Maximum Principle Based Algorithms for Deep Learning

Qianxiao Li*[1,2], Long Chen†[5], Cheng Tai‡[2,4,5], and Weinan E§[2,3,4,5]

[1] *Institute of High Performance Computing, Singapore*
[2] *The Program in Applied and Computational Mathematics, Princeton University, USA*
[3] *Department of Mathematics, Princeton University, USA*
[4] *Beijing Institute of Big Data Research, Beijing, China*
[5] *Center for Data Science, Peking University, Beijing, China*

Abstract

The continuous dynamical system approach to deep learning is explored in order to devise alternative frameworks for training algorithms. Training is recast as a control problem and this allows us to formulate necessary optimality conditions in continuous time using the Pontryagin's maximum principle (PMP). A modification of the method of successive approximations is then used to solve the PMP, giving rise to an alternative training algorithm for deep learning. This approach has the advantage that rigorous error estimates and convergence results can be established. We also show that it may avoid some pitfalls of gradient-based methods, such as slow convergence on flat landscapes near saddle points. Furthermore, we demonstrate that it obtains favorable initial convergence rate per-iteration, provided Hamiltonian maximization can be efficiently carried out - a step which is still in need of improvement. Overall, the approach opens up new avenues to attack problems associated with deep learning, such as trapping in slow manifolds and inapplicability of gradient-based methods for discrete trainable variables.

1 Introduction

Supervised learning using deep neural networks has become an increasingly successful tool in modern machine learning applications [1, 2, 3]. Efficient training methods of very deep neural networks, however, remain an active area of research. The most commonly applied training method is stochastic gradient descent [4, 5] and its variants [6, 7, 8, 9], where incremental updates to the trainable parameters are performed using gradient information computed via back-propagation [10, 11]. While efficient to implement, there are two main issues with these gradient based methods. First, the updates are incremental and tend to be slow, especially in the initial stages of the training. Second, the back-propagation and gradient updates are strictly sequential and hence training is not easily parallelized. These observations points to the question of whether there exists alternative training methods tailored to deep neural networks.

In this series of papers, we introduce an alternative approach by exploring the control theory viewpoint of deep learning [12, 13]. Our focus will be on ideas and algorithms derived from the powerful Pontryagin's maximum principle [14, 15], which has two major components: The Hamiltonian dynamics and the condition that at each time the optimal parameters maximize the Hamiltonian. The second component suggests that optimization can be performed independently at different times. One can also derive an explicit error control estimate based on the maximum principle (see Lemma 1 below).

In this first paper, we will consider the simplest context in which the deep neural networks are replaced by continuous (or discretized) dynamical systems, and devise numerical algorithms that are based on the optimality conditions in Pontryagin's maximum principle. This leads to a new approach for training deep learning models that have certain advantages, such as fast initial descent and ease for parallelization. An additional advantage is that one has a good control of the error through explicit estimates.

*qianxiao@math.princeton.edu
†xidonglc@pku.edu.cn
‡chengtai@pku.edu.cn
§weinan@math.princeton.edu

The rest of the paper is organized as follows. In Section 2, we present a dynamical systems viewpoint of function approximation and deep learning. We then discuss the necessary optimality conditions, which is the well-known Pontryagin's maximum principle. In Section 3 and 4, we discuss numerical methods to solve the necessary conditions and obtain error estimates and convergence guarantees. Using benchmarking examples, we then compare our method with traditional gradient based methods for optimizing deep neural networks in Sec. 5. In Section 6, we discuss and compare our work with existing literature. Conclusion and outlook is given in Section 7.

2 Function Approximation by Dynamical Systems

We start with a description of the (continuous) dynamical systems approach to machine learning (see [13]). The essential task of supervised learning is to approximate some function

$$F : \mathcal{X} \to \mathcal{Y}$$

which maps inputs in $\mathcal{X} \subset \mathbb{R}^d$ (e.g. images, time-series) to labels in $\mathcal{Y}$ (categories, numerical predictions). Given a collection of K sample input-label pairs $\{x^i, y^i = F(x^i)\}_{i=1}^K$, one aims to approximate F using these data points. In the dynamical systems framework, we consider the inputs $x = (x^1, \dots, x^K) \in \mathbb{R}^{d\times K}$ as the initial condition of a system of ordinary differential equations

$$\dot{X}^i_t = f(t, X^i_t, \theta_t), \quad X^i_0 = x^i, \quad 0 \le t \le T,$$

where $\theta : [0,T] \to \Theta \subset \mathbb{R}^p$, represents the control (training) parameters and $X_t = (X^1_t, \dots, X^K_t) \in \mathbb{R}^{d\times K}$ for all $t \in [0,T]$. The form of f is chosen as part of the machine learning model. For example, in deep learning, f is the composition of a linear transformation and a component-wise nonlinear function (the activation function). For the solution to (2) to exist for any θ, we shall assume that f is continuous in t, θ and continuously differentiable in x. For the i^{th} input sample, the prediction of the "network" is a deterministic transformation of the terminal state $g(X^i_T)$ for some $g : \mathbb{R}^d \to \mathcal{Y}$, which we can view collectively as a function of the initial state (input) x^i and the control parameters (weights) θ. The dynamics (2) are decoupled across samples except for the dependence on the control θ. We shall consider quite a general space of controls

$$\mathcal{U} := \{\theta : [0,T] \to \Theta : \theta \text{ Lebesgue measurable}\}.$$

The aim is to select θ from $\mathcal{U}$ so that $g(X^i_T)$ most closely resembles y^i for $i = 1, \dots, K$. To this end, we define a loss function $\Phi : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ which is minimized when its arguments are equal, and we consider minimizing $\sum_i \Phi(g(x^i_T), y^i)$. Since g is fixed, we shall absorb it into the definition of Φ. Also, we define $\Phi_i(\cdot) := \Phi(\cdot, y_i)$. Then, the supervised learning problem in our framework is

$$\begin{aligned} &\min_{\theta\in\mathcal{U}} \sum_{i=1}^K \Phi_i(X^i_T) + \int_0^T L(\theta_t)dt, \\ &\dot{X}^i_t = f(t, X^i_t, \theta_t), \quad X^i_0 = x^i, \quad 0 \le t \le T, \quad i = 1, \dots, K, \end{aligned} \tag{1}$$

where $L : \Theta \to \mathbb{R}$ is a running cost, or the regularizer[1]. We note here that alternatively, we can formulate the supervised learning problem more generally in terms of optimal control in function spaces, see Appendix A.

Problem (1) is a special case of a class of general optimal control problem for ordinary differential equations [16]. The advantage of this formulation is that we can write down and study the optimality conditions of (1) entirely in continuous time and derive numerical algorithms that can subsequently be discretized. In other words, we *optimize, then discretize*, as opposed to the traditional reverse approach in deep learning.

As was suggested in [13], deep residual networks [17] can be considered as the forward Euler discretization of the continuous approach described above. In this connection, the algorithms presented in this paper can

[1]We can also make L depend on X_t, but for simplicity of presentation and the fact that most current machine learning models do not regularize the states, we shall omit this general case.

also be formulated in the context of deep residual networks. For general deep neural networks, although one can also formulate similar algorithms, it is not clear at this moment that PMP holds and these algorithms are valid (e.g. converge to the right solution) in the general setting. This issue will be studied in future work.

The optimization problem (1) can be solved by first discretizing it into a discrete problem (a feed-forward neural network) and then applying back propagation and gradient descent approaches commonly used in deep learning. However, here we will present an alternative approach. Hereafter, for simplicity of notation we shall set $K = 1$ drop the scripts i on all functions, noting that analogous results can be obtained in the general case since the dynamics and loss functions are decoupled across samples. Equivalently, we can think of this as effectively concatenating all K sample inputs into a single input vector of dimension $d \times K$ and redefine our dynamics accordingly. Hence, all results remain valid if we perform full-batch training. The case of mini-batch training is discussed in Sec. 4.3.

2.1 Pontryagin's Maximum Principle

In this section, we introduce a set of necessary conditions for optimal solutions of (1), known as the Pontryagin's Maximum Principle (PMP) [14, 15]. This shall pave way for an alternative numerical algorithm to train (1) and its discrete-time counter-part.

To begin with, we define the *Hamiltonian* $H : [0,T] \times \mathbb{R}^d \times \mathbb{R}^d \times \Theta \to \mathbb{R}$ given by

$$H(t, x, p, \theta) := p \cdot f(t, x, \theta) - L(\theta).$$

Theorem 1 (Pontryagin's Maximum Principle). *Let $\theta^* \in \mathcal{U}$ be an essentially bounded optimal control, i.e. a solution to* (1)*, and X^* the corresponding optimally controlled process and ess $sup_{t\in[0,T]} \|\theta^*_t\|_\infty < \infty$. Then, there exists an absolutely continuous co-state process $P^* : [0,T] \to \mathbb{R}^d$ such that the Hamilton's equations*

$$\dot{X}^*_t = \nabla_p H(t, X^*_t, P^*_t, \theta^*_t), \qquad X^*_0 = x, \tag{2}$$

$$\dot{P}^*_t = -\nabla_x H(t, X^*_t, P^*_t, \theta^*_t), \qquad P^*_T = -\nabla\Phi(X^*_T), \tag{3}$$

are satisfied. Moreover, for each $t \in [0,T]$, we have the Hamiltonian maximization condition

$$H(t, X^*_t, P^*_t, \theta^*_t) \geq H(t, X^*_t, P^*_t, \theta) \text{ for all } \theta \in \Theta \tag{4}$$

Proof. The proof of the PMP its variants can be found in any optimal control theory reference, e.g. [18, 16, 19]. Various generalizations can be found in [20] and references therein. For example, the requirement of the continuity of f with respect to t can be replaced by a much weaker measurability requirement if one assumes more conditions on ∇f. □

A few remarks are in order. First, Eq. (2), (3) and (4) allow us to solve for the unknowns X^*, P^*, θ^* simultaneously as a function of t. In this sense, the resulting optimal control θ^* is *open-loop* and is not in a feed-back form $\theta^*_t = \theta^*(X^*_t)$. The latter is of closed-loop type and are typically obtained from dynamic programming and the Hamilton-Jacobi-Bellman formalism [21]. In this sense, the PMP gives a weaker control. However, open-loop solutions are sufficient for neural network applications, where the trained weights and biases are fixed and only depend on the layer number and not the inputs.

PMP can be regarded as a (highly non-trivial) generalization of the calculus of variations to non-smooth settings (since we only assume θ^* to be measurable). Perhaps more familiar to the optimization community, the PMP is related to the Karush-Kuhn-Tucker (KKT) conditions for non-linear constrained optimization. Indeed, we can view (1) as a non-linear program over the function space $\mathcal{U}$ where the constraint is the ODE (2). In this sense, the co-state process P^* plays the role of a continuous-time analogue of Lagrange multipliers. The key difference between the PMP and the KKT conditions (besides the lack of inequality constraints on the state) is the Hamiltonian maximization condition (4), which is stronger than a typical first-order condition that assumes smoothness with respect to θ (e.g. $\nabla_\theta H = 0$). In particular, the PMP says that H is not only stationary, but globally maximized at an optimal control - which is a much stronger statement if H is not concave. Moreover, the PMP makes minimal assumptions on the parameter space Θ; the PMP holds even when f is non-smooth with respect to θ, or worse, when Θ is a discrete subset of $\mathbb{R}^p$.

Last, we emphasize that the PMP is only a necessary condition, hence there can be cases where solutions to the PMP is not actually globally optimal for (1). Nevertheless, in practice the PMP is often strong enough to give good solution candidates, and when certain convexity assumptions are satisfied the PMP becomes sufficient [22]. In the next section, we will discuss numerical methods that can be used to solve the PMP.

3 Method of Successive Approximations

Now, our strategy is to devise numerical algorithms for training (1) via solving the PMP (Eq. (2), (3) and (4)). We derive and analyze algorithms entirely in continuous time, which allows us to characterize errors estimates and convergence in a more transparent fashion.

There are many methods for the numerical solution of the PMP, including two-point boundary value problem method [11, 23], and collocation methods [24] coupled with general non-linear programming techniques [25, 26]. See [27] for a more recent review. However, many of these methods concern small-scale problems typically encountered in control applications (e.g. trajectory optimization of spacecrafts) and do not scale well to modern machine learning problems with a large number of state and control variables. One exception is the method of successive approximations (MSA) [28], which is an iterative method based on alternating propagation and optimization steps. We first introduce the simplest form of the MSA.

3.1 Basic MSA

Observe that (2) is simply the equation

$$\dot{X}^*_t = f(t, X^*_t, \theta^*_t)$$

and is independent of the co-state P^*. Therefore, we may proceed in the following manner. First, we make an initial guess of the optimal control $\theta^0 \in \mathcal{U}$. For each $k = 0, 1, 2, \ldots$, we first solve (2)

$$\dot{X}^{\theta^k}_t = f(t, X^{\theta^k}_t, \theta^k_t), \quad X^{\theta^k}_0 = x. \tag{5}$$

for X^{θ^k}, which then allows us to solve (3)

$$\dot{P}^{\theta^k}_t = -\nabla_x H(t, X^{\theta^k}_t, P^{\theta^k}_t, \theta^k_t), \quad P^{\theta^k}_T = -\nabla\Phi(X^{\theta^k}_T) \tag{6}$$

to get P^{θ^k}. Finally, we use the maximization condition (4) to set

$$\theta^{k+1}_t = \arg\max_{\theta \in \Theta} H(t, X^{\theta^k}_t, P^{\theta^k}_t, \theta)$$

for $t \in [0, T]$. The algorithm is summarized in Alg. 1.

Algorithm 1 Basic MSA

```
1: Initialize: $\theta^0 \in \mathcal{U}$
2: for $k = 0$ to #Iterations do
3:     Solve $\dot{X}^{\theta^k}_t = f(t, X^{\theta^k}_t, \theta^k_t), \quad X^{\theta^k}_0 = x$
4:     Solve $\dot{P}^{\theta^k}_t = -\nabla_x H(t, X^{\theta^k}_t, P^{\theta^k}_t, \theta^k_t), \quad P^{\theta^k}_T = -\nabla\Phi(X^{\theta^k}_T)$
5:     Set $\theta^{k+1}_t = \arg\max_{\theta\in\Theta} H(t, X^{\theta^k}_t, P^{\theta^k}_t, \theta)$ for each $t \in [0, T]$
```

As is the case with the maximum principle, MSA consists of two major components: the forward-backward Hamiltonian dynamics and the maximization for the optimal parameters at each time. An important feature of MSA is that the Hamiltonian maximization step is decoupled for each $t \in [0, T]$. In the language of deep learning, the optimization step is decoupled for different network layers and only the Hamiltonian ODEs (Step 3,4 of Alg. 1) involve propagation through the layers. This allows the parallelization of the maximization step, which is typically the most time-consuming step.

It has been shown that the basic MSA converges for a restricted class of linear quadratic regulators [29]. However, in general it tends to diverge if a bad initial θ^0 is chosen [29, 28]. Our goal now is to modify the basic MSA to control its divergent behavior. Before we do so, it is important to understand why the MSA diverges, and in particular, the relationship between the maximization step in Alg. 1 and the optimization problem (1).

3.2 Error Estimate for the Basic MSA

For each $\theta \in \mathcal{U}$, let us denote

$$J(\theta) := \Phi(X^{\theta}_T) + \int_0^T L(\theta_t) dt,$$

where X^θ satisfies (5). Our goal is to minimize $J(\theta)$. We show in the following Lemma the relationship between the values of J and the Hamiltonian maximization step. We start by making the following assumptions.

(A1) Φ is twice continuously differentiable, with Φ and $\nabla\Phi$ satisfying a Lipschitz condition, i.e. there exists $K > 0$ such that

$$|\Phi(x) - \Phi(x')| + \|\nabla\Phi(x) - \nabla\Phi(x')\| \leq K\|x - x'\|$$

for all $x, x' \in \mathbb{R}^d$.

(A2) $f(t, \cdot, \theta)$ is twice continuously differentiable in x, with $f, \nabla_x f$ satisfying a Lipschitz condition in x uniformly in θ and t, i.e. there exists $K > 0$ such that

$$\|f(t, x, \theta) - f(t, x', \theta)\| + \|\nabla_x f(t, x, \theta) - \nabla_x f(t, x', \theta)\|_F \leq K\|x - x'\|$$

for all $x, x' \in \mathbb{R}^d$ and $t \in [0, T]$.

With these assumptions, we have the following estimate:

Lemma 1. *Suppose (A1)-(A2) holds. Then, there exists a constant $C > 0$ such that for any $\theta, \phi \in \mathcal{U}$,*

$$\begin{aligned} J(\phi) \leq & J(\theta) - \int_0^T \Delta_{\phi,\theta} H(t) dt \\ & + C \int_0^T \|f(t, X^\theta_t, \phi_t) - f(t, X^\theta_t, \theta_t)\|^2 dt \\ & + C \int_0^T \|\nabla_x H(t, X^\theta_t, P^\theta_t, \phi_t) - \nabla_x H(t, X^\theta_t, P^\theta_t, \theta_t)\|^2 dt, \end{aligned}$$

where X^θ, P^θ satisfy Eq. (5), (6) *respectively and $\Delta H_{\phi,\theta}$ denotes the change in Hamiltonian*

$$\Delta H_{\phi,\theta}(t) := H(t, X^\theta_t, P^\theta_t, \phi_t) - H(t, X^\theta_t, P^\theta_t, \theta_t).$$

Proof. See Appendix B for the proof and discussion on relaxing the assumptions. □

In essence, Lemma 1 says that the Hamiltonian maximization step in MSA (step 5 in Alg. 1) is in some sense the optimal descent direction for J. However, the last two terms on the right hand side indicates that this descent can be nullified if substituting ϕ for θ incurs too much error in the Hamiltonian dynamics (step 3,4 in Alg. 1). In other words, the last two integrals measure the degree of satisfaction of the Hamiltonian dynamics (2), (3), which can be viewed as a feasibility condition, when one replaces θ by ϕ. Hence, we shall hereafter refer to these errors as *feasibility errors*. The divergence of the basic MSA happens when the feasibility errors blow up. Armed with this understanding, we can then modify the basic MSA to ensure convergence.

3.3 Extended PMP and Extended MSA

As discussed previously in Lemma 1, the decrement of J is ensured if we can control the feasibility errors in the Hamiltonian dynamics in steps 3,4 of Alg. 1. To this end, we employ a similar idea to augmented Lagrangians [30]. Fix some $\rho > 0$ and introduce the augmented Hamiltonian

$$\begin{aligned}\tilde{H}(t,x,p,\theta,v,q) :=& H(t,x,p,\theta) - \frac{1}{2}\rho\|v - f(t,x,\theta)\|^2 \\ & - \frac{1}{2}\rho\|q + \nabla_x H(t,x,p,\theta)\|^2. \end{aligned} \tag{7}$$

Then, we have the following set of alternative necessary conditions for optimality:

Proposition 1 (Extended PMP)**.** *Suppose that θ^* is a solution to the optimal control problem* (1)*. Then, there exists an absolutely continuous co-state process P^* such that the tuple $(X_t^*, P_t^*, \theta_t^*)$ satisfies the necessary conditions*

$$\dot{X}_t^* = \nabla_p \tilde{H}(t, X_t^*, P_t^*, \theta_t^*, \dot{X}_t^*, \dot{P}_t^*), \qquad X_0^* = x, \tag{8}$$

$$\dot{P}_t^* = -\nabla_x \tilde{H}(t, X_t^*, P_t^*, \theta_t^*, \dot{X}_t^*, \dot{P}_t^*), \qquad P_T^* = -\nabla_x \Phi(X_T^*), \tag{9}$$

$$\tilde{H}(t, X_t^*, P_t^*, \theta_t^*, \dot{X}_t^*, \dot{P}_t^*) \geq \tilde{H}(t, X_t^*, P_t^*, \theta, \dot{X}_t^*, \dot{P}_t^*), \qquad \theta \in \Theta, t \in [0,T]. \tag{10}$$

Proof. If θ^* is optimal, then by the PMP there exists a co-state process P^* such that (2), (3) and (4) are satisfied. Then, for all $t \in [0,T]$ and $\theta \in \Theta$ we have

$$\begin{aligned}\nabla_x \tilde{H}(t, X_t^*, P_t^*, \theta, \dot{X}_t^*, \dot{P}_t^*) =& \nabla_x H(t, X_t^*, P_t^*, \theta), \\ \nabla_p \tilde{H}(t, X_t^*, P_t^*, \theta, \dot{X}_t^*, \dot{P}_t^*) =& \nabla_p H(t, X_t^*, P_t^*, \theta),\end{aligned}$$

which implies that (8) and (9) are satisfied. Lastly, we can write

$$\begin{aligned}&\tilde{H}(t, X_t^*, P_t^*, \theta, \dot{X}_t^*, \dot{P}_t^*) \\ =& H(t, X_t^*, P_t^*, \theta) - \frac{1}{2}\rho\|\dot{X}_t^* - f(t, X_t^*, \theta)\|^2 - \frac{1}{2}\rho\|\dot{P}_t^* + \nabla_x H(t, X_t^*, P_t^*, \theta)\|^2\end{aligned}$$

For each t, θ^* maximizes all three terms on the RHS simultaneously, and hence (10) is also satisfied. □

Compared with the usual PMP, the extended PMP is a weaker necessary condition. However, the advantage is that maximization of $\tilde{H}$ naturally penalizes errors in the Hamiltonian dynamical equations, and hence we should expect MSA applied to the extended PMP to converge for large enough ρ. Note that the Hamiltonian equation steps do not change (since the added terms have no effect on optimal solutions) and the only change is the maximization step. The extended MSA (E-MSA) algorithm is summarized in Alg. 2.

To establish convergence, define

$$\mu_k := \int_0^T \Delta H_{\theta^{k+1},\theta^k}(t) dt \geq 0$$

If $\mu_k = 0$, then from the Hamiltonian maximization step (10) we must have

$$\begin{aligned}0 =& -\mu_k \leq -\frac{1}{2}\rho \int_0^T \|f(t, X_t^{\theta^k}, \theta_t^{k+1}) - f(t, X_t^{\theta^k}, \theta_t^k)\|^2 dt \\ & - \frac{1}{2}\rho \int_0^T \|\nabla_x H(t, X_t^{\theta^k}, P_t^{\theta^k}, \theta_t^{k+1}) - \nabla_x H(t, X_t^{\theta^k}, P_t^{\theta^k}, \theta_t^k)\|^2 dt. \leq 0\end{aligned}$$

and so

$$\max_\theta \tilde{H}(X_t^{\theta^k}, P_t^{\theta^k}, \theta, \dot{X}_t^{\theta^k}, \dot{P}_t^{\theta^k}) = \tilde{H}(X_t^{\theta^k}, P_t^{\theta^k}, \theta_k, \dot{X}_t^{\theta^k}, \dot{P}_t^{\theta^k})$$

i.e. $(X^{\theta^k}, P^{\theta^k}, \theta^k)$ satisfy the extended PMP. In other words, the quantity $\mu_k \geq 0$ measures the distance from a solution of the extended PMP, and if it equals 0, then we have a solution. We now prove the following result that guarantees the convergence of the extended MSA (Alg. 2).

Theorem 2. *Let (A1)-(A2) be satisfied and $\theta^0 \in \mathcal{U}$ be any initial measurable control with $J(\theta^0) < +\infty$. Suppose also that $\inf_{\theta\in\mathcal{U}} J(\theta) > -\infty$. Then, for ρ large enough, we have under Alg. 2,*

$$J(\theta^{k+1}) - J(\theta^k) \leq -D\mu_k.$$

for some constant $D > 0$ and

$$\lim_{k\to 0} \mu_k = 0$$

i.e. the extended MSA algorithm converges to a solution of the extended PMP.

Proof. Using Lemma 1 with $\theta \equiv \theta^k, \phi \equiv \theta^{k+1}$, we have

$$\begin{aligned} J(\theta^{k+1}) - J(\theta^k) \leq & -\mu_k + C\int_0^T \|f(t, X_t^{\theta^k}, \theta_t^{k+1}) - f(t, X_t^{\theta^k}, \theta_t^k)\|^2 dt \\ & + C\int_0^T \|\nabla_x H(t, X_t^{\theta^k}, P_t^{\theta^k}, \theta_t^{k+1}) - \nabla_x H(t, X_t^{\theta^k}, P_t^{\theta^k}, \theta_t^k)\|^2 dt \end{aligned}$$

From the Hamiltonian maximization step in Alg. 2, we know that

$$\begin{aligned} H(t, X_t^{\theta^k}, P_t^{\theta^k}, \theta_t^k) \leq & H(t, X_t^{\theta^k}, P_t^{\theta^k}, \theta_t^{k+1}) \\ & - \frac{1}{2}\rho\|f(t, X_t^{\theta^k}, \theta_t^{k+1}) - f(t, X_t^{\theta^k}, \theta_t^k)\|^2 \\ & - \frac{1}{2}\rho\|\nabla_x H(t, X_t^{\theta^k}, P_t^{\theta^k}, \theta_t^{k+1}) - \nabla_x H(t, X_t^{\theta^k}, P_t^{\theta^k}, \theta_t^k)\|^2. \end{aligned}$$

Hence, we have

$$J(\theta^{k+1}) - J(\theta^k) \leq -(1 - \frac{2C}{\rho})\mu_k.$$

Pick $\rho > 2C$, then we indeed have $J(\theta^{k+1}) - J(\theta^k) \leq -D\mu_k$ with $D = (1 - \frac{2C}{\rho}) > 0$. Moreover, we can rearrange and sum the above expression to get

$$\sum_{k=0}^{M} \mu_k \leq D^{-1}(J(\theta^0) - J(\theta^{M+1})) \leq D^{-1}(J(\theta^0) - \inf_{\theta\in\mathcal{U}} J(\theta))$$

and hence $\sum_{k=0}^{\infty} \mu_k < +\infty$, which implies $\mu_k \to 0$ and the extended MSA converges to a solution of the extended PMP. ☐

Algorithm 2 Extended MSA (E-MSA)

```
1: Initialize: $\theta^0 \in \mathcal{U}$. Hyper-parameter: $\rho$
2: for $k = 0$ to #Iterations do
3:     Solve $\dot{X}_t^{\theta^k} = f(t, X_t^{\theta^k}, \theta_t^k)$,    $X_0^{\theta^k} = x$
4:     Solve $\dot{P}_t^{\theta^k} = -\nabla_x H(t, X_t^{\theta^k}, P_t^{\theta^k}, \theta_t^k)$,    $P_T^{\theta^k} = -\nabla\Phi(X_T^{\theta^k})$
5:     Set $\theta_t^{k+1} = \arg\max_{\theta\in\Theta} \tilde{H}(t, X_t^{\theta^k}, P_t^{\theta^k}, \theta, \dot{X}_t^{\theta^k}, \dot{P}_t^{\theta^k})$ for each $t \in [0, T]$
```

4 Discrete-Time Formulation

In the previous section, we discussed the PMP and MSA in the continuous-time setting, where we showed that an appropriately extended version (E-MSA) converges to a solution of an extended PMP. Here, we shall discuss the discretized versions of PMP, MSA and E-MSA, as well as their connections to deep residual networks and back-propagation.

4.1 Discrete-Time PMP and Discrete-Time MSA

Applying Euler-discretization to Eq. (2), we get

$$x_{n+1} = x_n + \delta f_n(x_n, \vartheta_n), \quad x_0 = x,$$

for $n = 0, \dots, N-1$, with $\delta = T/N$ (step-size), $x_n := X_{n\delta}$, $\vartheta_n := \theta_{n\delta}$ and $f_n(\cdot) := f(n\delta, \cdot)$. Then, the discrete-time analogue of the control problem (1) is

$$\begin{aligned} &\min_{\{\vartheta_0,\dots,\vartheta_{N-1}\}\in\Theta^N} \Phi(x_N) + \delta \sum_{n=0}^{N-1} L(\vartheta_n), \\ &x_{n+1} = x_n + \delta f_n(x_n, \vartheta_n), \quad x_0 = x, \quad 0 \le n \le N-1. \end{aligned} \tag{11}$$

Observe that barring the constant δ, this is exactly the supervised learning problem for deep residual networks[2]. Therefore, when suitably discretized, one expects that the E-MSA provides a means to train residual neural networks via the solution of the extended PMP.

We now write down formally the discretized form of the PMP. Let us use the shorthand $g_n(x_n, \vartheta_n) := x_n + \delta f_n(x_n, \vartheta_n)$. Define the scaled discrete Hamiltonian

$$H_n(x, p, \vartheta) = p \cdot g_n(x, \vartheta) - \delta L(\vartheta).$$

Then, a discrete-time PMP is the following set of conditions:

$$\begin{aligned} &x^*_{n+1} = g_n(x^*_n, \vartheta^*_n) && x^*_0 = x, \\ &p^*_n = \nabla_x H_n(x^*_n, p^*_{n+1}, \vartheta_n) && p^*_N = -\nabla_x \Phi(x^*_N), \\ &H_n(x^*_n, p^*_{n+1}, \vartheta^*_n) \ge H_n(x^*_n, p^*_{n+1}, \vartheta). && \vartheta \in \Theta, \quad n = 0, \dots, N-1. \end{aligned}$$

It is not clear whether these conditions strictly holds for general g_n. In fact the issue whether PMP holds for discrete dynamical systems is a complicated one and there are known counterexamples.[32, 33, 34]. Nevertheless, they must hold approximately for small time step size and this is the situation we will consider in the current paper. We expect Lemma 1, which implies monotonicity of the E-MSA algorithm, to hold in the discrete-time case under appropriate conditions. We leave a rigorous analysis of these statements to future work. For numerical experiments presented in the next section, we shall almost always work with residual networks that can be regarded as discretizations of continuous networks so that the PMP holds approximately at least [35].

For completeness, we summarize the discrete-time version of E-MSA in Alg. 3, where $\tilde{H}$ in this case is defined as in (7). Note that for residual networks ($g_n = x_n + \delta f_n$), this is equivalent to a forward Euler discretization on the state equation and a backward Euler discretization on the co-state equation in Alg. 2. As before, the for-loop in step 9 is decoupled across layers and can be carried out in parallel.

Algorithm 3 Discrete-time E-MSA

```
1: Initialize: $\vartheta^0_n \in \Theta_n$, $n = 0, \dots, N-1$. Hyper-parameter: $\rho$
2: **for** $k = 0$ to #Iterations **do**
3:     Set $x^{\theta^k}_0 = x$
4:     **for** $n = 0$ to $N-1$ **do**
5:         $x^{\vartheta^k}_{n+1} = g_n(x^{\vartheta^k}_n, \vartheta^k_n)$
6:     Set $p^{\vartheta^k}_N = -\nabla\Phi(x^{\vartheta^k}_N)$
7:     **for** $n = N-1$ to $0$ **do**
8:         $p^{\vartheta^k}_n = \nabla_x H_n(x^{\vartheta^k}_n, p^{\vartheta^k}_{n+1}, \vartheta^k_n)$
9:     **for** $n = 0$ to $N-1$ **do**
10:        Set $v(\vartheta) = x^{\vartheta^k}_{n+1} - g_n(x^{\vartheta^k}_n, \vartheta)$ and $q(\vartheta) = p^{\vartheta^k}_n - \nabla_x H_n(x^{\vartheta^k}_n, p^{\vartheta^k}_{n+1}, \vartheta)$
11:        Solve $\vartheta^{k+1}_n = \arg\max_{\vartheta\in\Theta_n} \tilde{H}_n(x^{\vartheta^k}_n, p^{\vartheta^k}_{n+1}, \vartheta, v(\vartheta), q(\vartheta))$
```

[2]If we pick ReLU activations [31], then δ can be absorbed into ϑ

4.2 Relationship to Gradient Descent with Back-propagation

We note an interesting relationship of the MSA with classical gradient descent with back-propagation [10, 11]. We have shown in Lemma 1 that the divergence of MSA can be attributed to the large errors in the Hamiltonian dynamics terms caused by the maximization step, which involve drastic changes in parameter values. Assuming each Θ_n is a continuum and g_n, L are differentiable in ϑ_n, a simple fix is to make the maximization step "soft": we replace step 11 in Alg. 3 with a gradient ascent step:

$$\vartheta_n^{k+1} = \vartheta_n^k + \eta \nabla_\vartheta H_n(x_n^{\vartheta^k}, p_{n+1}^{\vartheta^k}, \vartheta_n^k), \tag{12}$$

for some small learning rate η. We now show that in the discrete-time setting, this is equivalent to the classical gradient descent with back-propagation.

Proposition 2. *The basic MSA in discrete-time (Alg. 3 with $\rho = 0$) with step 11 replaced by* (12) *is equivalent to gradient descent with back-propagation.*

Proof. Recall that the Hamiltonian is

$$H_n := p_{n+1} \cdot g(x_n, \vartheta_n) - \delta L(\vartheta_n),$$

and the total loss function is $J(\vartheta) = \Phi(x_N) - \delta \sum_{n=0}^{N-1} L(\vartheta_n)$. It is easy to see that $p_n = -\nabla_{x_n}\Phi(x_N)$ by working backwards from $n = N$ and the fact that $\nabla_{x_n} x_{n+1} = \nabla_x g_n(x_n, \vartheta_n)$. Then,

$$\begin{aligned} \nabla_{\vartheta_n} J(\vartheta) =& \nabla_{x_{n+1}}\Phi(x_N) \cdot \nabla_{\vartheta_n} x_{n+1} - \delta \nabla_{\vartheta_n} L(\vartheta_n) \\ =& - p_{n+1} \cdot \nabla_{\vartheta_n} g(x_n, \vartheta_n) - \delta \nabla_{\vartheta_n} L(\vartheta_n) \\ =& - \nabla_{\vartheta_n} H_n \end{aligned}$$

Hence, (12) is simply the gradient descent step

$$\vartheta_n^{k+1} = \vartheta_n^k - \eta \nabla_{\vartheta_n} J(\vartheta^k).$$

□

As the proposition shows, gradient descent with back-propagation can be seen as a modification of the basic MSA by replacing the Hamiltonian maximization step with a gradient ascent step. However, we note that the PMP (and MSA convergence) holds, at least in continuous-time, even when differentiability with respect to ϑ is not satisfied, and hence is more general than the classical back-propagation. In fact, the PMP formalism shows that the back-propagation of information through a deep network is handled by the co-state equation and there is no requirement or relationship to the gradients with respect to the trainable parameters. In other words, optimization is performed at each layer separately (with or without gradient information), and propagation is independent of optimization.

4.3 A Remark on Mini-batch Algorithms

So far, our discussion has focused on full-batch algorithms, where the input x represents the full set of training inputs. As modern supervised learning tasks typically involve a large number of training samples, usually the optimization problem has to be solved in mini-batches, where at each iteration we sub-sample m input-label pairs and optimize the parameters θ (or ϑ in discrete time) based on losses evaluated on these pairs. In the context of continuous-time PMP, we can write the batch version of the three necessary conditions as

$$\begin{aligned} &\dot{X}_t^{i,*} = \nabla_p H(t, X_t^{i,*}, P_t^{i,*}, \theta_t^*), && X_0^{i,*} = x^i, \\ &\dot{P}_t^{i,*} = -\nabla_x H(t, X_t^{i,*}, P_t^{i,*}, \theta_t^*), && P_T^{i,*} = -\nabla \Phi^i(X_T^{i,*}), \\ &\theta_t^* = \arg\max_{\theta\in\Theta} \sum_{i=1}^{M} H(t, X_t^{i,*}, P_t^{i,*}, \theta), && t \in [0, T], \end{aligned}$$

for samples $i = 1, \dots, M$. We omit for brevity the equivalent expressions for discrete-time. In particular, notice that the propagation steps are decoupled across samples, and hence can be carried out independently. The only difference is the maximization step, where in a mini-batch setting we would evaluate instead

$$\arg\max_{\theta \in \Theta} \sum_{i=1}^{m} H(t, X_t^{i,*}, P_t^{i,*}, \theta)$$

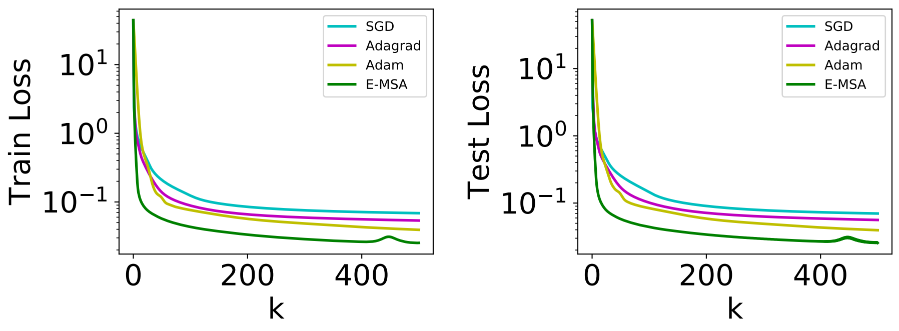
If m is large enough and the samples are independently and identically drawn, then uniform law of large numbers [36] holds under fairly general conditions and ensures that the mini-batch sum of Hamiltonians converges uniformly in θ to the full-batch sum. Hence, maximization performed on the mini-batch sum should be close to the actual maximization on the full Hamiltonian. Rigorous error estimates for the mini-batch version of our algorithm shall be left as future work, and we use instead numerical results in Sec. 5 to demonstrate that the algorithm can also be carried out in a mini-batch fashion.

5 Numerical Experiments

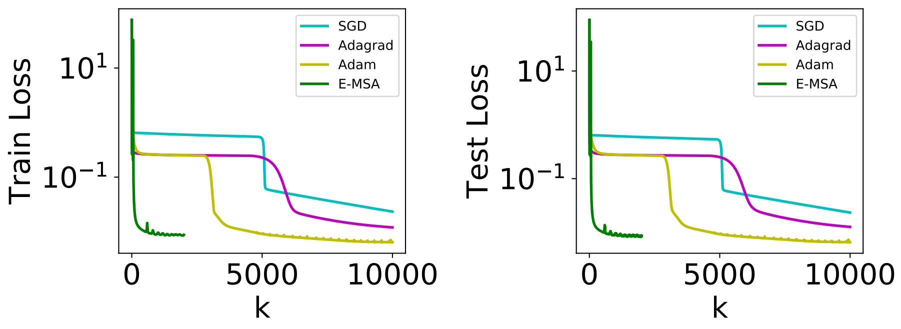
In this section, we investigate the performance of E-MSA compared with the usual gradient-based approaches, namely stochastic gradient descent and its variants: Adagrad [6] and Adam [8]. To illustrate key properties of E-MSA, we shall begin by investigating some synthetic examples. First, we consider a simple one-dimensional function approximation problem where we want to approximate $F(x) = \sin(x)$ for $x \in [-\pi, \pi]$ using a continuous time dynamical system. Let $T = 5$ and consider

$$\dot{X}_t = f(X_t, \theta_t) = \tanh(W_t X_t + b_t),$$

where $\theta_t = (W_t, b_t) \in \mathbb{R}^{5\times 5} \times \mathbb{R}^5$, i.e. a continuous analogue of a fully connected feed forward neural networks with 5 nodes per layer. To match dimensions, we shall concatenate the input x to form a five dimensional vector of identical components, which is now the initial condition to the dynamical system on $\mathbb{R}^d$. The output of the network is $\sum_{i=1}^{5} X_T^i$. and we define the loss function due to one sample to be $\Phi(X_T) = (\sum_{i=1}^{5} X_T^i - \sin(x))^2$. For multiple samples, we average the loss function over all samples in the usual way. We apply E-MSA with discretization size $\delta = 0.25$ (giving 20 layers) and compute the Hamiltonian maximization step using 10 iterations of limited memory BFGS method (L-BFGS) [37]. In Fig. 1(a), we compare the results with gradient descent based optimization approaches, where we observe that E-MSA has favorable convergence rate per-iteration. More interestingly, it is well-known that gradient descent may suffer slow convergence at flat regions or near saddle-points, where the gradients become very small and optimization may stall for a long time. This often occurs as a result of poor initialization of weights and biases [38]. Here, we simulate this scenario by initializing all weights and biases (W_t, b_t) to be 0 and observe the optimization process. We see from Fig. 1(b) that gradient descent based methods are more easily stalled at flat regions compared with E-MSA, where the Hamiltonian maximization can quickly escape the locally flat regions. We calculated numerically the eigenvalues of the Hessian at this point, which confirms that this is indeed very close to a saddle point.



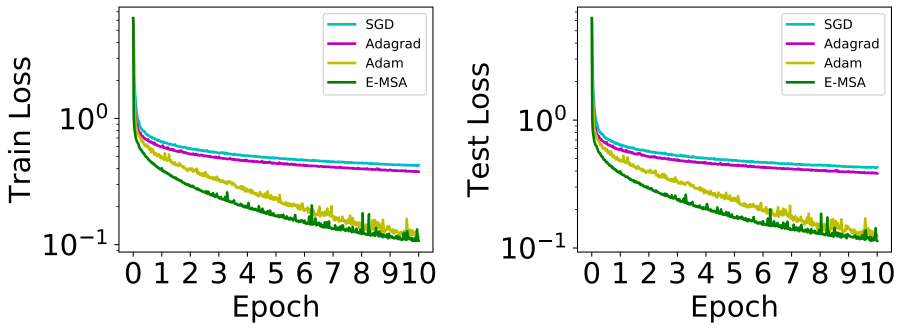
(a)



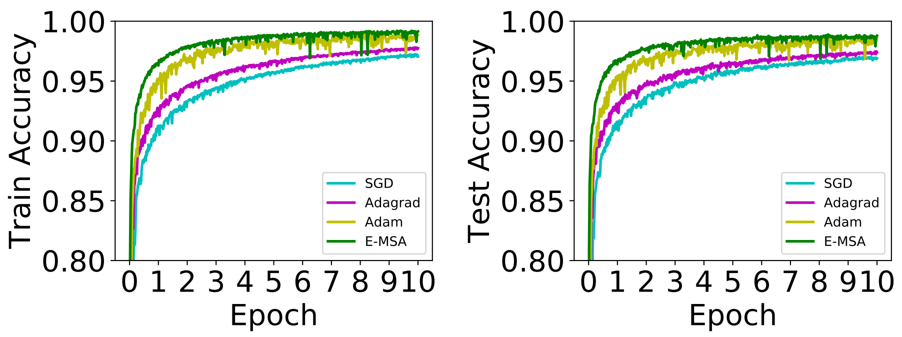
(b)

Figure 1: Comparison of E-MSA with gradient-based methods for approximating the sine function with a continuous, 5-dimensional dynamical system. A training and test set of 1000 samples each are used. (a) Loss function vs iterations for a good initialization, where weights are initialized with truncated random normal variables with standard deviation 0.1 and biases are initialized as constants equal to 0.1. We see that E-MSA has good convergence rate per iteration. (b) We use a poor initialization by setting all weights and biases to 0. We observe that gradient descent based methods tend to become stuck whereas E-MSA are better at escaping these slow manifolds, provided that ρ is well chosen (=1.0 in this case).

Next, we consider a familiar supervised learning test problem: the MNIST dataset [39] for handwritten digit recognition, with 55000 training samples and 10000 test samples. We employ a continuous dynamical system that resembles a (residual) convolution neural network [40] when discretized. More concretely, at each t we consider the map $f(t, x, \theta) = \tanh(W \star x + b)$ where W is a 3×3 convolution filter with 32 input and output channels. To match dimensions, we introduce two projection layers at the input (consisting of convolution, point-wise non-linearities followed by 2×2 max-pooling). We also use a fully-connected classification layer as the final layer, with softmax cross-entropy loss. Note that the input projection layers and fully-connected output layers are not of residual form, but we can nevertheless apply Alg. 3 with the appropriate g. We use a total of 10 layers (2 projections, 1 fully-connected and 7 residual layers with $\delta = 0.5$, i.e $T = 3.5$). The model is trained with mini-batch sizes of 100 using E-MSA and gradient based methods, namely SGD, Adagrad [6], and Adam [8]. For E-MSA, we approximately solve the Hamiltonian maximization step using either 10 iterations of L-BFGS. Note that since we have decoupled the layers through the PMP, the L-BFGS step used to maximize H is tractable since it involves much fewer parameters than directly minimizing J. Fig. 2 compares the performance of E-MSA with the other gradient based methods, where we observe that E-MSA has good performance per-iteration, especially at early stages of training. However, we also show in Fig. 3 that the wall-clock performance of our methods are not currently competitive, because the Hamiltonian maximization step is time consuming and the performance gains per iteration is outweighed by the running time. Note that wall-clock times are compared on the CPU for fairness since we did not use a GPU implementation of L-BFGS. As a further test, we train the same model on a different dataset, the fashion MNIST dataset [41], where we again observe similar phenomena (see Fig. 4).

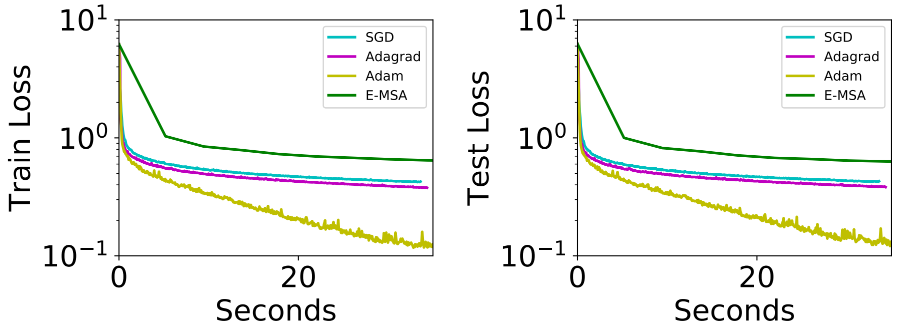


(a)



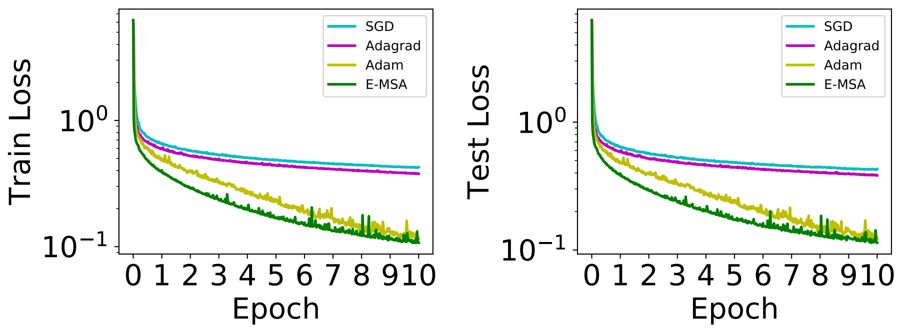
(b)

Figure 2: Comparison of E-MSA with gradient-based methods for the residual CNN on the MNIST dataset. Mini-batch size of 100 is used so that each epoch of training consists of 550 iterations. (a) Train and test Loss vs epoch. (b) Train and test accuracy vs epoch. For each case, we tuned the associated hyper-parameters on a coarse grid for optimal performance. We observe that per-iteration, E-MSA performs favorably, at least at early times. This shows that if the augmented Hamiltonian can be efficiently maximized, we may obtain good performance.

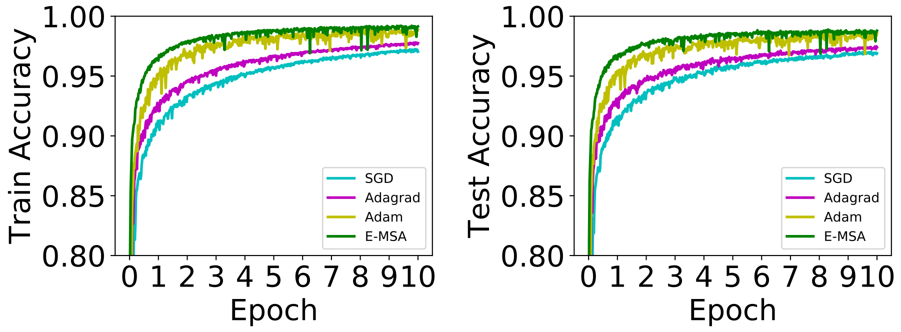


(a)

Figure 3: Comparison of E-MSA with gradient-based methods for the residual CNN on the MNIST dataset on a wall-clock basis. We observe that currently, the gains per iteration is outweighed by the additional computational costs. Note that we did not use a GPU implementation for the L-BFGS algorithm used to maximize the augmented Hamiltonian, hence the wall-clock time for E-MSA is expected to be improved. Nevertheless, we expect that more efficient Hamiltonian maximization algorithms must be developed for E-MSA to out-perform gradient-based methods in terms of wall-clock efficiency.



(a)



(b)

Figure 4: Comparison of E-MSA with gradient-based methods for the residual CNN on the fashion MNIST dataset. We use the same network structure and mini-batch sizes as in Fig. 2. The hyper-parameters have to be slightly re-tuned. (a) Train and test Loss vs epoch. (b) Train and test accuracy vs epoch. Again, we observe E-MSA performs favorably per-iteration.

6 Discussion and Related Work

We commence this section by highlighting the distinguishing features of E-MSA from traditional gradient based training methods. First, the formulations of PMP and E-MSA do not involve gradient information with respect to the trainable parameters. In fact, Thm. 1 and Alg. 2 remain valid even when the trainable parameters can only take values in a discrete set. Second, due to a more drastic argmax step taken at each iteration, E-MSA tends to have better convergence rates at the early steps of training, as observed in our numerical experiments (Sec. 5). Third, in the PMP formalism, the Hamiltonian equations for the state and co-state are the "forward and backward propagations", whereas given the state and co-state values, the optimization step is decoupled across layers. This allows one to potentially parallelize the often time-consuming optimization step. Moreover, from Lemma 1, we show that as long as the the Hamiltonian is sufficiently increased in a layer without causing too much loss in the Hamiltonian dynamics feasibility conditions, we can ensure decrement of the loss function. This is the reason why we can use a small number of iterations of L-BFGS at each step. Moreover, this suggests that the argmax updates need not happen synchronously, i.e. the optimization in each layer can be a separate thread or process that computes the argmax and updates that layer's parameters independent of other layers. The propagation may also potentially be allowed to happen asynchronously as long as updates are sufficiently frequent. We leave a rigorous analysis of an asynchronous version of the current approach to future work. In summary, the main strength of the PMP (over e.g. solving the KKT conditions using gradient methods) is that PMP says that at the optimum, the Hamiltonian is not only stationary (KKT), but globally maximized. This hints that heuristic global optimization methods can be applied to H to obtain algorithms that are very different in behavior compared with gradient based approaches. Again, Lemma 1 ensures that such heuristic global maximization need only be approximate.

As it currently stands, our experiments in Sec. 5 demonstrate that the Hamiltonian maximization step in E-MSA gives very different behavior compared with gradient based methods. When the Hamiltonian is sufficiently maximized, we indeed obtain favorable performance compared with gradient descent based methods. Furthermore, we saw in Fig. 1 that Hamiltonian maximization may avoid pitfalls such as a very flat landscape. Overall, the key to whether E-MSA (and other methods based on solving the PMP) will eventually constitute a replacement for gradient based algorithm lies in the question of whether efficient Hamiltonian maximization can be performed at reasonable computational costs. Although this is still a non-convex optimization problem, it is much simpler than the original training problem because: (1) Optimization in the layers are decoupled and hence parameter space is greatly reduced; (2) The Hamiltonian is formally similar across different layers, loss functions and models, so specialized algorithms may be designed; (3) The Hamiltonian does not need to be maximized exactly, thus fast heuristic methods [42] or learning [43, 44, 45] can potentially be used to perform this. All these are worthy of future exploration in order to make E-MSA truly competitive.

Next, we put our work in perspective by discussing related work in the optimal control, optimization and deep learning literature. First, the work on numerical algorithms for the solution of optimal control problem is abundant (see e.g. [27] for a survey). Many of the state-of-the-art techniques in the control theory literature assume a moderately small problem size, so that conventional non-linear programming techniques [25, 26] as well as shooting [23] and collocation methods [24] produce efficient algorithms. This is usually not the case for large-scale machine learning problems, where often, the only scalable approach is to rely on iterative updates to the parameters. This is the reason for our focus on the MSA algorithms [28], as they are straight-forward to implement and typically have linear scaling in computational complexity with respect to the input and parameter sizes. The basic MSA is discussed in [46], and a number of improved variants are discussed in [28] and references therein. For example, a popular improvement is based on needle-perturbations, where controls are varied on small intervals at each iteration. While convergent, the main issue with the needle-perturbation approach is the requirement of a sufficiently fine mesh (i.e. many layers in the discretized network), which impacts computational speed. A possible solution is the use of adaptive meshes, which is a future direction we plan to investigate. Our variant of the MSA presented in this work differs from classical approaches [28] mainly in the sense that we solve a weaker sufficient condition (extended PMP, Prop. 1), which then allows us to control errors in the Hamiltonian dynamical equations at every iteration without going into finer mesh-sizes. The regularization terms proportional to ρ is similar to the heuristic modifications suggested in [47] by regularizing the distance between θ^k and θ^{k+1}, but we do not have to assume convexity of Θ or that f is Lipschitz in θ.

In the optimization literature, our work shares some similarity with the recently proposed ADMM methods [48] for training deep neural networks, where the authors also considered necessary conditions with Lagrange multipliers that can decouple optimization across layers. The main difference in our work is that the PMP gives a stronger necessary condition (Hamiltonian maximization) that also applies to general parameter spaces (e.g., discrete, or bounded with non-linear constraints). Our modification of the basic MSA in terms of the augmented Hamiltonian is inspired by the method of augmented Lagrangians often applied in constrained optimization [30]. The idea of viewing a initially discrete system as the discretization of a continuous-time system has been explored in [49] in the form of stochastic optimization. Our current work is also in this flavor, but for neural network models.

In deep learning, there are a few works that share our perspective of deep neural networks as a discretization of a dynamical system, which then allows one to formulate the training of a deep neural network as an optimal control problem in continuous time. We note that the connection between the PMP and back-propagation has been pointed out qualitatively in [12] and well-understood in the development of back-propagation [11, 50], although to the best of our knowledge, this work is the first attempt to translate numerical algorithms for the PMP into training algorithms for deep learning that goes beyond gradient descent. The treatment of machine learning as function approximation via a dynamical system has been pointed out in [13]. The recent work of [51, 52] also propose the dynamical systems viewpoint, and the authors used continuous-time tools to address stability issues in inference and training of deep neural networks. In contrast, our work focuses on the optimization aspects centered around the PMP. We also mention other recent approaches to decouple optimization in deep neural networks, such as synthetic gradients [44, 45] and proximal back-propagation [53].

7 Conclusion and Outlook

In this paper, we discuss the viewpoint that deep residual neural networks can be viewed as discretization of a continuous-time dynamical system, and hence supervised deep learning can be regarded as solving an optimal control problem in continuous time. We explore a concrete consequence of this connection, by modifying the classical method of successive approximations for solving optimal control problems (in particular the PMP) into a method for solving a weaker sufficient condition (extended PMP). We prove the convergence of the resulting algorithm (E-MSA) and test it on various benchmark problems, where we observe that the E-MSA algorithm performs favorably on a per-iteration basis, especially at early stages of training, compared with gradient-based approaches such as SGD, Adagrad and Adam.

There are many avenues of future research. On the algorithmic side, it is necessary to further improve the computational efficiency of the E-MSA, in particular the Hamiltonian maximization step. Moreover, adaptive selection of ρ depending on iteration number and/or layer can be explored, e.g. by designing adaptive tuning schemes using control theoretic tools [49]. Also, it is desirable to formulate and analyze the PMP and E-MSA from a discrete-time perspective in order to broaden the method's application. From a modeling perspective, viewing deep neural networks as continuous-time dynamical systems is useful in the sense that it allows one to think of neural network architectures as dynamical objects. Indeed, at each training iteration of the E-MSA, we do not have to use the same discretization scheme to compute the Hamiltonian dynamical equations. Also, as the PMP and E-MSA assumes little structure on the parameter space Θ, it will also be interesting to apply the E-MSA to train neural networks that have discrete weights (e.g. those that can only take on binary values). Such networks have the advantage of fast inference speed and small memory requirement. However, training such networks is a challenge and most existing techniques rely on approximating or thresholding the derivatives [54, 55]. With the PMP and MSA, we may be able to directly train discrete networks in a principled way.

Acknowledgment

The work of W. E is supported in part by Major Program of NNSFC under grant 91130005, ONR grant N00014-13-1-0338, DOE grants de-sc0008626 and de-sc0009248 Q. Li is supported by the Agency for Science, Technology and Research, Singapore.

A Function Space Formulation

In this section, we give an alternative, non-rigorous formulation of the supervised learning problem as a optimal control problem on function spaces. We consider the same basic problem setting: we would like to approximate some function $F : \mathcal{X} \to \mathcal{Y}$ where $\mathcal{X} \subset \mathbb{R}^d$. Here, we take a simplifying assumption that $\mathcal{Y} = \mathbb{R}$ and $F \in L^2(\mathbb{R}^d)$. Now, we may begin with an approximator $F_0 \in L^2(\mathbb{R}^d)$ and consider improving it by evolving its input by a controllable dynamical system. That is, we define

$$u(t,x) := F_0(X_t(x))$$

where $X_t(x)$ is the flow map of the dynamical system

$$\dot{X}_t = f(t, X_t, \theta_t), \qquad X_0 = x,$$

and as before, $\theta_t \in \Theta$ for all t. By chain rule, it is easy to see that $u(t,\cdot) \in L^2(\mathbb{R}^d)$ evolves according to the partial differential equation

$$u_t(t,x) = f(t,x,\theta) \cdot \nabla_x u(t,x), \qquad u(0,\cdot) = F_0(\cdot). \tag{13}$$

We shall assume that all required regularities are satisfied. The supervised learning problem is then to find $\theta \in \mathcal{U}$ such that $u(T,x)$ is as close to F as possible. To this end, we may define the loss function $\Phi : L^2(\mathbb{R}) \times L^2(\mathbb{R}) \to \mathbb{R}$ (e.g. $\Phi(u,v) = \|u - v\|^2_{L^2}$) and pose the optimization problem

$$\begin{aligned} &\min_{\theta \in \mathcal{U}} \Phi(u(T,\cdot), F) + \int_0^T L(\theta_t) dt, \\ &u_t(t,x) = f(t,x,\theta) \cdot \nabla_x u(t,x), \qquad u(0,\cdot) = F_0(\cdot). \end{aligned} \tag{14}$$

As before, L is a regularizer on the trainable parameters. Now, (14) is an optimal control on $L^2(\mathbb{R})$. In particular, notice that the evolution equation (13) is in fact the adjoint to the Liouville's equation, which models the evolution of phase-space densities induced by a dynamical system.

We now write down a set of necessary conditions for optimality, in the form of the Pontryagin's maximum principle, for the present function-space control problem (14). Define the Hamiltonian functional $H : [0,T] \times L^2(\mathbb{R}) \times L^2(\mathbb{R}) \times \Theta \to \mathbb{R}$

$$H(t,u,v,\theta) = \int_{\mathbb{R}^d} v(x) f(t,x,\theta) \cdot \nabla_x u(x) dx - L(\theta).$$

Then, the Pontryagin's maximum principle for this system takes the form: let $\theta^* \in \mathcal{U}$ be an optimal control, then there exists a co-state process $v(t,\cdot) \in L^2(\mathbb{R}^d)$ such that

$$\begin{aligned} &u^*_t(t,x) = f(t,x,\theta^*_t) \cdot \nabla_x u^*(t,x), && u^*(0,x) = F_0(x) \\ &v^*_t(t,x) = -\nabla_x (f(t,x,\theta^*_t) v^*(t,x)), && v^*(T,x) = -D_u \Phi(u^*(T,x), F(x)) \\ &H(t, u^*(t,\cdot), v^*(t,\cdot), \theta^*_t) \geq H(t, u^*(t,\cdot), v^*(t,\cdot), \theta) && \theta \in \Theta, t \in [0,T] \end{aligned}$$

Note that the co-state v^* satisfies the Liouville's equation with a specified boundary condition. The PMP for similar functional optimal control problems has been studied in [56, 57]. We remark also that more general cases can be considered, e.g. we do not start with a fixed F_0, but a set $\Omega \subset L^2$ as a set of possible initial conditions for the state equation (e.g. fully connected layer with trainable weights forms such a subset, in fact subspace, of L^2). In this case, we may replace the initial condition on u with a transversality condition on $v(0,\cdot)$.

In summary, the advantage of this formulation is that we make no explicit reference to the training data and formulate the entire problem as a control problem on abstract functional spaces. Of course, in practice, to implement an MSA-like algorithm, the terminal condition of the co-state will depend on the target function F, which we can only access through the sampled data. A rigorous analysis of this function space control formulation and its consequences will be explored in future work.

B Proof of Lemma 1

Without loss of generality, we can assume the running loss (regularization loss) $L(\theta) \equiv 0$. This is because we can always define a new coordinate $Z \in \mathbb{R}$ such that

$$\dot{Z}_t = L(\theta_t), \qquad Z_0 = 0$$

and set the new terminal loss to $\Phi(X_T) + Z_T$. By redefining $X \leftarrow (X, Z)$ and f accordingly we arrive at the same form of the optimal control problem (1) but with $L \equiv 0$. With this redefinition, let us denote the loss function incurred by control θ as

$$J(\theta) := \Phi(X_T^\theta).$$

i.e. we seek to minimize $J(\theta)$.

First, observe that assumptions (A1)-(A2) in the main text implies that the second derivatives of f and Φ are bounded by K. Provided that P_t^θ is bounded, they also imply that the second derivatives of H with respect to x and p are bounded when evaluated on $X_t^\theta, P_t^\theta, \theta_t$. We first establish the boundedness of P_t^θ.

Lemma 2. *Assume that (A1)-(A2) hold. Then, there exists a constant $K > 0$ such that for any θ,*

$$\|P_t^\theta\| \leq K$$

for all $t \in [0, T]$.

Proof. Using (6) and setting $\tau := T - t$, $\tilde{P}_\tau^\theta := P_{T-\tau}^\theta$ we get

$$\dot{\tilde{P}}_\tau^\theta = \tilde{P}_\tau^\theta \cdot \nabla_x f(t, X_{T-\tau}^\theta, T - \tau), \qquad \tilde{P}_0^\theta = -\nabla\Phi(X_T^\theta)$$

Using (A1)-(A2), we have $\|P_T^\theta\| = \|\nabla_x \Phi(X_T^\theta)\| \leq K$ and $\|\nabla_x f(t, X_t^\theta, \theta_t)\|_F \leq K$. Hence,

$$\|\dot{\tilde{P}}_\tau^\theta\| \leq K\|\tilde{P}_\tau^\theta\|$$

and by Gronwall's inequality,

$$\|\tilde{P}_\tau^\theta\| \leq Ke^{KT}.$$

This proves the claim since it holds for any τ. □

We now prove Lemma 1. The approach here is similar to that employed in [58].

Proof of Lemma 1. From (5), we have for any $\theta \in \mathcal{U}$,

$$I(X^\theta, P^\theta, \theta) := \int_0^T P_t^\theta \cdot \dot{X}_t^\theta - H(t, X_t^\theta, P_t^\theta, \theta_t)dt \equiv 0$$

Denote $\delta X_t = X_t^\phi - X_t^\theta$ and $\delta P_t = P_t^\phi - P_t^\theta$, then we have

$$\begin{aligned} 0 \equiv &I(X^\phi, P^\phi, \phi) - I(X^\theta, P^\theta, \theta) \\ = &\int_0^T P_t^\theta \cdot \delta\dot{X}_t + \delta P_t \cdot \dot{X}_t^\theta + \delta P_t \cdot \delta\dot{X}_t dt \\ &- \int_0^T H(t, X_t^\phi, P_t^\phi, \phi_t) - H(t, X_t^\theta, P_t^\theta, \theta_t)dt \end{aligned} \tag{15}$$

Now, by integration by parts

$$\int_0^T P_t^\theta \cdot \delta\dot{X}_t dt = P_t^\theta \cdot \delta X_t \Big|_0^T - \int_0^T \dot{P}_t^\theta \cdot \delta X_t dt \tag{16}$$

$$\int_0^T \delta P_t \cdot \delta\dot{X}_t dt = \delta P_t \cdot \delta X_t \Big|_0^T - \int_0^T \delta\dot{P}_t \cdot \delta X_t dt \tag{17}$$

Using (5), (6) and (16), we have

$$\begin{aligned}
&\int_0^T P_t^\theta \cdot \delta \dot{X}_t + \delta P_t \cdot \dot{X}_t^\theta dt \\
=&P_t^\theta \cdot \delta X_t)\Big|_0^T + \int_0^T \left(f(t, X_t^\theta; \theta_t) \cdot \delta P + \nabla_x H(t, X_t^\theta, P_t^\theta, \theta_t) \cdot \delta X \right) dt \\
=&P_t^\theta \cdot \delta X_t\Big|_0^T + \int_0^T \left(\nabla_z H(t, Z_t^\theta, \theta_t) \cdot \delta Z \right) dt \qquad (18)
\end{aligned}$$

where in the last line we defined $Z^\theta := (X^\theta, P^\theta)$. Similarly, from (17) we get

$$\begin{aligned}
\int_0^T \delta P_t \cdot \delta \dot{X}_t dt =& \frac{1}{2}\int_0^T \delta P_t \cdot \delta \dot{X}_t dt + \frac{1}{2}\int_0^T \delta P_t \cdot \delta \dot{X}_t dt \\
=&\frac{1}{2}\delta P_t \cdot \delta X_t\Big|_0^T \\
&+ \frac{1}{2}\int_0^T \left([\nabla_z H(t, Z_t^\phi, \phi_t) - \nabla_z H(t, Z_t^\theta, \theta_t)] \cdot \delta Z_t \right) dt \\
=&\frac{1}{2}\delta P_t \cdot \delta X_t\Big|_0^T \\
&+ \frac{1}{2}\int_0^T [\nabla_z H(t, Z_t^\theta, \phi_t) - \nabla_z H(t, Z_t^\theta, \theta_t)] \cdot \delta Z_t dt \\
&+ \frac{1}{2}\int_0^T \delta Z_t \cdot \nabla_z^2 H(t, Z_t^\theta + r_1(t)\delta Z_t, \phi_t) \cdot \delta Z_t dt \qquad (19)
\end{aligned}$$

where we have used Taylor's theorem in the last step with $r_1(t) \in [0,1]$. We now rewrite the boundary terms. Since $\delta X_0 = 0$, we have

$$\begin{aligned}
&(P_t^\theta + \frac{1}{2}\delta P_t) \cdot \delta X_t\Big|_0^T = (P_T^\theta + \frac{1}{2}\delta P_T) \cdot \delta X_T \\
=& - \nabla\Phi(X_T^\theta) \cdot \delta X_T - \frac{1}{2}(\nabla\Phi(X_T^\phi) - \nabla\Phi(X_T^\theta)) \cdot \delta X_T \\
=& - \nabla\Phi(X_T^\theta) \cdot \delta X_T - \frac{1}{2}\delta X_T \cdot \nabla^2\Phi(X_T^\theta + r_2\delta X_T, Y) \cdot \delta X_T \\
=& - (J(\phi) - J(\theta)) - \frac{1}{2}\delta X_T \cdot (\nabla^2\Phi(X_T^\theta + r_2\delta X_T) + \nabla^2\Phi(X_T^\theta + r_3\delta X_T)) \cdot \delta X_T \qquad (20)
\end{aligned}$$

for some $r_2, r_3 \in [0,1]$. Lastly, for each $t \in [0,T]$ we have

$$\begin{aligned}
H(t, Z_t^\phi, \phi_t) - H(t, Z_t^\theta, \theta_t) =& H(t, Z_t^\theta, \phi_t) - H(t, Z_t^\theta, \theta_t) \\
&+ \nabla_z H(t, Z_t^\theta, \phi_t) \cdot \delta Z_t \\
&+ \frac{1}{2}\delta Z_t \cdot \nabla_z^2 H(t, Z_t^\theta + r_4(t)\delta Z_t, \phi_t) \cdot \delta Z_t \qquad (21)
\end{aligned}$$

where $r_4(t) \in [0,1]$.

Substituting (18), (19), (20), (21) into (15), we obtain

$$\begin{aligned}
&J(\phi) - J(\theta) \\
=&\frac{1}{2}\delta X_T \cdot (\nabla^2\Phi(X_T^\theta + r_2\delta X_T) + \nabla^2\Phi(X_T^\theta + r_3\delta X_T)) \cdot \delta X_T \\
&- \int_0^T \Delta H_{\phi,\theta}(t) dt \\
&- \frac{1}{2}\int_0^T (\nabla_z H(t, Z^\theta, \phi) - \nabla_z H(t, Z^\theta, \theta)) \cdot \delta Z_t dt \\
&+ \frac{1}{2}\int_0^T \left(\delta Z_t \cdot [\nabla_z^2 H(t, Z_t^\theta + r_1(t)\delta Z_t, \phi_t) - \nabla_z^2 H(t, Z_t^\theta + r_4(t)\delta Z_t, \phi_t)] \cdot \delta Z_t \right) dt. \qquad (22)
\end{aligned}$$

It remains to estimate the terms. First, let us estimate δX and δP. By definition,

$$\delta \dot{X}_t = f(t, X_t^\phi, \phi_t) - f(t, X_t^\theta, \theta_t).$$

Integrating, we get

$$\delta X_t = \int_0^t f(t, X_s^\phi, \phi_s) - f(t, X_s^\theta, \theta_s) ds,$$

and so

$$\begin{aligned}
\|\delta X_t\| \leq & \int_0^t \|f(t, X_s^\phi, \phi_s) - f(t, X_s^\theta, \theta_s)\| ds \\
\leq & \int_0^t \|f(t, X_s^\phi, \phi_s) - f(t, X_s^\theta, \phi_s)\| ds \\
& + \int_0^t \|f(t, X_s^\theta, \phi_s) - f(t, X_s^\theta, \theta_s)\| ds \\
\leq & \int_0^T \|f(t, X_s^\theta, \phi_s) - f(t, X_s^\theta, \theta_s)\| ds \\
& + K \int_0^t \|\delta X_s\| dt. \qquad (23)
\end{aligned}$$

By Gronwall's inequality, we have

$$\|\delta X_t\| \leq e^{KT} \int_0^T \|f(t, X_s^\theta, \phi_s) - f(t, X_s^\theta, \theta_s)\| ds. \qquad (24)$$

To estimate δP, we use the same substitution as in Lemma 2 with $\tau = T - t$ and $\tilde{\cdot}_\tau = \cdot_{T-t}$. We get

$$\delta \tilde{P}_\tau = \delta \tilde{P}_0 + \int_0^\tau \nabla_x H(t, \tilde{X}_s^\phi, \tilde{P}_s^\phi, \tilde{\phi}_s) - \nabla_x H(t, \tilde{X}_s^\theta, \tilde{P}_s^\theta, \tilde{\theta}_s) ds,$$

and hence using Lemma 2 and assumptions (A1)-(A2),

$$\begin{aligned}
\|\delta \tilde{P}_\tau\| \leq & \|\delta \tilde{P}_0\| + \int_0^\tau \|\nabla_x H(t, \tilde{X}_s^\phi, \tilde{P}_s^\phi, \tilde{\phi}_s) - \nabla_x H(t, \tilde{X}_s^\theta, \tilde{P}_s^\theta, \tilde{\theta}_s)\| ds \\
\leq & K \|\delta X_T\| + K^2 \int_0^T \|\delta X_s\| ds + K \int_0^\tau \|\delta \tilde{P}_s\| ds \\
& + \int_0^T \|\nabla_x H(t, X_s^\theta, P_s^\theta, \phi_s) - \nabla_x H(t, X_s^\theta, P_s^\theta, \theta_s)\| ds \\
\leq & e^{KT} K (\|\delta X_T\| + K \int_0^T \|\delta X_s\| ds) \\
& + e^{KT} \int_0^T \|\nabla_x H(t, X_s^\theta, P_s^\theta, \phi_s) - \nabla_x H(t, X_s^\theta, P_s^\theta, \theta_s)\| ds \qquad (25)
\end{aligned}$$

Using estimate (24), we obtain

$$\begin{aligned}
\|\delta P_t\| \leq & e^{2KT} K (1 + KT) \int_0^T \|f(t, X_s^\theta, \phi_s) - f(t, X_s^\theta, \theta_s)\| ds \\
& + e^{KT} \int_0^T \|\nabla_x H(t, X_s^\theta, P_s^\theta, \phi_s) - \nabla_x H(t, X_s^\theta, P_s^\theta, \theta_s)\| ds \qquad (26)
\end{aligned}$$

Now, we substitute estimates (24) and (26) into (22) and rename constants for simplicity. Note that by assumptions (A1)-(A2) and Lemma 2, all the second derivative terms are bounded element-wise by K.

Hence, we have $|\delta Z_t \cdot A \cdot \delta Z_t| \leq K\|\delta Z\|^2$ for each A being a second derivative matrix. Thus we obtain

$$\begin{aligned}
J(\phi) - J(\theta) \leq & -\int_0^T \Delta H_{\phi,\theta}(t)dt \\
& + K\|\delta X_T\|^2 \\
& + K\int_0^T (\|\delta X_t\|^2 + \|\delta P_t\|^2)dt \\
& + \frac{1}{2}\int_0^T \|\delta X_t\|\|f(t, X_t^\theta, \phi_t) - f(t, X_t^\theta, \theta_t)\|dt \\
& + \frac{1}{2}\int_0^T \|\delta P_t\|\|\nabla_x H(t, X_t^\theta, P_t^\theta, \phi_t) - \nabla_x H(t, X_t^\theta, P_t^\theta, \theta_t)\|dt \\
\leq & -\int_0^T \Delta H_{\phi,\theta}(t)dt \\
& + C\left(\int_0^T \|f(t, X_t^\theta, \phi_t) - f(t, X_t^\theta, \theta_t)\|dt\right)^2 \\
& + C\left(\int_0^T \|\nabla_x H(t, X_t^\theta, P_t^\theta, \phi_t) - \nabla_x H(t, X_t^\theta, P_t^\theta, \theta_t)\|^2 dt\right)^2 \qquad (27) \\
\leq & -\int_0^T \Delta H_{\phi,\theta}(t)dt \\
& + C\int_0^T \|f(t, X_t^\theta, \phi_t) - f(t, X_t^\theta, \theta_t)\|^2 dt \\
& + C\int_0^T \|\nabla_x H(t, X_t^\theta, P_t^\theta, \phi_t) - \nabla_x H(t, X_t^\theta, P_t^\theta, \theta_t)\|^2 dt
\end{aligned}$$

□

Remark 1. *For applications, the global Lipschitz condition (A2) w.r.t. x on f may be restrictive. Note that this can be replaced by a local Lipschitz condition if we can show that X_t, $t \in [0, T]$ is bounded for all $\theta \in \mathcal{U}$. This is true if the parameter space Θ is bounded, which we can always safely assume in practice.*

References

[1] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[2] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[3] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[4] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[5] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[6] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[7] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[8] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[9] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in neural information processing systems*, pages 315–323, 2013.

[10] Henry J Kelley. Gradient theory of optimal flight paths. *Ars Journal*, 30(10):947–954, 1960.

[11] Arthur Earl Bryson. *Applied optimal control: optimization, estimation and control.* CRC Press, 1975.

[12] Yann LeCun. A theoretical framework for back-propagation. In *The Connectionist Models Summer School*, volume 1, pages 21–28, 1988.

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

[14] Vladimir G Boltyanskii, Revaz V Gamkrelidze, and Lev S Pontryagin. The theory of optimal processes. I. The maximum principle. Technical report, TRW Space Tochnology Labs, Los Angeles, California, 1960.

[15] Lev S Pontryagin. *Mathematical theory of optimal processes.* CRC Press, 1987.

[16] Dimitri P Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena scientific Belmont, MA, 1995.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[18] Michael Athans and Peter L Falb. *Optimal control: an introduction to the theory and its applications.* Courier Corporation, 2013.

[19] Daniel Liberzon. *Calculus of variations and optimal control theory: a concise introduction.* Princeton University Press, 2012.

[20] Francis Clarke. The maximum principle in optimal control, then and now. *Control and Cybernetics*, 34(3):709, 2005.

[21] Richard Bellman. *Dynamic programming.* Courier Corporation, 2013.

[22] Alberto Bressan and Benedetto Piccoli. Introduction to mathematical control theory. *AIMS series on applied mathematics, Philadelphia*, 2007.

[23] Sanford M Roberts and Jerome S Shipman. Two-point boundary value problems: shooting methods. 1972.

[24] John T Betts. Survey of numerical methods for trajectory optimization. *Journal of Guidance control and dynamics*, 21(2):193–207, 1998.

[25] Dimitri P Bertsekas. *Nonlinear programming.* Athena scientific Belmont, 1999.

[26] Mokhtar S Bazaraa, Hanif D Sherali, and Chitharanjan M Shetty. *Nonlinear programming: theory and algorithms.* John Wiley & Sons, 2013.

[27] Anil V Rao. A survey of numerical methods for optimal control. *Advances in the Astronautical Sciences*, 135(1):497–528, 2009.

[28] Felix L Chernousko and Alexey A Lyubushin. Method of successive approximations for solution of optimal control problems. *Optimal Control Applications and Methods*, 3(2):101–114, 1982.

[29] Vladimir V Aleksandrov. On the accumulation of perturbations in the linear systems with two coordinates. *Vestnik MGU*, 3, 1968.

[30] Magnus R Hestenes. Multiplier and gradient methods. *Journal of optimization theory and applications*, 4(5):303–320, 1969.

[31] Richard HR Hahnloser, Rahul Sarpeshkar, Misha A Mahowald, Rodney J Douglas, and H Sebastian Seung. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, 405(6789):947, 2000.

[32] Anatolii B Butkovsky. Necessary and sufficient optimality conditions for sampled-data control systems. *Avtomat. i Telemekh*, 24(8):1056–1064, 1963.

[33] R Jackson and F Horn. On discrete analogues of pontryagin's maximum principle. *International Journal of Control*, 1(4):389–395, 1965.

[34] Zbigniew Nahorski, Hans F Ravn, and René Victor Valqui Vidal. The discrete-time maximum principle: a survey and some new results. *International Journal of Control*, 40(3):533–554, 1984.

[35] Hubert Halkin. A maximum principle of the pontryagin type for systems described by nonlinear difference equations. *SIAM Journal on control*, 4(1):90–111, 1966.

[36] Robert I Jennrich. Asymptotic properties of non-linear least squares estimators. *The Annals of Mathematical Statistics*, 40(2):633–643, 1969.

[37] Dong C Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

[38] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147, 2013.

[39] Yann LeCun. The MNIST database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[40] Yann LeCun and Yoshua Bengio. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[41] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[42] Kwang Y Lee and Mohamed A El-Sharkawi. *Modern heuristic optimization techniques: theory and applications to power systems*, volume 39. John Wiley & Sons, 2008.

[43] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*, pages 3981–3989, 2016.

[44] Max Jaderberg, Wojciech M Czarnecki, Simon Osindero, Oriol Vinyals, Alex Graves, and Koray Kavukcuoglu. Decoupled neural interfaces using synthetic gradients. *arXiv preprint arXiv:1608.05343*, 2016.

[45] Wojciech M Czarnecki, Grzegorz Świrszcz, Max Jaderberg, Simon Osindero, Oriol Vinyals, and Koray Kavukcuoglu. Understanding synthetic gradients and decoupled neural interfaces. *arXiv preprint arXiv:1703.00522*, 2017.

[46] Ivan A Krylov and Felix L Chernousko. On the method of successive approximations for solution of optimal control problems. *J. Comp. Mathem. and Mathematical Physics*, 2(6), 1962.

[47] Alexey A Lyubushin. Modifications of the method of successive approximations for solving optimal control problems. *USSR Computational Mathematics and Mathematical Physics*, 22(1):29–34, 1982.

[48] Gavin Taylor, Ryan Burmeister, Zheng Xu, Bharat Singh, Ankit Patel, and Tom Goldstein. Training neural networks without gradients: A scalable admm approach. In *International Conference on Machine Learning*, pages 2722–2731, 2016.

[49] Qianxiao Li, Cheng Tai, and Weinan E. Stochastic modified equations and adaptive stochastic gradient algorithms. In *International Conference on Machine Learning*, pages 2101–2110, 2017.

[50] Atilim G Baydin, Barak A Pearlmutter, Alexey A Radul, and Jeffrey M Siskind. Automatic differentiation in machine learning: a survey. *arXiv preprint arXiv:1502.05767*, 2015.

[51] Eldad Haber and Lars Ruthotto. Stable architectures for deep neural networks. *arXiv preprint arXiv:1705.03341*, 2017.

[52] Bo Chang, Lili Meng, Eldad Haber, Lars Ruthotto, David Begert, and Elliot Holtham. Reversible architectures for arbitrarily deep residual neural networks. *arXiv preprint arXiv:1709.03698*, 2017.

[53] Thomas Frerix, Thomas Möllenhoff, Michael Moeller, and Daniel Cremers. Proximal backpropagation. *arXiv preprint arXiv:1706.04638*, 2017.

[54] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in Neural Information Processing Systems*, pages 3123–3131, 2015.

[55] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to + 1 or - 1. *arXiv preprint arXiv:1602.02830*, 2016.

[56] Nikolay Pogodaev. Optimal control of continuity equations. *Nonlinear Differential Equations and Applications*, 23(2):21, 2016.

[57] Souvik Roy and Alfio Borz. Numerical investigation of a class of liouville control problems. *J Sci Comput*, 73:178, 2017.

[58] Lev I Rozonoer. The maximum principle of ls pontryagin in optimal-system theory. *Automation and Remote Control*, 20(10):11, 1959.