.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1):99–134, 1998. ISSN 0004-3702. doi: https://doi.org/10.1016/S0004-3702(98)00023-X. URL https://www.sciencedirect.com/science/article/pii/S000437029800023X.

Steven Kapturowski, Georg Ostrovski, Will Dabney, John Quan, and Remi Munos. Recurrent experience replay in distributed reinforcement learning. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=r1lyTjAqYX.

Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

Vojtech Kovarík, Martin Schmid, Neil Burch, Michael Bowling, and Viliam Lisý. Rethinking formal models of partially observable multiagent decision making. *CoRR*, abs/1906.11110, 2019. URL http://arxiv.org/abs/1906.11110.

Adam Lerer, Hengyuan Hu, Jakob Foerster, and Noam Brown. Improving policies via search in cooperative partially observable games. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7187–7194, Apr. 2020. doi: 10.1609/aaai.v34i05.6208. URL https://ojs.aaai.org/index.php/AAAI/article/view/6208.

Matej Moravčík, Martin Schmid, Neil Burch, Viliam Lisý, Dustin Morrill, Nolan Bard, Trevor Davis, Kevin Waugh, Michael Johanson, and Michael Bowling. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 356(6337):508–513, 2017. doi: 10.1126/science.aam6960.

Siyuan Qiao, Chenxi Liu, Wei Shen, and Alan Loddon Yuille. Few-shot image recognition by predicting parameters from activations. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7229–7238, 2018.

L. Rabiner and B. Juang. An introduction to hidden markov models. *IEEE ASSP Magazine*, 3(1): 4–16, 1986. doi: 10.1109/MASSP.1986.1165342.

L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989. doi: 10.1109/5.18626.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. *CoRR*, abs/2102.12092, 2021. URL https://arxiv.org/abs/2102.12092.

S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. In *ICLR*, 2017.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, L. Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy P. Lillicrap, and David Silver. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588 7839:604–609, 2020.

David Silver, Richard S. Sutton, and Martin Müller. Sample-based learning and search with permanent and transient memories. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pp. 968–975, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605582054. doi: 10.1145/1390156.1390278. URL https://doi.org/10.1145/1390156.1390278.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy P. Lillicrap, Karen Simonyan, and Demis Hassabis. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *CoRR*, abs/1712.01815, 2017. URL http://arxiv.org/abs/1712.01815.

Samuel Sokota, Edward Lockhart, Finbarr Timbers, Elnaz Davoodi, Ryan D'Orazio, Neil Burch, Martin Schmid, Michael Bowling, and Marc Lanctot. Solving common-payoff games with approximate policy iteration. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(11): 9695–9703, May 2021. URL https://ojs.aaai.org/index.php/AAAI/article/view/17166.

Michal Šustr, Vojtěch Kovařík, and Viliam Lisỳ. Particle value functions in imperfect information games. In *AAMAS Adaptive and Learning Agents Workshop*, 2021.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL https://proceedings.neurips.cc/paper/2014/file/a14ac55a4f27472c5d894ec1c3c743d2-Paper.pdf.

Gerald Tesauro et al. Temporal difference learning and td-gammon. *Communications of the ACM*, 38(3):58–68, 1995.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

A Appendix

A.1 Rules of Hanabi

The following rule description is quoted from the Hanabi challenge paper (Bard et al., 2019):

> Hanabi is a game for two to five players, best described as a type of cooperative solitaire. Each player holds a hand of four cards (or five, when playing with two or three players). Each card depicts a rank (1 to 5) and a colour (red, green, blue, yellow, and white); the deck (set of all cards) is composed of a total of 50 cards,

10 of each colour: three 1s, two 2s, 3s, and 4s, and finally a single 5. The goal of the game is to play cards so as to form five consecutively ordered stacks, one for each colour, beginning with a card of rank 1 and ending with a card of rank 5. What makes Hanabi special is that, unlike most card games, players can only see their partners' hands, and not their own. Players take turns doing one of three actions: giving a hint, playing a card from their hand, or discarding a card. We call the player whose turn it is the active player.

Hints On their turn, the active player can give a hint to any other player. A hint consists of choosing a rank or colour, and indicating to another player all of their cards that match the given rank or colour. Only ranks and colors that are present in the player's hand can be hinted for. To make the game interesting, hints are in limited supply. The game begins with the group owning eight information tokens, one of which is consumed every time a hint is given. If no information tokens remain, hints cannot be given and the player must instead play or discard.

Discard Whenever fewer than eight information tokens remain, the active player can discard a card from their hand. The discarded card is placed face up (along with any unsuccessfully played cards), visible to all players. Discarding has two effects: the player draws a new card from the deck and an information token is recovered.

Play Finally, the active player may pick a card from their hand and attempt to play it. Playing a card is successful if the card is the next in the sequence of its colour to be played. If the play is successful, the card is placed on top of the corresponding stack. When a stack is completed (the 5 is played) the players also receive a new information token (if they have fewer than eight). The player can play a card even if they know nothing about it; but if the play is unsuccessful, the card is discarded (without yielding an information token) and the group loses one life, possibly ending the game. In either circumstances, a new card is drawn from the deck.

Game Over The game ends in one of three ways: either because the group has successfully played cards to complete all five stacks, when three lives have been lost, or after a player draws the last card from the deck and every player has taken one final turn. If the game ends before three lives are lost, the group scores one point for each card in each stack, for a maximum of 25.

A.2 Additional Experiments

We performed additional experiments to see the effect of the number of fine-tuning steps, the number of training epochs, and the size of the belief model. Note that we did not use the same networks for these results as for the HMM results in the main body so they are not directly comparable.

A.2.1 Number of Fine-Tuning Steps

In the main body, we used 10000 gradient steps for all of our experiments. In Table 6, we examine the effect of using fewer gradient steps in an HMM setting. We find that BFT can yield benefits even with as few as 100 gradient steps. However, the benefits appear to saturate after 1000 gradient steps. This, in part, may be because the belief model is already relatively good in the HMM case.

V0 Belief	2.08 ± 0.01
Seq2Seq	1.68 ± 0.01
Seq2Seq + BFT(100 steps)	1.64 ± 0.01
Seq2Seq + BFT(1000 steps)	1.61 ± 0.01
Seq2Seq + BFT(10000 steps)	1.61 ± 0.01

Table 6: Cross entropy per card with standard error for different settings, aggregated over 300 games.

A.2.2 AMOUNT OF OFFLINE TRAINING

In the main body, our belief model was trained for 400 epochs offline. In Table 7, we examine the performance of combining BFT with lower quality belief models in an HMM setting. We find that BFT can produce large improvements even when its original model performs poorly.

V0 Belief	2.08 ± 0.01
Seq2Seq(400 epochs)	1.68 ± 0.01
Seq2Seq(400 epochs) + BFT	1.61 ± 0.01
Seq2Seq(200 epochs)	1.74 ± 0.01
Seq2Seq(200 epochs) + BFT	1.63 ± 0.01
Seq2Seq(100 epochs)	1.82 ± 0.01
Seq2Seq(100 epochs) + BFT	1.70 ± 0.01

Table 7: Cross entropy per card with standard error for different settings, aggregated over 300 games.

A.2.3 NETWORK SIZE AND PART OF NETWORK TUNED

In the main body of the paper, the belief model's encoder was a two layer LSTM with 512 hidden units. Also, in the main body of the paper, we performed fine-tuning only on the encoder. In Table 8, we examine the effect of using smaller encoders and of fine-tuning the whole belief network, rather than just the encoder, in the HMM setting. We find that the benefits to performing fine-tuning may increase as the capacity of the belief network decreases, if fine-tuning is performed for the whole network. However, in the case that fine-tuning is performed only on the encoder, we did not observe this effect.

V0 Belief	2.08 ± 0.01
Seq2Seq(512 units)	1.68 ± 0.01
Seq2Seq(512 units) + BFT(LSTM)	1.61 ± 0.01
Seq2Seq(512 units) + BFT(Full)	1.58 ± 0.01
Seq2Seq(256 units)	1.76 ± 0.01
Seq2Seq(256 units) + BFT(LSTM)	1.65 ± 0.01
Seq2Seq(256 units) + BFT(Full)	1.63 ± 0.01
Seq2Seq(128 units)	1.87 ± 0.01
Seq2Seq(128 units) + BFT(LSTM)	1.84 ± 0.01
Seq2Seq(128 units) + BFT(Full)	1.69 ± 0.01

Table 8: Cross entropy per card with standard error for different settings, aggregated over 300 games.

A.3 DESCRIPTION OF ALGORITHM IMPLEMENTATION

We describe the instantiation of BFT we used for our Hanabi experiments. We used a Seq2Seq model with LSTMs as encoders and an LSTM as a decoder. In particular, our encoder took the previous control variable z^{t-1}, the current emission variable y^t, and a summary of the history $h(y^0, z^0, \ldots, y^{t-1})$ and output a next summary of the history $h(y^0, z^0, \ldots, y^t) = e(z^{t-1}, y^t, h(y^0, z^0, \ldots, y^{t-1}))$. Our decoder d took the summary of the history $h(y^0, z^0, \ldots, y^{t-1}))$ as input, and autoregressively modeled the Markov state x^t.

During fine-tuning time, we first sample Markov states using a) the decoder with offline parameters θ, b) the encoder using the fine-tuned parameters from the previous time step θ^{t-1}, and c) the hidden state produced two time steps ago using the the offline parameters θ. In steps 2) and 3), we propagate these samples forward to the current time step, as described in the main body of the paper. In the fourth step, we fine-tune the parameters of the encoder (with the parameters initialized as the offline parameters θ), using the decoder (with fixed offline parameters θ). and the fixed hidden state from the previous time step computed using offline parameters θ. The result of this fine tuning yields encoder parameters θ^t. For the FOSG setting, we omitted z as input in step 1) and step 4) (i.e., we only used it in step 2)).

1. Sample $X_1^{t-1}, \ldots, X_k^{t-1} \sim d_\theta \circ e_{\theta^{t-1}}(z^{t-2}, y^{t-1}, h_\theta(y^0, z^0, \ldots, y^{t-2})$.
2. Sample $X_i^t \sim \mathcal{X}(X_i^{t-1}, z^{t-1})$ for each $i = 1, \ldots, k$.
3. Sample $Y_i^t \sim \mathcal{Y}(X_i^t)$ for each $i = 1, \ldots, k$.
4. Set $\theta^t \leftarrow \min_{\theta'} \text{NLL}(X_i^t, d_\theta \circ e_{\theta'}(z^{t-1}, Y_i^t, h_\theta(y^0, z^0, \ldots, z^{t-2}, y^{t-1}))$ for $i = 1, \ldots, k$.

An alternative instantiation, used for BFT(Full) in Table 8, fine-tunes both the encoder and the decoder. In this case d_θ is replaced by $d_{\theta^{t-1}}$ in step 1), and d_θ is replaced by $d_{\theta'}$ in step 4). Only fine-tuning the encoder has the advantage of preserving the equivalence between the encoding and the belief state, which may be useful for training belief state-dependent value functions in future work. However, fine-tuning the whole network can sometimes result in better performance, as we found in Table 8.

RLSearch only requires samples from the current belief state. To provide these samples, at time t, we generated samples from $d_\theta \circ e_{\theta^t}(z^{t-1}, y_i^t, h_\theta(y^0, z^0, \ldots, z^{t-2}, y^{t-1}))$. Again, in the FOSG case we omitted the control variables. We sampled a large number of Markov states prior to the start of search, and then passed this empirical distribution of samples to our RLSearch implementation. Note that, if fine-tuning the whole network, d_θ should be replaced with d_{θ^t} here.

All of the hyperparameters we used will be included in our code release.