Large-Scale Multi-Agent Deep FBSDEs

Tianrong Chen [1] Ziyi Wang [2] Ioannis Exarchos [3] Evangelos A. Theodorou [2 4]

Abstract

In this paper we present a scalable deep learning framework for finding Markovian Nash Equilibria in multi-agent stochastic games using fictitious play. The motivation is inspired by theoretical analysis of Forward Backward Stochastic Differential Equations (FBSDE) and their implementation in a deep learning setting, which is the source of our algorithm's sample efficiency improvement. By taking advantage of the permutation-invariant property of agents in symmetric games, the scalability and performance is further enhanced significantly. We showcase superior performance of our framework over the state-of-the-art deep fictitious play algorithm on an inter-bank lending/borrowing problem in terms of multiple metrics. More importantly, our approach scales up to 3000 agents in simulation, a scale which, to the best of our knowledge, represents a new state-of-the-art. We also demonstrate the applicability of our framework in robotics on a belief space autonomous racing problem.

1. Introduction

Stochastic Differential Games (SDG) represent a framework for investigating scenarios where multiple players make decisions in a stochastic environment. The theory of differential games dates back to the seminal work of Isaacs (1965) studying two-player zero-sum dynamic games, with the stochastic extension first appearing in Kushner & Chamberlain (1969). A key step in the study of games is to obtain the Nash equilibrium among players (Osborne & Rubinstein, 1994). A Nash equilibrium represents the solution of non-cooperative games where two or more players are involved. At the equilibrium, each player cannot gain any benefit by modifying his/her own strategy given opponents' strategy. In the context of adversarial multi-objective games, the Nash equilibrium can be represented as a system of coupled Hamilton-Jacobi-Bellman (HJB) equations when the system satisfies the Markovian property. Analytical solutions exist only for few special cases. Therefore, obtaining the Nash equilibrium solution is usually done numerically, and this can become challenging as the number of states/agents increases. Despite extensive theoretical work (Buckdahn & Li, 2008; Ramachandran & Tsokos, 2012), the algorithmic part has received less attention and is limited to special cases of differential games (e.g., Duncan & Pasik-Duncan (2015)), or suffers from the curse of dimensionality (Kushner, 2002). Nevertheless, SDG has a variety of applications including in robotics, autonomy, economics and management. Relevant examples include Mataramvura & Øksendal (2008), which formulates portfolio management as a SDG in order to obtain a market portfolio that minimizes the convex risk measure of a terminal wealth index value, as well as Prasad & Sethi (2004), which investigates optimal advertising spending in duopolistic settings via SDG.

Researchers have also been solving differential games via Reinforcement Learning (RL) (Harmon et al., 1995). Kamalapurkar et al. (2016) seeks to combine differential game theory and RL to solve cooperative control problems in which agents are coupled by graphs. Multi-agent RL (MARL) is an extension of RL, which is a more complex task due to internal interactions between agents and external interactions with environment. Independent learning (Tan, 1993) is an approach which assumes other agents are part of environment, and it suffers from the unstable learning problem. Training agents by augmenting their states and actions is called centralized training and executing (Yang et al., 2019), and scalability of it is limited. Another method is centralized training and decentralized execute (CTDE), however the challenge therein lies on how to decompose value function in the execute phase for value-based MARL. Sunehag et al. (2018); Zhou et al. (2019); Rashid et al. (2018); Son et al. (2019); Mahajan et al. (2019) achieve remarkable success in this vein. However, RL usually adopts model-free learning scheme in discrete time. Here we leverage the HJB PDE which contains information of the dynamics to develop data-efficient algorithm in continuous time. The de-

[1] School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, USA [2] Center for Machine Learning, Georgia Institute of Technology, Atlanta, USA [3] Department of Computer Science, Stanford University, Stanford, USA [4] School of Aerospace Engineering, Georgia Institute of Technology, Atlanta, USA. Correspondence to: Tianrong Chen <tchen429@gatech.edu>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).

composition of value function is characterized by decoupled property of HJB PDE by modifying nonlinear Feynman-Kac Lemma (Karatzas & Shreve, 1991) in multi-agent scenario.

Another vein of related research marries the mathematical formulation of a differential game with nonlinear PDE. This motivates the algorithmic development for differential games that combines elements of PDE theory with deep learning. Recent encouraging results (Han et al., 2018; Raissi, 2018) in solving nonlinear PDEs with deep learning illustrate the scalability and numerical efficiency of neural networks. The transition from a PDE formulation to a trainable neural network is done via a system of Forward-Backward Stochastic Differential Equations (FBSDEs). Specifically, certain PDE solutions are linked to solutions of FBSDEs, and the latter can be solved using a suitably defined neural network architecture. This is known as the *Deep FBSDE* approach. Han et al. (2018); Pereira et al. (2019); Wang et al. (2019b) utilize various deep neural network architectures to solve such stochastic systems. However, these algorithms address single agent dynamical systems. Two-player zero-sum games using FBSDEs were initially developed in Exarchos et al. (2019) and transferred to a deep learning setting in Wang et al. (2019a). Recently, Hu (2019) brought deep learning into fictitious play to solve multi-agent non-zero-sum games. Han & Hu (2019) introduced the Deep FBSDEs to a multi-agent scenario and the concept of fictitious play. Furthermore, Han & Long (2020) gives the convergence analysis of the framework. Our work mainly focuses on stochastic differential game with homogeneous agents which can also be seen as the case of mean field game with finite number of agents. Under mild assumptions, the approximation error of mean field game is of order $N^{-1/(d+4)}$ (Carmona & Delarue, 2013) where N is the number of agents and d is the state dimension of individual agent. Our algorithm can mitigate the approximation error when the number of agents is moderate.

In this work, we first extend the theoretical analysis in Han & Long (2020) to the scenario where the forward process includes a drift term. Suggested by the theoretic result, we integrate importance sampling (Bender & Moseler, 2010) into the existing framework. Furthermore, by leveraging the property of symmetric game, we introduce an invariant feature representation to reduce the complexity and significantly increase the number of agents the framework can handle. The main contribution of our work is threefold:

1. We theoretically analyze FBSDE with drift term under fictitious play problem setup, and explain the efficiency of importance sampling analytically and numerically.
2. We introduce an efficient Deep FBSDE framework for solving stochastic multi-agent games via fictitious play that outperforms the current state-of-the-art in performance and runtime/memory efficiency on an inter-bank lending/borrowing example. We demonstrate that our approach scales to a much larger number of agents (up to 3,000 agents, compared to 100 in existing works). To the best of our knowledge, this represents a new state-of-the-art.
3. We showcase the applicability of our framework to robotics on a belief-space autonomous racing problem which has larger individual control and state space. The experiments demonstrate that the BSDE has decoupled property and provides the possibility of applications for competitive scenario, and the model can extract informative features with partially observed input data.

The rest of the paper is organized as follows: in §2 we present the notation and mathematical preliminaries. In §3 we introduce the Scaled Deep Fictitious Play FBSDE (SDFP-FBSDE) algorithm, with simulation results following in §4. We conclude the paper in §5.

2. Notation and Preliminaries

Table 1. Mathematical notation.

CHARACTERS	DEFINITIONS
$\boldsymbol{X}$	quantities for all agents
X^i/X_i	quantities for ith agent
$\boldsymbol{X}_{-i}$	quantities from all agents except ith
$\boldsymbol{x}$	realization of $\boldsymbol{X}$

Throughout the paper the notation will be:

1. Bold letters denote quantities for all players[1]
2. (sup)subscript i indicates quantities for player i.
3. Bold letters with (sup)subscript $-i$ indicates quantities for all players except i.
4. Lower case letter is the realization of the upper case random variable.

Table.1 contains examples for the notation defined above. The baseline used in this paper refers to Han & Hu (2019).

2.1. HJB PDE and FBSDE

Fictitious play is a learning rule (Brown, 1951) where each player presumes other players' strategies to be fixed. An N-player game can then be decoupled into N individual decision-making problems which can be solved iteratively over M stages. When each agent converges to a stationary strategy at stage $m-1$, this strategy will become the stationary strategy for other players at stage m. We consider a N-player non-cooperative SDG with dynamics:

$$\begin{aligned} \mathrm{d}\boldsymbol{X}_t &= \big(f(\boldsymbol{X}_t,t)+G(\boldsymbol{X}_t,t)\boldsymbol{U}(\boldsymbol{X}_t)\big)\mathrm{d}t+\Sigma(\boldsymbol{X}_t,t)\mathrm{d}\boldsymbol{W}_t \\ \boldsymbol{X}_{t_0} &= \boldsymbol{x}_{t_0}, \end{aligned} \tag{1}$$

[1] Agent and player are used interchangeably in this paper

where $\boldsymbol{X} = (X_1, X_2, \ldots, X_N)$ is a vector containing the state process of all agents generated by their controls $\boldsymbol{U} = (U_1, U_2, \ldots, U_N)$ defined on the space $\mathcal{X}$ and $\mathcal{U}$, with $X_i \in \mathcal{X}_i \subseteq \mathbb{R}^{n_x}$ and $U_i \in \mathcal{U}_i \subseteq \mathbb{R}^{n_u}$. Here, $f : \mathcal{X} \times [t_0, T] \rightarrow \mathcal{X}$ represents the drift dynamics, $G : \mathcal{X} \times [t_0, T] \rightarrow \mathcal{X} \times \mathcal{U}$ represents the actuator dynamics, and $\Sigma : \mathcal{X} \times [t_0, T] \rightarrow \mathcal{X} \times \mathbb{R}^{n_w}$ represents the diffusion term. The state process is also driven by an n_w-dimensional independent Brownian motion W_i, and denoted $\boldsymbol{W} = (W_1, W_2, \ldots, W_N)$.

Given the other agents' strategies, the stochastic optimal control problem for agent i under the fictitious play assumption is defined as minimizing the expectation of the cumulative cost functional J_t^i

$$J_t^i(\boldsymbol{X}, U_{i,m}; \boldsymbol{U}_{-i,m-1}) = \\ \mathbb{E}\left[g(\boldsymbol{X}_T) + \int_t^T C^i(\boldsymbol{X}_\tau, U_{i,m}(\boldsymbol{X}_\tau); \boldsymbol{U}_{-i,m-1})\mathrm{d}\tau\right], \tag{2}$$

where $g : \mathcal{X} \rightarrow \mathbb{R}^+$ is the terminal cost, and $C^i : \mathcal{X} \times \mathcal{U} \rightarrow \mathbb{R}^+$ is the running cost for the i-th player. In this paper we assume that the running cost is of the form $C^i(\boldsymbol{X}, U_i) = q^i(\boldsymbol{X}) + \frac{1}{2}U_i^{\mathrm{T}} R U_i + \boldsymbol{X}^{\mathrm{T}} Q U_i$. We use the double subscript $U_{i,m}$ to denote the control of agent i at stage m. We can define value function of each player as

$$\begin{aligned} V_t^i(\boldsymbol{X}_t) &= \inf_{u_{i,m} \in \mathcal{U}_i} \left[J_t^i(\boldsymbol{X}, U_{i,m}; \boldsymbol{U}_{-i,m-1})\right] \\ V_T^i(\boldsymbol{X}_T) &= g(\boldsymbol{X}_T). \end{aligned} \tag{3}$$

Assume that V_t^i in eq. (3) is once differentiable w.r.t. t and twice differentiable w.r.t. $\boldsymbol{X}$. Here we drop the functional dependencies for simplicity. Then, standard stochastic optimal control theory leads to the HJB PDE:

$$V_t^i + \inf_{u^i \in \mathcal{U}_i} \left\{V_x^{i\mathrm{T}}(f + G\boldsymbol{U}) + C^i\right\} + \frac{1}{2}\mathrm{tr}(V_{xx}^i \Sigma\Sigma^{\mathrm{T}}) = 0 \tag{4}$$

The function inside the infimum is known as the Hamiltonian function $\boldsymbol{H}(t, \boldsymbol{X}, \boldsymbol{U}, Z^i)$. Z^i is known as adjoint states and is defined as $Z^i = \Sigma^{\mathrm{T}} V_x^i$ in literature (Yong & Zhou, 1999). If the optimal control can be obtained, by plugging back to eq.4, one can have,

$$V_t^i + h + V_x^{i\mathrm{T}}(f + G\boldsymbol{U}_{0,*}) + \frac{1}{2}\mathrm{tr}(V_{xx}^i \Sigma\Sigma^{\mathrm{T}}) = 0, \tag{5}$$

where $h^i = C^{i*} + G\boldsymbol{U}_{*,0}$. The first and second subscript of $\boldsymbol{U}$ represents for the control taken by ith and $-i$th agent respectively. The $*$ denotes the optimal control computed from $U_i^* = -R^{-1}(G_i^{\mathrm{T}} V_x^i + Q_i^{\mathrm{T}} \boldsymbol{X})$ and the 'zero' represents for taking zero control $\boldsymbol{U}_{-i} = 0$. For instance, $\boldsymbol{U}_{0,*}$ denotes the augmentation of $U_i = 0$ and $\boldsymbol{U}_{-i} = \boldsymbol{U}_{-i}^*$ as $(U_1^*, \cdots, U_{i-1}^*, 0, U_{i+1}^*, \cdots, U_N^*)$ and $\boldsymbol{U}_{*,0} = (0, \cdots, 0, U_i^*, 0, \cdots, 0)$. The derivation is included in Appendix A for completeness. The value function V_t^i in the HJB PDE can be related to a set of FBSDEs by non-linear Fayman-Kac Lemma Karatzas & Shreve (1991),

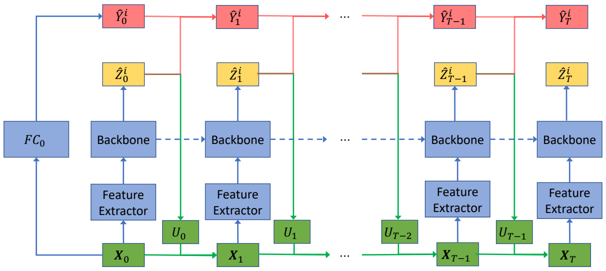

Figure 1. FBSDE Network for a single agent. The dashed arrow indicates hidden states propagation if LSTM is chosen as backbone. The dash arrow would disappear when FC is chosen.

$$\begin{aligned} &\mathrm{d}\boldsymbol{X}_t = (f + G\boldsymbol{U}_{0,*})\mathrm{d}t + \Sigma\mathrm{d}\boldsymbol{W}_t,\ \boldsymbol{X}_{t_0} = \boldsymbol{x}_{t_0} && \text{(FSDE)}, \\ &\mathrm{d}Y_t^i = -h_t^i\mathrm{d}t + Z_t^i\mathrm{d}W_t,\ Y_T^i = g(\boldsymbol{X}_T), && \text{(BSDE)}, \end{aligned} \tag{6}$$

where the backward process corresponds to the value function, and $t \in [t_0, T]$. The derivation can be found in Appendix B. Here we denote value function V as Y in order to be aligned with classic FBSDE literature.

Remark 1. *The problem of solving HJB PDE (5) will be transformed to solve a FBSDE system. (6). The ith player will provide zero control in the drift term in the forward process, and the rest of agents will execute the optimal policy according to the Value function.*

The HJB PDE (5) can be solved by Numerical PDE solvers based on finite element and finite difference methods (Greif, 2017). Howerver, these methods do not scale beyond very few dimensions, as they rely on discretization.

2.2. Deep Fictitious Play FBSDE Controller

In Han & Hu (2019), the FBSDE system in (6) is solved by a neural network where each agent has a Deep Fictitious Play FBSDE Controller (DFP-FBSDE) to generate the policy.

The neural network architecture is described as Fig.1, where the backbone is a unique fully connected neural network followed by a BatchNorm module (Ioffe & Szegedy, 2015) (hereafter we shorthand this backbone as FC) at each timestep. Since Han & Hu (2019) do not apply feature extraction, it is equivalent to setting feature extractor as identity mapping: $\hat{\boldsymbol{F}}_t = \boldsymbol{I} \cdot \boldsymbol{X}_t$. The output of the network is Z component in functions (6). The use of a neural network allows for the BSDE to be propagated forward (red path) from the initial condition predicted by another FC along with the FSDE (green path).

The training set of DFP-FBSDE contains randomly selected initial states $\boldsymbol{x}_0$, and the training labels are defined as the value function at terminal time Y^i_T(6) which depends on random variable $\boldsymbol{X}_T$. The training loss (24) is defined as the mean square error between the predicted value $\hat{Y}^i_T$ propagated by BSDE and the true terminal value. The metrics used on training/test data are summarized in Appendix C.

By collecting the FBSDEs from all agents, the final DFP-FBSDE framework will be formed and the architecture can be found in Appendix H.4. Furthermore, Pereira et al. (2019) extends Deep FBSDE to robotics applications and Han & Hu (2019) applys Deep FBSDE for searching Nash-Equlibrium with convergence proof (Han & Long, 2020).

3. SDFP-FBSDE

In this section we propose a novel SDFP-FBSDE algorithm. Motivated by the convergence analysis of DFP-FBSDE, we introduce an Importance Sampling term to facilitate exploration and accelerate convergence. Scalability is further increased by Invariant Layer Embeddings, leading to an order of magnitude lower time and memory complexity.

3.1. Motivation of Importance Sampling (IS)

We first analyze FBSDE (6) with drift term by extending previous work (Han & Long, 2020) where the drift term is ignored. The analysis focuses on one representative agent. Therefore, the agent index is omitted in this subsection.

Assumption 1. *There exists a measurable function $\phi : [0,T] \times \mathcal{X} \rightarrow \mathcal{X}$ and $\Gamma : [0,T] \times \mathcal{U} \rightarrow \mathcal{X}$ so that $\Sigma(t,\boldsymbol{X})\phi(t,\boldsymbol{X}) = f(t,\boldsymbol{X})$, and $\Sigma(t,\boldsymbol{X})\Gamma(t,\boldsymbol{X}) = G(t,\boldsymbol{X})$.*

Assumption 2. *For a general FBSDE system,*

$$\begin{aligned} \boldsymbol{X}^{t,\boldsymbol{x}}_T &= \boldsymbol{x} + \int_t^T \mu_s \mathrm{d}s + \int_t^T \Sigma_s \mathrm{d}\boldsymbol{W}_s && \textit{(FSDE)}, \\ Y^{T,\boldsymbol{x}}_t &= g(\boldsymbol{X}^{t,\boldsymbol{x}}_T) - \int_t^T H_s \mathrm{d}s + \int_t^T Z_s \mathrm{d}\boldsymbol{W}_s && \textit{(BSDE)}, \end{aligned} \tag{7}$$

The drift function μ_s, H_s and diffusion function Σ in (7), terminal objective function $g(\cdot)$, and control function $U(\cdot)$ satisfy Lipschitz continuous properties with Lipschitz constants $\mu_x, \mu_u, \Sigma_x, H_x, H_z, g_x, u_x$ respectively. The detailed and formal description can be found in Appendix D.1.

Lemma 1. *Denote $(\boldsymbol{X}^{t,\boldsymbol{x}}_s, Y^{t,\boldsymbol{x}}_s, Z^{t,\boldsymbol{x}}_s)_{t\leq s\leq T}$ as the solution for the FBSDE system (7) satisfying assumptions 1 and 2. Denote the difference of Y component at two different states $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ as:*

$$\begin{aligned} &\delta\boldsymbol{X}_t = \boldsymbol{X}^{t_0,\boldsymbol{x}_1}_t - \boldsymbol{X}^{t_0,\boldsymbol{x}_2}_t, \delta Y_t = Y^{t_0,\boldsymbol{x}_1}_t - Y^{t_0,\boldsymbol{x}_2}_t. \\ &\delta Z_t = Z^{t_0,\boldsymbol{x}_1}_t - Z^{t_0,\boldsymbol{x}_2}_t \end{aligned} \tag{8}$$

Then we can have:

$$|\delta Y_T|^2 \leq L_1|\boldsymbol{x}_1 - \boldsymbol{x}_2|^2, |\delta Y_{t_0}|^2 \leq L_2|\boldsymbol{x}_1 - \boldsymbol{x}_2|^2, \tag{9}$$

Where L_1 and L_2 are defined as:

$$\begin{aligned} L_1 &= g_x e^{\xi T} \\ L_2 &= e^{H_z(T-t_0)}\left[g_x e^{\xi(T-t_0)} + H_x \frac{e^{\xi(T-t_0)}-1}{\Sigma_x^{-1}H_z^{-1}}\right], \\ \xi &= I + \mu_x + \mu_u u_x + \Sigma_x, \end{aligned} \tag{10}$$

Following arguments in (Ma et al., 2002), one further has,

$$||Z_t||^2_S \leq ||\Sigma||^2_S ||\nabla_x Y_t||^2_S \leq M_\Sigma L_2 \tag{11}$$

Where $\mu_x, \mu_u, u_x, \Sigma_x, H_x, H_z, g_x$ are Lipschitz constants defined in Assumption.2. M_Σ is the upper bound of Σ. The proof can be found in Appendix D.2.

Lemma 1 bridges the connection between the states and their corresponding value functions. Note that the training labels Y_T are not fixed since it is a function of $\boldsymbol{X}_T$ driven by stochastic diffusion $\boldsymbol{W}_t$ which is distinct at each training iteration. Furthermore, the Lemma 1 suggests that scope of the training dataset $[\boldsymbol{x}_0, Y^i_T]$ and initial values functions Y_{t_0} can be controlled by Lipschtiz constants $\mu_x, \mu_u, H_x, H_z, u_x$, since Lipschtiz constants Σ_x, g_x are determined as long as the stochastic dynamics and terminal loss function are defined.

Increasing data complexity is crucial element for improving model performance in RL and supervised learning. In RL, agents are encouraged to explore more states by adding the entropy term in cost function (Haarnoja et al., 2018) or adding artificial stochasticity in observed states (Fortunato et al., 2017) to gain complex training data for the learning of action-value function. In supervised learning, connections between deep neural networks' generalization and complexity of dataset also has attracted considerable attention as a principled tool to explain deep learning(DL) (Schmidt et al., 2018),(Rozen et al., 2019). In practice, Data-Augmentation is an efficient approach adopted to improve the generalization performance of a deep learning model (Cubuk et al., 2018),(Fawzi et al., 2016). Motivated by the success of previous work in DL and RL community, we would like to modify the FBSDE system (6) to improve the complexity of training label δY_T and δY_{t_0} (9) by encouraging the exploration of agents while solving the same HJB PDE (5).

Theorem 1. *(Bender & Moseler, 2010): Let $(X^{i,t,x}_s, Y^{i,t,x}_s, Z^{i,s,x}_s)$ be the solution of the FBSDE system (6) for ith agent, and let $K_s : [0,T] \times \Omega \rightarrow \mathbb{R}^{n_x}$ be any bounded and square integrable process for ith agent. Consider the forward process whose drift term is modified by K_s*

$$\mathrm{d}\tilde{\boldsymbol{X}}_s = [\mu_s + \Sigma K_s]\mathrm{d}s + \Sigma \mathrm{d}\boldsymbol{W}_s, \; \tilde{\boldsymbol{X}}_t = \boldsymbol{x}_t, \tag{12}$$

along with the corresponding BSDE

$$\mathrm{d}\tilde{Y}^i_s = [-h^i_s + \tilde{Z}_s K_s]\mathrm{d}s + \tilde{Z}^i_s \mathrm{d}W_s, \; Y^i_T = g(\boldsymbol{X}_T). \tag{13}$$

Here we denote $(\tilde{X}_s^{i,t,x}, \tilde{Y}_s^{i,t,x}, \tilde{Z}_s^{i,s,x})$ as the solution for modified FBSDE system (12,13). For all $s \in [t, T]$, $(\tilde{X}_s^{i,t,x}, \tilde{Y}_s^{i,t,x}, \tilde{Z}_s^{i,s,x}) = (X_s^{i,t,x}, Y_s^{i,t,x}, Z_s^{i,s,x})$ a.s. If $(\tilde{Y}_s^{i,t,x}, \tilde{Z}_s^{i,s,x})$ are defined as $(\tilde{V}^i, \Sigma^{\mathrm{T}}\tilde{V}_x^i)$ with $\tilde{V}^i$ being the solution to 5, then $V^i \equiv \tilde{V}^i$ a.e.

Theorem 1 is also known as IS in the literature (Exarchos & Theodorou, 2018). By modifying the FSDE and BSDE simultaneously, the HJB PDE will be solved identically almost everywhere. We further bound $|\delta Y_T|^2$, $|\delta Y_{t_0}|^2$ and $||Z_t||_S^2$ with constant $\tilde{L}_1$ and $\tilde{L}_2$ when IS is applied in the FBSDE system in Lemma 2. The detailed description and proof can be found in Appendix D.3.

Theorem 2. *Denote $(\tilde{\boldsymbol{X}}_s^{t,\boldsymbol{x}}, \tilde{Y}_s^{t,\boldsymbol{x}}, \tilde{Z}_s^{t,\boldsymbol{x}})_{t \le s \le T}$ is the solution for the FBSDE system with IS (12,13), and $(\boldsymbol{X}_s^{t,\boldsymbol{x}}, Y_s^{t,\boldsymbol{x}}, Z_s^{t,\boldsymbol{x}})_{t \le s \le T}$ is the solution for the FBSDE system (7). and they satisfy the assumption (1,2). Then given the identical training data $\boldsymbol{X}_0$ for FBSDE w/ and w/o IS, combining Lemma 1 and Lemma 2, one can have,*

$$\begin{aligned} \max|\delta Y_T|^2 &\le \max|\delta \tilde{Y}_T|^2, \\ \max|\delta Y_0|^2 &\le \max|\delta \tilde{Y}_0|^2 \end{aligned} \tag{14}$$

Theorem 2 hints that, given identical training data , the model equipped with IS could obtain more diverse and complex training dataset, which means the generalization performance can potentially be enhanced. Fig.2 testifies the performance difference for the model with and without IS over identical dataset whose size is limited on the inter-bank problem (§4.1). The metric is chosen as the evaluation loss (24) and Relative Square Error (RSE) loss (23). We can find IS improve the performance for both backbones. In order to highlight the influence of IS term ΣK_s and $\tilde{Z} K_s$, we detached/stopped the gradient of them. In other words, the IS term will be treated as a constant during training phase, and the rapid convergence is not the result of the additional gradient path introduced by IS.

As a value-based optimal decision algorithm, a natural question will be how the policy performs based on the learnt value function. Proposition 1 demonstrates the convergence rate of Deep FBSDE policies in fictitious play.

Proposition 1. *(Han & Long, 2020) Denote $\delta \boldsymbol{u}_t^m = \boldsymbol{u}_t^m - \boldsymbol{u}_t^*$ as difference between the policy obtained at mth stage of fictitious play at time step t and the optimal policy, then,*

$$\int_0^T \mathbb{E}|\delta \boldsymbol{u}_t^{m+1}|^2 \le \eta(\lambda) \int_0^T \mathbb{E}|\delta \boldsymbol{u}_t^m|^2 \tag{15}$$

Where $\eta(\lambda)$ is the convergence rate.

The theoretical analysis of proposition 1 uncovers the convergence property of fictitious play of FBSDE system. However, choice of constant λ is tricky in order to obtain strict converge: $0 \le \eta(\lambda) < 1$. We demonstrate the numerical value of $\frac{1}{\eta(\lambda)}$ for Deep FBSDE model w/ and w/o IS with FC backbone (Han et al., 2018) and LSTM backbone (Pereira et al., 2019) in Fig.3. For the same backbone, IS accelerates the convergence of the policy significantly.

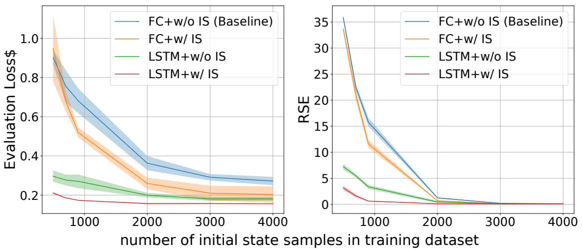

Figure 2. Performance difference of DFP-FBSDE w/ and w/o importance samping over limited training dataset. The simulation is executed on 100 agents inter-bank game.

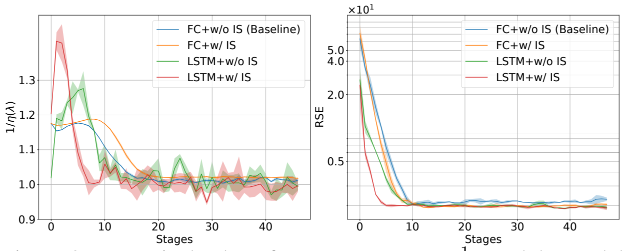

Figure 3. Numerical value of convergence rate $\frac{1}{\eta(\lambda)}$ and the model performance evalued by RSE. The simulation is executed on 100 agents inter-bank game.

3.2. Mitigating Curse of Many Agents

Scalability is a crucial criterion of RL. In DFP-FBSDE, each agent is equipped with policy computed by a distinct neural network. As the number of agents increases, the number of neural network copies would increase correspondingly. Meanwhile, the size of each neural network should be enlarged to gain enough capacity to capture the representation of many agents, leading to the infamous curse of many agents (Wang et al., 2020). This limits the scalability of prior works. However, one can mitigate the curse of many agents in this case by taking advantage of the symmetric game setup. We summarize merits of symmetric game as:

1. Since all agents have the same dynamics and cost function, only one copy of the network is needed. The strategies of other agents can be inferred by applying the same network.
2. Thanks to the symmetric property, we can apply the IL embedding (Zaheer et al., 2017) to extract invariant features.

Sharing one network: Here we introduce some techniques to improve the time and memory complexity. When sharing same network, we can query the policy of other agents simply by inferring own freeze network at $m - 1$ stage. It's

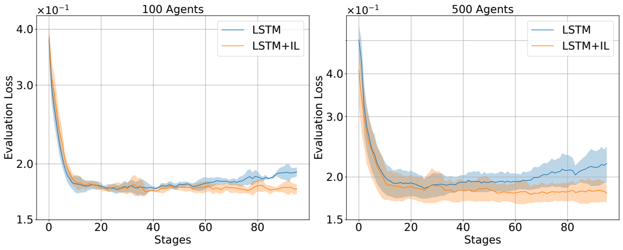

Figure 4. Comparison between DFP-FBSDE w/ and w/o IL. The backbone is chosen as LSTM. The simulation is inter-bank game.

important to note that querying the policy of other agents should not introduce additional gradient paths, which significantly reduces the memory complexity.

When querying other agents' strategy, one can either iterate through each agent or feed all agents' states to the network in a batch. Here we denote them as iterative scheme and batch scheme respectively. The latter approach reduces the time complexity by adopting the parallel nature of modern GPU but requires $\mathcal{O}(N^2)$ memory complexity rather than $\mathcal{O}(N)$ for the first approach.

Invariant Layers (IL): A comprehensive introduction of invariant network can be found in (Zaheer et al., 2017). Later applications in robotics (Shi et al., 2020) and RL (Liu et al., 2020), (Wang et al., 2020) verify the success of this technique. The memory complexity can be further reduced with an IL embedding (Zaheer et al., 2017) in our work. The IL utilizes an averaging function along with the features in the same set (known as feature averaging) to render the network invariant to permutation of agents. We apply the IL on $\boldsymbol{X}_{-i}$ which is insensitive to permutation and concatenate the resulting features to the features extracted from X_i. However, vanilla IL embedding does not reduce the memory complexity, because the dimension of feature space is commonly high even though the dimension of averaged feature is small. Thanks to the symmetric problem setup, one can apply a technique to reduce the IL memory complexity form $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$. A detailed introduction to the IL and our implementation techniques can be found in Appendix E.

Furthermore, the invariant representation, especially for feature averaging, can increase the performance of deep learning model theoretically (Lyle et al., 2020) when features have the invariant property. Learning the invariant representation for $\boldsymbol{X}_{-i}$ will become harder when the number of agents increases. This is illustrated in fig.4 where the DFP-FBSDE model equipped with IL can handle the complex invariant representation, and the performance gap becomes even larger when the number of agents increase.

3.3. Algorithm

Following the analysis in §3.1 and §3.2, here we introduce the SDFP-FBSDE algorithm which is scalable up to 3000 agents and has appreciable time and memory complexity by integrating IS and IL. We empirically verify the time and memory complexity shown in Fig.5.

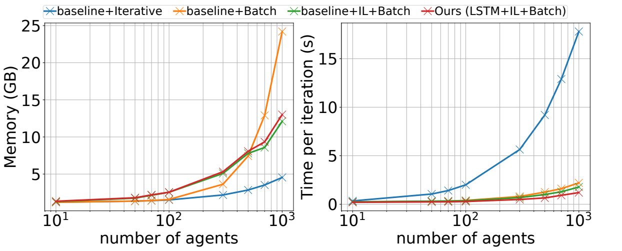

Figure 5. Time and memory complexity comparison between batch, iterate and IL+batch implementations. Time complexity is measured by per-iteration time. IL stands for invariant layer.

The neural network structure is same as shown in Fig.1. However, the feature extractor module (the blue boxes) will be replaced by an IL whose details can be found in Appendix E.2. Meanwhile, the forward and backward process of $\boldsymbol{X}$ and V_t^i will be modified according to the Theorem 1. Inspired by previous work (Theodorou et al., 2010), the IS term ($K_s = \Gamma \bar{\boldsymbol{U}}^i$) term is selected as the control calculated from previous run of the algorithm. LSTM will be chosen as the backbone in our algorithm from experimental result. Notably, our algorithm will also improve the performance for the framework with FC backbone as used in (Han & Hu, 2019), and the comparison will be elaborated in the simulation section. The full algorithm can be found in Appendix K.

4. Simulation

In this section, we demonstrate the capability of SFDP-FBSDE on two different systems in simulation. We first apply the framework to an inter-bank lending/borrowing problem, which is a classical multi-player non-cooperative game with an analytical solution. We compare against both the analytical solution and prior work (Han & Hu, 2019). We also apply the framework to a variation of the problem for which no analytical solution exists. Finally, we showcase the general applicability of our framework in an autonomous racing problem in belief space and discuss how BSDE plays an importance role in a SDG. All experiment configurations can be found in Appendix F. we plot the results of 3 repeated runs with different seeds with the line and shaded region showing the mean and mean±standard deviation respectively. The hyperparameters and dynamics coefficients used in the inter-bank experiments are the same as (Han & Hu, 2019) unless otherwise noted. Technical differences between Baseline (Han & Hu, 2019) and SDFP-FBSDE are shown in Table.4. The hardware used to run all simulations is included in Appendix J.

ALGO	BACKBONE	BATCH	IL	IS
BASELINE	FC+BN	✗	✗	✗
SDFP	LSTM	✓	✓	✓

Table 2. Technical differences between baseline (Han & Hu, 2019) and SDFP. Batch stands for the batch scheme.

FRAMEWORK	RSE	TIME (HR)
BASELINE	0.0423	2.23
SFDP	0.0105	1.55

Table 3. Comparison with Han & Hu (2019) on the 10-agent inter-bank problem.

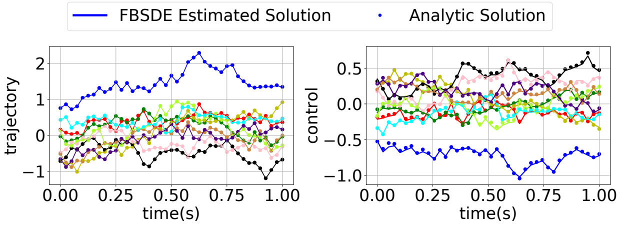

Figure 6. Comparison of SDFP and analytical solution for the inter-bank problem. Both the state (*left*) and control (*right*) trajectories are aligned with the analytical solution (represented by dots).

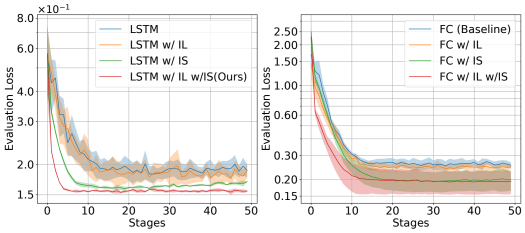

Figure 7. Ablation experiments on LSTM and FC backbone.

4.1. Inter-bank lending/borrowing problem

We first consider an inter-bank lending/ borrowing model (Carmona et al., 2013) where the dynamics of the log-monetary reserves of N banks is

$$
\begin{aligned}
\mathrm{d}X_t^i &= \left[a(\bar{X}_t - X_t^i) + U_t^i\right]\mathrm{d}t \\
&\quad + \sigma(\rho \mathrm{d}W_t^0 + \sqrt{1-\rho^2}\mathrm{d}W_t^i), \\
\bar{X}_t &= \frac{1}{N}\sum_{i=1}^{N} X_t^i, i \in \mathbb{I}.
\end{aligned} \tag{16}
$$

The state $X_t^i \in \mathbb{R}$ denotes the log-monetary reserve of bank i at time $t > 0$. The control u_t^i denotes the cash flow to/from a central bank, where as $a(\bar{X} - X_t^i)$ denotes the lending/borrowing rate of bank i from all other banks. The system is driven by N independent standard Brownian motion W_t^i, which denotes the idiosyncratic noise, and a common noise W_t^0. The cost function has the form,

$$
C_t^i(\boldsymbol{X}, U_i; \boldsymbol{U}_{-i}) = \frac{1}{2}U_i^2 - qU_i(\bar{X} - X_i) + \frac{\epsilon}{2}(\bar{X} - X_i)^2. \tag{17}
$$

The constants q, ϵ can be found in Appendix F. The derivation of the FBSDEs and analytical solution are in Appendix E. We compare the result of our algorithm on a 10-agent problem with analytical solution and the baseline with the same hyperparameters. Fig. 6 shows the performance of SDFP-FBSDE compared with analytical solution. The state and control trajectories outputted by SDFP-FBSDE solution are aligned closely with the analytical solution. Table 3 shows the numerical performance compared against prior work by Han & Hu (2019). Our method outperforms by RSE (23) and computation wall time.

Ablation Experiments: In order to verify the effect of combination of IL and IS introduced in §3.1 and §3.2, we perform the ablation experiments on the 500 agents inter-bank problem. Fig.7 shows the model equipped with IS and IL would obtain superior convergence rate and better results on evaluation set regardless of the choice of backbone.

High Dimension Experiments: Convinced by the ablation experiments, we conducted comparison between baseline and our proposed algorithm on different numbers of agents (up to 3000). Because of the limitation of graphical memory of hardware, the maximum number of agents for baseline is 700. The result is demonstrated in Fig.8. SDFP-FBSDE outperforms the baseline by cumulative total cost (25) and RSE (23). As been shown in FBSDE theory (6), the augmented control is characterized by $\boldsymbol{U}_{0,*}$. This means more agents will lead to more actuator and better explorations hence the RSE loss would decrease when number of agent is smaller than 300. When the number of agents keeps growing, the network would have difficulty of learning many agents representation, thus leading to the increase of RSE. Our method mitigates the curse of many agent, and postpones the RSE turning point to 1000 agents. Notably, SDFP-FBSDE has only marginally larger RSE loss at 3000 agents compared with baseline at 300 agents. The left subplot in Fig.8 shows our algorithm found more reasonable control sequence than baseline. The margin between SDFP-FBSDE and baseline is even larger on evaluation loss (24). Since this plot is not informative because of the large difference of scale, we defer it to Appendix H.1.

Superlinear Experiments: We also consider a variant of dynamics in equation (16):

$$
\begin{aligned}
\mathrm{d}X_t^i &= \left[a(\bar{X} - X_t^i)^3 + U_t^i\right]\mathrm{d}t \\
&\quad + \sigma(\rho \mathrm{d}W_t^0 + \sqrt{1-\rho^2}\mathrm{d}W_t^i), \\
\bar{X}_t &= \frac{1}{N}\sum_{i=1}^{N} X_t^i, i \in \mathbb{I}.
\end{aligned} \tag{18}
$$

Due to the nonlinearity in the drift term, analytical solution or simple numerical representation of the Nash equilibrium

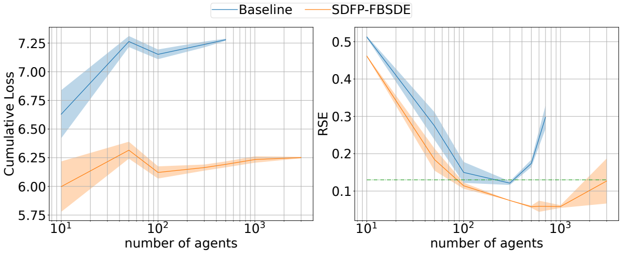

Figure 8. Comparison of SDFP-FBSDE and Baseline for inter-bank problem with different number of agents evaluated on cumulative loss(25) and RSE(23).

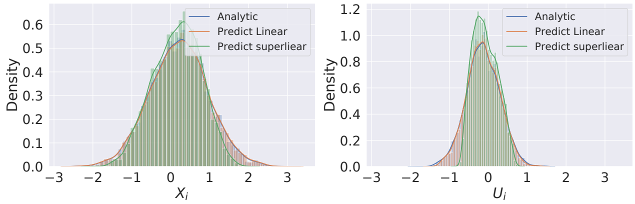

Figure 9. Terminal time step state $\boldsymbol{X}$ and control $\boldsymbol{U}$ distribution of ith agent for linear and superlinear dynamics. The analytic and predicted distribution for linear case are very similar.

does not exist (Han & Hu, 2019). The drift rate a is set to 1.0 to compensate for the vanishing drift term caused by super-linearity. Heuristically, the distribution of control and state should be more concentrated than that of the linear dynamics. We compare the state and control of agent i at terminal time against analytical solution and Deep FBSDE solution of the linear dynamics with the same coefficients. Fig. 9 is generated by evaluating the trained Deep FBSDE model with a batch size of 50000. Fig.9 shows that the solution from superlinear dynamics is more concentrated as expected. The terminal control distribution verifies that the superlinear drift term pushes the state back to the average faster than linear dynamics and thus requires less control effort. Since the numerical solution is not available, we compare the cumulative loss (25) loss and evaluation loss (24) between baseline and our algorithm (see Appendix H.2).

4.2. Extension to Partially Observed Game in Robotics

In this section, we demonstrate the general applicability of our framework on an autonomous racing example in belief space, and show how BSDE influences the game when competition is triggered experimentally. we consider a autonomous racing problem with race car dynamics

$$\dot{X}_i = [v\cos\theta, v\sin\theta, u_{\text{acc}} - c_{\text{drag}}v, u_{\text{steer}}v/L]^{\text{T}} \tag{19}$$

where $X_i = [x, y, v, \theta]^{\text{T}}$ represent the x, y position, forward velocity and heading of the ith vehicle. Here we assume $x, y, v, u_{\text{acc}} \in \mathbb{R}, u_{\text{steer}} \in [-1, 1]$. The goal of each player is to drive faster than the opponent, stay on the track and avoid collision. The running cost C^i is defined in Appendix I.2. During the game, players have access to partially observed

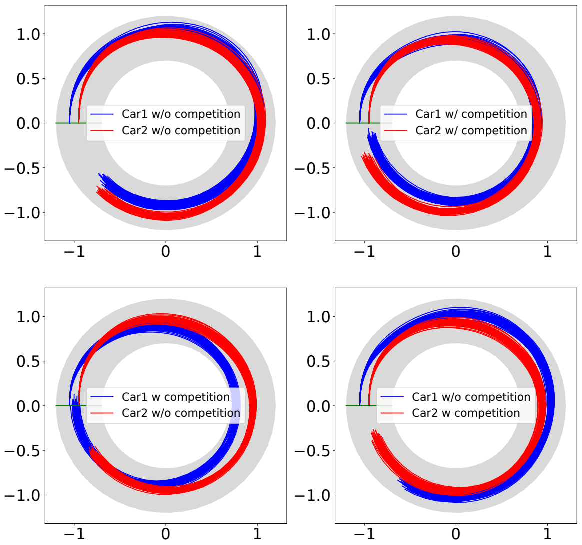

Figure 10. The plots contains 64 trials of racing. The performance varies with respect to the competition loss.

global augmented states estimated by an Extended Kalman Filter (EKF) and opponent's controller in the past games. Additionally, we assume that stochasticity enters the system through the control channels and have a continuous-time noisy observation model with full-state observation. The FBSDE derivation of belief space stochastic dynamics is included in Appendix I. We focus on the 2-car racing scenario for the interpretability of results, since no analytical solution exists for the problem. With the 2-car setup, interesting behaviors arise when the cost function of each car is altered. Since all the trials complete 1 lap, here we only show the first 8 seconds' result for neatness.

Non-cooperative and Non-competitive Case We first test the capability of our framework on a partially observed learning problem where all states are estimated with observation noise and additive noise in the system. Both cars can stay in the track as expected. Since there is no competition between the two cars, they demonstrate similar behaviors. The plot of trajectories with 64 trails of games can be found on the top left of Fig.10. The EKF posterior estimation is not drawn in Fig.10 for neatness. For single trails plot with EKF posterior can be found in Appendix H.3.

Competitive Case When we add competition loss (see Appendix I.2) on the cars, the propagation of BSDE (6) will be modified accordingly. Therefore, the racing game demonstrates interesting properties. When competition loss is applied on both cars, both of them try to cut the corner in order to occupy the leading position as shown on the top right of Fig. 10, and both of them travel longer distance compared with the ones do not have competitive loss. When competition loss is present in only one of the two cars, then the one with competition loss dominates the game as shown in the botton subplots of Fig.10.

5. Conclusion

In this paper, we first extend the theoretical analysis from (Han & Long, 2020) and introduce importance sampling to improve the sample efficiency and convergence rate. To further push our work to handle larger number of agents with appreciable time and memory complexity, batch query scheme and invariant layer implementation are proposed. The scalability of our algorithm, along with a detailed sensitivity analysis, is demonstrated in an inter-bank borrowing/lending example. Our framework achieves better performance in different metrics and scales to significantly higher dimensions than the state-of-the-art. The general applicability of our framework is showcased on a belief space racing problem in the partially observed scenario.

Acknowledgement

This research is supported under ARO W911NF2010151. We would also like to thank Professor Matthieu Bloch and Guan-Horng Liu for the helpful discussions.

References

Bender, C. and Moseler, T. Importance sampling for backward sdes. *Stochastic Analysis and Applications*, 28(2): 226–253, 2010.

Brown, G. W. Iterative solution of games by fictitious play. *Activity analysis of production and allocation*, 13(1):374–376, 1951.

Buckdahn, R. and Li, J. Stochastic differential games and viscosity solutions of hamilton–jacobi–bellman–isaacs equations. *SIAM Journal on Control and Optimization*, 47(1):444–475, 2008.

Carmona, R. and Delarue, F. Probabilistic analysis of mean-field games. *SIAM Journal on Control and Optimization*, 51(4):2705–2734, 2013.

Carmona, R., Fouque, J.-P., and Sun, L.-H. Mean field games and systemic risk. *Available at SSRN 2307814*, 2013.

Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., and Le, Q. V. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018.

Duncan, T. and Pasik-Duncan, B. Some stochastic differential games with state dependent noise. *54th IEEE Conference on Decision and Control, Osaka, Japan*, December 15-18, 2015.

Exarchos, I. and Theodorou, E. A. Stochastic optimal control via forward and backward stochastic differential equations and importance sampling. *Automatica*, 87:159–165, 2018.

Exarchos, I., Theodorou, E., and Tsiotras, P. Stochastic differential games: A sampling approach via FBSDEs. *Dynamic Games and Applications*, 9(2):486–505, 2019.

Fawzi, A., Samulowitz, H., Turaga, D., and Frossard, P. Adaptive data augmentation for image classification. In *2016 IEEE International Conference on Image Processing (ICIP)*, pp. 3688–3692. Ieee, 2016.

Fortunato, M., Azar, M. G., Piot, B., Menick, J., Osband, I., Graves, A., Mnih, V., Munos, R., Hassabis, D., Pietquin, O., et al. Noisy networks for exploration. *arXiv preprint arXiv:1706.10295*, 2017.

Greif, C. Numerical methods for hamilton-jacobi-bellman equations. 2017.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pp. 1861–1870. PMLR, 2018.

Han, J. and Hu, R. Deep fictitious play for finding markovian nash equilibrium in multi-agent games. *arXiv preprint arXiv:1912.01809*, 2019.

Han, J. and Long, J. Convergence of the deep bsde method for coupled fbsdes. *Probability, Uncertainty and Quantitative Risk*, 5(1):1–33, 2020.

Han, J., Jentzen, A., and Weinan, E. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018.

Harmon, M. E., Baird III, L. C., and Klopf, A. H. Reinforcement learning applied to a differential game. *Adaptive behavior*, 4(1):3–28, 1995.

Hu, R. Deep fictitious play for stochastic differential games. *arXiv preprint arXiv:1903.09376*, 2019.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. PMLR, 2015.

Isaacs, R. *Differential Games: A Mathematical Theory with Applications to Warfare and Pursuit, Control and Optimization*. New York: Willey, 1965.