Supplementary Material

A. Notation Summary

Table 6. Abbreviation.

OCT/OCP	Optimal Control Theory/Programming
MPDG	Multi-Player Dynamic Game
CG	Cooperative Game
OLNE	Open-loop Nash Equilibria
FNE	Feedback Nash Equilibria
GR	Group Rationality
IR	Individual Rationality

Table 7. Terminology mapping.

	MPDG	**Training generic (non-Markovian) DNNs**	
t	Stage order	Computation order from input to output	Layer index (t, n)
n	Player index	Index of parallel layers aligned at t	
$f_{t,n}$	-	Layer module indexed by (t, n)	
F_t	Shared dynamics	Joint propagation rule of layers $\{f_{t,n} : n \in [N]\}$	
$\theta_{t,n}$	Action committed at stage t by Player n	Trainable parameter of layer $f_{t,n}$	
$\boldsymbol{z}_{t,n}$	-	Pre-activation vector of layer $f_{t,n}$	
$\boldsymbol{x}_t$	State at stage t	Augmentation of pre-activation vectors of layers $\{f_{t,n} : n \in [N]\}$	
$\ell_{t,n}$	Cost incurred at stage t for Player n	Weight decay for layer $f_{t,n}$	
ϕ_n	Cost incurred at final stage T for Player n	Lost w.r.t. network output (*e.g.* cross entropy in classification)	

Table 8. Dynamic game theoretic terminology w.r.t. different optimality principles.

OLNE	$\eta^{\text{O}}_{t,n}$	Open-loop information structure
	$H_{t,n}$	Optimality objective (Hamiltonian) for OLNE
	$\boldsymbol{p}_{t,n}$	Co-state at stage t for Player n
FNE	$\eta^{\text{C}}_{t,n}$	Feedback information structure
	$Q_{t,n}$	Optimality objective (Isaacs-Bellman objective) for FNE
	$V_{t,n}$	Value function for FNE
	$\mathbf{k}_{t,n}$	Open gain of the locally optimal update for FNE
	$\mathbf{K}_{t,n}$	Feedback gain of the locally optimal update for FNE
GR	$\eta^{\text{O-CG}}_{t,n}$	Cooperative open-loop information structure
	$\eta^{\text{C-CG}}_{t,n}$	Cooperative feedback information structure
	P_t	Optimality objective (group Bellman objective) for GR
	W_t	Value function for GR
	$\widetilde{\mathbf{k}}_t$	Open gain of the locally optimal update for GR
	$\widetilde{\mathbf{K}}_t$	Feedback gain of the locally optimal update for GR

B. OCP Characterization of Training Feedforward Networks

The optimality principle to OCP (2), or equivalently the training process of feedforward networks, can be characterized by Dynamic Programming (DP) or Pontryagin Principle (PP). We synthesize the related results below.

Theorem 7 (Bellman (1954); Pontryagin et al. (1962)).
(DP) *Define a value function V_t computed recursively by the Bellman equation (17), starting from $V_T(\boldsymbol{z}_T) = \phi(\boldsymbol{z}_T)$,*

$$V_t(\boldsymbol{z}_t) = \min_{\pi_t} Q_t(\boldsymbol{z}_t, \theta_t), \quad \textit{where} \quad Q_t(\boldsymbol{z}_t, \theta_t) := \ell_t(\theta_t) + V_{t+1}(f_t(\boldsymbol{z}_t, \theta_t)) \tag{17}$$

is called the Bellman objective. $\pi_t \equiv \theta_t(\boldsymbol{z}_t)$ is an arbitrary mapping from $\boldsymbol{z}_t$ to θ_t. Let π_t^* be the minimizer to (17), then $\{\pi_t^* : t \in [T]\}$ is the optimal feedback policy to (2).

(PP) *The optimal trajectory $\pi_t^* \equiv \theta_t^*(\boldsymbol{z}_t^*)$ along (17) obeys*

$$\boldsymbol{z}_{t+1}^* = \nabla_{\boldsymbol{p}_{t+1}} H_t\left(\boldsymbol{z}_t^*, \boldsymbol{p}_{t+1}^*, \theta_t^*\right), \quad \boldsymbol{z}_0^* = \boldsymbol{z}_0, \tag{18a}$$

$$\boldsymbol{p}_t^* = \nabla_{\boldsymbol{z}_t} H_t\left(\boldsymbol{z}_t^*, \boldsymbol{p}_{t+1}^*, \theta_t^*\right), \quad \boldsymbol{p}_T^* = \nabla_{\boldsymbol{z}_T} \phi\left(\boldsymbol{z}_T^*\right), \tag{18b}$$

$$\theta_t^* = \arg\min_{\theta_t} H_t\left(\boldsymbol{z}_t^*, \boldsymbol{p}_{t+1}^*, \theta_t\right), \tag{18c}$$

where (18b) is the adjoint equation for the co-state $\boldsymbol{p}_t^*$ and

$$H_t\left(\boldsymbol{z}_t, \boldsymbol{p}_{t+1}, \theta_t\right) := \ell_t(\theta_t) + \boldsymbol{p}_{t+1}^{\mathsf{T}} f_t(\boldsymbol{z}_t, \theta_t)$$

is the discrete-time Hamiltonian.

Theorem 7 provides an OCP characterization of training feedforward networks. First, notice that the time-varying OCP objectives (Q_t, H_t) are constructed through some backward processes similar to the Back-propagation (BP). Indeed, one can verify that the adjoint equation (18b) gives the exact BP dynamics. Similarly, the dynamics of V_t in (17) also relate to BP under some conditions (Liu et al., 2021). The parameter update, $\theta_t \leftarrow \theta_t - \delta\theta_t$, for standard training methods can be seen as solving the discrete-time Hamiltonian H_t with different precondition matrices (Li et al., 2017a). On the other hand, DDPNOpt (Liu et al., 2021) minimizes the time-dependent Bellman objective Q_t with $\theta_t \leftarrow \theta_t - \delta\pi_t$. This elegant connection is, however, limited to the interpretation between feedforward networks and Markovian dynamical systems (1).

C. Missing Derivations in Section 3

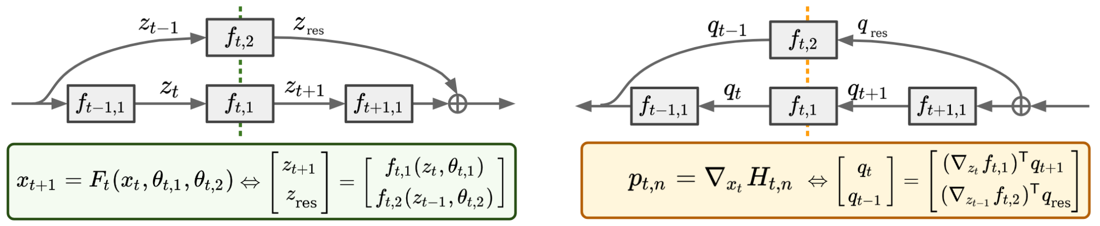
z_{t-1}
$f_{t,2}$
z_{res}
$f_{t-1,1}$
z_t
$f_{t,1}$
z_{t+1}
$f_{t+1,1}$
$\boldsymbol{x}_{t+1} = F_t(\boldsymbol{x}_t, \theta_{t,1}, \theta_{t,2}) \Leftrightarrow \begin{bmatrix} z_{t+1} \\ z_{\text{res}} \end{bmatrix} = \begin{bmatrix} f_{t,1}(z_t, \theta_{t,1}) \\ f_{t,2}(z_{t-1}, \theta_{t,2}) \end{bmatrix}$
q_{t-1}
$f_{t,2}$
q_{res}
$f_{t-1,1}$
q_t
$f_{t,1}$
q_{t+1}
$f_{t+1,1}$
$\boldsymbol{p}_{t,n} = \nabla_{\boldsymbol{x}_t} H_{t,n} \Leftrightarrow \begin{bmatrix} q_t \\ q_{t-1} \end{bmatrix} = \begin{bmatrix} (\nabla_{z_t} f_{t,1})^{\mathsf{T}} q_{t+1} \\ (\nabla_{z_{t-1}} f_{t,2})^{\mathsf{T}} q_{\text{res}} \end{bmatrix}$

Figure 12. Forward propagation (left) and Back-propagation (right) of a residual block and how each quantity connects to OLNE optimality.

Proof of Proposition 2. Expand the expression of the Hamiltonian in OLNE:

$$H_{t,n}(\boldsymbol{x}_t, \boldsymbol{p}_{t+1,n}, \theta_{t,1}, \cdots, \theta_{t,N}) := \ell_{t,n}(\theta_{t,1}, \cdots, \theta_{t,N}) + F_t(\boldsymbol{x}_t, \theta_{t,1}, \cdots, \theta_{t,N})^{\mathsf{T}} \boldsymbol{p}_{t+1,n},$$

where $\boldsymbol{p}_{t,n}$ is the co-state whose dynamics obey

$$\boldsymbol{p}_{t,n} = \nabla_{\boldsymbol{x}_t} H_{t,n}, \quad \boldsymbol{p}_{T,n} = \nabla_{\boldsymbol{x}_T} \phi_n(\boldsymbol{x}_T).$$

Recall §3.1 where we demonstrate that for training generic DNNs, one shall consider $\ell_{t,n} := \ell_{t,n}(\theta_{t,n})$ and $\phi_n := \phi$. Hence, the dynamics of $\boldsymbol{p}_{t,n}$ become

$$\boldsymbol{p}_{t,n} = \nabla_{\boldsymbol{x}_t} H_{t,n}, \quad \boldsymbol{p}_{T,n} = \nabla_{\boldsymbol{x}_T} \phi(\boldsymbol{x}_T), \quad \text{where } H_{t,n} = \ell_{t,n}(\theta_{t,n}) + F_t(\boldsymbol{x}_t, \theta_{t,1}, \cdots, \theta_{t,N})^{\mathsf{T}} \boldsymbol{p}_{t+1,n}. \tag{19}$$

Our goal is to show that (19) gives the exact Back-propagation dynamics. First, notice that the terminal condition of (19), *i.e.* $\boldsymbol{p}_{T,n} = \nabla_{\boldsymbol{x}_T}\phi$, is already the gradient w.r.t. the network output without any manipulation. Next, to show that $\boldsymbol{p}_{t,n}$ corresponds to the Back-propagation at stage t, consider, for instance, the computation graphs of the residual block in Fig. 12, where we replot Fig. 2 together with its Back-propagation dynamic and denote $\boldsymbol{q}$ as the gradient w.r.t. the activation vector $\boldsymbol{z}$. Then, it can be shown by induction that $\boldsymbol{p}_{t,n}$ augments all "$\boldsymbol{q}$"s aligned at stage t. Indeed, suppose $\boldsymbol{p}_{t+1,n}$ is the augmentation of the Back-propagation gradients at stage $t{+}1$, *i.e.* $\boldsymbol{p}_{t+1,n} \coloneqq [\boldsymbol{q}_{t+1}, \boldsymbol{q}_{\text{res}}]^{\mathsf{T}}$, then the co-state at the current stage t can be expanded as

$$\boldsymbol{p}_{t,n} = \nabla_{\boldsymbol{x}_t} H_{t,n} = \nabla_{\boldsymbol{x}_t} F_t^{\mathsf{T}} \boldsymbol{p}_{t+1,n} = \begin{bmatrix} \nabla_{\boldsymbol{z}_t} f_{t,1} & \nabla_{\boldsymbol{z}_{t-1}} f_{t,1} \\ \nabla_{\boldsymbol{z}_t} f_{t,2} & \nabla_{\boldsymbol{z}_{t-1}} f_{t,2} \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} \boldsymbol{q}_{t+1} \\ \boldsymbol{q}_{\text{res}} \end{bmatrix} = \begin{bmatrix} \nabla_{\boldsymbol{z}_t} f_{t,1}{}^{\mathsf{T}} \boldsymbol{q}_{t+1} \\ \nabla_{\boldsymbol{z}_{t-1}} f_{t,2}{}^{\mathsf{T}} \boldsymbol{q}_{\text{res}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{q}_t \\ \boldsymbol{q}_{t-1} \end{bmatrix},$$

which augments all "$\boldsymbol{q}$"s at stage t. Once we connect $\boldsymbol{p}_{t,n}$ to the Back-propagation dynamics, it can be verified that

$$H_\theta^{t,n} \equiv \nabla_{\theta_{t,n}} H_{t,n} = \nabla_{\theta_{t,n}} \ell_{t,n} + \nabla_{\theta_{t,n}} F_t^{\mathsf{T}} \boldsymbol{p}_{t+1,n}.$$

is indeed the gradient w.r.t. the parameter $\theta_{t,n}$ of each layer $f_{t,n}$. Therefore, taking the iterative update $\theta_{t,n} \leftarrow \theta_{t,n} - H_\theta^{t,n}$ is equivalent to descending along the SGD direction, up to a learning rate scaling. Similarly, setting different precondition matrices $\boldsymbol{M}$ will recover other standard methods. Hence, we conclude the proof.

□

Optimality principle for $\eta_{t,n}^{\text{O-CG}}$. For the completeness, below we provide the optimality principle for the cooperative open-loop information structure $\eta_{t,n}^{\text{O-CG}}$.

Definition 8 (Cooperative optimality principle by $\eta_{t,n}^{\text{O-CG}}$)**.** *A set of strategy, $\{\theta_{t,n}^* : \forall t, n\}$, provides an open-loop optimal solution to the joint optimization (4) if*

$$\theta_{t,1}^*, \cdots, \theta_{t,N}^* = \operatorname*{arg\,min}_{\theta_{t,n}: n \in [N]} \widetilde{H}_t(\boldsymbol{x}_t, \widetilde{\boldsymbol{p}}_{t+1}, \theta_{t,1}, \cdots, \theta_{t,N}),$$
$$\textit{where} \quad \theta_{t,n}^* \equiv \theta_{t,n}^*(\eta_{t,n}^{\textit{O-CG}}) \quad \textit{and} \quad \widetilde{H}_t \coloneqq \textstyle\sum_{n=1}^N \ell_{t,n} + F_t^{\mathsf{T}} \widetilde{\boldsymbol{p}}_{t+1}$$

is the "group" Hamiltonian at stage t. Similar to OLNE, the joint co-state $\widetilde{\boldsymbol{p}}_t$ can be simulated by

$$\widetilde{\boldsymbol{p}}_t = \nabla_{\boldsymbol{x}_t} \widetilde{H}_t, \quad \widetilde{\boldsymbol{p}}_T = \textstyle\sum_{n=1}^N \nabla_{\boldsymbol{x}_T} \phi_n.$$

In this work, we focus on solving the optimality principle inherited in $\eta_{t,n}^{\text{C-CG}}$ as a representative of the CG optimality. Since $\eta_{t,n}^{\text{O-CG}} \subset \eta_{t,n}^{\text{C-CG}}$, the latter captures richer information and tends to perform better in practice, as evidenced by Fig. 3.

D. Missing Derivations in Section 4

D.1. Complete Derivation of the Iterative Updates

Derivation of FNE update. Our goal is to approximately solve the Isaacs-Bellman recursion (6) only up to second-order. Recall that the second-order expansion of $Q_{t,n}$ at some fixed point $(\boldsymbol{x}_t, \theta_{t,n})$ takes the form

$$Q_{t,n} \approx \frac{1}{2} \begin{bmatrix} \mathbf{1} \\ \delta\boldsymbol{x}_t \\ \delta\theta_{t,n} \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} Q_{t,n} & Q_{\boldsymbol{x}}^{t,n\,\mathsf{T}} & Q_\theta^{t,n\,\mathsf{T}} \\ Q_{\boldsymbol{x}}^{t,n} & Q_{\boldsymbol{xx}}^{t,n} & Q_{\theta\boldsymbol{x}}^{t,n\,\mathsf{T}} \\ Q_\theta^{t,n} & Q_{\theta\boldsymbol{x}}^{t,n} & Q_{\theta\theta}^{t,n} \end{bmatrix} \begin{bmatrix} \mathbf{1} \\ \delta\boldsymbol{x}_t \\ \delta\theta_{t,n} \end{bmatrix}, \quad \text{where} \quad \begin{aligned} Q_{\boldsymbol{x}}^{t,n} &= F_{\boldsymbol{x}}^t{}^{\mathsf{T}} V_{\boldsymbol{x}}^{t+1,n} \\ Q_\theta^{t,n} &= F_\theta^t{}^{\mathsf{T}} V_{\boldsymbol{x}}^{t+1,n} + \ell_\theta^{t,n} \\ Q_{\theta\theta}^{t,n} &= F_\theta^t{}^{\mathsf{T}} V_{\boldsymbol{xx}}^{t+1,n} F_\theta^t + \ell_{\theta\theta}^{t,n} \\ Q_{\theta\boldsymbol{x}}^{t,n} &= F_\theta^t{}^{\mathsf{T}} V_{\boldsymbol{xx}}^{t+1,n} F_{\boldsymbol{x}}^t \\ Q_{\boldsymbol{xx}}^{t,n} &= F_{\boldsymbol{x}}^t{}^{\mathsf{T}} V_{\boldsymbol{xx}}^{t+1,n} F_{\boldsymbol{x}}^t \end{aligned} \tag{20}$$

follow standard chain rule (recall $Q_{t,n} \coloneqq \ell_{t,n} + V_{t+1,n} \circ F_t$) with the linearized dynamics $F_\theta^t \equiv \nabla_{\theta_{t,n}} F_t$ and $F_{\boldsymbol{x}}^t \equiv \nabla_{\boldsymbol{x}_t} F_t$. The expansion (20) is a standard quadratic programming, and its analytic solution is given by

$$-\delta\pi_{t,n}^* \equiv -\delta\theta_{t,n}^*(\delta\boldsymbol{x}_t) = -(Q_{\theta\theta}^{t,n})^\dagger (Q_\theta^{t,n} + Q_{\theta\boldsymbol{x}}^{t,n} \delta\boldsymbol{x}_t) =: -(\mathbf{k}_{t,n} + \mathbf{K}_{t,n} \delta\boldsymbol{x}_t).$$

Substituting this solution back to the Isaacs-Bellman recursion gives us the local expression of $V_{t,n}$,

$$V_{t,n} \approx Q_{t,n} - \frac{1}{2}(Q^{t,n}_{\theta})^{\mathsf{T}}(Q^{t,n}_{\theta\theta})^{\dagger}Q^{t,n}_{\theta}. \tag{21}$$

Therefore, the local derivatives of $V_{t,n}$ can be computed by

$$\begin{aligned} V^{t,n}_{\boldsymbol{x}} &= Q^{t,n}_{\boldsymbol{x}} - Q^{t,n}_{\boldsymbol{x}\theta}(Q^{t,n}_{\theta\theta})^{\dagger}Q^{t,n}_{\theta} = Q^{t,n}_{\boldsymbol{x}} - Q^{t,n}_{\boldsymbol{x}\theta}\mathbf{k}_{t,n} \\ V^{t,n}_{\boldsymbol{x}\boldsymbol{x}} &= Q^{t,n}_{\boldsymbol{x}\boldsymbol{x}} - Q^{t,n}_{\boldsymbol{x}\theta}(Q^{t,n}_{\theta\theta})^{\dagger}Q^{t,n}_{\theta\boldsymbol{x}} = Q^{t,n}_{\boldsymbol{x}\boldsymbol{x}} - Q^{t,n}_{\boldsymbol{x}\theta}\mathbf{K}_{t,n}. \end{aligned}$$

Derivation of GR update. We will adopt the same terminology $\boldsymbol{u} \equiv \theta_{t,1}, \boldsymbol{v} \equiv \theta_{t,2}$. Following the procedure as in the FNE case, we can perform the second-order expansion of P_t at some fixed point $(\boldsymbol{x}_t, \boldsymbol{u}, \boldsymbol{v})$. The analytic solution to the corresponding quadratic programming is given by

$$-\begin{bmatrix} \delta\pi^*_{t,1} \\ \delta\pi^*_{t,2} \end{bmatrix} = -\begin{bmatrix} P^t_{\boldsymbol{u}\boldsymbol{u}} & P^t_{\boldsymbol{u}\boldsymbol{v}} \\ P^t_{\boldsymbol{v}\boldsymbol{u}} & P^t_{\boldsymbol{v}\boldsymbol{v}} \end{bmatrix}^{\dagger} \left(\begin{bmatrix} P^t_{\boldsymbol{u}} \\ P^t_{\boldsymbol{v}} \end{bmatrix} + \begin{bmatrix} P^t_{\boldsymbol{u}\boldsymbol{x}} \\ P^t_{\boldsymbol{v}\boldsymbol{x}} \end{bmatrix} \delta\boldsymbol{x}_t \right), \tag{22}$$

where the block-matrices inversion can be expanded using the Schur complement.

$$\begin{bmatrix} P^t_{\boldsymbol{u}\boldsymbol{u}} & P^t_{\boldsymbol{u}\boldsymbol{v}} \\ P^t_{\boldsymbol{v}\boldsymbol{u}} & P^t_{\boldsymbol{v}\boldsymbol{v}} \end{bmatrix}^{\dagger} = \begin{bmatrix} (\overbrace{P^t_{\boldsymbol{u}\boldsymbol{u}} - P^t_{\boldsymbol{u}\boldsymbol{v}}P^{t\,\dagger}_{\boldsymbol{v}\boldsymbol{v}}P^t_{\boldsymbol{v}\boldsymbol{u}}}^{\widetilde{P}^t_{\boldsymbol{u}\boldsymbol{u}}})^{\dagger} & -\widetilde{P}^{t\,\dagger}_{\boldsymbol{u}\boldsymbol{u}}P^t_{\boldsymbol{u}\boldsymbol{v}}P^{t\,\dagger}_{\boldsymbol{v}\boldsymbol{v}} \\ -\widetilde{P}^{t\,\dagger}_{\boldsymbol{v}\boldsymbol{v}}P^t_{\boldsymbol{v}\boldsymbol{u}}P^{t\,\dagger}_{\boldsymbol{u}\boldsymbol{u}} & (\underbrace{P^t_{\boldsymbol{v}\boldsymbol{v}} - P^t_{\boldsymbol{v}\boldsymbol{u}}P^{t\,\dagger}_{\boldsymbol{u}\boldsymbol{u}}P^t_{\boldsymbol{u}\boldsymbol{v}}}_{\widetilde{P}^t_{\boldsymbol{v}\boldsymbol{v}}})^{\dagger} \end{bmatrix}. \tag{23}$$

Hence, (22) becomes

$$\begin{aligned} \begin{bmatrix} \delta\pi^*_{t,1} \\ \delta\pi^*_{t,2} \end{bmatrix} &= \begin{bmatrix} \widetilde{P}^{t\,\dagger}_{\boldsymbol{u}\boldsymbol{u}}(P^t_{\boldsymbol{u}} - P^t_{\boldsymbol{u}\boldsymbol{v}}P^{t\,\dagger}_{\boldsymbol{v}\boldsymbol{v}}P^t_{\boldsymbol{v}}) \\ \widetilde{P}^{t\,\dagger}_{\boldsymbol{v}\boldsymbol{v}}(P^t_{\boldsymbol{v}} - P^t_{\boldsymbol{v}\boldsymbol{u}}P^{t\,\dagger}_{\boldsymbol{u}\boldsymbol{u}}P^t_{\boldsymbol{u}}) \end{bmatrix} + \begin{bmatrix} \widetilde{P}^{t\,\dagger}_{\boldsymbol{u}\boldsymbol{u}}(P^t_{\boldsymbol{u}\boldsymbol{x}} - P^t_{\boldsymbol{u}\boldsymbol{v}}P^{t\,\dagger}_{\boldsymbol{v}\boldsymbol{v}}P^t_{\boldsymbol{v}\boldsymbol{x}}) \\ \widetilde{P}^{t\,\dagger}_{\boldsymbol{v}\boldsymbol{v}}(P^t_{\boldsymbol{v}\boldsymbol{x}} - P^t_{\boldsymbol{v}\boldsymbol{u}}P^{t\,\dagger}_{\boldsymbol{u}\boldsymbol{u}}P^t_{\boldsymbol{u}\boldsymbol{x}}) \end{bmatrix} \delta\boldsymbol{x}_t \\ &= \begin{bmatrix} \widetilde{P}^{t\,\dagger}_{\boldsymbol{u}\boldsymbol{u}}(P^t_{\boldsymbol{u}} - P^t_{\boldsymbol{u}\boldsymbol{v}}\mathbf{I}_t) \\ \widetilde{P}^{t\,\dagger}_{\boldsymbol{v}\boldsymbol{v}}(P^t_{\boldsymbol{v}} - P^t_{\boldsymbol{v}\boldsymbol{u}}\mathbf{k}_t) \end{bmatrix} + \begin{bmatrix} \widetilde{P}^{t\,\dagger}_{\boldsymbol{u}\boldsymbol{u}}(P^t_{\boldsymbol{u}\boldsymbol{x}} - P^t_{\boldsymbol{u}\boldsymbol{v}}\mathbf{L}_t) \\ \widetilde{P}^{t\,\dagger}_{\boldsymbol{v}\boldsymbol{v}}(P^t_{\boldsymbol{v}\boldsymbol{x}} - P^t_{\boldsymbol{v}\boldsymbol{u}}\mathbf{K}_t) \end{bmatrix} \delta\boldsymbol{x}_t \\ &=: \begin{bmatrix} \widetilde{\mathbf{k}}_t \\ \widetilde{\mathbf{I}}_t \end{bmatrix} + \begin{bmatrix} \widetilde{\mathbf{K}}_t \\ \widetilde{\mathbf{L}}_t \end{bmatrix} \delta\boldsymbol{x}_t, \end{aligned}$$

where we denote the non-cooperative iterative update for Player 1 and 2 respectively by

$$\begin{aligned} \delta\boldsymbol{u}_t(\delta\boldsymbol{x}_t) &= \mathbf{k}_t + \mathbf{K}_t\delta\boldsymbol{x}_t, \quad \text{where } \mathbf{k}_t := P^{t\,\dagger}_{\boldsymbol{u}\boldsymbol{u}}P^t_{\boldsymbol{u}} \quad \text{and} \quad \mathbf{K}_t := P^{t\,\dagger}_{\boldsymbol{u}\boldsymbol{u}}P^t_{\boldsymbol{u}\boldsymbol{x}}, \\ \delta\boldsymbol{v}_t(\delta\boldsymbol{x}_t) &= \mathbf{I}_t + \mathbf{L}_t\delta\boldsymbol{x}_t, \quad \text{where } \mathbf{I}_t := P^{t\,\dagger}_{\boldsymbol{v}\boldsymbol{v}}P^t_{\boldsymbol{v}} \quad \text{and} \quad \mathbf{L}_t := P^{t\,\dagger}_{\boldsymbol{v}\boldsymbol{v}}P^t_{\boldsymbol{v}\boldsymbol{x}}. \end{aligned}$$

Substituting this solution back to the GR Bellman equation gives the local expression of W_t,

$$W_t \approx P_t - \frac{1}{2}\begin{bmatrix} P^t_{\boldsymbol{u}} \\ P^t_{\boldsymbol{v}} \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} P^t_{\boldsymbol{u}\boldsymbol{u}} & P^t_{\boldsymbol{u}\boldsymbol{v}} \\ P^t_{\boldsymbol{v}\boldsymbol{u}} & P^t_{\boldsymbol{v}\boldsymbol{v}} \end{bmatrix}^{\dagger} \begin{bmatrix} P^t_{\boldsymbol{u}} \\ P^t_{\boldsymbol{v}} \end{bmatrix}. \tag{24}$$

Finally, taking the derivatives yields the formula for updating the derivatives of W_t,

$$W^t_{\boldsymbol{x}} = P^t_{\boldsymbol{x}} - \frac{1}{2}\left(P^t_{\boldsymbol{x}\boldsymbol{u}}\widetilde{\mathbf{k}}_t + P^t_{\boldsymbol{x}\boldsymbol{v}}\widetilde{\mathbf{I}}_t + \widetilde{\mathbf{K}}^{\mathsf{T}}_t P^t_{\boldsymbol{u}} + \widetilde{\mathbf{L}}^{\mathsf{T}}_t P^t_{\boldsymbol{v}}\right) \quad \text{and} \quad W^t_{\boldsymbol{x}\boldsymbol{x}} = P^{t,n}_{\boldsymbol{x}\boldsymbol{x}} - P^t_{\boldsymbol{x}\boldsymbol{u}}\widetilde{\mathbf{K}}_t - P^t_{\boldsymbol{x}\boldsymbol{v}}\widetilde{\mathbf{L}}_t, \tag{25}$$

which is much complex than (10).

D.2. Kronecker Factorization and Proof of Proposition 5

We first provide some backgrounds for the Kronecker factorization (KFAC; Martens & Grosse (2015)). KFAC relies on the fact that for an affine mapping layer, *i.e.* $\boldsymbol{z}_{t+1} = f_t(\boldsymbol{z}_t, \theta_t) := \boldsymbol{W}_t\boldsymbol{z}_t + \boldsymbol{b}_t,\ \theta_t := \text{vec}([\boldsymbol{W}_t, \boldsymbol{b}_t])$, the gradient of the training objective L w.r.t. the parameter θ_t admits a compact factorization,

$$\nabla_{\theta_t} L = \nabla_{\theta_t} f_t^{\mathsf{T}} \nabla_{\boldsymbol{z}_{t+1}} L = \boldsymbol{z}_t \otimes \nabla_{\boldsymbol{z}_{t+1}} L,$$

where $\otimes$ denotes the Kronecker product. With this, the layer-wise Fisher information matrix, or equivalently the Gauss-Newton (GN) matrix, for classification can be approximated with

$$\mathbb{E}[\nabla_{\theta_t} L \nabla_{\theta_t} L^\mathsf{T}] = \mathbb{E}[(\boldsymbol{z}_t \otimes \nabla_{\boldsymbol{z}_{t+1}} L)(\boldsymbol{z}_t \otimes \nabla_{\boldsymbol{z}_{t+1}} L)^\mathsf{T}] \approx \mathbb{E}[\boldsymbol{z}_t \boldsymbol{z}_t^\mathsf{T}] \otimes \mathbb{E}[\nabla_{\boldsymbol{z}_{t+1}} L \nabla_{\boldsymbol{z}_{t+1}} L^\mathsf{T}].$$

We can adopt this factorization to our setup by first recalling from our proof of Proposition 2 (see Appendix C) that $(\nabla_{\theta_t} L, \nabla_{\boldsymbol{z}_{t+1}} L)$ are interchangeable with $(H^t_\theta, \boldsymbol{p}_{t+1})$, or equivalently $(H^t_\theta, H^{t+1}_{\boldsymbol{z}})$. Hence, the GN approximation of $\mathbb{E}[H^t_{\theta\theta}]$ can be factorized by

$$\mathbb{E}[H^t_\theta {H^t_\theta}^\mathsf{T}] \approx \mathbb{E}[\boldsymbol{z}_t \boldsymbol{z}_t^\mathsf{T}] \otimes \mathbb{E}[\boldsymbol{p}_{t+1} \boldsymbol{p}_{t+1}^\mathsf{T}] = \mathbb{E}[\boldsymbol{z}_t \boldsymbol{z}_t^\mathsf{T}] \otimes \mathbb{E}[H^{t+1}_{\boldsymbol{z}} H^{t+1\,\mathsf{T}}_{\boldsymbol{z}}]. \tag{26}$$

Equation (26) suggests that KFAC factorizes the parameter curvature with two smaller matrices using the activation state $\boldsymbol{z}_t$ and the derivative of *some optimality* (in this case the Hamiltonian H) w.r.t. $\boldsymbol{z}_{t+1}$. The main advantage of this factorization is to exploit the following formula,

$$(\boldsymbol{A} \otimes \boldsymbol{B})^\dagger \mathrm{vec}(\boldsymbol{W}) = (\boldsymbol{A}^\dagger \otimes \boldsymbol{B}^\dagger)\mathrm{vec}(\boldsymbol{W}) = \mathrm{vec}(\boldsymbol{B}^\dagger \boldsymbol{W} \boldsymbol{A}^{-\mathsf{T}}), \tag{27}$$

which allows one to efficiently inverse the parameter curvature with two smaller matrices.

Now, let us proceed to the proof of Proposition 5. First notice that for the shared dynamics considered in Fig. 2, we have

$$F_t(\boldsymbol{x}_t, \boldsymbol{u}, \boldsymbol{v}) := \begin{bmatrix} f_{t,1}(\boldsymbol{z}_1, \boldsymbol{u}) \\ f_{t,2}(\boldsymbol{z}_2, \boldsymbol{v}) \end{bmatrix} = \begin{bmatrix} f_{t,1}(\cdot, \boldsymbol{u}) & \mathbf{0} \\ \mathbf{0} & f_{t,2}(\cdot, \boldsymbol{v}) \end{bmatrix} \begin{bmatrix} \boldsymbol{z}_1 \\ \boldsymbol{z}_2 \end{bmatrix},$$

which resembles the affine mapping concerned by KFAC. This motivates the following approximation,

$$\mathbb{E}[P^t_\theta {P^t_\theta}^\mathsf{T}] \approx \mathbb{E}[\boldsymbol{x}_t \boldsymbol{x}_t^\mathsf{T}] \otimes \mathbb{E}[W^{t+1}_{\boldsymbol{x}} {W^{t+1}_{\boldsymbol{x}}}^\mathsf{T}]. \tag{28}$$

Similar to (26), this approximation (28) factorizes the GN matrix with the MPDG state $\boldsymbol{x}_t$ and the derivative of an optimality (in this case it becomes the GR value function W_{t+1}) w.r.t. $\boldsymbol{x}_{t+1}$.

If we denote the derivatives w.r.t. the outputs of $f_{t,1}$ and $f_{t,2}$ by $\boldsymbol{g}_1$ and $\boldsymbol{g}_2$, *i.e.* $W^{t+1}_{\boldsymbol{x}} := [\boldsymbol{g}_1, \boldsymbol{g}_2]^\mathsf{T}$, and rewrite $\boldsymbol{x}_t := [\boldsymbol{z}_1, \boldsymbol{z}_2]^\mathsf{T}$, then (28) can be expanded by

$$\begin{aligned} \mathbb{E}[\boldsymbol{x}_t {\boldsymbol{x}_t}^\mathsf{T}] &= \begin{bmatrix} \mathbb{E}[\boldsymbol{z}_1 {\boldsymbol{z}_1}^\mathsf{T}] & \mathbb{E}[\boldsymbol{z}_1 {\boldsymbol{z}_2}^\mathsf{T}] \\ \mathbb{E}[\boldsymbol{z}_2 {\boldsymbol{z}_1}^\mathsf{T}] & \mathbb{E}[\boldsymbol{z}_2 {\boldsymbol{z}_2}^\mathsf{T}] \end{bmatrix} =: \begin{bmatrix} A_{\boldsymbol{uu}} & A_{\boldsymbol{uv}} \\ A_{\boldsymbol{vu}} & A_{\boldsymbol{vv}} \end{bmatrix} \\ \mathbb{E}[W^{t+1}_{\boldsymbol{x}} {W^{t+1}_{\boldsymbol{x}}}^\mathsf{T}] &= \begin{bmatrix} \mathbb{E}[\boldsymbol{g}_1 {\boldsymbol{g}_1}^\mathsf{T}] & \mathbb{E}[\boldsymbol{g}_1 {\boldsymbol{g}_2}^\mathsf{T}] \\ \mathbb{E}[\boldsymbol{g}_2 {\boldsymbol{g}_1}^\mathsf{T}] & \mathbb{E}[\boldsymbol{g}_2 {\boldsymbol{g}_2}^\mathsf{T}] \end{bmatrix} =: \begin{bmatrix} B_{\boldsymbol{uu}} & B_{\boldsymbol{uv}} \\ B_{\boldsymbol{vu}} & B_{\boldsymbol{vv}} \end{bmatrix}. \end{aligned}$$

Their inverse matrices are given by the Schur component.

$$\begin{aligned} \begin{bmatrix} A_{\boldsymbol{uu}} & A_{\boldsymbol{uv}} \\ A_{\boldsymbol{vu}} & A_{\boldsymbol{vv}} \end{bmatrix}^\dagger &= \begin{bmatrix} \widetilde{A}^\dagger_{\boldsymbol{uu}} & -\widetilde{A}^\dagger_{\boldsymbol{uu}} A_{\boldsymbol{uv}} A^\dagger_{\boldsymbol{vv}} \\ -\widetilde{A}^\dagger_{\boldsymbol{vv}} A_{\boldsymbol{vu}} A^\dagger_{\boldsymbol{uu}} & \widetilde{A}^\dagger_{\boldsymbol{vv}} \end{bmatrix}, \quad \text{where} \begin{cases} \widetilde{A}_{\boldsymbol{uu}} := A_{\boldsymbol{uu}} - A_{\boldsymbol{uv}} A^\dagger_{\boldsymbol{vv}} A^\mathsf{T}_{\boldsymbol{uv}} \\ \widetilde{A}_{\boldsymbol{vv}} := A_{\boldsymbol{vv}} - A_{\boldsymbol{vu}} A^\dagger_{\boldsymbol{uu}} A^\mathsf{T}_{\boldsymbol{vu}} \end{cases} \\ \begin{bmatrix} B_{\boldsymbol{uu}} & B_{\boldsymbol{uv}} \\ B_{\boldsymbol{vu}} & B_{\boldsymbol{vv}} \end{bmatrix}^\dagger &= \begin{bmatrix} \widetilde{B}^\dagger_{\boldsymbol{uu}} & -\widetilde{B}^\dagger_{\boldsymbol{uu}} B_{\boldsymbol{uv}} B^\dagger_{\boldsymbol{vv}} \\ -\widetilde{B}^\dagger_{\boldsymbol{vv}} B_{\boldsymbol{vu}} B^\dagger_{\boldsymbol{uu}} & \widetilde{B}^\dagger_{\boldsymbol{vv}} \end{bmatrix}, \quad \text{where} \begin{cases} \widetilde{B}_{\boldsymbol{uu}} := B_{\boldsymbol{uu}} - B_{\boldsymbol{uv}} B^\dagger_{\boldsymbol{vv}} B^\mathsf{T}_{\boldsymbol{uv}} \\ \widetilde{B}_{\boldsymbol{vv}} := B_{\boldsymbol{vv}} - B_{\boldsymbol{vu}} B^\dagger_{\boldsymbol{uu}} B^\mathsf{T}_{\boldsymbol{vu}} \end{cases} \end{aligned} \tag{29}$$

With all these, the cooperative open gain can be computed with the formula (27),

$$\left(\mathbb{E}[\boldsymbol{x}_t \boldsymbol{x}_t^\mathsf{T}] \otimes \mathbb{E}[W^{t+1}_{\boldsymbol{x}} {W^{t+1}_{\boldsymbol{x}}}^\mathsf{T}]\right)^\dagger \mathrm{vec}(\begin{bmatrix} P^t_{\boldsymbol{u}} & \mathbf{0} \\ \mathbf{0} & P^t_{\boldsymbol{v}} \end{bmatrix}) = \mathrm{vec}\left(\begin{bmatrix} B_{\boldsymbol{uu}} & B_{\boldsymbol{uv}} \\ B_{\boldsymbol{vu}} & B_{\boldsymbol{vv}} \end{bmatrix}^\dagger \begin{bmatrix} P^t_{\boldsymbol{u}} & \mathbf{0} \\ \mathbf{0} & P^t_{\boldsymbol{v}} \end{bmatrix} \begin{bmatrix} A_{\boldsymbol{uu}} & A_{\boldsymbol{uv}} \\ A_{\boldsymbol{vu}} & A_{\boldsymbol{vv}} \end{bmatrix}^{-\mathsf{T}}\right). \tag{30}$$

Substituting (29) into (30), after some algebra we will arrive at *the KFAC of the cooperative open gain* suggested in (15).

$$\begin{aligned} \widetilde{\mathbf{k}}_t \approx& \mathrm{vec}(\widetilde{B}^\dagger_{\boldsymbol{uu}} P^t_{\boldsymbol{u}} \widetilde{A}^{-\mathsf{T}}_{\boldsymbol{uu}} + \widetilde{B}^\dagger_{\boldsymbol{uu}} B_{\boldsymbol{uv}} B^\dagger_{\boldsymbol{vv}} P^t_{\boldsymbol{v}} (\widetilde{A}^\dagger_{\boldsymbol{uu}} A_{\boldsymbol{uv}} A^\dagger_{\boldsymbol{vv}})^\mathsf{T}) \\ =& \mathrm{vec}(\widetilde{B}^\dagger_{\boldsymbol{uu}} (P^t_{\boldsymbol{u}} + B_{\boldsymbol{uv}} B^\dagger_{\boldsymbol{vv}} P^t_{\boldsymbol{v}} A^{-\mathsf{T}}_{\boldsymbol{vv}} A^\mathsf{T}_{\boldsymbol{uv}}) \widetilde{A}^{-\mathsf{T}}_{\boldsymbol{uu}}) \\ =& \mathrm{vec}(\widetilde{B}^\dagger_{\boldsymbol{uu}} (P^t_{\boldsymbol{u}} + B_{\boldsymbol{uv}} \mathrm{vec}^\dagger(\mathbf{I}_t) A^\mathsf{T}_{\boldsymbol{uv}}) \widetilde{A}^{-\mathsf{T}}_{\boldsymbol{uu}}), \end{aligned} \tag{31}$$

where the last equality follows by another KFAC approximation $\mathbf{I}_t \approx (A_{\boldsymbol{vv}} \otimes B_{\boldsymbol{vv}})^\dagger \text{vec}(P^t_{\boldsymbol{v}}) = \text{vec}(B^\dagger_{\boldsymbol{vv}} P^t_{\boldsymbol{v}} A^{-\mathsf{T}}_{\boldsymbol{vv}})$. Finally, recalling the expression, $\widetilde{\mathbf{k}}_t := \widetilde{P}^{t\,\dagger}_{\boldsymbol{uu}}(P^t_{\boldsymbol{u}} - P^t_{\boldsymbol{uv}}\mathbf{I}_t)$, from (11) and rewriting (31) by

$$\begin{aligned}\widetilde{\mathbf{k}}_t \approx &\text{vec}(\widetilde{B}^\dagger_{\boldsymbol{uu}}(P^t_{\boldsymbol{u}} + B_{\boldsymbol{uv}}\text{vec}^\dagger(\mathbf{I}_t)A^{\mathsf{T}}_{\boldsymbol{uv}})\widetilde{A}^{-\mathsf{T}}_{\boldsymbol{uu}}) \\ =&(\widetilde{A}_{\boldsymbol{uu}} \otimes \widetilde{B}_{\boldsymbol{uu}})^\dagger \text{vec}(P^t_{\boldsymbol{u}} + B_{\boldsymbol{uv}}\text{vec}^\dagger(\mathbf{I}_t)A^{\mathsf{T}}_{\boldsymbol{uv}})\end{aligned}$$

imply the KFAC representation $\widetilde{P}^t_{\boldsymbol{uu}} \approx \widetilde{A}_{\boldsymbol{uu}} \otimes \widetilde{B}_{\boldsymbol{uu}}$ in (14). Hence we conclude the proof. □

D.3. Proof of Theorem 6

We first show that setting $P^t_{\boldsymbol{uv}} := \mathbf{0}$ in the update (11) yields (9). To begin, observe that when $P^t_{\boldsymbol{uv}}$ vanishes, the cooperative gains $(\widetilde{\mathbf{k}}_t, \widetilde{\mathbf{K}}_t)$ appearing in (11) degenerate to $\widetilde{\mathbf{k}}_t = P^{t\,\dagger}_{\boldsymbol{uu}} P^t_{\boldsymbol{u}}$ and $\widetilde{\mathbf{K}}_t = P^{t\,\dagger}_{\boldsymbol{uu}} P^t_{\boldsymbol{ux}}$. Therefore, it is sufficient to prove the following result.[4]

Lemma 9. *Suppose $Q_{t,n}$ in (6) and P_t in (7) are expanded up to second-order along the same local trajectory $\{(\boldsymbol{x}_t, \theta_{t,n}, \cdots, \theta_{t,N}) : \forall t \in [T]\}$, then we will have the following relations when $P^t_{\boldsymbol{uv}} := \mathbf{0}$ at all stages.*

$$\forall t, \quad Q^{t,1}_\theta = P^t_{\boldsymbol{u}}, \quad Q^{t,1}_{\theta\theta} = P^t_{\boldsymbol{uu}}, \quad Q^{t,1}_{\theta\boldsymbol{x}} = P^t_{\boldsymbol{ux}}, \quad Q^{t,2}_\theta = P^t_{\boldsymbol{v}}, \quad Q^{t,2}_{\theta\theta} = P^t_{\boldsymbol{vv}}, \quad Q^{t,2}_{\theta\boldsymbol{x}} = P^t_{\boldsymbol{vx}}, \tag{32}$$

where $(\boldsymbol{u}_t, \boldsymbol{v}_t) \equiv (\theta_{t,1}, \theta_{t,2})$ denotes the actions for Player 1 and 2. Furthermore, we have

$$\forall t, \quad W_t = \textstyle\sum_{n=1}^N V_{t,n}. \tag{33}$$

Proof. We will proceed the proof by induction. At the terminal stage $T-1$, we have

$$P_{T-1} = \sum_{n=1}^2 \ell_{T-1,n} + W_T \circ F_{T-1} = \sum_{n=1}^2 (\ell_{T-1,n} + \phi_n \circ F_{T-1}) = \sum_{n=1}^2 Q_{T-1,n},$$

since $\phi_n = V_{T,n}$. This implies that when solving the second-order expansion for $\pi_{T-1,1}$ and $\pi_{T-1,2}$, we will have

$$\min_{\pi_{T-1,1}, \pi_{T-1,2}} P_{T-1} = \min_{\pi_{T-1,1}} Q_{T-1,1} + \min_{\pi_{T-1,2}} Q_{T-1,2}$$

since the cross-correlation matrix $P^{T-1}_{\boldsymbol{uv}}$ is discarded. Therefore, all equalities in (32) hold at this stage. Furthermore, substituting $P^{T-1}_{\boldsymbol{uv}} := \mathbf{0}$ into (24) yields the following GR value function

$$\begin{aligned}W_{T-1} &= P_{T-1} - \frac{1}{2}\left((P^{T-1}_{\boldsymbol{u}})^{\mathsf{T}}(P^{T-1}_{\boldsymbol{uu}})^\dagger P^{T-1}_{\boldsymbol{u}} + (P^{T-1}_{\boldsymbol{v}})^{\mathsf{T}}(P^{T-1}_{\boldsymbol{vv}})^\dagger P^{T-1}_{\boldsymbol{v}}\right) \\ &= \sum_{n=1}^2 \left(Q_{T-1,n} - \frac{1}{2}(Q^{T-1,n}_\theta)^{\mathsf{T}}(Q^{T-1,n}_{\theta\theta})^\dagger Q^{T-1,n}_\theta\right) = \sum_{n=1}^2 V_{T-1,n}.\end{aligned}$$

So (33) also holds. Now, suppose (32, 33) hold at $t+1$, then

$$P_t = \sum_{n=1}^2 \ell_{t,n} + W_{t+1} \circ F_t = \sum_{n=1}^2 (\ell_{t,n} + V_{t+1,n} \circ F_t) = \sum_{n=1}^2 Q_{t,n}.$$

Together with $P^t_{\boldsymbol{uv}} := \mathbf{0}$, we can see that all equalities in (32) hold. Furthermore, it implies that

$$W_t = P_t - \frac{1}{2}\left(P^{t\,\mathsf{T}}_{\boldsymbol{u}} P^{t\,\dagger}_{\boldsymbol{uu}} P^t_{\boldsymbol{u}} + P^{t\,\mathsf{T}}_{\boldsymbol{v}} P^{t\,\dagger}_{\boldsymbol{vv}} P^t_{\boldsymbol{v}}\right) = \sum_{n=1}^2 \left(Q_{t,n} - \frac{1}{2}(Q^{t,n}_\theta)^{\mathsf{T}}(Q^{t,n}_{\theta\theta})^\dagger Q^{t,n}_\theta\right) = \sum_{n=1}^2 V_{t,n}.$$

Hence we conclude the proof. □

Next, we proceed to the second case, which suggests that running (9) with $(Q^{t,n}_{\theta\boldsymbol{x}}, Q^{t,n}_{\theta\theta}) := (\mathbf{0}, \boldsymbol{I})$ yields SGD. Since the FNE update in this case degenerates to $\delta\pi^*_{t,n} = Q^{t,n}_\theta$, it is sufficient to prove the following lemma.

[4] We consider the two-player setup for simplicity, yet the methodology applies to the multi-player setup.

Lemma 10. *Suppose $H_{t,n}$ in (5) and $Q_{t,n}$ in (6) are expanded up to second-order along the same local trajectory $\{(\boldsymbol{x}_t, \theta_{t,n}, \cdots, \theta_{t,N}) : \forall t \in [T]\}$, then we will have the following relations when $(Q^{t,n}_{\theta\boldsymbol{x}}, Q^{t,n}_{\theta\theta}) := (\mathbf{0}, \boldsymbol{I})$ for all stages.*

$$\forall t, \quad Q^{t,n}_{\theta} = H^{t,n}_{\theta}, \quad V^{t,n}_{\boldsymbol{x}} = \boldsymbol{p}_{t,n}. \tag{34}$$

Proof. Again, we will proceed the proof by induction. First, notice that $V^{T,n}_{\boldsymbol{x}} = \phi^n_{\boldsymbol{x}} = \boldsymbol{p}_{T,n}$ no matter whether or not $Q^{t,n}_{\theta\boldsymbol{x}}$ and $Q^{t,n}_{\theta\theta}$ degenerate. At the terminal stage $T-1$, we have

$$Q^{T-1,n}_{\theta} = \ell^{T-1,n}_{\theta} + (F^{T-1}_{\theta})^{\mathsf{T}} V^{T,n}_{\boldsymbol{x}} = \ell^{T-1,n}_{\theta} + (F^{T-1}_{\theta})^{\mathsf{T}} \boldsymbol{p}_{T,n} = H^{T-1,n}_{\theta}.$$

Also, when $Q^{T-1,n}_{\theta\boldsymbol{x}} := \mathbf{0}$, (10) becomes

$$V^{T-1,n}_{\boldsymbol{x}} = Q^{T-1,n}_{\boldsymbol{x}} = (F^{T-1}_{\boldsymbol{x}})^{\mathsf{T}} V^{T,n}_{\boldsymbol{x}} = (F^{T-1}_{\boldsymbol{x}})^{\mathsf{T}} \boldsymbol{p}_{T,n} = H^{T-1,n}_{\boldsymbol{x}} = \boldsymbol{p}_{T-1,n}.$$

Hence, (34) holds at $T-1$. Now, suppose these relations hold at $t+1$, then

$$Q^{t,n}_{\theta} = \ell^{t,n}_{\theta} + (F^{t}_{\theta})^{\mathsf{T}} V^{t+1,n}_{\boldsymbol{x}} = \ell^{t,n}_{\theta} + (F^{t}_{\theta})^{\mathsf{T}} \boldsymbol{p}_{t+1,n} = H^{t,n}_{\theta}$$

and similarly

$$V^{t,n}_{\boldsymbol{x}} = Q^{t,n}_{\boldsymbol{x}} = (F^{t}_{\boldsymbol{x}})^{\mathsf{T}} V^{t+1,n}_{\boldsymbol{x}} = (F^{t}_{\boldsymbol{x}})^{\mathsf{T}} \boldsymbol{p}_{t+1,n} = H^{t,n}_{\boldsymbol{x}} = \boldsymbol{p}_{t,n}.$$

Hence, we conclude the proof. □

Finally, the last case follows readily by combining Lemma 9 and 10, so we conclude all proofs. □

E. More on the Experiments

All experiments are run with Pytorch on the GPU machines, including GTX 1080 TI, GTX 2070, and TITAN RTX. We preprocessed all datasets with standardization. We also perform data augmentation when training CIFAR100. Below we detail the setup for each experiment.

Classification (Table 2 and 3). For CIFAR10 and CIFAR100, we use standard implementation of ResNet18 from `https://pytorch.org/hub/pytorch_vision_resnet/`. As for SVHN and MNIST, the residual network consists of 3 residual blocks. The residual block shares a similar architecture in Fig. 2 except with the identity shortcut mapping and without BN. We use 3×3 kernels for all convolution filters. The number of feature maps in the convolution filters is set to 12 and 16 respectively for MNIST and SVHN. Meanwhile, the inception network consists of a convolution layer followed by an inception block (see Fig. 6), another convolution layer, and two fully-connected layers. Regarding the hyper-parameters used in baselines, we select them from an appropriate search space detailed in Table 9. We use the implementation in `https://github.com/Thrandis/EKFAC-pytorch` for EKFAC and implement our own EMSA in PyTorch since the official code released from Li et al. (2017a) does not support GPU parallelization.

Table 9. Hyper-parameter search in Table 2

Standard Baselines	Learning Rate (LR)
SGD	(7e-2, 5e-1)
Adam & RMSprop	(7e-4, 1e-2)
EKFAC	(1e-2, 3e-1)

Ablation study (Fig. 5) Each grid in Fig. 5 corresponds to a distinct combination of baseline and dataset. Its numerical value reports the performance difference between the following two training processes.

- Accuracy of the baseline run with the best-tuned configuration which we report in Table 2 and 3.
- Accuracy of DGNOpt with its parameter curvature set to the precondition matrix implied by the above best-tuned setup.

For instance, suppose the learning rate of EKFAC on MNIST is best-tuned to 0.01, then we simply set $Q^{t,n}_{\theta\theta} \approx 0.01 \times Q^{t,n}_{\theta} {Q^{t,n}_{\theta}}^{\mathsf{T}}$ for all t. From Theorem 6, these two training procedures only differ in the presence of $Q^{t,n}_{\theta\boldsymbol{x}}$, which allows EKFAC to adjust its update based on the change of $\boldsymbol{x}_t \in \eta^{\mathsf{C}}_{t,n}$.

Runtime and memory complexity (Fig. 7). The numerical values are measured on the GTX 2070.

Feedback analysis (Fig. 8). We use the same inception-based network in Table 3.

Remark for EMSA (Footnote 3). Extended Method of Successive Approximations (EMSA) was originally proposed by Li et al. (2017a) as an OCP-inspired method for training *feedforward* networks. It considers the following minimization,

$$\theta_t^* = \arg\min H_t^\rho\left(\boldsymbol{z}_t, \boldsymbol{z}_{t+1}, \boldsymbol{p}_t, \boldsymbol{p}_{t+1}, \theta_t\right),$$
$$\text{where } H_t^\rho\left(\boldsymbol{z}_t, \boldsymbol{z}_{t+1}, \boldsymbol{p}_t, \boldsymbol{p}_{t+1}, \theta_t\right) := H_t\left(\boldsymbol{z}_t, \boldsymbol{p}_{t+1}, \theta_t\right) + \frac{1}{2}\rho \left\|\boldsymbol{z}_{t+1} - f_t(\boldsymbol{z}_t, \theta_t)\right\|_2 + \frac{1}{2}\rho \left\|\boldsymbol{p}_t - \nabla_{\boldsymbol{z}_t} H_t\right\|_2 \tag{35}$$

essentially augments the original Hamiltonian H_t with the feasibility constraints on both forward states and backward co-states. EMSA solves the minimization (35) with L-BFGS per layer at each training iteration. In Table 2 and 3, we extend their formula to accept $H_{t,n}$. Due to the feasibility constraints, the resulting modified Hamiltonian $H_{t,n}^\rho$ depends additionally on $\boldsymbol{x}_{t+1}$ and $\boldsymbol{p}_{t,n}$; hence being different from the original Hamiltonian $H_{t,n}$. As a result, the ablation analysis using Theorem 6 is not applicable for EMSA.

Cooperative training (Fig. 9, Fig. 11, and Table 4). The network consists of 4 convolutions followed by 2 fully-connected layers, and is activated by ReLU. We use 3×3 kernels with 32 feature maps for all convolutions and set the batch size to 128.

Adaptive alignment with bandit (Fig. 10 and Table 5). We use the same ResNet18 as in classification for CIFAR10, and a smaller residual network with 1 residual block for SVHN. The residual block shares the same architecture as in Fig. 2 except without BN. All convolution layers use 3×3 kernels with 12 feature maps. Again, the batch size is set to 128. Note that in this experiment we use a slightly larger learning rate compared with the one used in Table 2. While DGNOpt achieves better final accuracies for both setups, in practice, the former tends to amplify the stabilization when we enlarge the information structure during training. Hence, it differentiates DGNOpt from other baselines.

Alg. 2 presents the pseudo-code of how DGNOpt can be integrated with any generic bandit-based algorithm (marked as blue). For completeness, we also provide the pseudo-code of EXP3++ in Alg. 3. We refer readers to Seldin & Slivkins (2014) for the definition of $\xi_k(m)$ and η_k (do not confuse with $\eta_{t,n}$ in the main context).

Algorithm 2 DGNOpt with Multi-Armed Bandit (MAB)

```
Input: dataset $\mathcal{D}$, network $\{f_i(\cdot,\theta_i)\}$, number of alignments $M$
Initialize the multi-armed bandit MAB.init($M$)
repeat
    Draw an alignment $m \leftarrow$ MAB.sample().
    Construct $F \equiv \{F_t : t \in [T]\}$ according to $m$.
    Compute $\boldsymbol{x}_t$ by propagating $\boldsymbol{x}_0 \sim \mathcal{D}$ through $F$
    for $t = T{-}1$ to 0 do                         ▷ Solve FNE or GR
        Solve the update $\delta\pi^*_{t,n}$ with (9) or (11)
        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)
    end for
    Set $\boldsymbol{x}'_0 = \boldsymbol{x}_0$
    for $t = 0$ to $T{-}1$ do                        ▷ Update parameter
        Apply $\theta_{t,n} \leftarrow \theta_{t,n} - \delta\pi^*_{t,n}(\delta\boldsymbol{x}_t)$ with $\delta\boldsymbol{x}_t = \boldsymbol{x}'_t - \boldsymbol{x}_t$
        Compute $\boldsymbol{x}'_{t+1} = F_t(\boldsymbol{x}'_t, \theta_{t,1}, \cdots, \theta_{t,N})$
    end for
    Compute the accuracy $r$ on validation set.
    Run MAB.update($r$).
until converges
```

Algorithm 3 EXP3++ (Seldin & Slivkins, 2014)

```
function init($M$)
    $(k, M) \leftarrow (1, M)$
    $\forall m, L_k(m) = 0$
end function

function sample()
    $\forall m, \epsilon_k(m) = \min\{\frac{1}{2M}, \frac{1}{2}\sqrt{\frac{\ln M}{kM}}, \xi_k(m)\}$
    $\forall m, \rho_k(m) = e^{-\eta_k L_k(m)} / \sum_{m'} e^{-\eta_k L_k(m')}$
    $\forall m, \tilde{\rho}_k(m) = (1 - \sum_{m'} \epsilon_k(m'))\rho_k(m) + \epsilon_k(m)$
    Sample action according to $\tilde{\rho}_k(m)$
end function

function update($r^m_k$)
    $\ell^m_k = (1 - r^m_k)/\tilde{\rho}_k(m)$
    $L_{k+1}(m) = L_k(m) + \ell^m_k$
    $k \leftarrow k + 1$
end function
```

Additional Experiments.

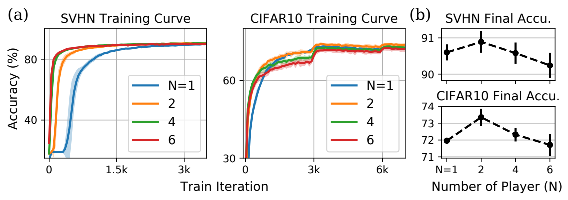

Figure 13. (a) Training curve and (b) final accuracy as we vary the number of player (N) as a hyper-parameter of game-extended EKFAC. Similar to Fig. 9, we also observe that N=2 gives the best final accuracy on both datasets.