An Optimal Control Approach to Deep Learning and Applications to Discrete-Weight Neural Networks

Qianxiao Li [1] **Shuji Hao** [1]

Abstract

Deep learning is formulated as a discrete-time optimal control problem. This allows one to characterize necessary conditions for optimality and develop training algorithms that do not rely on gradients with respect to the trainable parameters. In particular, we introduce the discrete-time method of successive approximations (MSA), which is based on the Pontryagin's maximum principle, for training neural networks. A rigorous error estimate for the discrete MSA is obtained, which sheds light on its dynamics and the means to stabilize the algorithm. The developed methods are applied to train, in a rather principled way, neural networks with weights that are constrained to take values in a discrete set. We obtain competitive performance and interestingly, very sparse weights in the case of ternary networks, which may be useful in model deployment in low-memory devices.

1. Introduction

The problem of training deep feed-forward neural networks is often studied as a nonlinear programming problem (Bazaraa et al., 2013; Bertsekas, 1999; Kuhn & Tucker, 2014)

$$\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

where $\boldsymbol{\theta}$ represents the set of trainable parameters and J is the empirical loss function. In the general unconstrained case, necessary optimality conditions are given by the condition $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}^*) = 0$ for an optimal set of training parameters $\boldsymbol{\theta}^*$. This is largely the basis for (stochastic) gradient-descent based optimization algorithms in deep learning (Robbins & Monro, 1951; Duchi et al., 2011; Zeiler, 2012; Kingma & Ba, 2014). When there are additional constraints, e.g. on the trainable parameters, one can instead employ projected versions of the above algorithms. More broadly, necessary conditions for optimality can be derived in the form of the Karush-Kuhn-Tucker conditions (Kuhn & Tucker, 2014). Such approaches are quite general and typically do not rely on the structures of the objectives encountered in deep learning. However, in deep learning, the objective function J often has a specific structure; It is derived from feeding a batch of inputs recursively through a sequence of trainable transformations, which can be adjusted so that the final outputs are close to some fixed target set. This process resembles an optimal control problem (Bryson, 1975; Bertsekas, 1995; Athans & Falb, 2013) that originates from the study of the calculus of variations.

In this paper, we exploit this optimal control viewpoint of deep learning. First, we introduce the discrete-time Pontryagin's maximum principle (PMP) (Halkin, 1966), which is an extension the central result in optimal control due to Pontryagin and coworkers (Boltyanskii et al., 1960; Pontryagin, 1987). This is an alternative set of necessary conditions characterizing optimality, and we discuss the extent of its validity in the context of deep learning. Next, we introduce the discrete method of successive approximations (MSA) based on the PMP to optimize deep neural networks. A rigorous error estimate is proved that elucidates the dynamics of the MSA, and aids us in designing optimization algorithms under rather general conditions. We apply our method to train a class of unconventional networks, i.e. those with discrete-valued weights, to illustrate the usefulness of this approach. In the process, we discover that in the case of ternary networks, our training algorithm obtains trained models that are very sparse, which is an attractive feature in practice.

The rest of the paper is organized as follows: In Sec. 2, we introduce the optimal control viewpoint and the discrete-time Pontryagin's maximum principle. We then introduce the method of successive approximation in Sec. 3 and prove our main estimate, Theorem 2. In Sec. 4, we derive algorithms based on the developed theory to train binary and ternary neural networks. Finally, we end with a discussion on related work and a conclusion in Sec. 5 and 6 respectively. Various details on proofs and algorithms are provided in Appendix A-D, which also contains a link to a software implementation of our algorithms that reproduces all exper-

[1]Institute of High Performance Computing, Singapore. Correspondence to: Qianxiao Li <liqix@ihpc.a-star.edu.sg>.

Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018. Copyright 2018 by the author(s).

iments in this paper.

Hereafter, we denote the usual Euclidean norm by $\|\cdot\|$ and the corresponding induced matrix norm by $\|\cdot\|_2$. The Frobenius norm is written as $\|\cdot\|_F$. Throughout this work, we use a bold-faced version of a variable to represent a collection of the same variable, but indexed additionally by t, e.g. $\boldsymbol{\theta} := \{\theta_t : t = 0, \dots, T-1\}$.

2. The Optimal Control Viewpoint

In this section, we formalize the problem of training a deep neural network as an optimal control problem. Let $T \in \mathbb{Z}_+$ denote the number of layers and $\{x_{s,0} \in \mathbb{R}^{d_0} : s = 0, \dots, S\}$ represent a collection of fixed inputs (images, time-series). Here, $S \in \mathbb{Z}_+$ is the sample size. Consider the dynamical system

$$x_{s,t+1} = f_t(x_{s,t}, \theta_t), \quad t = 0, 1, \dots, T-1, \qquad (1)$$

where for each t, $f_t : \mathbb{R}^{d_t} \times \Theta_t \to \mathbb{R}^{d_{t+1}}$ is a transformation on the state. For example, in typical neural networks, it can represent a trainable affine transformation or a non-linear activation (in which case it is not trainable and f_t does not depend on θ). We assume that each trainable parameter set Θ_t is a subset of an Euclidean space. The goal of training a neural network is to adjust the weights $\boldsymbol{\theta} := \{\theta_t : t = 0, \dots, T-1\}$ so as to minimize some loss function that measures the difference between the final network output $x_{s,T}$ and some true targets y_s of $x_{s,0}$, which are fixed. Thus, we may define a family of real-valued functions $\Phi_s : \mathbb{R}^{d_T} \to \mathbb{R}$ acting on $x_{s,T}$ (y_s are absorbed into the definition of Φ_s) and the average loss function is $\sum_s \Phi_s(x_{s,T})/S$. In addition, we may consider some regularization terms for each layer $L_t : \mathbb{R}^{d_t} \times \Theta_t \to \mathbb{R}$ that has to be simultaneously minimized. In typical applications, regularization is only performed for the trainable parameters so that $L_t(x, \theta) \equiv L_t(\theta)$, but here we will consider the slightly more general case where it is also possible to regularize the state at each layer. In summary, we wish to solve the following problem

$$\min_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} J(\boldsymbol{\theta}) := \frac{1}{S} \sum_{s=1}^{S} \Phi_s(x_{s,T}) + \frac{1}{S} \sum_{s=1}^{S} \sum_{t=0}^{T-1} L_t(x_{s,t}, \theta_t)$$

subject to:

$$x_{s,t+1} = f_t(x_{s,t}, \theta_t),\ t = 0, \dots, T-1,\ s \in [S] \qquad (2)$$

where we have defined for shorthand $\boldsymbol{\Theta} := \{\Theta_0 \times \cdots \times \Theta_{T-1}\}$ and $[S] := \{1, \dots, S\}$. One may recognize problem (2) as a classical fixed-time, variable-terminal-state optimal control problem in discrete time (Ogata, 1995), in fact a special one with almost decoupled dynamics across samples in $[S]$.

2.1. The Pontryagin's Maximum Principle

Maximum principles of the Pontryagin type (Boltyanskii et al., 1960; Pontryagin, 1987) usually consist of necessary conditions for optimality in the form of the maximization of a certain Hamiltonian function. The distinguishing feature is that it does not assume differentiability (or even continuity) of f_t with respect to θ. Consequently the optimality condition and the algorithms based on it need not rely on gradient-descent type updates. This is an attractive feature for certain classes of applications.

Let $\boldsymbol{\theta}^* = \{\theta_0, \dots, \theta_{T-1}\} \in \boldsymbol{\Theta}$ be a solution of (2). We now outline informally the Pontryagin's maximum principle (PMP) that characterizes $\boldsymbol{\theta}^*$. First, for each t we define the Hamiltonian function $H_t : \mathbb{R}^{d_t} \times \mathbb{R}^{d_{t+1}} \times \Theta_t \to \mathbb{R}$ by

$$H_t(x, p, \theta) := p \cdot f_t(x, \theta) - \tfrac{1}{S} L_t(x, \theta). \qquad (3)$$

One can show the following necessary conditions.

Theorem 1 (Discrete PMP, Informal Statement). *Let f_t and Φ_s, $s = 1, \dots, S$ be sufficiently smooth in x. Assume further that for each t and $x \in \mathbb{R}^{d_t}$, the sets $\{f_t(x, \theta) : \theta \in \Theta_t\}$ and $\{L_t(x, \theta) : \theta \in \Theta_t\}$ are convex. Then, there exists co-state processes $\boldsymbol{p}_s^* := \{p_{s,t}^* : t = 0, \dots, T\}$, such that following holds for $t = 0, \dots, T-1$ and $s \in [S]$:*

$$x_{s,t+1}^* = \nabla_p H_t(x_{s,t}^*, p_{s,t+1}^*, \theta_t^*), \quad x_{s,0}^* = x_{s,0} \qquad (4)$$

$$p_{s,t}^* = \nabla_x H_t(x_{s,t}^*, p_{s,t+1}^*, \theta_t^*), \quad p_{s,T}^* = -\tfrac{1}{S} \nabla \Phi_s(x_{s,T}^*) \qquad (5)$$

$$\sum_{s=1}^{S} H_t(x_{s,t}^*, p_{s,t+1}^*, \theta_t^*) \geq \sum_{s=1}^{S} H_t(x_{s,t}^*, p_{s,t+1}^*, \theta), \forall \theta \in \Theta_t \qquad (6)$$

The full statement of Theorem 1 involve explicit smoothness assumptions and additional technicalities (such as the inclusion of an abnormal multiplier). In Appendix A, we state these assumptions and give a sketch of its proof based on Halkin (1966).

Let us discuss the PMP in detail. The state equation (4) is simply the forward propagation equation (1) under the optimal parameters $\boldsymbol{\theta}^*$. Eq. (5) defines the evolution of the co-state $\boldsymbol{p}_s^*$. To draw an analogy with nonlinear programming, the co-state can be interpreted as a set of Lagrange multipliers that enforces the constraint (1) when the optimization problem (2) is regarded as a joint optimization problem in $\boldsymbol{\theta}$ and $\boldsymbol{x}_s$, $s \in [S]$. In the optimal control and PMP viewpoint, it is perhaps more appropriate to think of the dynamics (5) as the evolution of the normal vector of a separating hyper-plane, which separates the set of reachable states and the set of states where the objective function takes on values smaller than the optimum (see Appendix A).

The Hamiltonian maximization condition (6) is the centerpiece of the PMP. It says that an optimal solution $\boldsymbol{\theta}^*$ must *globally maximize the (summed) Hamiltonian* for *each* layer $t = 0, \dots, T-1$. Let us contrast this statement with usual first-order optimality conditions of the form $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}^*) = 0$. A key observation is that in Theorem 1, we did not make reference to the derivative of any quantity with respect $\boldsymbol{\theta}$. In fact, the PMP holds even if f_t is not differentiable, or even continuous, with respect to θ, as long as the convexity assumptions are satisfied. On the other hand, if we assume for each t: 1) f_t is differentiable with respect to θ; 2) H_t is concave in θ; and 3) θ_t^* lies in the interior of Θ_t, then the Hamiltonian maximization condition (6) is equivalent to the condition $\nabla_\theta \sum_s H_t = 0$ for all t, which one can then show is equivalent to $\nabla_{\boldsymbol{\theta}} J = 0$ (See Appendix C, proof of Prop. C.1). In other words, the PMP can be viewed as a stronger set of necessary conditions (at optimality, H_t is not just stationary, but globally maximized) and has meaning in more general scenarios, e.g. when stationarity with respect to $\boldsymbol{\theta}$ is not achievable due to constraints, or not defined due to non-differentiability.

Remark 1. *It may occur that $\sum_s H_t(x_{s,t}^*, p_{s,t+1}^*, \theta)$ is constant for all $\theta \in \Theta_t$, in which case the problem is singular (Athans & Falb, 2013). In such cases, the PMP is trivially satisfied by any θ and so the PMP does not tell us anything useful. This may arise especially in the case where there are no regularization terms.*

2.2. The Convexity Assumption

The most crucial assumption in Theorem 1 is the convexity of the sets $\{f_t(x,\theta) : \theta \in \Theta_t\}$ and $\{L_t(x,\theta) : \theta \in \Theta_t\}$ for each fixed x [1]. We now discuss how restrictive these assumptions are with regard to deep neural networks. Let us first assume that the admissable sets Θ_t are convex. Then, the assumption with respect to L_t is not restrictive since most regularizers (e.g. ℓ_1, ℓ_2) satisfy it. Let us consider the convexity of $\{f_t(x,\theta) : \theta \in \Theta_t\}$. In classical feed-forward neural networks, there are two types of layers: trainable ones and non-trainable ones. Suppose layer t is non-trainable (e.g. $f(x_t,\theta_t) = \sigma(x_t)$ where σ is a non-linear activation function), then for each x the set $\{f_t(x,\theta) : \theta \in \Theta_t\}$ is a singleton, and hence trivially convex. On the other hand, in trainable layers f_t is usually affine in θ. This includes fully connected layers, convolution layers and batch normalization layers (Ioffe & Szegedy, 2015). In these cases, as long as the admissable set Θ_t is convex, we again satisfy the convexity assumption. Residual networks also satisfy the convexity constraint if one introduces auxiliary variables (see Appendix A.1). When the set Θ_t is not convex, then it is in general not true that the PMP constitute necessary conditions.

[1]Note that this is in general unrelated to the convexity, in the sense of functions, of f_t with respect to either x or θ. For example, the scalar function $f(x,\theta) = \theta^3 \sin(x)$ is evidently non-convex in both arguments, but $\{f(x,\theta) : \theta \in \mathbb{R}\}$ is convex for each x. On the other hand $\{\theta x : \theta \in \{-1,1\}\}$ is non-convex because of a non-convex admissible set.

Finally, we remark that in the original derivation of the PMP for continuous-time control systems (Boltyanskii et al., 1960) (i.e. $\dot{x}_{s,t} = f_t(x_{s,t},\theta_t), t \in [0,T]$ in place of Eq. (1)), the convexity condition can be removed due to the "convexifying" effect of integration with respect to time (Halkin, 1966; Warga, 1962). Hence, the convexity condition is purely an artifact of discrete-time dynamical systems.

3. The Method of Successive Approximations

The PMP (Eq. (4)-(6)) gives us a set of necessary conditions an optimal solution to (2) must satisfy. However, it does not tell us how to find one such solution. The goal of this section is to discuss algorithms for solving (2) based on the maximum principle.

On closer inspection of Eq. (4)-(6), one can see that they each represent a manifold in solution space consisting of all possible $\boldsymbol{\theta}$, $\{\boldsymbol{x}_s, s \in [S]\}$ and $\{\boldsymbol{p}_s, s \in [S]\}$, and the intersection of these three manifolds must contain an optimal solution, if one exists. Consequently, an iterative projection method that successively projects a guessed solution onto each of the manifolds is natural. This is the method of successive approximations (MSA), which was first introduced to solve continuous-time optimal control problems (Krylov & Chernousko, 1962; Chernousko & Lyubushin, 1982). Let us now outline a discrete-time version.

Start from an initial guess $\boldsymbol{\theta}^0 := \{\theta_t^0, t = 0,\dots,T-1\}$. For each sample s, we define $\boldsymbol{x}_s^{\boldsymbol{\theta}^0} := \{x_{s,t}^{\boldsymbol{\theta}^0} : t = 0,\dots,T\}$ by the dynamics

$$x_{s,t+1}^{\boldsymbol{\theta}^0} = f_t(x_{s,t}^{\boldsymbol{\theta}^0}, \theta_t^0), \quad x_{s,0}^{\boldsymbol{\theta}^0} = x_{s,0}, \tag{7}$$

for $t = 0,\dots,T-1$. Intuitively, this is a projection onto the manifold defined by Eq. (4). Next, we perform the projection onto the manifold defined by Eq. (5), i.e. we define $\boldsymbol{p}_s^{\boldsymbol{\theta}^0} := \{p_{s,t}^{\boldsymbol{\theta}^0} : t = 0,\dots,T\}$ by the backward dynamics

$$p_{s,t}^{\boldsymbol{\theta}^0} = \nabla_x H(x_{s,t}^{\boldsymbol{\theta}^0}, p_{s,t+1}^{\boldsymbol{\theta}^0}, \theta_t^0), \quad p_{s,T}^{\boldsymbol{\theta}^0} = -\tfrac{1}{S}\nabla\Phi_s(x_{s,T}^{\boldsymbol{\theta}^0}), \tag{8}$$

for $t = T-1,\dots,0$. Finally, we project onto manifold defined by Eq. (6) by performing Hamiltonian maximization to obtain $\boldsymbol{\theta}^1 := \{\theta_t^1 : t = 0,\dots,T-1\}$ with

$$\theta_t^1 = \operatorname*{arg\,max}_{\theta\in\Theta_t} \sum_{s=1}^{S} H_t(x_{s,t}^{\boldsymbol{\theta}^0}, p_{s,t+1}^{\boldsymbol{\theta}^0}, \theta). \quad t = 0,\dots,T-1. \tag{9}$$

The steps (7)-(9) are then repeated until convergence. We summarize the basic MSA algorithm in Alg. 1.

Let us contrast the MSA with gradient-descent based methods. Similar to the formulation of the PMP, at no point did we take the derivative of any quantity with respect to θ. Hence, we can in principle apply this to problems that

Algorithm 1 Basic MSA

```
Initialize: $\boldsymbol{\theta}^0 = \{\theta^0_t \in \Theta_t : t = 0 \dots, T-1\}$;
for $k = 0$ to #Iterations do
    $x^{\boldsymbol{\theta}^k}_{s,t+1} = f_t(x^{\boldsymbol{\theta}^k}_{s,t}, \theta^k_t)$, $x^{\boldsymbol{\theta}^k}_{s,0} = x_{s,0}, \forall s, t$;
    $p^{\boldsymbol{\theta}^k}_{s,t} = \nabla_x H_t(x^{\boldsymbol{\theta}^k}_{s,t}, p^{\boldsymbol{\theta}^k}_{s,t+1}, \theta^k_t), p^{\boldsymbol{\theta}^k}_{s,T} = -\frac{1}{S}\nabla\Phi_s(x_{s,T})$, $\forall s, t$;
    $\theta^{k+1}_t = \arg\max_{\theta\in\Theta_t} \sum_{s=1}^S H_t(x^{\boldsymbol{\theta}^k}_{s,t}, p^{\boldsymbol{\theta}^k}_{s,t+1}, \theta)$ for $t = 0, \dots, T-1$;
end for
```

are not differentiable with respect to $\boldsymbol{\theta}$. However, the catch is that the Hamiltonian maximization step (9) may not be trivial to evaluate. Nevertheless, observe that the maximization step is *decoupled* across different layers of the neural network, and hence it is a much smaller problem than the original optimization problem, and its solution method can be parallelized. Alternatively, as seen in Sec. 4, one can exploit cases where the maximization step has explicit solutions.

The basic MSA (Alg. 1 can be shown to converge for problems where f_t is linear and the costs Φ_s, L_t are quadratic (Aleksandrov, 1968). In general, however, unless a good initial condition is given, the MSA may diverge. Let us understand the nature of such phenomena by obtaining rigorous error estimates per-iteration of Eq. (7)-(9).

3.1. An Error Estimate for the MSA

In this section, we derive a rigorous error estimate for the MSA, which can help us understand its dynamics. Let us define $W_t := \mathrm{conv}\{x \in \mathbb{R}^{d_t} : \exists \boldsymbol{\theta} \text{ and } s \text{ s.t. } x^{\boldsymbol{\theta}}_{s,t} = x\}$, where $x^{\boldsymbol{\theta}}_t$ is defined according to Eq. (7). This is the convex hull of all states reachable at layer t by some initial sample and some choice of the values for the trainable parameters. Let us now make the following assumptions:

(A1) Φ_s is twice continuously differentiable, with Φ_s and $\nabla\Phi_s$ satisfying a Lipschitz condition, i.e. there exists $K > 0$ such that for all $x, x' \in W_T$ and $s \in [S]$

$$|\Phi_s(x) - \Phi_s(x')| + \|\nabla\Phi_s(x) - \nabla\Phi_s(x')\| \le K\|x - x'\|$$

(A2) $f_t(\cdot, \theta)$ and $L_t(\cdot, \theta)$ are twice continuously differentiable in x, with $f_t, \nabla_x f_t, L_t, \nabla_x L_t$ satisfying Lipschitz conditions in x uniformly in t and θ, i.e. there exists $K > 0$ such that

$$\begin{aligned}&\|f_t(x,\theta) - f_t(x',\theta)\| + \|\nabla_x f_t(x,\theta) - \nabla_x f_t(x',\theta)\|_2 \\ &+ |L_t(x,\theta) - L_t(x',\theta)| + \|\nabla_x L_t(x,\theta) - \nabla_x L_t(x',\theta)\| \\ &\le K\|x - x'\|\end{aligned}$$

for all $x, x' \in W_t, \theta \in \Theta_t$ and $t = 0, \dots, T-1$.

Again, let us discuss these assumptions with respect to neural networks. Note that both assumptions are more easily satisfied if each W_t is bounded, which is usually implied by the boundedness of Θ_t. Although this is not typically true in principle, we can safely assume this in practice by truncating weights that are too large in magnitude. Consequently, (A1) is not very restrictive, since many commonly employed loss functions (mean-square, soft-max with cross-entropy) satisfy these assumptions. In (A2), the regularity assumption on L_t is again not an issue, because we mostly take L_t to be independent of x. On the other hand, the regularity of f_t with respect to x is sometimes restrictive. For example, ReLU activations does not satisfy (A2) due to non-differentiability. Nevertheless, any suitably mollified version (like Soft-plus) does satisfy it. Moreover, tanh and sigmoid activations also satisfy (A2). Finally, unlike in Theorem 1, we do not assume the convexity of the sets $\{f_t(x,\theta) : \theta \in \Theta_t\}$ and $\{L_t(x,\theta) : \theta \in \Theta_t\}$, and hence the results in this section applies to discrete-weight neural networks considered in Sec. 4. With the above assumptions, we prove the following estimate.

Theorem 2 (Error Estimate for Discrete MSA). *Let assumptions (A1) and (A2) be satisfied. Then, there exists a constant $C > 0$, independent of S, $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$, such that for any $\boldsymbol{\theta}, \boldsymbol{\phi} \in \boldsymbol{\Theta}$, we have*

$$J(\boldsymbol{\phi}) - J(\boldsymbol{\theta})$$

$$\le -\sum_{t=0}^{T-1}\sum_{s=1}^{S} H_t(x^{\boldsymbol{\theta}}_{s,t}, p^{\boldsymbol{\theta}}_{s,t+1}, \phi_t) - H_t(x^{\boldsymbol{\theta}}_{s,t}, p^{\boldsymbol{\theta}}_{s,t+1}, \theta_t) \quad (10)$$

$$+ \frac{C}{S}\sum_{t=0}^{T-1}\sum_{s=1}^{S} \|f_t(x^{\boldsymbol{\theta}}_{s,t}, \phi_t) - f_t(x^{\boldsymbol{\theta}}_{s,t}, \theta_t)\|^2 \quad (11)$$

$$+ \frac{C}{S}\sum_{t=0}^{T-1}\sum_{s=1}^{S} \|\nabla_x f_t(x^{\boldsymbol{\theta}}_{s,t}, \phi_t) - \nabla_x f_t(x^{\boldsymbol{\theta}}_{s,t}, \theta_t)\|_2^2, \quad (12)$$

$$+ \frac{C}{S}\sum_{t=0}^{T-1}\sum_{s=1}^{S} \|\nabla_x L_t(x^{\boldsymbol{\theta}}_{s,t}, \phi_t) - \nabla_x L_t(x^{\boldsymbol{\theta}}_{s,t}, \theta_t)\|^2, \quad (13)$$

where $\boldsymbol{x}^{\boldsymbol{\theta}}_s$, $\boldsymbol{p}^{\boldsymbol{\theta}}_s$ are defined by Eq. (7) and (8).

Proof. The proof follows from elementary estimates and a discrete Gronwall's lemma. See Appendix B. □

Theorem 2 relates the decrement of the total objective function J with respect to the iterative projection steps of the MSA. Intuitively, Theorem 2 says that the Hamiltonian maximization step (9) is generally the right direction, because a large magnitude of (10) results in higher loss improvement. However, whenever we change the parameters from $\boldsymbol{\theta}$ to $\boldsymbol{\phi}$ (e.g. during the maximization step (9)), we incur non-negative penalty terms (11)-(13). Observe that these penalty terms vanish if $\boldsymbol{\phi} = \boldsymbol{\theta}$, or more generally, when the state and co-state equations (Eq. (7), (8)) are still satisfied when $\boldsymbol{\theta}$ is replaced by $\boldsymbol{\phi}$. In other words, these terms measure the distance from manifolds defined by the state and co-state equations when the parameter changes. Alg. 1 diverges when these penalty terms dominate the gains from (10). This insight can point us in the right direction of developing convergent modifications of the basic MSA. We shall now discuss this in the context of some specific applications.

4. Neural Networks with Discrete Weights

We now turn to the application of the theory developed in the previous section on the MSA, which is a PMP-based numerical method for training deep neural networks. As discussed previously, the main strength of the PMP and MSA formalism is that we do not rely on gradient-descent type updates. This is particularly useful when one considers neural networks with (some) trainable parameters that can only take values in a discrete set. Then, any small gradient update to the parameters will almost always be infeasible. In this section, we will consider two such cases: *binary networks*, where weights are restricted to $\{-1,+1\}$; and *ternary networks*, where weights are selected from $\{-1,+1,0\}$. These networks are potentially useful for low-memory devices as storing the trained weights requires less memory. In this section, we will modify the MSA so that we can train these networks in a principled way.

4.1. Binary Networks

Binary neural networks are those with binary trainable layers, e.g. in the fully connected case,

$$f_t(x,\theta) = \theta x \tag{14}$$

where $\theta \in \Theta_t = \{-1,+1\}^{d_t \times d_{t+1}}$ is a binary matrix. A similar form of f_t holds for convolution neural networks after reshaping, except that Θ_t is now the set of Toeplitz binary matrices. Hereafter, we will consider the fully connected case for simplicity of exposition. It is also natural to set the regularization to 0 since there is in general no preference between $+1$ or -1. Thus, the Hamiltonian has the form

$$H_t(x,p,\theta) = p \cdot \theta x.$$

Consequently, the Hamiltonian maximization step (9) has explicit solution, given by

$$\operatorname*{arg\,max}_{\theta\in\Theta_t} \sum_{s=1}^{S} H_t(x^{\boldsymbol{\theta}^k}_{s,t}, p^{\boldsymbol{\theta}^k}_{s,t+1}, \theta) = \operatorname{sign}(M^{\boldsymbol{\theta}^k}_t)$$

where $M^{\boldsymbol{\theta}}_t := \sum_{s=1}^{S} p^{\boldsymbol{\theta}}_{s,t+1} (x^{\boldsymbol{\theta}}_{s,t})^T$. Note that the sign function is applied element-wise. If $[M^{\boldsymbol{\theta}}_t]_{ij} = 0$, then the argmax is arbitrary. Using Theorem 2 with the form of f_t given by (14) and the fact that $L_t \equiv 0$, we get

$$\begin{aligned} J(\boldsymbol{\phi}) - J(\boldsymbol{\theta}) \leq & - \sum_{t=0}^{T-1}\sum_{s=1}^{S} H_t(x^{\boldsymbol{\theta}^k}_{s,t}, p^{\boldsymbol{\theta}^k}_{s,t+1}, \theta) \\ & + \frac{C}{S}\sum_{t=0}^{T-1}(1+\sum_{s=1}^{S}\|x^{\boldsymbol{\theta}}_{s,t}\|^2)\|\phi_t - \theta_t\|_F^2, \end{aligned}$$

Note that we have used the inequality $\|\cdot\|_2 \leq \|\cdot\|_F$. Assuming that $\|x^{\theta}_{s,t}\|$ is $\mathcal{O}(1)$, we may then decrease J by not only maximizing the Hamiltonian, but also penalizing the difference $\|\phi_t - \theta_t\|_F$, i.e. for each k and t we set

$$\theta^{k+1}_t = \operatorname*{arg\,max}_{\theta\in\Theta_t} \left[\sum_{s=1}^{S} H_t(x^{\boldsymbol{\theta}^k}_{s,t}, p^{\boldsymbol{\theta}^k}_{s,t+1}, \theta) - \rho_{k,t}\|\theta - \theta^k\|_F^2 \right] \tag{15}$$

for some penalization parameters $\rho_{k,t} > 0$. This again has the explicit solution

$$[\theta^{k+1}_t]_{ij} = \begin{cases} \operatorname{sign}([M^{\boldsymbol{\theta}^k}_t]_{ij}) & |[M^{\boldsymbol{\theta}^k}_t]_{ij}| \geq 2\rho_{k,t} \\ [\theta^k_t]_{ij} & \text{otherwise} \end{cases} \tag{16}$$

Therefore, we simply replace the parameter update step in Alg. 1 with (16). Furthermore, to deal with mini-batches, we keep a moving average of $M^{\boldsymbol{\theta}^k}_t$ across different mini-batches and use the averaged value to update our parameters. It is found empirically that this further stabilizes the algorithm. Note that the assumption $\|x^{\boldsymbol{\theta}}_{s,t}\|$ is $\mathcal{O}(1)$ can be achieved by normalization, e.g. batch-normalization (Ioffe & Szegedy, 2015). We summarize the algorithm in Alg. 2. Further algorithmic details are found in Appendix D, where we also discuss the choice of hyper-parameters and the convergence of the algorithm for a simple binary regression problem. A rigorous proof of convergence in the general case is beyond the scope of this work, but we demonstrate via experiments below that the algorithm performs well on the tested benchmarks.

We apply Alg. 2 to train binary neural networks on various benchmark datasets and compare the results from previous work on training binary-weight neural networks (Courbariaux et al., 2015). We consider a fully-connected neural network on MNIST (LeCun, 1998), as well as (shallow) convolutional networks on CIFAR-10 (Krizhevsky & Hinton, 2009) and SVHN (Netzer et al., 2011) datasets. The network structures are mostly identical to those considered in Courbariaux et al. (2015) for ease of comparison. Complete implementation and model details are found in Appendix D. The graphs of training/testing loss and error rates are shown in Fig. 1. We observe that our algorithm performs well in terms of an optimization algorithm, as measured by the training loss and error rates. For the harder datasets

Algorithm 2 Binary MSA

```
Initialize: $\boldsymbol{\theta}^0$, $\overline{\mathbf{M}}^0$;
Hyper-parameters: $\rho_{k,t}$, $\alpha_{k,t}$;
**for** $k = 0$ **to** #Iterations **do**
    $x^{\boldsymbol{\theta}^k}_{s,t+1} = f_t(x^{\boldsymbol{\theta}^k}_{s,t}, \theta^k_t) \qquad \forall s, t$
        with $x^{\boldsymbol{\theta}^k}_{s,0} = x_{s,0}$;
    $p^{\boldsymbol{\theta}^k}_{s,t} = \nabla_x H_t(x^{\boldsymbol{\theta}^k}_{s,t}, p^{\boldsymbol{\theta}^k}_{s,t+1}, \theta^k_t) \qquad \forall s, t$
        with $p^{\boldsymbol{\theta}^k}_{s,T} = -\frac{1}{S}\nabla\Phi_s(x_{s,T})$;
    $\overline{M}^{k+1}_t = \alpha_{k,t}\overline{M}^k_t + (1-\alpha_{k,t})\sum_{s=1}^S p^{\boldsymbol{\theta}^k}_{s,t+1}(x^{\boldsymbol{\theta}^k}_{s,t})^T$
    $[\theta^{k+1}_t]_{ij} = \begin{cases} \text{sign}([\overline{M}^{k+1}_t]_{ij}) & |[\overline{M}^{k+1}_t]_{ij}| \geq 2\rho_{k,t} \\ [\theta^k_t]_{ij} & \text{otherwise} \end{cases}$
    $\forall t$, $i$, and $j$;
**end for**
```

(CIFAR-10 and SVHN), we have rapid convergence but worse test loss and error rates at the end, possibly due to overfitting. We note that in (Courbariaux et al., 2015), many regularization strategies are performed. We expect that similar techniques must be employed to improve the testing performance. However, these issues are out of the scope of the optimization framework of this paper. Note that we also compared the results of BinaryConnect without regularization strategies such as stochastic binarization, but the results are similar in that our algorithm converges very fast with very low training losses, but sometimes overfits.

4.2. Ternary Networks

We shall consider another case where the network weights are allowed to take on values in $\{-1, +1, 0\}$. In this case, our goal is to explore the sparsification of the network. To this end, we shall take $L_t(x, \theta) = \lambda_t \|\theta\|_F^2$ for some parameter λ_t. Note that since the weights are restricted to the ternary set, any component-wise ℓ_p regularization for $p > 0$ are identical. The higher the λ_t values, the more sparse the solution will be.

As in Sec. 4.1, we can write down the Hamiltonian for a fully connected ternary layer as

$$H_t(x, p, \theta) = p \cdot \theta x - \tfrac{1}{S}\lambda_t \|\theta\|_F^2.$$

The derivation of the ternary algorithm then follows directly from those in Sec. 4.1, but with the new form of Hamiltonian above and that $\Theta_t = \{-1, +1, 0\}^{d_t \times d_{t+1}}$. Maximizing the augmented Hamiltonian (15) with H_t as defined above, we obtain the ternary update rule

$$[\theta^{k+1}_t]_{ij} = \begin{cases} +1 & [M^{\boldsymbol{\theta}^k}_t]_{ij} \geq \rho_{k,t}(1 - 2[\theta^k_t]_{ij}) + \lambda_t \\ -1 & [M^{\boldsymbol{\theta}^k}_t]_{ij} \leq -\rho_{k,t}(1 + 2[\theta^k_t]_{ij}) - \lambda_t \\ 0 & \text{otherwise.} \end{cases} \tag{17}$$

We replace the parameter update step in Alg. 2 by (17) to obtain the MSA algorithm for ternary networks. For completeness, we give the full ternary algorithm in Alg. 3. We now test the ternary algorithm on the same benchmarks used in Sec. 4.1 and the results are shown in Fig. 2. Observe that the performance on training and testing datasets is similar to the binary case (Fig. 1), but the ternary networks achieve high degrees of sparsity in the weights, with only 0.5-2.5% of the trained weights being non-zero, depending on the dataset. This potentially offers significant memory savings compared to its binary or full floating precision counterparts.

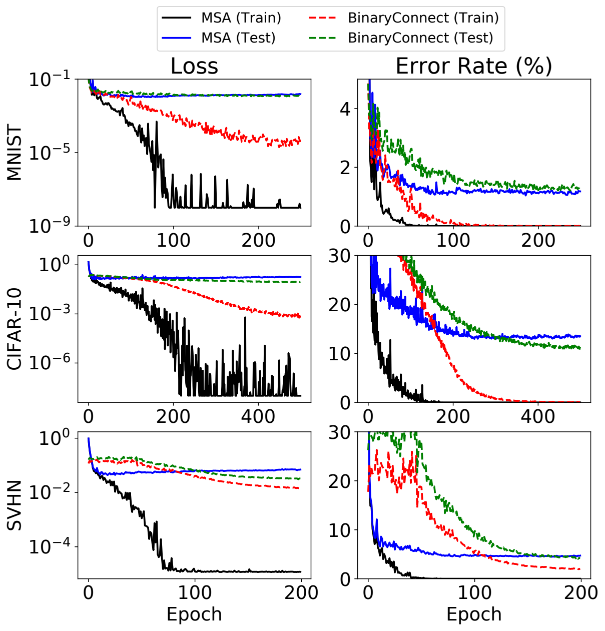

Figure 1. Comparison of binary MSA (Alg. 2) with BinaryConnect (Courbariaux et al., 2015) (with binary variables for inference). We observe that MSA has good convergence in terms of the training loss and error rates, showing that it is an efficient optimization algorithm. Note that to avoid broken lines, when the loss equals 0 exactly, we replace it by 1e-8 on the log-scale. The test loss for the bigger datasets (CIFAR10, SVHN) eventually becomes worse due to overfitting, hence some regularization techniques is needed for applications which are prone to overfitting.

5. Discussion and Related Work

We begin with a discussion of the results presented thus far. We first introduced the viewpoint that deep learning can be regarded as a discrete-time optimal control problem. Consequently, an important result in optimal control theory, the Pontryagin's maximum principle, can be applied to give a set of necessary conditions for optimality. These are in general stronger conditions than the usual optimality conditions based on the vanishing of first-order partial derivatives. Moreover, they apply to broader contexts such as problems with constraints on the trainable parameters or problems that are non-differentiable in the trainable parameters. However, we note that specific assumptions regarding the convexity

Algorithm 3 Ternary MSA

```
Initialize: $\boldsymbol{\theta}^0$, $\overline{\mathbf{M}}^0$;
Hyper-parameters: $\rho_{k,t}$, $\alpha_{k,t}$;
for $k = 0$ to #Iterations do
    $x^{\boldsymbol{\theta}^k}_{s,t+1} = f_t(x^{\boldsymbol{\theta}^k}_{s,t}, \theta^k_t) \qquad \forall s, t$
        with $x^{\boldsymbol{\theta}^k}_{s,0} = x_{s,0}$;
    $p^{\boldsymbol{\theta}^k}_{s,t} = \nabla_x H_t(x^{\boldsymbol{\theta}^k}_{s,t}, p^{\boldsymbol{\theta}^k}_{s,t+1}, \theta^k_t) \qquad \forall s, t$
        with $p^{\boldsymbol{\theta}^k}_{s,T} = -\frac{1}{S}\nabla\Phi_s(x_{s,T})$;
    $\overline{M}^{k+1}_t = \alpha_{k,t}\overline{M}^k_t + (1-\alpha_{k,t})\sum_{s=1}^S p^{\boldsymbol{\theta}^k}_{s,t+1}(x^{\boldsymbol{\theta}^k}_{s,t})^T$
    $[\theta^{k+1}_t]_{ij} = \begin{cases} +1 & [\overline{M}^{k+1}_t]_{ij} \geq \rho_{k,t}(1-2[\theta^k_t]_{ij}) + \lambda_t \\ -1 & [\overline{M}^{k+1}_t]_{ij} \leq -\rho_{k,t}(1+2[\theta^k_t]_{ij}) - \lambda_t \\ 0 & \text{otherwise.} \end{cases}$
    $\forall t$, $i$, and $j$;
end for
```

of some sets must be satisfied. We showed that they are justified for conventional neural networks, but not necessarily so for all neural networks (e.g. binary, ternary networks).

Next, based on the PMP, we introduced an iterative projection technique, the discrete method of successive approximations (MSA), to find an optimal solution of the learning problem. A rigorous error estimate (Theorem 2) is derived for the discrete MSA, which can be used to both understand its dynamics and to derive useful algorithms. This should be viewed as the main theoretical result of the present paper. Note that the usual back-propagation with gradient descent can be regarded as a simple modification of the MSA, if differentiability conditions are assumed (see Appendix C). Nevertheless, we note that Theorem 2 itself does not assume any regularity conditions with respect to the trainable parameters. Moreover, neither does it require the convexity conditions in Theorem 1, and hence applies to a wider range of neural networks, including those in Sec. 4. All results up to this point apply to general neural networks (assuming that the respective conditions are satisfied), and are not specific to the applications presented subsequently.

In the last part of this work, we apply our results to devise training algorithms for discrete-weight neural networks, i.e. those with trainable parameters that can only take values in a discrete set. Besides potential applications in model deployment in low-memory devices, the main reasons for choosing such applications are two-fold. First, gradient-descent updates are not applicable by itself because small updates to parameters are prohibited by the discrete equality constraint on the trainable parameters. However, our method based on the MSA is applicable since it does not perform gradient-descent updates. Second, in such applications the potentially expensive Hamiltonian maximization steps in the MSA have explicit solutions. This makes MSA an attractive optimization method for problems of this nature. In Sec 4, we demonstrate the effectiveness of our methods on various benchmark datasets. Interestingly, the ternary

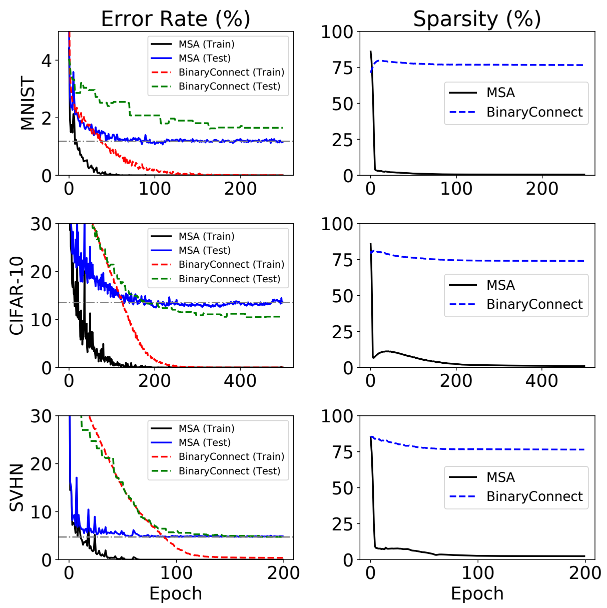

Figure 2. Performance of the ternary MSA (Alg. 3) vs. BinaryConnect with a simple thresholding procedure described in (Li et al., 2016). In the second column of plots, we show the sparsity of the networks (defined as the percentage of all weights that are non-zero) as training proceeds. Observe that the MSA algorithm finds solutions with comparable error rates with the binary case (whose final test error is plotted as a gray horizontal line) but are very sparse. In comparison, the simple thresholding of BinaryConnect does not produce sparse solutions. In fact, the final sparsities for MSA are approximately: MNIST: <1.0%; CIFAR-10: 0.9%; SVHN: 2.4%. It is expected that sparser solutions can be found by adjusting the penalty parameters λ_t.

network exhibits extremely sparse weights that perform almost as well as its binary counter-part (see Fig. 2). Also, the phenomena of overfitting in Fig. 1 and 2 are interesting as overfitting is generally less common in stochastic gradient based optimization approaches. This seems to suggest that the MSA based methods optimize neural networks in a rather different way.

Let us now put our work in the context of the existing literature. First, the optimal control approach we adopt is quite different from the prevailing viewpoint of nonlinear programming (Bertsekas, 1999; Bazaraa et al., 2013; Kuhn & Tucker, 2014) and the analysis of the derived gradient-based algorithms (Moulines, 2011; Shamir & Zhang, 2013; Bach & Moulines, 2013; Xiao & Zhang, 2014; Shalev-Shwartz & Zhang, 2014) for the training of deep neural networks. In particular, the PMP (Thm. 1) and the MSA error estimate (Thm. 2) do not assume differentiability and do not characterize optimality via gradients (or sub-gradients) with respect to trainable parameters. In this sense, it is a stronger and more robust condition, albeit sometimes requiring different assumptions. The optimal control and dynamical systems viewpoint has been discussed in the context of deep learning in E (2017); Li et al. (2018) and dynamical systems based discretization schemes has been introduced in Haber & Ruthotto (2017); Chang et al. (2017). Most of these works have theoretical basis in continuous-time dynamical systems. In particular, Li et al. (2018) analyzed continuous-time analogues of neural networks in the optimal control framework and derived MSA-based algorithms in continuous time. In contrast, the present work presents a discrete-time formulation, which is natural in the usual context of deep learning. The discrete PMP turns out to be more subtle, as it requires additional assumptions of convexity of reachable sets (Thm. 1). Note also that unlike the estimates derived in Li et al. (2018), Thm. 2 holds rigorously for discrete-time neural networks. The present method for stabilizing the MSA is also different from that in Li et al. (2018), where augmented Lagrangian type of modifications are employed. The latter would not be effective here because weights cannot be updated infinitesimally without violating the binary/ternary constraint. Moreover, the present methods that rely on explicit solutions of Hamiltonian maximization are fast (comparable to SGD) on a wall-clock basis.

In the deep learning literature, the connection between optimal control and deep learning has been qualitative discussed in LeCun (1988) and applied to the development of automatic differentiation and back-propagation (Bryson, 1975; Baydin et al., 2015). However, there are relatively fewer works relating optimal control algorithms to training neural networks beyond the classical gradient-descent with back-propagation. Optimal control based strategies in hyper-parameter tuning has been discussed in Li et al. (2017b).

In the continuous-time setting, the Pontryagin's maximum principle and the method of successive approximations have a long history, with a large body of relevant literature including, but not limited to Boltyanskii et al. (1960); Pontryagin (1987); Bryson (1975); Bertsekas (1995); Athans & Falb (2013); Krylov & Chernousko (1962); Aleksandrov (1968); Krylov & Chernousko (1972); Chernousko & Lyubushin (1982); Lyubushin (1982). The discrete-time PMP have been studied in Halkin (1966); Holtzman (1966a); Holtzman & Halkin (1966); Holtzman (1966b); Canon et al. (1970), where Theorem 1 and its extensions are proved. To the best of our knowledge, the discrete-time MSA and its quantitative analysis have not been performed in either the deep learning or the optimal control literature.

Sec. 4 concerns the application of the MSA, in particular Thm. 2, to develop training algorithms for binary and ternary neural networks. There are a number of prior work exploring the training of similar neural networks, such as Courbariaux et al. (2015); Hubara et al. (2016); Rastegari et al. (2016); Tang et al. (2017); Li et al. (2016); Zhu et al. (2016). Theoretical analysis for the case of convex loss functions is carried out in Li et al. (2017a). Our point of numerical comparison for the binary MSA algorithm is Courbariaux et al. (2015), where optimization of binary networks is based on shadow variables with full floating-point precision that is iteratively truncated to obtain gradients. We showed in Sec. 4.1 that the binary MSA is competitive as a training algorithm, but is in need of modifications to reduce overfitting for certain datasets. Training ternary networks has been discussed in Hwang & Fan (1967); Kim et al. (2014); Li et al. (2016); Zhu et al. (2016). The difference in our ternary formulation is that we explore the sparsification of networks using a regularization parameter. In this sense it is related to compression of neural networks (e.g. Han et al. (2015)), but our approach trains a network that is naturally ternary, and compression is achieved during training by a regularization term. Generally, a contrasting aspect of our approach from the aforementioned literature is that the theory of optimal control, together with Theorem. 2, provide a theoretical basis for the development of our algorithms. Nevertheless, further work is required to rigorously establish the convergence of these algorithms. We also mention a recent work (Yin et al., 2018) which analyzes quantized networks and develops algorithms based on relaxing the discrete-weight constraints into continuous regularizers. Lastly, there are also analyses of quantized networks from a statistical-mechanical viewpoint (Baldassi et al., 2015; 2016a;b; 2017).

6. Conclusion and Outlook

In this paper, we have introduced the discrete-time optimal control viewpoint of deep learning. In particular, the PMP and the MSA form an alternative theoretical and algorithmic basis for deep learning that may apply to broader contexts. As an application of our framework, we considered the training of binary and ternary neural networks, in which we develop effective algorithms based on optimal control.

There are certainly many avenues of future work. An interesting mathematical question is the applicability of the PMP for discrete-weight neural networks, which does not satisfy the convexity assumptions in Theorem 1. It will be desirable to find the condition under which rigorous statements can be made. Another question is to establish the convergence of the algorithms presented.

References

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

Athans, M. and Falb, P. L. *Optimal control: an introduction* *to the theory and its applications*. Courier Corporation, 2013.

Bach, F. and Moulines, E. Non-strongly-convex smooth stochastic approximation with convergence rate O(1/n). In *Advances in Neural Information Processing Systems*, pp. 773–781, 2013.

Baldassi, C., Ingrosso, A., Lucibello, C., Saglietti, L., and Zecchina, R. Subdominant dense clusters allow for simple learning and high computational performance in neural networks with discrete synapses. *Physical review letters*, 115(12):128101, 2015.

Baldassi, C., Borgs, C., Chayes, J. T., Ingrosso, A., Lucibello, C., Saglietti, L., and Zecchina, R. Unreasonable effectiveness of learning neural networks: From accessible states and robust ensembles to basic algorithmic schemes. *Proceedings of the National Academy of Sciences*, 113(48):E7655–E7662, 2016a.

Baldassi, C., Gerace, F., Lucibello, C., Saglietti, L., and Zecchina, R. Learning may need only a few bits of synaptic precision. *Physical Review E*, 93(5):052313, 2016b.

Baldassi, C., Gerace, F., Kappen, H. J., Lucibello, C., Saglietti, L., Tartaglione, E., and Zecchina, R. On the role of synaptic stochasticity in training low-precision neural networks. *arXiv preprint arXiv:1710.09825*, 2017.

Baydin, A. G., Pearlmutter, B. A., Radul, A. A., and Siskind, J. M. Automatic differentiation in machine learning: a survey. *arXiv preprint arXiv:1502.05767*, 2015.

Bazaraa, M. S., Sherali, H. D., and Shetty, C. M. *Nonlinear programming: theory and algorithms*. John Wiley & Sons, 2013.

Bertsekas, D. P. *Dynamic programming and optimal control*, volume 1. Athena scientific Belmont, MA, 1995.

Bertsekas, D. P. *Nonlinear programming*. Athena scientific Belmont, 1999.

Boltyanskii, V. G., Gamkrelidze, R. V., and Pontryagin, L. S. The theory of optimal processes. I. The maximum principle. Technical report, TRW Space Technology Labs, Los Angeles, California, 1960.

Bryson, A. E. *Applied optimal control: optimization, estimation and control*. CRC Press, 1975.

Canon, M. D., Cullum Jr, C. D., and Polak, E. *Theory of optimal control and mathematical programming*. McGraw-Hill Book Company, 1970.

Chang, B., Meng, L., Haber, E., Ruthotto, L., Begert, D., and Holtham, E. Reversible architectures for arbitrarily deep residual neural networks. *arXiv preprint arXiv:1709.03698*, 2017.

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

Courbariaux, M., Bengio, Y., and David, J.-P. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in Neural Information Processing Systems*, pp. 3123–3131, 2015.

Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

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

Haber, E. and Ruthotto, L. Stable architectures for deep neural networks. *arXiv preprint arXiv:1705.03341*, 2017.

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

Han, S., Mao, H., and Dally, W. J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

Holtzman, J. Convexity and the maximum principle for discrete systems. *IEEE Transactions on Automatic Control*, 11(1):30–35, 1966a.

Holtzman, J. On the maximum priciple for nonlinear discrete-time systems. *IEEE Transactions on Automatic Control*, 11(2):273–274, 1966b.

Holtzman, J. M. and Halkin, H. Discretional convexity and the maximum principle for discrete systems. *SIAM Journal on Control*, 4(2):263–275, 1966.

Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., and Bengio, Y. Binarized neural networks. In *Advances in neural information processing systems*, pp. 4107–4115, 2016.

Hwang, C. and Fan, L. A discrete version of pontryagin's maximum principle. *Operations Research*, 15(1):139–146, 1967.

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

Kim, J., Hwang, K., and Sung, W. X1000 real-time phoneme recognition vlsi using feed-forward deep neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pp. 7510–7514. IEEE, 2014.

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

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. *Technical Report. University of Toronto*, 2009.

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

Krylov, I. A. and Chernousko, F. L. An algorithm for the method of successive approximations in optimal control problems. *USSR Computational Mathematics and Mathematical Physics*, 12(1):15–38, 1972.

Kuhn, H. W. and Tucker, A. W. Nonlinear programming. In *Traces and emergence of nonlinear programming*, pp. 247–258. Springer, 2014.

LeCun, Y. A theoretical framework for back-propagation. In *The Connectionist Models Summer School*, volume 1, pp. 21–28, 1988.

LeCun, Y. The MNIST database of handwritten digits. *http://yann.lecun.com/exdb/mnist/*, 1998.

Li, F., Zhang, B., and Liu, B. Ternary weight networks. *arXiv preprint arXiv:1605.04711*, 2016.

Li, H., De, S., Xu, Z., Studer, C., Samet, H., and Goldstein, T. Training quantized nets: A deeper understanding. In *Advances in Neural Information Processing Systems*, pp. 5813–5823, 2017a.

Li, Q., Tai, C., and E, W. Stochastic modified equations and adaptive stochastic gradient algorithms. In *International Conference on Machine Learning*, pp. 2101–2110, 2017b.

Li, Q., Chen, L., Tai, C., and E, W. Maximum principle based algorithms for deep learning. *Journal of Machine Learning Research*, 18:1–29, 2018.

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

Moulines, Eric and, F. R. Non-asymptotic analysis of stochastic approximation algorithms for machine learning. In *Advances in Neural Information Processing Systems*, pp. 451–459, 2011.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, pp. 5, 2011.

Ogata, K. *Discrete-time control systems*, volume 2. Prentice Hall Englewood Cliffs, NJ, 1995.

Pontryagin, L. S. *Mathematical theory of optimal processes*. CRC Press, 1987.

Rastegari, M., Ordonez, V., Redmon, J., and Farhadi, A. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pp. 525–542. Springer, 2016.

Robbins, H. and Monro, S. A stochastic approximation method. *The annals of mathematical statistics*, pp. 400–407, 1951.

Shalev-Shwartz, S. and Zhang, T. Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. *Mathematical Programming*, pp. 1–41, 2014.

Shamir, O. and Zhang, T. Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes. In *ICML (1)*, pp. 71–79, 2013.

Tang, W., Hua, G., and Wang, L. How to train a compact binary neural network with high accuracy? In *AAAI*, pp. 2625–2631, 2017.

Warga, J. Relaxed variational problems. *Journal of Mathematical Analysis and Applications*, 4(1):111–128, 1962.

Xiao, L. and Zhang, T. A proximal stochastic gradient method with progressive variance reduction. *SIAM Journal on Optimization*, 24(4):2057–2075, 2014.

Yin, P., Zhang, S., Lyu, J., Osher, S., Qi, Y., and Xin, J. Binaryrelax: A relaxation approach for training deep neural networks with quantized weights. *arXiv preprint arXiv:1801.06313*, 2018.

Zeiler, M. D. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

Zhu, C., Han, S., Mao, H., and Dally, W. J. Trained ternary quantization. *arXiv preprint arXiv:1612.01064*, 2016.