Dynamic Game Theoretic Neural Optimizer

Guan-Horng Liu [1 2] **Tianrong Chen** [3] **Evangelos A. Theodorou** [1 2]

Abstract

The connection between training deep neural networks (DNNs) and optimal control theory (OCT) has attracted considerable attention as a principled tool of algorithmic design. Despite few attempts being made, they have been limited to architectures where the layer propagation resembles a Markovian dynamical system. This casts doubts on their flexibility to modern networks that heavily rely on non-Markovian dependencies between layers (*e.g.* skip connections in residual networks). In this work, we propose a novel dynamic game perspective by viewing each *layer* as a *player* in a dynamic game characterized by the DNN itself. Through this lens, different classes of optimizers can be seen as matching different types of Nash equilibria, depending on the implicit information structure of each (p)layer. The resulting method, called Dynamic Game Theoretic Neural Optimizer (DGNOpt), not only generalizes OCT-inspired optimizers to richer network class; it also motivates a new training principle by solving a multi-player cooperative game. DGNOpt shows convergence improvements over existing methods on image classification datasets with residual and inception networks. Our work marries strengths from both OCT and game theory, paving ways to new algorithmic opportunities from robust optimal control and bandit-based optimization.

1. Introduction

Attempts from different disciplines to provide a fundamental understanding of deep learning have advanced rapidly in recent years. Among those, interpretation of DNNs as discrete-time nonlinear dynamical systems has received tremendous focus. By viewing each layer as a distinct time step, it motivates principled analysis from numerical equations (Weinan, 2017; Lu et al., 2017) to physics (Greydanus et al., 2019). For instance, casting residual networks (He et al., 2016) as a discretization of ordinary differential equations enables fundamental reasoning on the loss landscape (Lu et al., 2020) and inspires new architectures with numerical stability or continuous limit (Chang et al., 2018; Chen et al., 2018).

Terminology Mapping

🟩	Deep Neural Net	Multi-Player Dynamic Game
🟧	Training Optimizer	Solving Nash Equilibria
- - -	Computation Order	Stage Sequence
▭	Layer Module	Player
(L, θ)	Objective & Parameters	Payoff & Actions

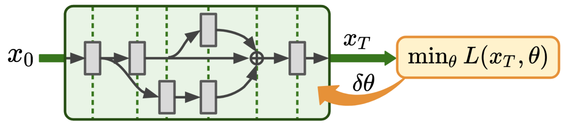


Figure 1. Dynamic game perspective of generic DNN training process, where we treat layer modules as players in a dynamic game and solve for the related Nash equilibria (Best viewed in color).

This dynamical system viewpoint also motivates control-theoretic analysis, which further recasts the network weight as control. With that, the training process can be viewed as an optimal control problem, as both methodologies aim to optimize some variables (weights *v.s.* controls) subjected to the chain structure (network *v.s.* dynamical system). This connection has lead to theoretical characterization of the learning process (Weinan et al., 2018; Hu et al., 2019; Liu & Theodorou, 2019) and practical methods for hyper-parameter adaptation (Li et al., 2017b) or computational acceleration (Gunther et al., 2020; Zhang et al., 2019).

Development of algorithmic progress, however, remains relatively limited. This is because OCT-inspired training methods, by construction, are restricted to network class that resembles Markovian state-space models (Liu et al., 2021; Li & Hao, 2018; Li et al., 2017a). This raises questions of their flexibility and scalability to training modern architectures composed of complex dependencies between layers. It is unclear whether this interpretation of dynamical system and optimal control remains suitable, or how it should be adapted, under those cases.

In this work, we address the aforementioned issues using dynamic game theory, a discipline of interactive decision making (Yeung & Petrosjan, 2006) built upon optimal con-

[1]Center for Machine Learning [2]School of Aerospace Engineering [3]School of Electrical and Computer Engineering, Georgia Institute of Technology, USA. Correspondence to: Guan-Horng Liu <ghliu@gatech.edu>.

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

trol and game theory. Specifically, we propose to treat each layer as a player in a dynamic game connected through the network propagation. The additional dimension gained from multi-player allows us to generalize OCT-inspired methods to accept a much richer network class. Further, introducing game-theoretic analysis, *e.g. information structure*, provides a novel algorithmic connection between different classes of training methods from a Nash equilibria standpoint (Fig. 1).

Unlike prior game-related works, which typically cast the whole network as a player competing over training iteration (Goodfellow et al., 2014; Balduzzi et al., 2018), the (p)layers in our dynamic game interact along the network propagation. This naturally leads to a coalition game since all players share the same objective. The resulting cooperative training scheme urges the network to yield group optimality, or Pareto efficiency (Pardalos et al., 2008). As we will show through experiments, this improves convergence of training modern architectures, as richer information flows between layers to compute the updates. We name our method Dynamic Game Theoretic Neural Optimizer (**DGNOpt**).

Notably, casting the network as a realization of the game has appeared in analyzing the convergence of Back-propagation (Balduzzi, 2016) or contribution of neurons (Stier et al., 2018; Ghorbani & Zou, 2020). Our work instead focuses on developing game-theoretic training methods and how they can be connected to, or generalize, existing optimizers. In summary, we present the following contributions.

- We draw a novel algorithmic characterization from the Nash equilibria perspective by framing the training process as solving a multi-player dynamic game.
- We propose **DGNOpt**, a game-theoretic optimizer that generalizes OCT-inspired methods to richer network class and encourages cooperative updates among layers with an enlarged information structure.
- Our method achieves competitive results on image classification with residual and inception nets, enabling rich applications from robust control and bandit analysis.

2. Preliminaries

Notation: Given a real-valued function $\mathcal{F}_s$ indexed by $s \in \mathcal{S}$, we shorthand its derivatives evaluated on $(\boldsymbol{x}_s, \theta_s)$ as $\nabla_{\boldsymbol{x}_s}\mathcal{F}_s \equiv \mathcal{F}^s_{\boldsymbol{x}}$, $\nabla^2_{\boldsymbol{x}_s}\mathcal{F}_s \equiv \mathcal{F}^s_{\boldsymbol{x}\boldsymbol{x}}$, and $\nabla_{\boldsymbol{x}_s}\nabla_{\theta_s}\mathcal{F}_s \equiv \mathcal{F}^s_{\boldsymbol{x}\theta}$, etc. Throughout this work, we will preserve $n \in \{1, \cdots, N\}$ as the player index and $t \in \{0, 1, \cdots, T-1\}$ as the propagation order along the network, or equivalently the stage sequence of the game (see Fig. 1). We will abbreviate them as $n \in [N]$ and $t \in [T]$ for brevity. Composition of functions is denoted by $f(g(\cdot)) \equiv (f \circ g)(\cdot)$. We use $\dagger$, $\odot$ and $\otimes$ to denote pseudo inversion, Hadamard and Kronecker product. A complete notation table can be found in Appendix A.

2.1. Training Feedforward Nets with Optimal Control

Let the layer propagation rule in feedforward networks (*e.g.* fully-connected and convolution networks) with depth T be

$$\boldsymbol{z}_{t+1} = f_t(\boldsymbol{z}_t, \theta_t), \quad t \in [T], \tag{1}$$

where $\boldsymbol{z}_t$ and θ_t represent the vectorized hidden state and parameter at each layer t. For instance, $\theta_t := \mathrm{vec}([\boldsymbol{W}_t, \boldsymbol{b}_t])$ for a fully-connected layer, $f_t(\boldsymbol{z}_t, \theta_t) := \sigma(\boldsymbol{W}_t \boldsymbol{z}_t + \boldsymbol{b}_t)$, with nonlinear activation $\sigma(\cdot)$. Equation (1) can be interpreted as a discrete-time Markovian model propagating the state $\boldsymbol{z}_t$ with the tunable variable θ_t. With that, the training process, *i.e.* finding optimal parameters $\{\theta_t : t \in [T]\}$ for all layers, can be described by Optimal Control Programming (OCP),

$$\min_{\theta_t : t \in [T]} L := \left[\phi(\boldsymbol{z}_T) + \sum_{t=0}^{T-1} \ell_t(\theta_t)\right] \quad \textit{s.t. } (1). \tag{2}$$

The objective L consists of a loss ϕ incurred by the network prediction $\boldsymbol{z}_T$ (*e.g.* cross-entropy in classification) and the layer-wise regularization ℓ_t (*e.g.* weight decay). Despite (2) considers only one data point $\boldsymbol{z}_0$, it can be easily modified to accept batch training (Weinan et al., 2018). Hence, minimizing L sufficiently describes the training process.

Equation (2) provides an OCP characterization of training feedforward networks. First, the optimality principles to OCP, according to standard optimal control theory, typically involve solving some time-dependent objectives recursively from the terminal stage T. Previous works have shown that these backward processes relate closely to the computation of Back-propagation (Li et al., 2017a; Liu et al., 2021). Further, the parameter update of each layer, $\theta_t \leftarrow \theta_t - \delta\theta_t$, can be seen as solving these layer-wise OCP objectives with certain approximations. To ease the notational burden, we leave a thorough discussion in Appendix B. We stress that this intriguing connection is, however, limited to the particular network class described by (1).

2.2. Multi-Player Dynamic Game (MPDG)

Following the terminology in Yeung & Petrosjan (2006), in a discrete-time N-player dynamic game, Player n commits to the action $\theta_{t,n}$ at each stage t and seeks to minimize

$$L_n(\bar{\theta}_n; \bar{\theta}_{\neg n}) := \left[\phi_n(\boldsymbol{x}_T) + \sum_{t=0}^{T-1} \ell_{t,n}(\theta_{t,1}, \cdots, \theta_{t,N})\right]$$
$$\textit{s.t. } \boldsymbol{x}_{t+1} = F_t(\boldsymbol{x}_t, \theta_{t,1}, \cdots, \theta_{t,N}), \quad \theta_{t,n} \equiv \theta_{t,n}(\eta_{t,n}), \tag{3}$$

where $\bar{\theta}_n := \{\theta_{t,n} : t \in [T]\}$ denotes the action sequence for Player n throughout the game. The set $\neg n := \{i \in [N] : i \neq n\}$ includes all players except Player n. The key components that characterize an MPDG (3) are detailed as follows.

- **Shared dynamics F_t.** The stage-wise propagation rule for $\boldsymbol{x}_t$, affected by actions across all players $\theta_{t,n}, \forall n$.

- **Payoff/Cost L_n.** The objective for each player that accumulates the costs $(\phi_n, \ell_{t,n})$ incurred at each stage.
- **Information structure $\eta_{t,n}$.** A set of information available to Player n at t for making the decision $\theta_{t,n}$.

The Nash equilibria $\{(\bar\theta_1^*, \cdots, \bar\theta_N^*)\}$ to (3) is a set of stationary points where no player has the incentive to deviate from the decision. Mathematically, this can be described by

$$L_n(\bar\theta_n^*; \bar\theta_{\neg n}^*) \le L_n(\bar\theta_n; \bar\theta_{\neg n}^*), \quad \forall n \in [N],\ \forall \bar\theta_n \in \bar\Theta_n,$$

where $\bar\Theta_n$ denotes the set of admissible actions for Player n. When the players agree to cooperate upon an agreement on a set of strategies and a mechanism to distribute the payoff/cost, a cooperative game (CG) of (3) will be formed. CG requires additional optimality principles to be satisfied. This includes *(i)* group rationality (GR), which requires all players to optimize their joint objective,

$$L^* := \min_{\bar\theta_1, \cdots, \bar\theta_N} \textstyle\sum_{n=1}^N L_n(\bar\theta_n; \bar\theta_{\neg n}), \tag{4}$$

and *(ii)* individual rationality (IR), which requires the cost distributed to each player from L^* be at most the cost he/she will suffer if plays against others non-cooperatively. Intuitively, IR justifies the participation of each player in CG.

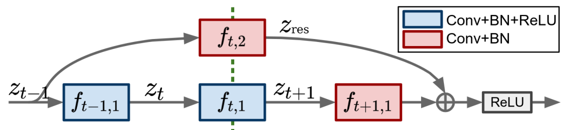
3. Dynamic Game Theoretic Perspective

3.1. Formulating DNNs as Dynamic Games

In this section, we draw a novel perspective between the three components in MPDG (3) and the training process of generic (*i.e.* non-Markovian) DNNs. Given a network composed of the layer modules $\{f_i(\cdot, \theta_i)\}$, where θ_i denotes the trainable parameters of layer f_i similar to (1), we treat each layer as a player in MPDG. The network can be converted into the form of F_t by indexing $i := (t, n)$, where t represents the sequential order from network input to prediction, and n denotes the index of layers aligned at t. Fig. 2 demonstrates such an example for a residual block. When the network propagation collapses from multiple paths to a single one, we can consider either duplicated players sharing the same path or dummy players with null action space. Hence, *w.l.o.g.* we will treat N as fixed over t. Notice that the assignment $i := (t, n)$ may not be unique. We will discuss its algorithmic implication later in §5.2.

Once the shared dynamics is constructed, the payoff L_n can be readily linked to the training objective. Since $\ell_{t,n}$ corresponds to the weight decay for layer $f_{t,n}$, it follows that $\ell_{t,n} := \ell_{t,n}(\theta_{t,n})$. Also, we will have $\phi_n := \phi$ whenever all (p)layers share the same task,[1] *e.g.* in classification. In short, the network architecture and training objective respectively characterize the structure of a dynamic game and its payoff.

[1] One of the examples for multi-task objective is the *auxiliary loss* used in deep reinforcement learning (Jaderberg et al., 2016).



$$x_{t+1} = F_t(x_t, \theta_{t,1}, \theta_{t,2}) \Leftrightarrow \begin{bmatrix} z_{t+1} \\ z_{\text{res}} \end{bmatrix} = \begin{bmatrix} f_{t,1}(z_t, \theta_{t,1}) \\ f_{t,2}(z_{t-1}, \theta_{t,2}) \end{bmatrix}$$

Figure 2. Example of representing a residual block as F_t. Note that $\boldsymbol{x}_t$ augments all hidden states across parallel paths.

3.2. Information Structure and Nash Optimality

We now turn into the role of information structure $\eta_{t,n}$. Standard game-theoretic analysis suggests that $\eta_{t,n}$ determines the type of Nash equilibria inherited in the MPDG (Petrosjan, 2005). Below we introduce several variants that are of our interests, starting from the one with the least structure.

Definition 1 (Open-loop Nash equilibrium (OLNE))**.** *Let $\eta_{t,n}^{O} := \{\boldsymbol{x}_0\}$ be the open-loop information structure. Then a set of action, $\{\theta_{t,n}^* : \forall t, n\}$, provides an OLNE to (3) if*

$$\theta_{t,n}^* = \arg\min_{\theta_{t,n}} H_{t,n}(\boldsymbol{x}_t, \boldsymbol{p}_{t+1,n}, \theta_{t,n}, \theta_{t,\neg n}^*), \tag{5}$$

where $\theta_{t,n}^* \equiv \theta_{t,n}^*(\eta_{t,n}^{O})$ and $H_{t,n} := \ell_{t,n} + F_t^{\mathsf{T}} \boldsymbol{p}_{t+1,n}$ is the Hamiltonian for Player n at stage t. The co-state $\boldsymbol{p}_{t,n}$ is a vector of the same size as $\boldsymbol{x}_t$ and can be simulated from the backward adjoint process, $(\boldsymbol{p}_{t,n}, \boldsymbol{p}_{T,n}) := (H_{\boldsymbol{x}}^{t,n}, \phi_{\boldsymbol{x}}^n)$.

The Hamiltonian objective $H_{t,n}$ varies for each (t, n) and depends on the proceeding stage via co-state $\boldsymbol{p}_{t+1,n}$. When $N{=}1$, (5) degenerates to the celebrated Pontryagin principle (Pontryagin et al., 1962), which provides the necessary condition to OCP (2). This motivates the following result.

Proposition 2. *Solving $\theta_{t,n}^* {=} \arg\min H_{t,n}$ with the iterative update, $\theta_{t,n} \leftarrow \theta_{t,n} - \boldsymbol{M}^\dagger H_\theta^{t,n}$, recovers the descent direction of standard training methods. Specifically, setting*

$$\boldsymbol{M} := \begin{cases} \boldsymbol{I} \\ \operatorname{diag}\left(\sqrt{H_\theta^{t,n} \odot H_\theta^{t,n}}\right) \\ H_\theta^{t,n} H_\theta^{t,n\,\mathsf{T}} \end{cases} \textit{yields} \begin{cases} \text{SGD} \\ \text{RMSprop} \\ \text{Gauss-Newton} \end{cases}.$$

Proposition 2 provides a similar OCP characterization (*c.f.* §2.1) except for a more generic network class represented by F_t. It also gives our first game-theoretic interpretation of DNN training: *standard training methods implicitly match an OLNE defined upon the network propagation.* The proof (see Appendix C) relies on constructing a set of co-state $\boldsymbol{p}_{t,n}$ such that $H_\theta^{t,n} \equiv \nabla_{\theta_{t,n}} H_{t,n}$ gives the exact gradient w.r.t. the parameter of layer $f_{t,n}$. The degenerate information structure $\eta_{t,n}^{O}$ implies that optimizers of this class utilize minimal knowledge available from the game (*i.e.* network) structure. This is in contrast to the following Nash equilibrium which relies on richer information.

Table 1. Dynamic game theoretic perspective of DNN training.

Nash Equilibria	Information Structure	Optimality Principle	Class of Optimizer
OLNE	$\eta^{\text{O}}_{t,n}$	$\min H_{t,n}$ in (5)	Baselines
FNE	$\eta^{\text{C}}_{t,n}$	$\min Q_{t,n}$ in (6)	DGNOpt (ours)
GR	$\eta^{\text{C-CG}}_{t,n}$	$\min P_t$ in (7)	DGNOpt (ours)

Definition 3 (Feedback Nash equilibrium (FNE)). *Let $\eta^{\text{C}}_{t,n}$:=$\{\boldsymbol{x}_s : s \leq t\}$ be the closed-loop information structure. Then a set of strategy, $\{\pi^*_{t,n} : \forall t, n\}$, is called a FNE to (3) if it is the solution to the Isaacs-Bellman equation (6).*

$$\begin{aligned} V_{t,n}(\boldsymbol{x}_t) &= \min_{\pi_{t,n}} Q_{t,n}(\boldsymbol{x}_t, \pi_{t,n}, \pi^*_{t,\neg n}), \\ V_{T,n} = \phi_n, &\quad \textit{where} \quad Q_{t,n} := \ell_{t,n} + V_{t+1,n} \circ F_t \end{aligned} \tag{6}$$

is the Isaacs-Bellman objective for Player n at stage t. Also, $\pi_{t,n} \equiv \theta_{t,n}(\boldsymbol{x}_t; \eta^{C}_{t,n})$ denotes any arbitrary mapping from $\boldsymbol{x}_t$ to $\theta_{t,n}$, conditioned on the closed-loop structure $\eta^{C}_{t,n}$.

For the closed-loop information structure $\eta^{\text{C}}_{t,n}$, each player has complete access to all preceding states until the current stage t. Consequently, it is preferable to solve for a state-dependent, *i.e. feedback*, strategy $\pi^*_{t,n}$ rather than a state-independent action $\theta^*_{t,n}$ as in OLNE. Similar to (5), the Isaacs-Bellman objective $Q_{t,n}$ is constructed for each (t, n), except now carrying a *function* $V_{t,n}(\cdot)$ backward from the terminal stage, rather than the co-state. This *value function* $V_{t,n}$ summarizes the optimal cost-to-go for Player n from each state $\boldsymbol{x}_t$, provided all afterward stages are minimized accordingly. When N=1, (6) collapses to standard Dynamic Programming (DP; Bellman (1954)), which is an alternative optimality principle parallel to the Pontryagin. For nontrivial N>1, solving the FNE optimality (6) provides a game-theoretic extension for previous DP-inspired training methods, *e.g.* Liu et al. (2021), to generic (*i.e.* non-Markovian) architectures.

3.3. Cooperative Game Optimality

Now, let us consider the CG formulation. When a cooperative agreement is reached, each player will be aware of how others react to the game. This can be mathematically expressed by the following information structures,

$$\eta^{\text{O-CG}}_{t,n} := \{\boldsymbol{x}_0, \theta^*_{t,\neg n}\} \text{ and } \eta^{\text{C-CG}}_{t,n} := \{\boldsymbol{x}_s, \pi^*_{t,\neg n} : s \leq t\},$$

which enlarge $\eta^{\text{O}}_{t,n}$ and $\eta^{\text{C}}_{t,n}$ with additional knowledge from other players, $\neg n$. We can characterize the inherited optimality principles similar to OLNE and FNE. Take $\eta^{\text{C-CG}}_{t,n}$ for instance, the joint optimization in GR (4) requires

Definition 4 (Cooperative feedback solution). *A set of strategy, $\{\pi^*_{t,n} : \forall t, n\}$, provides an optimal feedback solution to the joint optimization (4) if it solves*

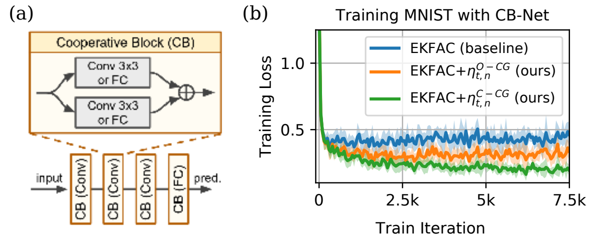


Figure 3. (a) The cooperative-block network and (b) its training performance on MNIST when the optimizer (EKFAC) is exposed to different information structure $\eta_{t,n}$. Note that by Proposition 2, the EKFAC baseline utilizes only the open-loop structure $\eta^{\text{O}}_{t,n}$.

$$\begin{aligned} W_t(\boldsymbol{x}_t) &= \min_{\pi_{t,n}:n\in[N]} P_t(\boldsymbol{x}_t, \pi_{t,1}, \cdots, \pi_{t,N}), \\ W_T = \textstyle\sum_{n=1}^N \phi_n, &\quad \textit{where } P_t := \textstyle\sum_{n=1}^N \ell_{t,n} + W_{t+1} \circ F_t \end{aligned} \tag{7}$$

is the "group-rational" Bellman objective at stage t. $\pi_{t,n} \equiv \theta_{t,n}(\boldsymbol{x}_t; \eta^{\text{C-CG}}_{t,n})$ denotes arbitrary mapping from $\boldsymbol{x}_t$ to $\theta_{t,n}$, conditioned on the cooperative closed-loop structure $\eta^{\text{C-CG}}_{t,n}$.

Notice that (7) is the GR extension of (6). Both optimality principles solve for a set of feedback strategies, except the former considers a joint objective P_t summing over all players. Hence, it is sufficient to carry a joint value function W_t backward. We leave the discussion on $\eta^{\text{O-CG}}_{t,n}$ in Appendix C.

To emphasize the importance of information structure, consider the architecture in Fig. 3a, where each pair of parallel layers shares the same input and output; hence the network resembles a two-player dynamic game with $F_t := f_{t,1}+f_{t,2}$. As shown in Fig. 3b, providing different information structures to the same optimizer, EKFAC (George et al., 2018), greatly affects the training. Having richer information tends to achieve better performance. Additionally, the fact that

$$\eta^{\text{O}}_{t,n} \subset \eta^{\text{C}}_{t,n} \subset \eta^{\text{C-CG}}_{t,n} \quad \text{and} \quad \eta^{\text{O}}_{t,n} \subset \eta^{\text{O-CG}}_{t,n} \subset \eta^{\text{C-CG}}_{t,n} \tag{8}$$

also implies an algorithmic connection between different classes of optimizers, which we will explore in §4.3.

Table 1 summarizes our game-theoretic analysis. Each information structure suggests its own Nash equilibria and optimality principle, which characterizes a distinct class of training methods. We already established the connection between baselines and $H_{t,n}$ in Proposition 2. In the next section, we will derive methods for solving $(Q_{t,n}, P_t)$.

4. Training DNN by Solving Dynamic Game

In this section, we derive a new *second-order* method, called Dynamic Game Theoretic Neural Optimizer (**DGNOpt**), that solves (6) and (7) as an alternative to training DNNs. While we will focus on the residual network for its popularity and algorithmic simplicity when deriving the analytic update, we stress that our methodology applies to other architectures. A full derivation is left in Appendix D.

4.1. Iterative Update via Linearization

Computing the game-theoretic objectives $(Q_{t,n}, P_t)$ requires knowing $(F_t, \ell_{t,n}, \phi_n)$. Despite they are well-defined from §3.1, carrying $Q_{t,n}$ or P_t as a stage-varying function is computationally impractical even on a relatively low-dimensional system (Tassa et al., 2012), let alone DNNs. Since the goal is to derive an incremental update given partial (*e.g.* mini-batch) data at each training iteration, we can consider solving them approximately via *linearization*.

Iterative methods via linearization have been widely used in real-time OCP (Pan et al., 2015; Tassa et al., 2014). We adopt a similar methodology for its computational efficiency and algorithmic connection to existing training methods (shown later). First, consider solving the FNE recursion (6) by $\pi^*_{t,n} \approx \arg\min Q_{t,n}$. We begin by performing second-order Taylor expansion on $Q_{t,n}$ w.r.t. to the variables that are *observable* to Player n at stage t according to $\eta^{\text{C}}_{t,n}$.

$$Q_{t,n} \approx \frac{1}{2}\begin{bmatrix} \mathbf{1} \\ \delta \boldsymbol{x}_t \\ \delta\theta_{t,n}\end{bmatrix}^{\mathsf{T}} \begin{bmatrix} Q_{t,n} & {Q^{t,n}_{\boldsymbol{x}}}^{\mathsf{T}} & {Q^{t,n}_{\theta}}^{\mathsf{T}} \\ Q^{t,n}_{\boldsymbol{x}} & Q^{t,n}_{\boldsymbol{x}\boldsymbol{x}} & {Q^{t,n}_{\theta\boldsymbol{x}}}^{\mathsf{T}} \\ Q^{t,n}_{\theta} & Q^{t,n}_{\theta\boldsymbol{x}} & Q^{t,n}_{\theta\theta}\end{bmatrix} \begin{bmatrix} \mathbf{1} \\ \delta \boldsymbol{x}_t \\ \delta\theta_{t,n}\end{bmatrix}$$

Note that $\delta\theta_{t,\neg n}$ does not appear in the above quadratic expansion since it is unobservable according to $\eta^{\text{C}}_{t,n}$. The derivatives of $Q_{t,n}$ w.r.t. different arguments follow standard chain rule (recall $Q_{t,n} := \ell_{t,n} + V_{t+1,n} \circ F_t$), with the dynamics linearized at some fixed point $(\boldsymbol{x}_t, \theta_{t,n})$, *e.g.*

$$Q^{t,n}_{\theta} = \ell^{t,n}_{\theta} + {F^t_\theta}^{\mathsf{T}} V^{t+1,n}_{\boldsymbol{x}}, \; Q^{t,n}_{\theta\boldsymbol{x}} = {F^t_\theta}^{\mathsf{T}} V^{t+1,n}_{\boldsymbol{x}\boldsymbol{x}} F^t_{\boldsymbol{x}}.$$

The analytic solution to this quadratic expression is given by $\pi^*_{t,n} = \theta_{t,n} - \delta\pi^*_{t,n}$, with the incremental update $\delta\pi^*_{t,n}$ being

$$\begin{gathered}\delta\pi^*_{t,n} = \mathbf{k}_{t,n} + \mathbf{K}_{t,n}\delta\boldsymbol{x}_t. \\ \mathbf{k}_{t,n} := (Q^{t,n}_{\theta\theta})^{\dagger} Q^{t,n}_{\theta} \quad \text{and} \quad \mathbf{K}_{t,n} := (Q^{t,n}_{\theta\theta})^{\dagger} Q^{t,n}_{\theta\boldsymbol{x}}\end{gathered} \tag{9}$$

are called the open and feedback gains. The superscript † denotes the pseudo inversion. Note that $\delta\pi^*_{t,n}$ is only *locally* optimal around the region where the quadratic expansion remains valid. Since $\boldsymbol{x}_t$ augments preceding hidden states (*e.g.* $\boldsymbol{x}_t := [\boldsymbol{z}_t, \boldsymbol{z}_{t-1}]^{\mathsf{T}}$ in Fig. 2), (9) implies that preceding hidden states contribute to the update via linear superposition.

Substituting the incremental update $\delta\pi^*_{t,n}$ back to the FNE recursion (6) yields the local expression of the value function $V_{t,n}$, which will be used to compute the preceding update $\delta\pi^*_{t-1,n}$. Since the computation depends on $V_{t,n}$ only through its local derivatives $V^{t,n}_{\boldsymbol{x}}$ and $V^{t,n}_{\boldsymbol{x}\boldsymbol{x}}$, it is sufficient to propagate these quantities rather than the function itself. The propagation formula is summarized in (10). This procedure (line 4-7 in Alg. 1) repeats recursively backward from the terminal to initial stage, similar to Back-propagation.

$$\begin{aligned} V^{t,n}_{\boldsymbol{x}} &= Q^{t,n}_{\boldsymbol{x}} - Q^{t,n}_{\boldsymbol{x}\theta}\mathbf{k}_{t,n}, & V^{T,n}_{\boldsymbol{x}} &= \phi^n_{\boldsymbol{x}}, \\ V^{t,n}_{\boldsymbol{x}\boldsymbol{x}} &= Q^{t,n}_{\boldsymbol{x}\boldsymbol{x}} - Q^{t,n}_{\boldsymbol{x}\theta}\mathbf{K}_{t,n}, & V^{T,n}_{\boldsymbol{x}\boldsymbol{x}} &= \phi^n_{\boldsymbol{x}\boldsymbol{x}}. \end{aligned} \tag{10}$$

Algorithm 1 Dynamic Game Theoretic Neural Optimizer

```
1: Input: dataset $\mathcal{D}$, network $F \equiv \{F_t : t \in [T]\}$
2: repeat
3:   Compute $\boldsymbol{x}_t$ by propagating $\boldsymbol{x}_0 \sim \mathcal{D}$ through $F$
4:   for $t = T-1$ to $0$ do        ▷ Solve FNE or GR
5:     Solve the update $\delta\pi^*_{t,n}$ with (9) or (11)
6:     Solve $(V^{t,n}_{\boldsymbol{x}}, V^{t,n}_{\boldsymbol{x}\boldsymbol{x}})$ or $(W^t_{\boldsymbol{x}}, W^t_{\boldsymbol{x}\boldsymbol{x}})$ with (10) or (25)
7:   end for
8:   Set $\boldsymbol{x}'_0 = \boldsymbol{x}_0$
9:   for $t = 0$ to $T-1$ do        ▷ Update parameter
10:    Apply $\theta_{t,n} \leftarrow \theta_{t,n} - \delta\pi^*_{t,n}(\delta\boldsymbol{x}_t)$ with $\delta\boldsymbol{x}_t = \boldsymbol{x}'_t - \boldsymbol{x}_t$
11:    Compute $\boldsymbol{x}'_{t+1} = F_t(\boldsymbol{x}'_t, \theta_{t,1}, \cdots, \theta_{t,N})$
12:  end for
13: until converges
```

Derivation for CG follows similar steps except we consider solving the GR recursion (7) by $\pi^*_{t,n} \approx \arg\min P_t$. Since all players' actions are now observable from $\eta^{\text{C-CG}}_{t,n}$, we need to expand P_t w.r.t. all arguments. For notational simplicity, let us denote $\boldsymbol{u} \equiv \theta_{t,1}, \boldsymbol{v} \equiv \theta_{t,2}$ in Fig. 2. In the case when each player minimizes P_t independently without knowing the other, we know the non-cooperative update for Player 2 admits the form[2] of $\delta\boldsymbol{v}_t = \mathbf{I}_t + \mathbf{L}_t\delta\boldsymbol{x}_t$. Now, the locally-optimal cooperative update for Player 1 can be written as

$$\begin{gathered}\delta\pi^*_{t,1} = \widetilde{\mathbf{k}}_t + \widetilde{\mathbf{K}}_t\delta\boldsymbol{x}_t, \text{ where} \\ \widetilde{\mathbf{k}}_t := {\widetilde{P}^t_{\boldsymbol{u}\boldsymbol{u}}}^{\dagger}(P^t_{\boldsymbol{u}} - P^t_{\boldsymbol{u}\boldsymbol{v}}\mathbf{I}_t), \; \widetilde{\mathbf{K}}_t := {\widetilde{P}^t_{\boldsymbol{u}\boldsymbol{u}}}^{\dagger}(P^t_{\boldsymbol{u}\boldsymbol{x}} - P^t_{\boldsymbol{u}\boldsymbol{v}}\mathbf{L}_t).\end{gathered} \tag{11}$$

Similar equations can be derived for Player 2. We will refer $\widetilde{P}^t_{\boldsymbol{u}\boldsymbol{u}} := P^t_{\boldsymbol{u}\boldsymbol{u}} - P^t_{\boldsymbol{u}\boldsymbol{v}}{P^t_{\boldsymbol{v}\boldsymbol{v}}}^{\dagger}P^t_{\boldsymbol{v}\boldsymbol{u}}$ as the *cooperative precondition*. The update (11), despite seemly complex, exhibits intriguing properties. For one, notice that computing the cooperative open gain $\widetilde{\mathbf{k}}_t$ for Player 1 involves the non-cooperative open gain $\mathbf{I}_t$ from Player 2. In other words, each player adjusts the strategy after knowing the companion's action. Similar interpretation can be drawn for the feedbacks $\widetilde{\mathbf{K}}_t$ and $\mathbf{L}_t$. Propagation of $(W^t_{\boldsymbol{x}}, W^t_{\boldsymbol{x}\boldsymbol{x}})$ follows similarly as (10) once all players' updates are computed. We leave the complete formula in (25) (see Appendix D.1) since it is rather tedious.

Let us discuss the role of $\delta\boldsymbol{x}_t$ and how to compute them. Conceptually, $\delta\boldsymbol{x}_t$ can be any deviation away from the fixed point $\boldsymbol{x}_t$ where we expand the objectives, $Q_{t,n}$ or P_t. In MPDG application, it is typically set to the state difference when the parameter updates are applied until stage t,

$$\delta\boldsymbol{x}_t := (F_{t-1} \circ \cdots \circ F_0)(\boldsymbol{x}_0, \{\theta + \delta\pi^*\}_{<t,\forall n}) - \boldsymbol{x}_t, \tag{12}$$

where $\{\delta\pi^*\}_{<t,\forall n} := \{\delta\pi^*_{s,n} : s<t, \forall n\}$ collects all players' updates until stage t. In this view, the feedback compensates all changes, including those that may cause instability, cascading from the preceding layers; hence it tends to robustify the training process (Pantoja, 1988; Liu et al., 2021).

[2] Similar to (9), we have $\mathbf{I}_t := {P^t_{\boldsymbol{v}\boldsymbol{v}}}^{\dagger}P^t_{\boldsymbol{v}}$ and $\mathbf{L}_t := {P^t_{\boldsymbol{v}\boldsymbol{v}}}^{\dagger}P^t_{\boldsymbol{v}\boldsymbol{x}}$.

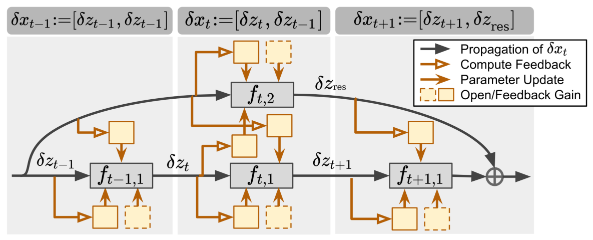

$\delta x_{t-1} := [\delta z_{t-1}, \delta z_{t-1}]$
$\delta x_t := [\delta z_t, \delta z_{t-1}]$
$\delta x_{t+1} := [\delta z_{t+1}, \delta z_{\text{res}}]$
Propagation of δx_t
Compute Feedback
Parameter Update
Open/Feedback Gain
$f_{t,2}$
δz_{res}
δz_{t-1}
$f_{t-1,1}$
δz_t
$f_{t,1}$
δz_{t+1}
$f_{t+1,1}$

Figure 4. How $\delta\pi^*_{t,n}$ is applied (*c.f.* line 8-12 in Alg. 1) to a residual block. We compute $\delta \boldsymbol{x}_t$ with a forward propagation (12) and simultaneously update the parameter. The solid yellow box denotes the feedback dependent on the preceding hidden states $\boldsymbol{z}_{s\leq t}$.

Alg. 1 presents the pseudo-code of DGNOpt, which consists of *(i)* the same forward propagation through the network (line 3), *(ii)* a distinct game-theoretic backward process that solves either FNE or GR optimality (line 4-7), and *(iii)* an additional forward pass that applies the feedback updates $\delta\pi^*_{t,n}$ (line 8-12; also Fig. 4). We stress that Alg. 1 accepts any generic DNN so long as it can be represented by F_t.

4.2. Curvature Approximation

Naively inverting the parameter curvature, *i.e.* $(Q^{t,n}_{\theta\theta})^\dagger$ and $\widetilde{P}^{t\ \dagger}_{\boldsymbol{uu}}$, can be computationally inefficient and sometimes unstable for practical training. To mitigate the issue, we adopt curvature amortizations (Kingma & Ba, 2014; Hinton et al., 2012) used in DNN training. These methods naturally fit into our framework by recalling Proposition 2 that different baselines differ in how they estimate the curvature $H^{t,n}_{\theta\theta}$ for the preconditioned update $H^{t.n\ \dagger}_{\theta\theta} H^{t,n}_{\theta}$. With this in mind, we can estimate the FNE parameter curvature $Q^{t,n}_{\theta\theta}$ with

$$Q^{t,n}_{\theta\theta} \approx Q^{t,n}_{\theta} Q^{t,n\,\mathsf{T}}_{\theta} \quad \text{or} \quad \operatorname{diag}(\sqrt{Q^{t,n}_{\theta} \odot Q^{t,n}_{\theta}}), \tag{13}$$

which resembles the Gauss-Newton (GN) matrix or its adaptive diagonal matrix (as appeared in RMSProp and Adam).

As for $\widetilde{P}^{t\ \dagger}_{\boldsymbol{uu}}$, which contains an *inner* inversion $P^{t\ \dagger}_{\boldsymbol{vv}}$ inside $\widetilde{P}^{t}_{\boldsymbol{uu}}$, we propose a new approximation inspired by the Kronecker factorization (KFAC; Martens & Grosse (2015)). KFAC factorizes the GN matrix with two smaller-size matrices. We leave the complete discussion on KFAC, as well as the proof of the following result, in Appendix D.2.

Proposition 5 (KFAC for $\widetilde{P}^t_{\boldsymbol{uu}}$)**.** *Suppose $P^t_{\boldsymbol{uu}}$ and $P^t_{\boldsymbol{vv}}$ are factorized with some vectors $\boldsymbol{z}_1, \boldsymbol{z}_2, \boldsymbol{g}_1, \boldsymbol{g}_2$ by*

$$P^t_{\boldsymbol{uu}} \approx \mathbb{E}[\boldsymbol{z}_1 \boldsymbol{z}_1^\mathsf{T}] \otimes \mathbb{E}[\boldsymbol{g}_1 \boldsymbol{g}_1^\mathsf{T}] =: A_{\boldsymbol{uu}} \otimes B_{\boldsymbol{uu}},$$
$$P^t_{\boldsymbol{vv}} \approx \mathbb{E}[\boldsymbol{z}_2 \boldsymbol{z}_2^\mathsf{T}] \otimes \mathbb{E}[\boldsymbol{g}_2 \boldsymbol{g}_2^\mathsf{T}] =: A_{\boldsymbol{vv}} \otimes B_{\boldsymbol{vv}},$$

where the expectation is taken over the mini-batch data. Let $A_{\boldsymbol{uv}} := \mathbb{E}[\boldsymbol{z}_1 \boldsymbol{z}_2^\mathsf{T}]$ and $B_{\boldsymbol{uv}} := \mathbb{E}[\boldsymbol{g}_1 \boldsymbol{g}_2^\mathsf{T}]$, then the cooperative precondition matrix in (11) can be factorized by

$$\begin{aligned} \widetilde{P}^t_{\boldsymbol{uu}} &\approx \widetilde{A}_{\boldsymbol{uu}} \otimes \widetilde{B}_{\boldsymbol{uu}} \\ &= (A_{\boldsymbol{uu}} - A_{\boldsymbol{uv}} A^\dagger_{\boldsymbol{vv}} A^\mathsf{T}_{\boldsymbol{uv}}) \otimes (B_{\boldsymbol{uu}} - B_{\boldsymbol{uv}} B^\dagger_{\boldsymbol{vv}} B^\mathsf{T}_{\boldsymbol{uv}}). \end{aligned} \tag{14}$$

In practice, we set $(\boldsymbol{z}_1, \boldsymbol{z}_2, \boldsymbol{g}_1, \boldsymbol{g}_2) := (\boldsymbol{z}_t, \boldsymbol{z}_{t-1}, W^{t+1}_{\boldsymbol{z}_t}, W^{t+1}_{\boldsymbol{z}_{t-1}})$ for the residual block in Fig. 2 or 4. With Proposition 5, we can compute the update, take $\widetilde{\mathbf{k}}_t$ for instance, by

$$\widetilde{\mathbf{k}}_t = \operatorname{vec}(\widetilde{B}^\dagger_{\boldsymbol{uu}} (P^t_{\boldsymbol{u}} - B_{\boldsymbol{uv}} \operatorname{vec}^\dagger(\mathbf{I}_t) A^\mathsf{T}_{\boldsymbol{uv}}) \widetilde{A}^{-\mathsf{T}}_{\boldsymbol{uu}}), \tag{15}$$

where $\operatorname{vec}^\dagger$ is the inverse operation of vectorization (vec).

Another computation source comes from the curvature w.r.t. the MPDG state, *i.e.* $V^{t,n}_{\boldsymbol{xx}}$ and $W^t_{\boldsymbol{xx}}$. Here, we approximate them with low-rank matrices using either Gauss-Newton or their top eigenspace. These are rather reasonable approximations since it has been constantly observed that these Hessians are highly degenerate for DNNs (Wu et al., 2020; Papyan, 2019; Sagun et al., 2017). With all these, we are able to train modern DNNs by solving their corresponding dynamic games, (6) or (7), with a runtime comparable to other first and second-order methods (see Fig. 7).

4.3. Algorithmic Connection

Finally, let us discuss an intriguing algorithmic equivalence. Recall the subset relation among the information structures in (8). Manipulating these structures allows one to traverse between different game optimality principles. For instance, masking $\pi^*_{t,\neg n}$ in $\eta^{\text{C-CG}}_{t,n}$ makes it degenerate to $\eta^{\text{C}}_{t,n}$, which implies the FNE and GR optimality become equivalent. Through this lens, one may wonder if a similar algorithmic relation can be drawn for these iterative updates. This is indeed the case as shown below (proof left in Appendix D.3).

Theorem 6 (Algorithmic equivalence)**.**

- *(11) with $P^t_{\boldsymbol{uv}} := \mathbf{0}$ gives (9)*
- *(9) with $(Q^{t,n}_{\theta\boldsymbol{x}}, Q^{t,n}_{\theta\theta}) := (\mathbf{0}, \boldsymbol{I})$ gives SGD*
- *(11) with $(P^t_{\boldsymbol{uv}}, P^t_{\boldsymbol{ux}}, P^t_{\boldsymbol{uu}}) := (\mathbf{0}, \mathbf{0}, \boldsymbol{I})$ gives SGD*

Setting $Q^{t,n}_{\theta\theta}$ and $P^t_{\boldsymbol{uu}}$ to other precondition matrices, similar to (13), recovers other baselines.

The intuition behind Theorem 6 is that when higher-order (>2) expansions are discarded, setting $P^t_{\boldsymbol{uv}} := \mathbf{0}$ completely blocks the communication between two players; therefore we effectively remove $\pi^*_{t,\neg n}$ from $\eta^{\text{C-CG}}_{t,n}$. Similarly, forcing $Q^{t,n}_{\theta\boldsymbol{x}} := \mathbf{0}$ prevents Player n from observing how changing $\boldsymbol{x}_t$ may affect the payoff, hence one can at best achieve the same OLNE optimality as baselines. Theorem 6 implies that (9) and (11) generalize standard updates to richer information structure; thereby creating more complex updates.

5. Experiment

5.1. Evaluation on Classification Datasets

Datasets and networks. We verify the performance of DGNOpt on image classification datasets as they are suitable

Table 2. Accuracy (%) of residual-based networks (averaged over 6 random seeds)

Dataset	Baselines (*i.e.* min $H_{t,n}$ in OLNE)					**DGNOpt**
	SGD	RMSProp	Adam	EKFAC	EMSA	**(ours)**
MNIST	98.65	98.61	98.49	**98.77**	98.25	98.76
SVHN	88.58	88.96	89.20	88.75	87.40	**89.22**
CIFAR10	82.94	83.75	85.66	85.65	75.60	**85.85**
CIFAR100	71.78	71.65	71.96	71.95	62.63	**72.24**

Table 3. Accuracy (%) of inception-based networks (averaged over 4 random seeds)

Dataset	Baselines (*i.e.* min $H_{t,n}$ in OLNE)					**DGNOpt**
	SGD	RMSProp	Adam	EKFAC	EMSA	**(ours)**
MNIST	97.96	97.75	97.72	97.90	97.39	**98.03**
SVHN	87.61	86.14	86.84	88.89	82.68	**88.94**
CIFAR10	76.66	74.38	75.38	77.54	70.17	**77.72**

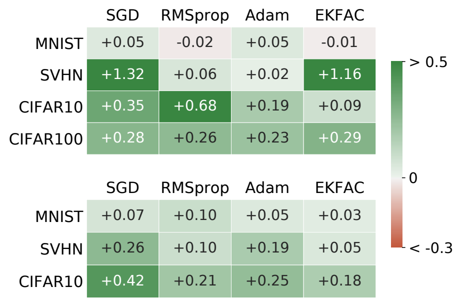


Figure 5. Accuracy (%) improvement (+) or degradation (-) when richer information structure, *i.e.* $\eta^{\text{O}}_{t,n} \rightarrow \{\eta^{\text{C}}_{t,n}, \eta^{\text{C-CG}}_{t,n}\}$, is used for each best-tuned baseline[3] in Table 2 (upper) and Table 3 (bottom). Color bar is scaled for best view.

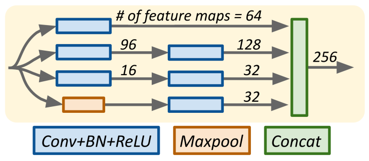


Figure 6. Architecture of the inception block.

testbeds for modern networks that contain non-Markovian dependencies. Specifically, we first consider residual-based networks given their popularity and our thorough discussions in §4. For larger datasets such as CIFAR10/100, we train ResNet18 with multi-stepsize learning rate decay. For MNIST and SVHN, we use residual networks composed of 3 residual blocks (see Fig. 2). Meanwhile, we also consider inception-based networks, which composed of an inception block (see Fig. 6) that resembles a 4-player dynamic game. All networks use ReLU activation and are trained with 128 batch size. Other setups are detailed in Appendix E.

Baselines. Motivated by our discussion in §3, we compare DGNOpt, which essentially solves FNE and GR, with methods involving OLNE either implicitly or explicitly. This includes standard training methods such as SGD, RMSprop, Adam, and EKFAC (George et al., 2018), which is an extension to the second-order method KFAC with eigenvalue-correction. To also compare against OCT-inspired methods, we include EMSA (Li et al., 2017a), which explicitly minimizes a modified Hamiltonian. Other OCT-based training methods mostly consider degenerate, *e.g.* discrete-weighted (Li & Hao, 2018) or Markovian (Liu et al., 2021), networks. In this view, DGNOpt generalizes those methods to both larger network class and richer information structure.

Performance and ablation study. Table 2 and 3 summarize the performance for the residual and inception networks. On most datasets, DGNOpt achieves competitive results against standard methods and outperforms EMSA by a large margin. Despite both originates from the OCT methodology, in practice EMSA often exhibits numerical instability for larger networks. On the contrary, DGNOpt leverages iteration-based linearization and amortized curvature, which greatly stabilizes the training.

On the other hand, DGNOpt distinguishes itself from standard baselines by considering a larger information structure. To validate the benefit of having this additional knowledge during training, we conduct an ablation study using the algorithmic connection built in Theorem 6. Specifically, we measure the performance difference when the best-tuned baselines, *i.e.* the ones we report in Table 2 and 3, are further allowed to utilize higher-level information. Algorithmically, this can be done by running DGNOpt with the parameter curvature replaced by the precondition matrix of each baseline. For instance, replacing all $Q^{t,n}_{\theta\theta}$ with identity matrices $\boldsymbol{I}$ while keeping other computation unchanged is equivalent to lifting SGD to accept the closed-loop structure $\eta^{\text{C}}_{t,n}$. From Theorem 6, these two training procedures now differ only in the presence of $Q^{t,n}_{\theta\boldsymbol{x}}$, which allows SGD to adjust its update based on the change of $\boldsymbol{x}_t \in \eta^{\text{C}}_{t,n}$. As shown in Fig. 5, enlarging the information structure tends to enhance the performance, or at least being innocuous. We highlight these improvements as the benefit gained from introducing dynamic game theory to the original OCT interpretation.

Overhead vs performance trade-offs. As shown in Fig. 7, DGNOpt enjoys a comparable runtime and memory complexity to standard methods on training ResNet18. Specifically, its per-iteration runtime is around $\pm 40\%$ compared to the second-order baseline, depending on the information structures (DGNOpt-FNE *v.s.* DGNOpt-GR). In practice, these gaps tend to vanish for smaller networks. The overhead introduced by DGNOpt enables the computation of *feedback updates* using a richer information structure. From the OCT standpoint, the feedback is known to play a key role in compensating the unstable disturbance along the propagation. Particularly, when problems inherit chained

[3] The ablation analysis in Fig. 5 applies Theorem 6 to methods that solve the exact Hamiltonian; hence excludes EMSA since it instead considers a modified Hamiltonian (see Appendix E).

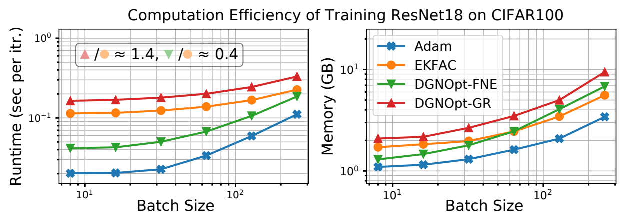


Figure 7. Our second-order method DGNOpt exhibits similar runtime (±40%) and memory (±30%) complexity compared to the second-order baseline EKFAC.

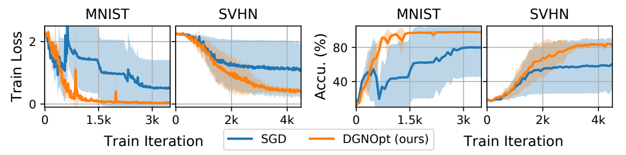


Figure 8. Training inception-based networks using larger step sizes, where MNIST (*resp.* SVHN) uses lr=1.0 (*resp.* lr=2.0).

Table 4. Convergence speed w.r.t. N. Numerical values report the training steps required to achieve certain accuracy on each dataset.

Achieved Performance	Number of Player (N) in SGD 1	2	4	6
80% in SVHN	5.14k	2.31k	1.25k	**0.8k**
60% in CIFAR10	3.62k	**2.97k**	**2.98k**	5.83k

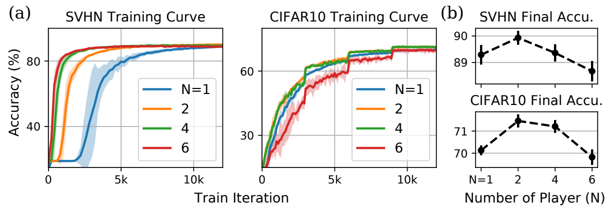


Figure 9. (a) Training curve and (b) final accuracy as we vary the number of player (N) as a hyperparameter of game-extended SGD.

constraints (*e.g.* DNNs), these feedback-enhanced methods often converge faster with superior numerical stability against standard methods (Murray & Yakowitz, 1984).

To validate the role of feedback in training modern DNNs, notice that one shall expect the effect of feedback becomes significant when a larger step size is taken. This is because (see (12)) larger $\delta\pi^*$ increases $\delta \boldsymbol{x}_t$, which amplifies the feedback $\mathbf{K}\delta\boldsymbol{x}_t$. Fig. 8 confirms our hypothesis, where we train the inception-based networks on MNIST and SVHN using relatively large learning rates. It is clear that utilizing feedback updates greatly stabilizes the training. While the SGD baseline struggles to make stable progress, DGNOpt converges almost flawlessly (with negligible overhead). As for well-tuned hyperparameter which often has a smaller step size, our ablation analysis in Fig. 5 suggests that having feedbacks throughout the stochastic training generally leads to better local minima.

5.2. Game-Theoretic Applications

Cooperative training with fictitious agents. Despite all the rigorous connection we have explored so far, it is perhaps unsatisfactory to see our multi-agent analysis degenerates when facing feedforward networks, since the number of player N becomes trivially 1. We can remedy this scenario by considering the following transformation.

$$F_t(\boldsymbol{z}_t, \theta_{t,1}, \cdots, \theta_{t,N}) := f_t(\boldsymbol{z}_t, \theta_t), \ \textstyle\sum_{n=1}^N \theta_{t,n} = \theta_t \quad (16)$$

In other words, we can divide the layer's weight (or player's action) into multiple parts, so that the MPDG framework remains applicable. Interestingly, the transformation of this kind resembles game-theoretic robust optimal control (Pan et al., 2015; Sun et al., 2018), where the controller (or player in our context) models external disturbances with *fictitious agents*, in order to enhance the robustness or convergence of the optimization process.

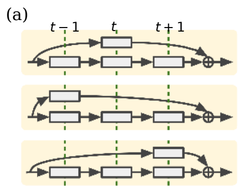
Fig. 9 and Table 4 provide the training results when SGD presumes different numbers of players interacting in a feedforward network consisting of 4 convolution and 2 fully-connected layers (see Appendix E). Notice that N=1 corresponds to the original method. For N>1, we apply the transformation (16) then solve for the cooperative update as in DGNOpt. We stress that these fictitious agents only appear during the training phase for computing the cooperative updates. At inference, actions from all players collapse back to θ_t by the summation in (16).

While it is clear that encouraging agents to cooperate during training can achieve better minima at a faster rate, having more agents, surprisingly, does not always imply better performance. In practice, the improvement can slow down or even degrade once N passes some critical values. This implies that N shall be treated as a hyper-parameter of these game-extended methods. Empirically, we find that N=2 provides a good trade-off between the final performance and convergence speed. We observe a consistent result for this setup on other optimizers (see Appendix E for EKFAC).

Adaptive alignment using multi-armed bandit. Finally, let us discuss an application of the bandit algorithm in our framework. In §3.1, we briefly mentioned that mapping from modern networks to the shared dynamics F_t most likely will not be unique. For instance (see Fig. 10a), placing the shortcut module of a residual block at different locations leads to different F_t; hence results in different DGNOpt updates. This is a distinct feature arising exclusively from our MPDG framework, since these alignments are unrecognizable to standard baselines. It naturally raises the following questions: what is the optimal strategy to align the (p)layers of the network in our dynamic game, and how do different aligning strategies affect training?

Table 5. Training result (accuracy %) of Fig. 10 on two datasets.

Dataset	EKFAC	DGNOpt + Aligning Strategy fixed	random	adaptive
SVHN	87.49	88.20	88.12	**88.33**
CIFAR10	84.67	85.20	85.27	**85.65**



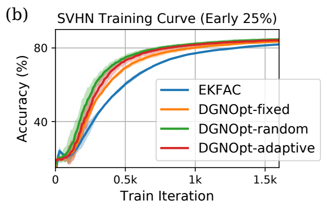


Figure 10. (a) Different alignments of the same residual block leads to distinct DGNOpt updates, yet they are unrecognizable to baselines. (b) Early phase of training with different aligning strategies.

To answer these questions, we compare the performance between three strategies, including *(i)* using a fixed alignment throughout training, *(ii)* random alignment at each iteration, and *(iii)* adaptive alignment using a multi-armed bandit. For the last case, we interpret pulling an arm as selecting one of the alignments and associate the round-wise reward with the validation accuracy at each iteration. Note that this is a *non-stationary* bandit problem since the reward distribution of each arm/alignment evolves as we train the network. We provide the pseudo-code of this procedure in Appendix E.

Fig. 10b and Table 5 report the results of DGNOpt using different aligning strategies. We also include the baseline when the information structure shrinks from $\eta^{\text{C-CG}}_{t,n}$ to $\eta^{\text{O}}_{t,n}$, similar to the ablation study in §5.1. In this case, all these DGNOpt variants degenerate to EKFAC. For the non-stationary bandit, we find EXP3++ (Seldin & Slivkins, 2014) to be sufficient in this application. While DGNOpt with fixed alignment already achieves faster convergence compared with the baseline, dynamic alignment using either random or adaptive strategy leads to further improvement (see Fig. 10b). Notably, having the adaptation throughout training also enhances the final accuracy. For CIFAR10 with ResNet18, the value is boost by 1% from baseline and 0.5% compared with the other two strategies. This sheds light on new algorithmic opportunities inspired by *architecture-aware* optimization.

6. Discussion

Comparison to Markovian-based OCT-inspired methods. As we briefly mentioned in §3.2, our DGNOpt (with FNE) can be seen as a game-theoretic extension of Liu et al. (2021), which is also an OCT-inspired method despite concerning only Markovian networks. It is natural to wonder whether these two methods are interchangeable since one can always force a non-Markovian system to be Markovian by lifting it into higher dimensions or aggregating the state. Here, we stress that our DGNOpt differs from Liu et al. (2021) in many significant ways. For one, forming a Markovian chain by grouping the non-Markovian layers into higher dimensions leads to a *degenerate* information structure. The (p)layers inside each Markovian group, $\{f_{t,n} : t^- < t < t^+\}$, only have access to $\boldsymbol{x}_{s \leq t^-}$ rather than full latest information $\boldsymbol{x}_{s \leq t}$ as in DGNOpt, since their dependencies are discarded. From the Nash standpoint, this leads to degenerate backward optimality and update rules. Indeed, in the limit when we simply group the whole network as single-step dynamics, we will recover $\eta^{\text{O}}_{t,n} \triangleq \{\boldsymbol{x}_0\}$ in baselines. In contrast, DGNOpt fully leverages the structural relation of the network, hence enables rich game-based applications, *e.g.* bandit or robust control, that are otherwise infeasible with Liu et al. (2021).

Degeneracy when partitioning parameters as players. In §5.2, we demonstrate a specific transformation, *i.e.* (16), that makes cooperative training possible for single-player feedforward networks while respecting our *layer-as-player* game formulation. This transformation may seem artificial at first glance compared to a naive alternative that directly partitions the parameters of each layer as distinct players. Unfortunately, the latter strategy yields *degenerate* cooperative optimality. To see it, notice that treating the $\pi_{t,n}$ appeared in the joint optimization (7) as the n^{th}-partitioned parameters of layer t is *equivalent* to solving the FNE optimality (6) with $N=1$ (so that the $\pi_{t,N=1}$ in (6) becomes the intact parameters of layer t). Hence, it collapses to the prior single-player non-cooperative method (Liu et al., 2021), which converges much slower than our DGNOpt (Fig. 11). Our proposed transformation (16) enables collaborative control and may be naturally extended to other robust formulations, *e.g.* minimax adversarial training.

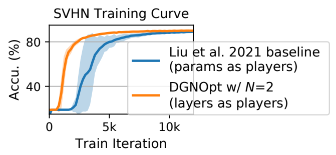


Figure 11. Convergence using different dynamic game formulations with the same setup as Fig. 9a.

7. Conclusion

In this work, we introduce a novel game-theoretic characterization by bridging the training process of DNN with a multi-agent dynamic game. The inspired optimizer, DGNOpt, generalizes previous OCT-based methods to generic network class and encourages cooperative updates to improve the performance. Our work pushes forward principled algorithmic design from OCT and game theory.

Acknowledgements

G.H. Liu was supported by CPS NSF Award #1932068, and T. Chen was supported by ARO Award #W911NF2010151. The authors thank C.H. Lin, Y. Pan, C.W. Kuo, M. Gandhi, E. Evans, and Z. Wang for many helpful discussions.

References

Balduzzi, D. Deep online convex optimization with gated games. *arXiv preprint arXiv:1604.01952*, 2016.

Balduzzi, D., Racaniere, S., Martens, J., Foerster, J., Tuyls, K., and Graepel, T. The mechanics of n-player differentiable games. *arXiv preprint arXiv:1802.05642*, 2018.

Bellman, R. The theory of dynamic programming. Technical report, Rand corp santa monica ca, 1954.

Chang, B., Meng, L., Haber, E., Ruthotto, L., Begert, D., and Holtham, E. Reversible architectures for arbitrarily deep residual neural networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Chen, T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, pp. 6572–6583, 2018.

George, T., Laurent, C., Bouthillier, X., Ballas, N., and Vincent, P. Fast approximate natural gradient descent in a kronecker factored eigenbasis. In *Advances in Neural Information Processing Systems*, pp. 9550–9560, 2018.

Ghorbani, A. and Zou, J. Neuron shapley: Discovering the responsible neurons. *arXiv preprint arXiv:2002.09815*, 2020.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Greydanus, S., Dzamba, M., and Yosinski, J. Hamiltonian neural networks. In *Advances in Neural Information Processing Systems*, pp. 15353–15363, 2019.

Gunther, S., Ruthotto, L., Schroder, J. B., Cyr, E. C., and Gauger, N. R. Layer-parallel training of deep residual neural networks. *SIAM Journal on Mathematics of Data Science*, 2(1):1–23, 2020.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hinton, G., Srivastava, N., and Swersky, K. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. 2012.

Hu, K., Kazeykina, A., and Ren, Z. Mean-field langevin system, optimal control and deep neural networks. *arXiv preprint arXiv:1909.07278*, 2019.

Jaderberg, M., Mnih, V., Czarnecki, W. M., Schaul, T., Leibo, J. Z., Silver, D., and Kavukcuoglu, K. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Li, Q. and Hao, S. An optimal control approach to deep learning and applications to discrete-weight neural networks. *arXiv preprint arXiv:1803.01299*, 2018.

Li, Q., Chen, L., Tai, C., and Weinan, E. Maximum principle based algorithms for deep learning. *The Journal of Machine Learning Research*, 18(1):5998–6026, 2017a.

Li, Q., Tai, C., and E, W. Stochastic modified equations and adaptive stochastic gradient algorithms. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2101–2110. JMLR. org, 2017b.

Liu, G.-H. and Theodorou, E. A. Deep learning theory review: An optimal control and dynamical systems perspective. *arXiv preprint arXiv:1908.10920*, 2019.

Liu, G.-H., Chen, T., and Theodorou, E. A. Ddpnopt: Differential dynamic programming neural optimizer. In *International Conference on Learning Representations*, 2021.

Lu, Y., Zhong, A., Li, Q., and Dong, B. Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations. *arXiv preprint arXiv:1710.10121*, 2017.

Lu, Y., Ma, C., Lu, Y., Lu, J., and Ying, L. A mean-field analysis of deep resnet and beyond: Towards provable optimization via overparameterization from depth. *arXiv preprint arXiv:2003.05508*, 2020.

Martens, J. and Grosse, R. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pp. 2408–2417, 2015.

Murray, D. and Yakowitz, S. Differential dynamic programming and newton's method for discrete optimal control problems. *Journal of Optimization Theory and Applications*, 43(3):395–414, 1984.

Pan, Y., Theodorou, E., and Bakshi, K. Robust trajectory optimization: A cooperative stochastic game theoretic approach. In *Robotics: Science and Systems*, 2015.

Pantoja, J. F. A. Differential dynamic programming and newton's method. *International Journal of Control*, 47 (5):1539–1553, 1988.

Papyan, V. Measurements of three-level hierarchical structure in the outliers in the spectrum of deepnet hessians. *arXiv preprint arXiv:1901.08244*, 2019.

Pardalos, P. M., Migdalas, A., and Pitsoulis, L. *Pareto optimality, game theory and equilibria*, volume 17. Springer Science & Business Media, 2008.

Petrosjan, L. A. Cooperative differential games. In *Advances in dynamic games*, pp. 183–200. Springer, 2005.

Pontryagin, L. S., Mishchenko, E., Boltyanskii, V., and Gamkrelidze, R. The mathematical theory of optimal processes. 1962.

Sagun, L., Evci, U., Guney, V. U., Dauphin, Y., and Bottou, L. Empirical analysis of the hessian of over-parametrized neural networks. *arXiv preprint arXiv:1706.04454*, 2017.

Seldin, Y. and Slivkins, A. One practical algorithm for both stochastic and adversarial bandits. In *International Conference on Machine Learning*, pp. 1287–1295. PMLR, 2014.

Stier, J., Gianini, G., Granitzer, M., and Ziegler, K. Analysing neural network topologies: a game theoretic approach. *Procedia Computer Science*, 126:234–243, 2018.

Sun, W., Pan, Y., Lim, J., Theodorou, E. A., and Tsiotras, P. Min-max differential dynamic programming: Continuous and discrete time formulations. *Journal of Guidance, Control, and Dynamics*, 41(12):2568–2580, 2018.

Tassa, Y., Erez, T., and Todorov, E. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4906–4913. IEEE, 2012.

Tassa, Y., Mansard, N., and Todorov, E. Control-limited differential dynamic programming. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1168–1175. IEEE, 2014.

Weinan, E. A proposal on machine learning via dynamical systems. *Communications in Mathematics and Statistics*, 5(1):1–11, 2017.

Weinan, E., Han, J., and Li, Q. A mean-field optimal control formulation of deep learning. *arXiv preprint arXiv:1807.01083*, 2018.

Wu, Y., Zhu, X., Wu, C., Wang, A., and Ge, R. Dissecting hessian: Understanding common structure of hessian in neural networks. *arXiv preprint arXiv:2010.04261*, 2020.

Yeung, D. W. and Petrosjan, L. A. *Cooperative stochastic differential games*. Springer Science & Business Media, 2006.

Zhang, D., Zhang, T., Lu, Y., Zhu, Z., and Dong, B. You only propagate once: Accelerating adversarial training via maximal principle. *arXiv preprint arXiv:1905.00877*, 2019.