DIFFERENTIABLE PROGRAMMING A LA MOREAU

Vincent Roulet, Zaid Harchaoui

University of Washington, Department of Statistics, Seattle, USA

ABSTRACT

The notion of a Moreau envelope is central to the analysis of first-order optimization algorithms for machine learning and signal processing. We define a compositional calculus adapted to Moreau envelopes and show how to apply it to deep networks, and, more broadly, to learning systems equipped with automatic differentiation and implemented in the spirit of differentiable programming.

***Index Terms*—** Moreau envelope, differentiable programming, mathematical optimization, deep learning.

Introduction

We consider nonlinear dynamical optimization problems, characterized by a function f, that, given τ mappings ϕ_t, an initial point x_0, and a sequence of variables $w = (w_1, \dots, w_\tau)$, outputs

$$f(w, x_0) = x_\tau, \quad \text{s.t. } x_t = \phi_t(w_t, x_{t-1}) \text{ for } t = 1, ..., \tau. \tag{1}$$

Such structure typically arises in, e.g., deep learning problems, where ϕ_t are layers and $w_1, \dots, w_\tau$ are the weights of all layers, and in nonlinear discrete control problems, where ϕ_t are nonlinear dynamics and $w_1, \dots, w_\tau$ represent a sequence of controls. Given a dynamical structure (1), the optimization problem then consists in solving $\min_w h(f(w, x_0))$ for h a cost on the output of the dynamical system.

Standard gradient-based optimization methods can be used to solve such problems. Obtaining the gradient then amounts to applying the chain-rule, which is nowadays usually implemented using automatic differentiation for deep networks and other complex models in a differentiable programming framework [1, 2, 3].

As differentiable programming stands out as a computational framework tailored for training models using first-order optimization, one may ask how the notion of Moreau envelope could fit into it and expand its scope.

Indeed, the notion of Moreau envelope [4, 5, 6, 7, 8, 9] has arisen as a central notion in the analysis of first-order optimization algorithms for machine learning [10, 11, 12]. To blend Moreau envelopes into differentiable programming, one needs to define a calculus adapted to Moreau envelopes. We propose a framework to define such a calculus and show how to integrate it within differentiable programming. We show how previous proposals of smoother alternatives to gradient back-propagation fit into our framework. We present numerical results in deep learning and nonlinear control and deep learning. Detailed proofs and additional details can be found in the longer version [13].

Related work. The computational building blocks we consider are similar to the ones considered in variants of gradient back-propagation, which can be traced back to the now called target propagation [14, 15, 16, 17]. Target propagation can be described as using approximate inverses of layers when computing the gradient of a deep network [18, 19, 20]. The moving targets that minimize the overall objective are back-propagated via approximate layer inverses. The layer weights are then updated by minimizing the distance between the output of the layer and the given moving target. These algorithms were found to be effective in some settings and were, for the most part, motivated by empirical observations. Penalized formulations of the training problem have also been considered to decouple the optimization of the weights in a distributed way or using an ADMM approach [21, 22, 23]. Finally, our framework encompasses the proximal back-propagation algorithm of [24] which mixes the classical gradient back-propagation and a proximal step to update the weights of a deep forward network, to get a proximal-type gradient back-propagation.

1. DIFFERENTIABLE PROGRAM FOR THE MOREAU ENVELOPE

Key to the minimization of dynamical systems of the form (1) is the availability of first-order information via automatic differentiation in a differentiable programming framework.

Algorithm 1 Forward pass

Inputs: Function f parameterized by $(\phi_t)_{t=1}^\tau$ in (1), input x_0, parameters $(w_t)_{t=1}^\tau$
for $t = 1, \dots, \tau$ **do**
    Compute $x_t = \phi_t(w_t, x_{t-1})$
    Store x_{t-1}, w_t, ϕ_t
end for
Output: Final result x_τ
Stored: Intermediate comput. $(x_{t-1}, w_t, \phi_t)_{t=1}^\tau$

Algorithm 2 Backward pass

Inputs: Stored $(x_{t-1}, w_t, \phi_t)_{t=1}^\tau$, last state x_τ and objective h.
Initialize $\lambda_\tau = \mathrm{BP}(h)(x_\tau, 1)$
for $t = \tau, \dots, 1$ **do**
    Compute $\lambda_{t-1} = \mathrm{BP}(\phi_t(w_t, \cdot))(x_t, \lambda_t)$
    Compute $g_t = \mathrm{BP}(\phi_t(\cdot, x_t))(w_t, \lambda_t)$
end for
Output: Oracle directions $(g_t)_{t=1}^\tau$.

Formally, a differentiable program $\mathcal{P}$ implements the evaluation of a function f and enables the computation of any gradient-vector product on the evaluated point, i.e.,

$$\mathcal{P} : \begin{cases} \mathbb{R}^d & \to \mathbb{R}^m \times (\mathbb{R}^m \to \mathbb{R}^d) \\ x & \mapsto (f(x), \lambda \mapsto \nabla f(x)\lambda) \end{cases}.$$

We aim to define a similar procedure for computing the gradient of the Moreau envelope of a function that can be be decomposed as in (1).

Moreau envelope. The Moreau envelope defines an oracle through the minimization of the function rather than using a linear approximation of the objective [4, 25]. Formally, for a real function $f : \mathbb{R}^d \to \mathbb{R}$ and $\alpha > 0$ such that $x \mapsto \alpha f(x) + \|x\|_2^2/2$ is strongly convex[1], the Moreau envelope is defined as

$$\mathrm{env}(\alpha f)(x) = \inf_{y \in \mathbb{R}^d} \left\{ \alpha f(x + y) + \|y\|_2^2/2 \right\}, \qquad (2)$$

and its gradient, called hereafter a Moreau gradient, is defined as

$$\nabla \mathrm{env}(\alpha f)(x) = -\operatorname*{argmin}_{y \in \mathbb{R}^d} \left\{ \alpha f(x + y) + \|y\|_2^2/2 \right\}.$$

The parameter α acts as a step-size for the oracle and is part of the definition of the oracle. In particular, note that $\alpha \nabla \mathrm{env}(f)(x) \neq \nabla \mathrm{env}(\alpha f)(x)$. The Moreau envelope and its gradient yield smooth surrogates of the function and its gradient at a cost of solving (2).

Approximate envelope. In practice, one usually approximates the Moreau envelope using an optimization algorithm; see, e.g., [11]. Namely, for f differentiable and $\alpha > 0$ such that $x \mapsto \alpha f(x) + \|x\|_2^2/2$ is strongly convex, the Moreau gradient can be approximated as

$$\nabla \mathrm{env}(\alpha f)(x) = -\lim_{k \to +\infty} \mathcal{A}_k \left(\alpha f(x + \cdot) + \|\cdot\|_2^2/2 \right),$$

where $\mathcal{A}_k(h)$ is the k^{th} output of an algorithm $\mathcal{A}$, such as gradient descent, applied to minimize a function h.

[1] The Moreau envelope is guaranteed to exist under weaker assumptions; we simplify the conditions here for the exposition.

Moreau gradient for multivariate functions. For a multivariate function $f : \mathbb{R}^d \to \mathbb{R}^m$, a classical gradient encodes the linear form $\lambda \to \nabla(\lambda^\top f)(x)$. Similarly, we define the Moreau gradient of f as the nonlinear form $\lambda \to \nabla \mathrm{env}(\lambda^\top f)(x)$. Note that if f is linear, the Moreau gradient coincides with the usual gradient-vector product, i.e., $\nabla \mathrm{env}(\lambda^\top f)(x) = \nabla f(x)\lambda$.

Our goal in the following is to define a numerical program $\mathcal{M}$ which implements the Moreau gradient for a chain of computations (1), i.e.,

$$\mathcal{M} : \begin{cases} \mathbb{R}^d & \to \mathbb{R}^m \times (\mathbb{R}^m \to \mathbb{R}^d) \\ x & \mapsto (f(x), \lambda \mapsto \nabla \mathrm{env}(\lambda^\top f)(x)). \end{cases}$$

Back-propagation. To implement Moreau gradients for functions of the form (1), we consider taking advantage of the structure of the problem just as a gradient oracle does by using automatic differentiation.

Formally, the forward algorithm evaluates the function while keeping in memory the intermediate computations and the associated inputs as presented in Algo. 1. During the backward pass in Algo. 2, we consider procedures that either use gradient-vector products, back-propagate the Moreau gradients or use regularized inversions of the intermediate computations. Namely, for a function f evaluated at x and a direction λ, we consider back-propagation procedures BP of the form

$$\mathrm{GBP}(f)(x, \lambda) = \nabla f(x)\lambda$$
$$\mathrm{MBP}(f)(x, \lambda) = -\operatorname*{argmin}_{y \in \mathbb{R}^d} \lambda^\top f(x + y) + \|y\|_2^2/2$$
$$\mathrm{IBP}(f)(x, \lambda) = -\operatorname*{argmin}_{y \in \mathbb{R}^d} \|f(x+y) - f(x) + \lambda\|_2^2 + \gamma\|y\|_2^2,$$

where IBP is a regularized inversion akin to the virtual target propagation rule motivated in the next section. Different back-propagation procedures can be used at different stages. For example, GBP can be used on lines 2 and 4, while IBP can be used on line 5, which recovers the algorithm ProxProp of [24]. Once oracle directions $(g_t)_{t=1}^\tau$ are computed, the weights/controls are updated as

$$w_t^{\text{next}} = w_t - \alpha g_t, \text{ for } t \in \{1, \dots, \tau\}. \qquad (3)$$

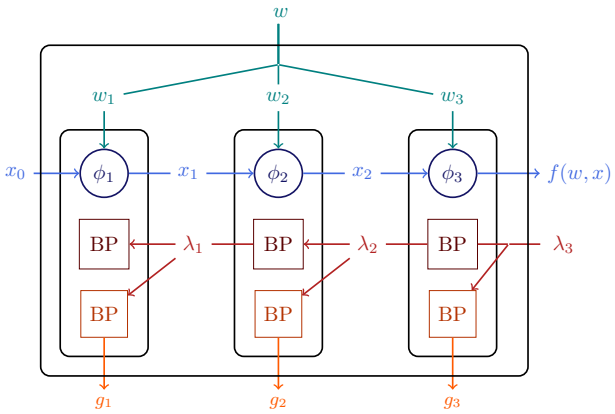

Fig. 1. Detailed computational process.

2. CHAIN RULES

Consider the computation of the Moreau gradient of a single composition of $g : \mathbb{R}^d \to \mathbb{R}^k$ and $f : \mathbb{R}^k \to \mathbb{R}$. Provided that f is convex, f, g are Lipschitz-continuous and smooth, and $\alpha > 0$ is small enough, denoting $(\alpha f)^*$ the convex conjugate of αf, we have

$$\begin{aligned}\nabla \operatorname{env}(\alpha f \circ g)(x) &= -\operatorname*{argmin}_{y\in\mathbb{R}^d} \left\{ \mu_*^\top g(x+y) + \frac{1}{2}\|y\|_2^2 \right\} \\ &= \nabla \operatorname{env}(\mu_*^\top g)(x), \qquad (4)\end{aligned}$$

$$\text{where } \mu_* = \operatorname*{argmax}_{\mu\in\mathbb{R}^k} -(\alpha f)^*(\mu) + \operatorname{env}(\mu^\top g)(x). \qquad (5)$$

Compare (4) to the classical gradient chain rule

$$\nabla(\alpha f \circ g)(x) = -\operatorname*{argmin}_{y\in\mathbb{R}^d} \left\{ \mu^\top \nabla g(x)^\top (x+y) + \frac{1}{2}\|y\|_2^2 \right\}$$

where $\mu = \nabla f(g(x))\lambda$. We retrieve the same structure, except that (i) for the Moreau gradient the dual direction μ_* is given by solving an optimization problem, (ii) the classical gradient minimizes a linearized approximation of the inner function along this direction, while for the Moreau gradient the inner function itself is used.

Denoting for x fixed, $e(\mu) = \operatorname{env}(\mu^\top g)(x)$, by approximating the solution of (5) by a proximal gradient step from 0, we get, for some $\beta > 0$,

$$\begin{aligned}\mu_* &\approx \operatorname*{argmax}_{\mu\in\mathbb{R}^k} \nabla e(0)^\top \mu - (\alpha f)^*(\mu) - \frac{1}{2\beta}\|\mu\|_2^2 \\ &= \beta \nabla \operatorname{env}((\alpha/\beta) f)(g(x)).\end{aligned}$$

More generally, for f multivariate, we consider approximating the Moreau gradient as

$$\begin{aligned}\nabla \operatorname{env}(\lambda^\top f \circ g)(x) &\approx \nabla \operatorname{env}(\hat{\mu}^\top g)(x) \\ \text{where} \quad \hat{\mu} &= \beta \nabla \operatorname{env}((\lambda/\beta)^\top f)(g(x)),\end{aligned}$$

which can be interpreted as a layer-wise proximal point method on the Lagrangian of the problem defining the Moreau gradient [13].

The computation of the Moreau gradient can also be formulated as solving

$$\min_{z\in\mathbb{R}^k} \alpha f(g(x) + z) + p(z)$$

for $p(z) = \min \left\{ \|y\|_2^2/2 : g(x+y) - g(x) = z \right\}$. An incremental proximal point method on the above problem, starting from $z = 0$, amounts then to computing

$$\begin{aligned}z_1 &= \operatorname*{argmin}_{z\in\mathbb{R}^k} \left\{ \alpha f(g(x) + z) + \frac{1}{2}\|z\|_2^2 \right\} \\ z_2 &= \operatorname*{argmin}_{z\in\mathbb{R}^k} \left\{ p(z) + \frac{1}{2}\|z - z_1\|_2^2 \right\} = g(x + \hat{y}) - g(x) \\ \hat{y} &= \operatorname*{argmin}_{y\in\mathbb{R}^d} \left\{ \|g(x+y) - g(x) - z_1\|_2^2 + \|y\|_2^2 \right\}.\end{aligned}$$

We consider then $\hat{y}$ as an approximation of the Moreau gradient whose computation can be summarized as

$$\begin{aligned}\nabla \operatorname{env}(\alpha f \circ g)(x) &\approx \hat{y} = \operatorname{IBP}(g)(x; \mu) \\ \text{where} \quad \mu &= \nabla \operatorname{env}(\alpha f)(g(x)).\end{aligned}$$

The computational complexity of using MBP or IBP is driven by the number of inner iterations used to approximate the minimizers. Namely, if we use K_{GD} steps of a gradient descent, the computational complexity of MBP or IBP is of the order of $\mathcal{T}(f) + K_{\mathrm{GD}}\mathcal{T}(\nabla f)$, where $\mathcal{T}(f)$ and $\mathcal{T}(\nabla f)$ are the complexities of computing the layer and its gradient respectively. In particular, by considering one step of a gradient descent, we retrieve the gradient-vector product, since we have

$$\begin{aligned}\nabla f(x)\lambda &= -\operatorname{GD}_1(\lambda^\top f(x + \cdot) + \|\cdot\|_2^2/2) \\ &= -\operatorname{GD}_1(\|f(x+\cdot) - f(x) - \lambda\|_2^2/2 + \|\cdot\|_2^2/2).\end{aligned}$$

The computations of IBP can also be replaced by an approximate inverse as done in target propagation.

3. NUMERICAL ILLUSTRATIONS

Nonlinear control of a swinging pendulum. We consider the control of a pendulum to make it swing-up after a finite time, which can be written as

$$\begin{aligned}\min_{w_1,\dots,w_\tau} \quad & h(x_\tau) \\ \text{s.t.} \quad & x_{t+1} = \phi_t(w_t, x_{t-1}) \text{ for } t \in \{1, \dots, \tau\},\end{aligned}$$

for x_0 fixed, where the formulations of ϕ_t, h are detailed in [13]. The horizon τ is usually large to ensure that the discretization scheme is accurate enough. As many compositions are involved, we are interested in the effects of using approximate Moreau gradients.

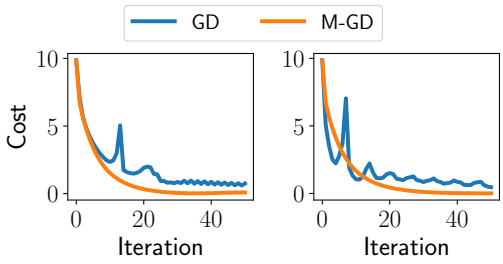

Fig. 2. Gradient descent (GD) vs Moreau gradient descent (M-GD) on the control of a pendulum. Left: horizon $\tau = 50$, Right: horizon $\tau = 100$

We compare a gradient descent to our Moreau gradient algorithm using MBP in lines 2, 4, 5 on the control of a pendulum in Fig. 2 for various horizons τ. More details on experimental settings can be found in [13].

Supervised classification with deep networks. For supervised classification with deep networks, we consider a mini-batch stochastic counterpart to the proposed algorithm. Namely, we compute approximate Moreau gradients for mini-batches written as

$$h_m(f_m(w, x_0)) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(y_i, \psi(w, x_{0,i})), \quad (6)$$

where $\mathcal{L}$ is a smooth loss function, $x_0 = (x_{0,1}; \ldots; x_{0,m})$ is a mini-batch of m samples, $h_m(\hat{y}) = \sum_{i=1}^{m} \mathcal{L}(y_i, \hat{y}_i)/m$ for $\hat{y} = (\hat{y}_1; \ldots; \hat{y}_m)$. Here f is the concatenation of a chain of computations applied to the mini-batch of inputs, i.e., $f(w, x_0) = (\psi(w, x_{0,1}); \ldots; \psi(w, x_{0,m}))$.

In Fig. 3, we compare plain mini-batch stochastic gradient descent against a mini-batch approximate stochastic Moreau gradient descent, i.e., updates (3) on mini-batches (6) using MBP in lines 2, 4, 5, to train a deep network on the image classification dataset CIFAR10 [26]. We consider a fully connected multi-layer neural network with hidden layer sizes $(4000, 1000, 4000)$ with a squared loss and a convolutional neural network architecture as presented by [24] with a logistic loss. The plots present the minimum of the loss or the test error obtained so far, i.e., on the y-axis we plot $y_k = \min_{i=0,\ldots,k} h_n(f_n(w^{(i)}, x_0))$, where n is the total number of samples in the train and $w^{(i)}$ the current set of parameters.

We observe that the mini-batch stochastic counterpart of the proposed algorithm compares favorably with a stochastic gradient descent in both cases.

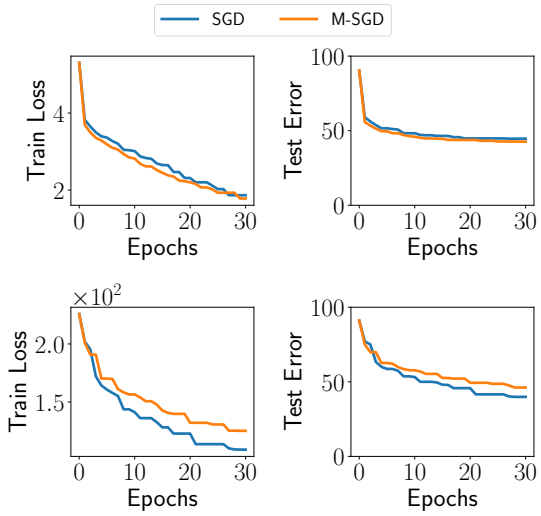

Fig. 3. Stochastic Gradient Descent (SGD) versus Stochastic Moreau Gradient Descent (M-SGD) on deep learning problems. Top: MLP, Bottom: Convnet

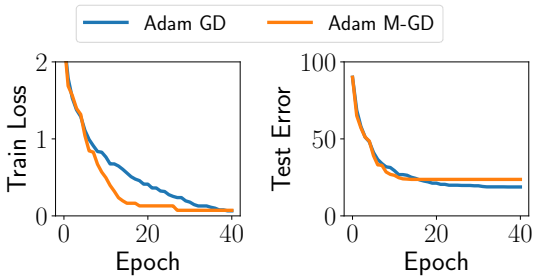

Fig. 4. Adam with Gradient oracle (Adam GD) versus Adam with Moreau gradients (Adam M-GD) on CIFAR with AllCNN architecture.

Stochastic algorithms with momentum. Moreau gradients define first-order oracles that can be incorporated in popular algorithms for stochastic training with momentum such as Adam [27]. We illustrate this by considering the Proximal BackPropagation algorithm of [24] which can be seen as a particular implementation of a Moreau gradient and apply it to the image classification dataset CIFAR10 using the AllCNN-C architecture [28] with a logistic loss[2]. In Fig. 4, we observe that an approach using Moreau gradients can optimize faster on the training loss, while an approach with classical gradients can generalize better in this experiment.

Acknowledgments. This work was supported by NSF CCF-1740551, NSF DMS-1839371, CIFAR program "Learning in Machines and Brains", and faculty research awards.

[2] A similar experiment was done by [24] on a smaller architecture.

4. REFERENCES

[1] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer, "Automatic differentiation in PyTorch," 2017.

[2] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015.

[3] Jérôme Bolte and Edouard Pauwels, "A mathematical model for automatic differentiation in machine learning," in *Advances in Neural Information Processing Systems*, 2020, vol. 33.

[4] Jean Jacques Moreau, "Fonctions convexes duales et points proximaux dans un espace hilbertien," *Comptes Rendus de l'Académie des Sciences*, vol. 255, 1962.

[5] Kösaku Yosida, *Functional analysis*, Springer Science & Business Media, 2012.

[6] Bernard Martinet, "Régularisation d'inéquations variationnelles par approximations successives. rev. française informat," *Recherche Opérationnelle*, vol. 4, pp. 154–158, 1970.

[7] Bernard Martinet, "Détermination approchée d'un point fixe d'une application pseudo-contractante," *CR Acad. Sci. Paris*, vol. 274, no. 2, pp. 163–165, 1972.

[8] R Tyrrell Rockafellar, "Monotone operators and the proximal point algorithm," *SIAM journal on control and optimization*, vol. 14, no. 5, pp. 877–898, 1976.

[9] Hédy Attouch, "Convergence de fonctions convexes, des sous-différentiels et semi-groupes associés," *CR Acad. Sci. Paris*, vol. 284, no. 539-542, pp. 13, 1977.

[10] John C. Duchi and Feng. Ruan, "Stochastic methods for composite and weakly convex optimization problems," *SIAM Journal on Optimization*, vol. 28, no. 4, pp. 3229–3259, 2018.

[11] Hongzhou Lin, Julien Mairal, and Zaid Harchaoui, "Catalyst acceleration for first-order convex optimization: from theory to practice," *Journal of Machine Learning Research*, vol. 18, no. 212, pp. 1–54, 2018.

[12] Dmitriy Drusvyatskiy and Courtney Paquette, "Efficiency of minimizing compositions of convex functions and smooth maps," *Mathematical Programming*, vol. 178, no. 1-2, pp. 503–558, 2019.

[13] Vincent Roulet and Zaid Harchaoui, "Differentiable programming à la Moreau," *arXiv preprint arXiv:2012.15458*, 2020.

[14] Yann Lecun, "A theoretical framework for back-propagation," in *1988 Connectionist Models Summer School, CMU, Pittsburg, PA*, 1988.

[15] Yann Le Cun, Conrad C Galland, and Geoffrey E Hinton, "GEMINI: gradient estimation through matrix inversion after noise injection," in *Advances in neural information processing systems*, 1989, pp. 141–148.

[16] Richard Rohwer, "The "moving targets" training algorithm," in *Advances in neural information processing systems*, 1990, pp. 558–565.

[17] Piotr Mirowski and Yann LeCun, "Dynamic factor graphs for time series modeling," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2009, pp. 128–143.

[18] Dong-Hyun Lee, Saizheng Zhang, Asja Fischer, and Yoshua Bengio, "Difference target propagation," in *Joint european conference on machine learning and knowledge discovery in databases*. Springer, 2015, pp. 498–515.

[19] Alexander Meulemans, Francesco Carzaniga, Johan Suykens, João Sacramento, and Benjamin F. Grewe, "A theoretical framework for target propagation," in *Advances in Neural Information Processing Systems 33*, 2020.

[20] Nasir Ahmad, Marcel A van Gerven, and Luca Ambrogioni, "Gait-prop: A biologically plausible learning rule derived from backpropagation of error," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[21] Miguel Carreira-Perpinan and Weiran Wang, "Distributed optimization of deeply nested systems," in *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics*, 2014.

[22] Gavin Taylor, Ryan Burmeister, Zheng Xu, Bharat Singh, Ankit Patel, and Tom Goldstein, "Training neural networks without gradients: A scalable admm approach," in *Proceedings of the 33rd International Conference on Machine Learning*, 2016.

[23] Akhilesh Gotmare, Valentin Thomas, Johanni Brea, and Martin Jaggi, "Decoupling backpropagation using constrained optimization methods," in *Credit Assignment in Deep Learning and Reinforcement Learning Workshop (ICML 2018 ECA)*, 2018.

[24] Thomas Frerix, Thomas Möllenhoff, Michael Moeller, and Daniel Cremers, "Proximal backpropagation," in *International Conference on Learning Representations*, 2018.

[25] Heinz H Bauschke and Patrick L Combettes, *Convex analysis and monotone operator theory in Hilbert spaces*, vol. 408, Springer, 2nd edition, 2017.

[26] Alex Krizhevsky, "Learning multiple layers of features from tiny images," Tech. Rep., University of Toronto, 2009.

[27] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations*, 2015.

[28] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller, "Striving for simplicity: The all convolutional net," in *International Conference on Learning Representations*, 2015.