An Inertial Newton Algorithm for Deep Learning

Camille Castera, Jérôme Bolte, Cédric Févotte, Edouard Pauwels

An Inertial Newton Algorithm for Deep Learning

Camille Castera
IRIT, Université de Toulouse,
CNRS, Toulouse, France
camille.castera@irit.fr

Jérôme Bolte*
Toulouse School of Economics
Université Toulouse 1 Capitole, France
jerome.bolte@ut-capitole.fr

Cédric Févotte*
IRIT, Université de Toulouse,
CNRS, Toulouse, France
cedric.fevotte@irit.fr

Edouard Pauwels*
IRIT, Université de Toulouse, CNRS,
DEEL, IRT Saint Exupery, Toulouse, France
edouard.pauwels@irit.fr

Abstract

We devise a learning algorithm for possibly nonsmooth deep neural networks featuring inertia and Newtonian directional intelligence only by means of a backpropagation oracle. Our algorithm, called INDIAN, has an appealing mechanical interpretation, making the role of its two hyperparameters transparent. An elementary phase space lifting allows both for its implementation and its theoretical study under very general assumptions. We handle in particular a stochastic version of our method (which encompasses usual mini-batch approaches) for nonsmooth activation functions (such as ReLU). Our algorithm shows high efficiency and reaches state of the art on image classification problems.

1 Introduction

Can we devise a learning algorithm for general/nonsmooth deep neural networks (DNNs) featuring inertia and Newtonian directional intelligence only by means of a backpropagation oracle?

In an optimization jargon: can we use second order ideas in time and space for *nonsmooth nonconvex* optimization by uniquely using a subgradient oracle?

Before providing answers to this daring question, let us have a glimpse at some of the fundamental optimization algorithms for training deep networks.

The backpropagation algorithm is, to this day, the fundamental block for training DNNs. It is an instance of the Stochastic Gradient Descent algorithm (SGD, (34)) and is as such powerful, flexible, capable of handling huge size problems, noise, and further comes with theoretical guarantees of many kinds. We refer to (13; 31) in a convex machine learning context and (14) for a recent account highlighting the importance of deep learning (DL) applications and their challenges. In the nonconvex setting, recent works of (2; 20) follow the "*Ordinary Differential Equations (ODE) approach*" introduced in (29), and further developed in (8; 26; 9; 12). SGD is however a raw first order algorithm requiring manual tuning and whose convergence rate can sometimes be low on some DL instances.

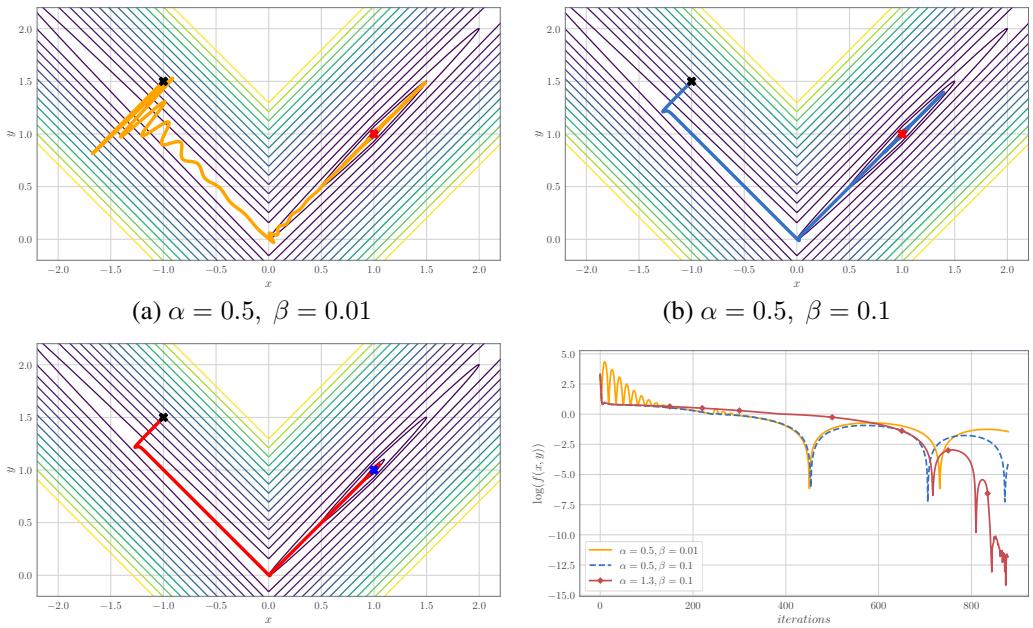
In the recent literature two improvement lines have been explored:

− use local geometry of empirical losses to improve over steepest descent directions,
− use past steps history to design clever steps in the present.

The first approach is akin to quasi-Newton methods while the second revolves around Polyak's inertial method (33). The latter is inspired by the following appealing mechanical thought-experiment.

*Last three authors are listed in alphabetical order.

Preprint. Under review.



(a) $\alpha = 0.5,\ \beta = 0.01$ (b) $\alpha = 0.5,\ \beta = 0.1$

(c) $\alpha = 1.3,\ \beta = 0.1$ (d) Objective function $\mathcal{J}$ (in log-scale)

Figure 1: Illustration of INDIAN applied to the nonsmooth function $f(x, y) = 100(y - |x|)^2 + |1 - x|$. Subplots (a-c) represent the trajectories of the parameters x and y in $\mathbb{R}^2$ for three choices of hyperparameters α and β, see Equation (1) for an intuitive explanation. Subplot (d) displays the values of the objective function $f(x, y)$ for the three settings considered.

Consider a heavy ball evolving on the graph of the loss (the loss "landscape"), subject to gravity and stabilized by some friction effects. Friction generates energy dissipation, so that the particle will eventually reach a steady state which one hopes to be a local minimum. These two approaches are already present in the DL literature: among the most popular algorithms for training DNNs, ADAGRAD (22) features local geometrical aspects while ADAM (24) combines inertial ideas with stepsizes similar to the ones of ADAGRAD. Stochastic Newton and quasi-Newton algorithms have been considered by (30; 15; 16) and recently reported to perform efficiently on several problems (10; 39). The work of (38) demonstrates that carefully tuned SGD and heavy-ball algorithms are competitive with concurrent methods.

But deviating from the simplicity of SGD also comes with major challenges because of the size and the severe absence of regularity in DL (differential regularity is generally absent, but even weaker regularity as semi-convexity or Clarke regularity are not available). All sorts of practical and theoretical hardships are met: defining/computing Hessian is delicate, inverting them is unthinkable at this day, first and second order Taylor approximation are useless, and one has to deal with shocks which are inherent to inertial approaches in a nonsmooth context ("corners/walls" indeed generate velocity discontinuity). This makes the study of ADAGRAD and ADAM in full generality quite difficult. Some recent progresses are reported in (7).

Our approach also blends inertial ideas with Newton's method. It is inspired by the following dynamical system introduced in (3):

$$\underbrace{\ddot{\theta}(t)}_{\text{Inertial term}} + \underbrace{\alpha\,\dot{\theta}(t)}_{\text{Friction term}} + \underbrace{\beta\,\nabla^2\mathcal{J}(\theta(t))\dot{\theta}(t)}_{\text{Newtonian effects}} + \underbrace{\nabla\mathcal{J}(\theta(t))}_{\text{Gravity effect}} = 0,\ t \geq 0, \tag{1}$$

where t is the time parameter which acts as a continuous epoch counter, $\mathcal{J}$ is a given loss function (usually empirical loss in DL applications) while $\nabla\mathcal{J}$ and $\nabla^2\mathcal{J}$ denote respectively the gradient of $\mathcal{J}$ and its Hessian.

To adapt this dynamics to DL and overcome the computational difficulties generated by second order objects occuring in (1), we combine a phase space lifting method with a small step discretization process typical to the stochastic approach. An important difficulty is met when dealing with networks

having nonsmooth activation functions as ReLU (23). Indeed subsampled versions of the algorithm, which are absolutely necessary in practice, must be treated with great care since the sum of subdifferentials no longer coincides with the subdifferential of the sum. We address this delicate issue by using new notions of steady states and by providing adequate calculus rules.

The resulting algorithm, called INDIAN, shows great efficiency in practice. For the same computational price than other tested methods (including ADAM and ADAGRAD), INDIAN avoids parasite oscillations, often achieves better training accuracy and shows robustness to hyperparameter setting. A first illustration of the behaviour of the induced dynamics is given in Figure 1 for a simple nonsmooth and nonconvex function in $\mathbb{R}^2$.

Our theoretical results are also strong and simple. Using Lyapunov analysis from (3) we combine tame nonsmooth results *à la* Sard and the differential inclusion approximation method (9) to characterize the asymptotics of our algorithm similarly as in (20; 2). This provides a strong theoretical ground to our study since we can prove that our method converges to a connected component of the set of steady states even in the ReLU case where the optimization problem is nonsmooth. The algorithm is described in details in Section 2 and its convergence proof is given in Section 3. Section 4 describes experimental results on synthetic and real datasets. Details of proofs and additional experiments can be found in the appendices, `python` notebooks that allows to reproduce our experiments are available here `https://github.com/camcastera/Code-for-an-Inertial-Newton-algorithm-for-DL/`.

2 INDIAN: an Inertial Newton algorithm for Deep Neural Networks

2.1 Neural networks with Lipschitz continuous prediction function and losses

We consider DNNs of a very general type represented by a locally Lipschitz function $f : (x, \theta) \in \mathbb{R}^M \times \mathbb{R}^P \mapsto y \in \mathbb{R}^D$ (e.g., a composition of feed-forward, convolutional, recurrent networks with ReLU, sigmoid, or tanh activation functions). The variable $\theta \in \mathbb{R}^P$ is the parameter of the model (P can be very large), while $x \in \mathbb{R}^M$ and $y \in \mathbb{R}^D$ represent input and output data. For instance, the vector x may embody an image while y is a label explaining its content. Consider further a dataset of N samples $(x_n, y_n)_{n=1,\dots,N}$. Training amounts to find a value of the parameter θ such that, for each input data x_n of the dataset, the output $f(x_n, \theta)$ of the model predicts the real value y_n with good accuracy.

To do so, we follow the traditional approach of minimizing an empirical risk loss function

$$\mathbb{R}^P \ni \theta \mapsto \mathcal{J}(\theta) = \sum_{n=1}^{N} l(f(x_n, \theta), y_n), \tag{2}$$

where $l : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ is a locally Lipschitz continuous dissimilarity measure.

2.2 Neural networks and tameness in a nutshell

Tameness refers to an ubiquitous geometrical property of losses/constraints encompassing most finite dimensional optimization problems met in practice. Prominent classes of tame objects are piecewise linear objects (with finitely many pieces), or semi-algebraic objects but the notion is much more general as we intent to convey below.

Sets or functions are called *tame* when they can be described by a finite number of basic formulas/inequalities/Boolean operations involving standard functions such as polynomial, exponential, or max functions. We refer to (5) for illustrations, recipes and examples within a general optimization setting or (20) for illustrations in the context of neural networks. One is referred to (21; 19; 36) for foundational material. To apprehend the strength behind tameness it is convenient to remember that it models nonsmoothness by confining the study to sets/functions which are union of smooth pieces in an inbuilt manner. This is the so-called *stratification* property of tame sets/functions. It was this property which motivated the vocable of *tame topology*, "*la topologie modérée*" wished for by Grothendieck, see (21).

All finite dimensional deep learning optimization models we are aware of yield tame losses $\mathcal{J}$. To understand this assertion and convey the universality of tameness assumptions, let us provide concrete

examples (see also (20)). If one assumes that the neural networks under consideration are built from the following traditional components:

− the function f has an arbitrary number of layers of arbitrary dimensions,
− the activation functions are among classical ones: ReLU, sigmoid, SQNL, RReLU, tanh, APL, soft plus, soft clipping, and many others,
− the dissimilarity function $l(x, y)$ is a standard loss such as ℓ_p norms, logistic loss or cross-entropy,

then one easily shows, by elementary quantifier elimination arguments, that the corresponding loss is tame.[2]

2.3 INDIAN and its generalized stochastic form INDIAN$_g$

Given a locally Lipschitz continuous function F between finite dimensional spaces, we define for each θ in the domain of F, its Clarke's subdifferential $\partial F(\theta)$ as the closed convex envelope of the limits of neighboring gradients (see (18) for a formal definition). This makes this set compact, convex and nonempty.

In order to compute the subdifferential of $\mathcal{J}$ and to cope with large datasets, $\mathcal{J}$ can be approximated by mini-batches, reducing the memory footprint and computational cost of evaluation. For any $\mathcal{B} \subset \{1, \dots, N\}$, set

$$\mathcal{J}_{\mathcal{B}} \colon \theta \mapsto \sum_{n \in \mathcal{B}} l(f(x_n, \theta), y_n),$$

and

$$D\mathcal{J}_{\mathcal{B}} = \sum_{n \in \mathcal{B}} \partial \left[l(f(x_n, \cdot), y_n) \right], \quad D\mathcal{J} = \sum_{n=1}^{N} \partial \left[(f(x_n, \cdot), y_n) \right]. \tag{3}$$

Observe that, for each $\mathcal{B}$, we have $D\mathcal{J}_{\mathcal{B}} \supset \partial \mathcal{J}_{\mathcal{B}}$ and that $\mathcal{J}_{\mathcal{B}}$ is differentiable almost everywhere with $D\mathcal{J}_{\mathcal{B}} = \partial \mathcal{J}_{\mathcal{B}} = \{\nabla \mathcal{J}_{\mathcal{B}}\}$, see (18). When $\mathcal{J}$ is tame the equalities hold on the complement of a finite union of manifolds of dimension strictly lower than P, see (19). For convenience, a point satisfying $D\mathcal{J}(\theta) \ni 0$ will be called *D-critical*. This vocable is motivated by favourable properties whose statements and proofs are postponed in an appendix: a good calculus along curves (see Lemmas 3 and 4) and the existence of a tame Sard's theorem (see Lemma 5). To our knowledge, this notion of steady state has not previously been used in the literature. The definition of $D\mathcal{J}$ stems from the unavoidable absence of a sum rule for Clarke subdifferentials (think about $0 = |\cdot| - |\cdot|$). This lack of linearity makes the traditional "*subgradient plus centered noise*" approach unfit to the study of mini-batch subsampling methods in DL.

We consider a sequence $(\mathcal{B}_k)_{k \in \mathbb{N}}$ of nonempty subsets of $\{1, \dots, N\}$ chosen independently, uniformly at random with replacement and a sequence of positive stepsizes $(\gamma_k)_{k \in \mathbb{N}}$. Starting from initial values $\theta_0 \in \mathbb{R}^P$ and $\psi_0 \in \mathbb{R}^P$, we consider the following iterative process:

$$\text{(INDIAN)} \quad \begin{cases} v_k & \in D\mathcal{J}_{\mathcal{B}_k}(\theta_k) \\ \theta_{k+1} & = \theta_k + \gamma_k \left((\frac{1}{\beta} - \alpha)\theta_k - \frac{1}{\beta}\psi_k - \beta v_k \right) \\ \psi_{k+1} & = \psi_k + \gamma_k \left((\frac{1}{\beta} - \alpha)\theta_k - \frac{1}{\beta}\psi_k \right) \end{cases} \tag{4}$$

where $\alpha > 0$ and $\beta > 0$ are parameters of the algorithm. Empirical experiments suggest that $\alpha = 0.5$ and $\beta = 0.1$ is a good choice of damping parameters for training DNNs. See the last section and the appendix for further details and explanation.

In practice $v_k \in D\mathcal{J}_{\mathcal{B}_k}(\theta_k)$ is usually computed with a backpropagation algorithm, as in the seminal work of (35). The whole process is a stochastic approximation of the deterministic dynamics obtained by choosing $\mathcal{B}_k \equiv \{1, \dots, N\}$, that is $\mathcal{J}_{\mathcal{B}_k} \equiv \mathcal{J}$ (batch version). This can be seen by observing that the vectors v_k above may be written $\tilde{v}_k + \eta_k$, where $\tilde{v}_k \in D\mathcal{J}(\theta_k)$ and η_k compensates for the missing subgradients and can be seen as a zero-mean noise.

[2] From now on we impose an o-minimal structure, so that an object is said to be tame if it belongs to this structure.

Hence, INDIAN admits the following general abstract stochastic formulation:

$$(\text{INDIAN}_g) \quad \begin{cases} w_k & \in D\mathcal{J}(\mu_k) \\ \mu_{k+1} & = \mu_k + \gamma_k \left((\frac{1}{\beta} - \alpha)\mu_k - \frac{1}{\beta}\phi_k - \beta w_k + \xi_k \right) \\ \phi_{k+1} & = \phi_k + \gamma_k \left((\frac{1}{\beta} - \alpha)\mu_k - \frac{1}{\beta}\phi_k \right) \end{cases} \tag{5}$$

where $(\xi_k)_{k\in\mathbb{N}}$ is a martingale difference noise sequence adapted to the filtration induced by (random) iterates up to k, and θ_0, ϕ_0 are arbitrary initial conditions.

2.4 INDIAN and INDIAN$_g$ converge

In order to establish convergence, we start with the following standing assumption.

Assumption 1 (Vanishing stepsizes)**.** *The stepsize sequence γ_k is positive, diverges $\sum \gamma_k = +\infty$ and satisfies $\gamma_k = o\left(\frac{1}{\log k}\right)$, that is $\limsup_{k\to+\infty} |\gamma_k \log k| = 0$.*

Typical admissible choices are $\gamma_k = C(k+1)^{-a}$ with $a \in (0, 1]$, $C > 0$. The main theoretical result of this paper follows.

Theorem 1 (INDIAN converges to the set of D-critical points of $\mathcal{J}$)**.** *Assume that $\mathcal{J}$ is locally Lipschitz continuous, tame and that the stepsizes satisfy Assumption 1. Set an initial condition (θ_0, ψ_0) and assume that there exists $M > 0$ such that $\sup_k \|(\theta_k, \psi_k)\| \leq M$ almost surely.*
Then, almost surely, any accumulation point $\bar{\theta}$ of a realization of the sequence $(\theta_k)_{k\in\mathbb{N}}$ satisfies $D\mathcal{J}(\bar{\theta}) \ni 0$. In addition $(\mathcal{J}(\theta_k))_{k\in\mathbb{N}}$ converges.

Remark 1. (a) [Stepsizes]: Assumption 1 offers much more flexibility than the usual $0(1/\sqrt{k})$ assumption commonly used for SGD. We leverage boundedness assumption, local Lipschitz continuity and finite sum structure of $\mathcal{J}$, so that the noise is actually uniformly bounded, hence sub-gaussian, allowing for much larger stepsizes than in the more common bounded second moment setting. See (9, Remark 1.5) and (8) for more details.
(b) [Convergence of INDIAN$_g$]: Apart from the uniform boundedness of the noise, we do not use the specific structure of DL losses. Thus our result actually holds for general locally Lipschitz continuous tame functions with finite sum structure and for the general stochastic algorithm INDIAN$_g$ under uniformly bounded martingale increment noise. Other variants could be considered depending on the assumptions on the noise, see (9).
(c) [Convergence to critical points]: Observe that when $\mathcal{J}$ is differentiable, limit points are simply critical points.
(d) [Local minima]: Let us mention that for general $\mathcal{J}$, being D-critical is a necessary condition for being a local minima.

3 Convergence proof

3.1 Underlying differential inclusion

To study algorithms with vanishing stepsizes such as (4), a powerful approach is to view them as the time discretization of a differential equation/inclusion, see (20) in the context of DL. Our algorithms can be seen as discretizations of the following dynamical system akin to the one considered by (3):

$$(\text{DIN}) \quad \begin{cases} \dot{\theta}(t) + \beta D\mathcal{J}(\theta(t)) + (\alpha - \frac{1}{\beta})\theta(t) + \frac{1}{\beta}\psi(t) \ni 0, \\ \dot{\psi}(t) + (\alpha - \frac{1}{\beta})\theta(t) + \frac{1}{\beta}\psi(t) \ni 0, \text{ a.e. on } (0, +\infty). \end{cases} \tag{6}$$

Given any initial condition (θ_0, ψ_0), general results ensure the existence of an absolutely continuous solution $t \to (\theta(t), \psi(t))$ to this system satisfying $\theta(0) = \theta_0$ and $\psi(0) = \psi_0$ (6). Recall that absolute continuity amounts to the fact that θ, ψ are differentiable almost everywhere with

$$\theta(t) = \int_0^t \dot{\theta}(s)ds, \ \psi(t) = \int_0^t \dot{\psi}(s)ds, \text{ for all } t \geq 0.$$

As explained in (3), when $\mathcal{J}$ is twice differentiable, (DIN) is equivalent to a second order system (avoiding the explicit use of the second order derivatives of $\mathcal{J}$):

$$\ddot{\theta}(t) + \alpha\dot{\theta}(t) + \beta\nabla^2\mathcal{J}(\theta(t))\dot{\theta}(t) + \nabla\mathcal{J}(\theta(t)) = 0. \tag{7}$$

The steady points of DIN are given by

$$\mathcal{S} = \left\{(\theta, \psi) \in \mathbb{R}^P \times \mathbb{R}^P : 0 \in D\mathcal{J}(\theta), \psi = (1 - \alpha\beta)\theta)\right\}. \tag{8}$$

Observe that the first coordinates of these points are D-critical for $\mathcal{J}$ and that conversely any D-critical point of $\mathcal{J}$ corresponds to a unique rest point in $\mathcal{S}$.

3.2 Proof of convergence for INDIANg

Definition 1 (Lyapunov function)**.** *Let A be a subset of $\mathbb{R}^P \times \mathbb{R}^P$, we say that $E : \mathbb{R}^P \to \mathbb{R}$ is a Lyapunov function for the set A and the dynamics* (6) *if*

(i) for any solution (θ, ψ) of (DIN) *with initial condition (θ_0, ψ_0), we have: $E(\theta(t), \psi(t)) \leq E(\theta_0, \psi_0)$ a.e. on $\mathbb{R}$.*

(ii) for any solution (θ, ψ) of (DIN) *with initial condition $(\theta_0, \psi_0) \notin A$, we have: $E(\theta(t), \psi(t)) < E(\theta_0, \psi_0)$ a.e. on $\mathbb{R}$.*

In practice, to establish that a functional is Lyapunov, one can simply use differentiation through chain rule results, see appendices (with in particular Lemma 3 first stated by (20)[Theorem 5.8] and based on the projection formula in (11)).

To build a Lyapunov function for the dynamics (6) and the set $\mathcal{S}$, consider the two following energy-like functions:

$$\begin{cases} E_{\min}(\theta(t), \psi(t)) & = (1 - \sqrt{\alpha\beta})^2 \mathcal{J}(\theta(t)) + \frac{1}{2}\left|(\alpha - \frac{1}{\beta})\theta(t) + \frac{1}{\beta}\psi(t)\right|^2 \\ E_{\max}(\theta(t), \psi(t)) & = (1 + \sqrt{\alpha\beta})^2 \mathcal{J}(\theta(t)) + \frac{1}{2}\left|(\alpha - \frac{1}{\beta})\theta(t) + \frac{1}{\beta}\psi(t)\right|^2. \end{cases} \tag{9}$$

Then the following lemma applies.

Lemma 1 (Differentiation along DIN trajectories)**.** *Let (θ, ψ) be a solution of* (6) *with initial condition (θ_0, ψ_0). For almost all $t > 0$, θ and ψ are differentiable at t,* (6) *holds, $\frac{\dot\theta(t) - \dot\psi(t)}{\beta} \in D\mathcal{J}(\theta(t))$ and*

$$\frac{\mathrm{d}E_{\min}}{\mathrm{d}t}(\theta(t), \psi(t)) = -\left|\sqrt{\alpha}\dot\theta(t) - \frac{1}{\sqrt{\beta}}\left(\dot\psi(t) - \dot\theta(t)\right)\right|^2$$
$$\frac{\mathrm{d}E_{\max}}{\mathrm{d}t}(\theta(t), \psi(t)) = -\left|\sqrt{\alpha}\dot\theta(t) + \frac{1}{\sqrt{\beta}}\left(\dot\psi(t) - \dot\theta(t)\right)\right|^2$$

Define $E = E_{\min} + E_{\max}$ and recall that $\mathcal{S} = \left\{(\theta, \psi) \in \mathbb{R}^P \times \mathbb{R}^P : 0 \in DJ(\theta), \psi = (1 - \alpha\beta)\theta)\right\}$. By a direct integration argument, we obtain the following.

Lemma 2 (E is Lyapunov function for (INDIAN) with respect to $\mathcal{S}$)**.** *For any $(\theta_0, \psi_0) \notin \mathcal{S}$ and any solution (θ, ψ) with initial condition (θ_0, ψ_0),*

$$E(\theta(t), \psi(t)) < E(\theta_0, \psi_0), \text{ for almost all } t > 0. \tag{10}$$

Proof of Theorem 1. Lemmas 1 and 2 entail that E is a Lyapunov function for the set $\mathcal{S}$ and the dynamics (6). Set $C = \{\theta \in \mathbb{R}^P : (\theta, \psi) \in \mathcal{S}\}$ which is a actually the set of D-critical points of $\mathcal{J}$. Using Lemma 5 in appendix A, $\mathcal{J}(C)$ is finite. Moreover, since $E(\theta, \psi) = (1 + 2\alpha\beta)\mathcal{J}(\theta)$ for all $(\theta, \psi) \in \mathcal{S}$, E takes a finite number of values on $\mathcal{S}$, and in particular, $E(\mathcal{S})$ has empty interior.

Denote by $\mathcal{L}$ the set of accumulation points of a realizations of the sequences $((\theta_k, \psi_k))_{k\in\mathbb{N}}$ produced by (4) starting at (θ_0, ψ_0) and $\mathcal{L}_1$ its projection on $\mathbb{R}^P \times \{0\}$. We have the 3 following properties:
$-$ By assumption, we have $\|(\theta_k, \psi_k)\| \leq M$ almost surely, for all $k \in \mathbb{N}$.
$-$ By local Lipschitz continuity $\partial\mathcal{J}_{\mathcal{B}}(\theta)$ is uniformly bounded for $\|\theta\| \leq M$ and any $\mathcal{B} \subset \{1, \ldots, N\}$, hence the centered noise $(\xi_k)_{k\in\mathbb{N}}$ is a uniformly bounded martingale difference sequence.
$-$ By Assumption 1, the sequence $(\gamma_k)_{k\in\mathbb{N}}$ are chosen such that $\gamma_k = o(\frac{1}{\log k})$ (see Remark 1) (a)).

Combining Theorem 3.6, Remark 1.5 and Proposition 3.27 of (9) to obtain that $\mathcal{L} \subset \mathcal{S}$ and $E(\mathcal{L})$ is a singleton. Hence $\mathcal{J}(\mathcal{L}_1)$ is also a singleton and the theorem follows. □

4 Experiments

In this section we provide some explanations for the 2D examples displayed in the introduction; these are meant to illustrate the versatility of INDIAN and the effect of hyperparameters. We then compare the performance of INDIAN with those of other algorithms on a DNN training for image recognition. Experiments were conducted with `Python 3.6`. For the DL experiment, we used `Keras 2.2.4` (17) with `Tensorflow 1.13.1` (1) as backend.

4.1 Understanding the role of hyperparameters

Both hyperparameters α and β can be seen as damping coefficients from the viewpoint of mechanics as discussed by (3) and sketched in the introduction. Recall the second-order time-continuous dynamics which serves as a model to INDIAN for twice differentiable $\mathcal{J}$:

$$\text{(DIN)} \quad \ddot{\theta}(t) + \alpha\,\dot{\theta}(t) + \beta\,\nabla^2\mathcal{J}(\theta(t))\dot{\theta}(t) + \nabla\mathcal{J}(\theta(t)) = 0. \tag{11}$$

This differential equation was inspired by Newton's Second Law of dynamics asserting that the acceleration of a material point coincides with the sum of forces applied to the particle. As recalled in the introduction three forces are at stake: the gravity and two friction terms. The parameter α calibrates the *viscous damping* intensity as in the Heavy Ball friction method of (33). It acts as a dissipation term but it can also be seen as a proximity parameter of the system with the usual gradient descent: the higher α is, the more DIN behaves as a pure gradient descent.[3] On the other hand the parameter β can be seen as a "Newton damping" which takes into account the geometry of the landscape to brake or accelerate the dynamics in an adaptive anisotropic fashion, see (4; 3) for further insights.

We now turn our attention to the cousin dynamics INDIAN, and illustrate the versatility of the hyperparameters α and β in this case. We proceed on a 2D visual nonsmooth ill-conditioned example à la Rosenbrock, see Figure 1. For this example, we aim at finding the minimum of the function $f(x, y) = 100(y - |x|)^2 + |1 - x|$. This function has a *V*-shaped valley, and a unique critical point at $(1, 1)$ which is also the global minimum. Starting from the point $(-1, 1.5)$ (the black cross), we apply INDIAN with constant steps $\gamma = 10^{-4}$. Figure 1 shows that when β is too small, the trajectory presents many transverse oscillations as well as longitudinal ones close to the critical point (subplot (a)). Then, increasing β significantly reduces transverse/parasite oscillations (subplot (b)). Finally, the longitudinal oscillations are reduced by choosing a higher α (subplot (c)). In addition, these behaviors are also reflected in the values of the objective function (subplot (d)).

The orange curve (first setting) presents large oscillations. Moreover, looking at the red curve, corresponding to plot (c), there is a short period between $20,000$ and $60,000$ iterations when the decrease is slower than for the other values of α and β, but still it presents fewer oscillations. In the longer term, the third setting ($\alpha = 1.3$, $\beta = 0.1$) provides remarkably good performance.

4.2 Training a DNN with INDIAN

Methodology. We now compare INDIAN with other popular algorithms used in deep learning (SGD, ADAM, ADAGRAD). We train a DNN for classification using the CIFAR-10 dataset (25). This dataset is composed of $60,000$ small colored images of size $32 \times 32 \times 3$ each associated with a label (airplane, cat, etc.). We split the dataset into $50,000$ images for the training part and $10,000$ for the test. Regarding the network, we use a sightly modified version of the LeNet-5 network of (27) as implemented by (37). It consists of two 2D-convolutional layers with max-pooling and three dense layers with ReLU activation function. The loss function used is the categorical cross-entropy. We compare our algorithm to both the classical stochastic gradient descent algorithm and the very popular ADAGRAD (22) and ADAM (24) algorithms. At each iteration, we compute the approximation of $\partial\mathcal{J}(\theta)$ on a subset $\mathcal{B} \subset \{1, \ldots, 50000\}$ of size 32. To do a fair comparison, each algorithm is initialized with the same random weights (following a normal distribution). To obtain more relevant results, this process is done for five different random initializations of θ. Given θ_0, ψ_0 is initialized such that the initial velocity is in the direction of $-\nabla\mathcal{J}(\theta_0)$, we use $\psi_0 = (1 - \alpha\beta)\theta_0 - (\beta^2 - \beta)\nabla\mathcal{J}(\theta_0)$.

[3] This is easier to see when one rescales $\mathcal{J}$ by α.

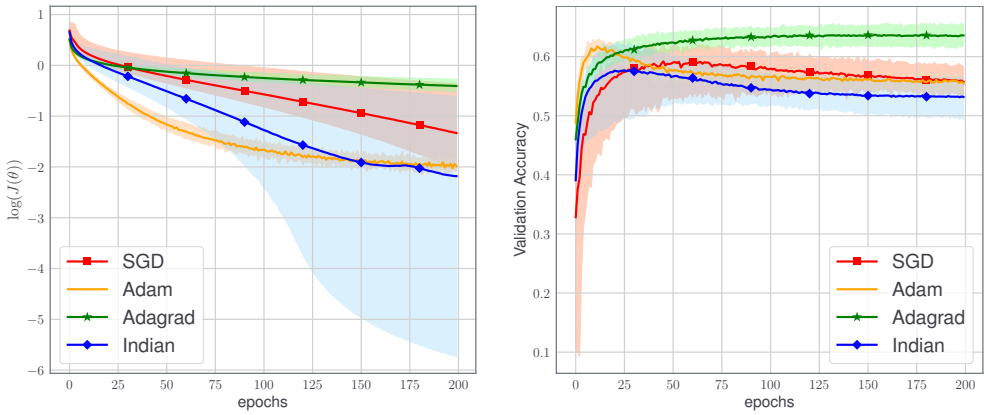


Figure 2: Optimization and accuracy results using LeNet with CIFAR-10. Left: logarithm of the loss function $\mathcal{J}(\theta)$. Right: percentage accuracy on test set. Comparison between SGD, ADAGRAD, ADAM, and INDIAN. For all algorithms we use a mini-batch of size 32 and 200 epochs. Solid lines represent mean values and pale surfaces represent the best and worst runs in terms of training loss and validation accuracy over five random initailizations.

The sequence of steps γ_k has to meet Assumption 1 and we chose the classical schedule $\gamma_k = \frac{\gamma_0}{\sqrt{k+1}}$ for both INDIAN and SGD, where γ_0 is the initial stepsize. Starting from γ_0, ADAGRAD and ADAM's steps follow an adaptive procedure based on past gradients, see (22; 24). For all four algorithms, choosing the right initial step length γ_0 is often critical in terms of efficiency. We use a grid-search for each algorithm and chose the initial stepsize that most decreases the loss function over four epochs (four complete passes over the data).

Based on Section 4.1 we used INDIAN with $\alpha = 0.5, \beta = 0.1$; there are however other possible choices of these parameters that provide satisfying results as shown in Figure 3, Appendix C.1. Besides performances on the empirical loss, we assess the accuracy of the estimated DNNs using the test dataset that contains $10,000$ images.

LeNet network is a reasonable benchmark that is however behind current state-of-the-art architectures. We may only expect $\sim 60\%$ accuracy with this network on CIFAR-10 (32; 28), as obtained in our results. Obtaining higher accuracy would involve more complex networks with careful tuning strategies, beyond the scope of this paper.

Results. Figure 2 shows that SGD is significantly slower than INDIAN whereas ADAM is faster at early training. However, ADAM fails to decrease $\mathcal{J}(\theta)$ below 10^{-2}, this behavior has been observed before (38). On the contrary INDIAN performs much better in the long run and can reach very low training error. ADAM can be interpreted as a special case of the Heavy Ball algorithm (7), which coincides with (7) when $\beta = 0$ and the friction coefficient α is not constant. Therefore we expect INDIAN to keep the favourable properties of ADAM, while introducing additional advantages thanks to the parameter β. That said, one may want to tune INDIAN such that it becomes as efficient as ADAM for early training. However to do so, it is necessary to take into account the fact that ADAM decreases the stepsizes in a non usual way, therefore we tried to use slow decreasing steps of the form Ck^{-a} in INDIAN which preserves convergence as stated in Theorem 1 because it meets Assumption 1. Figure 4 of Appendix C.2 shows that slower stepsize decay may lead to faster loss decrease in early training using INDIAN.

Finally, regarding validation accuracy, it appears that for the choice of parameters α and β we made on Figure 2, validation is not as good as for the other algorithms (especially ADAGRAD) though it remains comparable. Different values of hyperparameters α and β allow to tune INDIAN to obtain similar performances as ADAGRAD on our example (see Figure 3 (b) of Appendix C.1). This suggests that a trade-off between generalization and training can be found by tuning these hyperpameters.

5 Conclusion

We introduced a novel stochastic optimization algorithm featuring inertial and Newtonian behaviour motivated by applications to DL. We provided a powerful convergence analysis under weak hypotheses applicable to most DL problems. We would like to point out that, apart from SGD (20), the convergence of concurrent methods in such a general setting is still an open question. Our result seems moreover to be the first one able to handle the mini-batch subsampling approach for ReLU DNNs. Our experiments show that INDIAN is very competitive with state of the art algorithms for DL. We stress that these numerical manipulations were performed on substantial DL benchmarks with only minimal algorithm tuning (very classical stepsizes with a simple grid search on a few epochs to set the initial stepsize). This facilitates reproducibility and allows to stay as close as possible to the reality of DL applications in machine learning.

Acknowledgments

This work has been supported by the European Research Council (ERC FACTORY-CoG-6681839). Support from the ANR-3IA Artificial and Natural Intelligence Toulouse Institute is also gratefully acknowledged. Jérôme Bolte was partially supported by Air Force Office of Scientific Research, Air Force Material Command, USAF, under grant number FA9550-18-1-0226.
Part of the numerical experiments were done on the OSIRIM platform of IRIT, supported by the CNRS, the FEDER, the Occitanie region and the French government (http://osirim.irit.fr/site/fr).

We thank H. Attouch for useful discussions.

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In *Proceedings of USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 265–283, 2016.

[2] S. Adil. *Opérateurs monotones aléatoires et application à l'optimisation stochastique*. PhD Thesis, Paris Saclay, 2018.

[3] F. Alvarez, H. Attouch, J. Bolte, and P. Redont. A second-order gradient-like dissipative dynamical system with Hessian-driven damping: Application to optimization and mechanics. *Journal de Mathématiques Pures et Appliquées*, 81(8):747–779, 2002.

[4] F. D. Alvarez and J. C. Pérez. A dynamical system associated with Newton's method for parametric approximations of convex minimization problems. *Applied Mathematics and Optimization*, 38:193–217, 1998.

[5] H. Attouch, J. Bolte, P. Redont, and A. Soubeyran. Proximal alternating minimization and projection methods for nonconvex problems: An approach based on the Kurdyka-Łojasiewicz inequality. *Mathematics of Operations Research*, 35(2):438–457, 2010.

[6] J.-P. Aubin and A. Cellina. *Differential inclusions: set-valued maps and viability theory*. Springer, 2012.

[7] A. Barakat and P. Bianchi. Convergence of the ADAM algorithm from a dynamical system viewpoint. *arXiv:1810.02263*, 2018.

[8] M. Benaïm. Dynamics of stochastic approximation algorithms. In *Séminaire de Probabilités XXXIII*, pages 1–68. Springer, 1999.

[9] M. Benaïm, J. Hofbauer, and S. Sorin. Stochastic approximations and differential inclusions. *SIAM Journal on Control and Optimization*, 44(1):328–348, 2005.

[10] A. S. Berahas, R. Bollapragada, and J. Nocedal. An investigation of Newton-sketch and subsampled newton methods. *arXiv:1705.06211*, 2017.

[11] J. Bolte, A. Daniilidis, A. Lewis, and M. Shiota. Clarke subgradients of stratifiable functions. *SIAM Journal on Optimization*, 18(2):556–572, 2007.

[12] V. S. Borkar. *Stochastic approximation: A dynamical systems viewpoint*. Springer, 2009.

[13] L. Bottou and O. Bousquet. The tradeoffs of large scale learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 161–168, 2008.

[14] L. Bottou, F. E. Curtis, and J. Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018.

[15] R. H. Byrd, G. M. Chin, W. Neveitt, and J. Nocedal. On the use of stochastic hessian information in optimization methods for machine learning. *SIAM Journal on Optimization*, 21(3):977–995, 2011.

[16] R. H. Byrd, S. L. Hansen, J. Nocedal, and Y. Singer. A stochastic quasi-newton method for large-scale optimization. *SIAM Journal on Optimization*, 26(2):1008–1031, 2016.

[17] F. Chollet. Keras. `https://github.com/fchollet/keras`, 2015.

[18] F. H. Clarke. *Optimization and nonsmooth analysis*. SIAM, 1990.

[19] M. Coste. *An introduction to o-minimal geometry*. Istituti editoriali e poligrafici internazionali Pisa, 2000.

[20] D. Davis, D. Drusvyatskiy, S. Kakade, and J. D. Lee. Stochastic subgradient method converges on tame functions. *Foundations of Computational Mathematics*, 2019. (in press).

[21] L. van den Dries. *Tame topology and o-minimal structures*. Cambridge university press, 1998.

[22] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(7):2121–2159, 2011.

[23] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 315–323, 2011.

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

[25] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, Canadian Institute for Advanced Research, 2009.

[26] H. Kushner and G. G. Yin. *Stochastic approximation and recursive algorithms and applications*. Springer, 2003.

[27] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[28] W. Li. Convolutional neural networks for CIFAR-10. `https://github.com/BIGBALLON/cifar-10-cnn/tree/master/1_Lecun_Network`, 2018.

[29] L. Ljung. Analysis of recursive stochastic algorithms. *IEEE Transactions on Automatic Control*, 22(4):551–575, 1977.

[30] J. Martens. Deep learning via Hessian-free optimization. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 735–742, 2010.

[31] E. Moulines and F. R. Bach. Non-asymptotic analysis of stochastic approximation algorithms for machine learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 451–459, 2011.

[32] A. A. Pandey. LeNet-Cifar10. `https://github.com/amitpandey2194/LeNet-Cifar10/`, 2017.

[33] B. T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

[34] H. Robbins and S. Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(1):400–407, 1951.

[35] D. E. Rumelhart and G. E. Hinton. Learning representations by back-propagating errors. *Nature*, 323(9):533–536, 1986.

[36] M. Shiota. *Geometry of subanalytic and semialgebraic sets*, volume 150. Springer Science & Business Media, 2012.

[37] T. Thaman. LeNet-5 with Keras. `https://github.com/TaavishThaman/LeNet-5-with-Keras`, 2018.

[38] A. C. Wilson, R. Roelofs, M. Stern, N. Srebro, and B. Recht. The marginal value of adaptive gradient methods in machine learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 4148–4158, 2017.

[39] P. Xu, F. Roosta-Khorasani, and M. W. Mahoney. Second-order optimization for non-convex machine learning: An empirical study. *arXiv:1708.07827*, 2017.

This is the appendices for the paper "An Inertial Newton Algorithm for Deep Learning".

A Some variational results under sum rule failures

We recall a useful result of (20) which follows of the projection formula in (11).

Lemma 3 (Chain rule)**.** *Let $g : \mathbb{R}^P \to \mathbb{R}$ be a locally Lipschitz continuous, tame function, then g admits a chain rule, meaning that for all absolutely continuous curves $\theta : \mathbb{R}_+ \to \mathbb{R}^P$, and for almost all $t \in \mathbb{R}_+$,*

$$\frac{\mathrm{d}g}{\mathrm{d}t}(\theta(t)) = \langle \dot{\theta}(t), \partial g(\theta(t)) \rangle = \langle \dot{\theta}(t), v \rangle, \ \ \forall v \in \partial g(\theta(t)). \tag{12}$$

Consider now a function with an additive/composite structure (such as in deep learning):

$$f \colon \mathbb{R}^P \ni \theta \mapsto \sum_{i=1}^{n} f_i(\theta),$$

where each $f_i \colon \mathbb{R}^P \mapsto \mathbb{R}$ is locally Lipschitz and tame. We set for any $\theta \in \mathbb{R}^P$

$$Df(\theta) = \sum_{i=1}^{n} \partial f_i(\theta).$$

The following lemma applies and is a direct generalization of the above chain rule.

Lemma 4 (Chain rule)**.** *Let $c \colon [0,1] \mapsto \mathbb{R}^P$ be an absolutely continuous curve so that $t \mapsto f(c(t))$ is differentiable almost everywhere. If f is tame then, for almost all $t \in [0,1]$, and for all $v \in Df(c(t))$,*

$$\frac{d}{dt} f(c(t)) = \langle v, \dot{c}(t) \rangle .$$

Proof. By local Lipschitz continuity and absolute continuity, each f_i is differentiable almost everywhere and Lemma 3 can be applied:

$$\frac{d}{dt} f_i(c(t)) = \langle v_i, \dot{c}(t) \rangle \text{, for all } v_i \in \partial f_i(c(t)) \text{ and or almost all } t \in \mathbb{R}_+.$$

Thus

$$\frac{d}{dt} f(c(t)) = \sum_{i=1}^{n} \frac{d}{dt} f_i(c(t)) = \sum_{i=1}^{n} \langle v_i, \dot{c}(t) \rangle ,$$

for any $v_i \in \partial f_i(c(t))$, for all $i = 1, \ldots, n$, and for almost all $t \in \mathbb{R}_+$. This proves the desired result. □

Lemma 5 (A Sard's theorem for tame D-critical values)**.** *Set*

$$S = D - \text{crit} := \left\{ \theta \in \mathbb{R}^P : Df(\theta) \ni 0 \right\},$$

then $f(S)$ is finite.

Proof. The set S is tame and hence it has a finite number of connected components. It is sufficient to prove that f is constant on each connected component of S. Without loss of generality, assume that S is connected and consider $\theta_0, \theta_1 \in S$. By Whitney regularity (21, 4.15), there exist a tame continuous path γ joining θ_0 to θ_1. Because of the tame nature of the result, we should here conclude with only tame arguments and use the projection formula in (11), but for convenience of readers who are not familiar with this result we use Lemma 3. Since γ is tame, the monotonicity lemma gives the existence of a finite collection of real numbers $0 = a_0 < a_1 < \ldots < a_q = 1$, such that γ is C^1 on each segment (a_{j-1}, a_j), $j = 1, \ldots, q$. Applying Lemma 3 to each $\gamma_{|(a_i, a_{i+1})}$, we see that f is constant save perhaps on a finite number of points, it is thus constant by continuity. □

B Proof of Lemma 1

Proof. Define $E_\lambda(\theta,\psi) = \lambda\mathcal{J}(\theta) + \frac{1}{2}\left|(\alpha-\frac{1}{\beta})\theta + \frac{1}{\beta}\psi\right|^2$. We aim at choosing λ so that E_λ is a Lyapunov function. Because $\mathcal{J}$ is tame and locally Lipschitz continuous, using Lemma 3 we know that for any absolutely continuous trajectory $\theta : \mathbb{R}_+ \to \mathbb{R}^P$ and for almost all $t > 0$,

$$\frac{\mathrm{d}\mathcal{J}}{\mathrm{d}t}(\theta(t)) = \langle \dot\theta(t), D\mathcal{J}(\theta(t))\rangle = \langle \dot\theta(t), v(t)\rangle, \;\; \forall v(t) \in D\mathcal{J}(\theta(t)). \tag{13}$$

Let θ and ψ be solutions of (DIN). For almost all $t \in \mathbb{R}_+$, we can differentiate $E_\lambda(\theta,\psi)$ to obtain

$$\begin{aligned}\frac{\mathrm{d}E_\lambda}{\mathrm{d}t}(\theta(t),\psi(t)) =& \lambda\langle\dot\theta(t), v(t)\rangle + (\alpha-\frac{1}{\beta})\langle\dot\theta(t), (\alpha-\frac{1}{\beta})\theta(t) + \frac{1}{\beta}\psi(t)\rangle \\ &+ \frac{1}{\beta}\langle\dot\psi(t), (\alpha-\frac{1}{\beta})\theta(t) + \frac{1}{\beta}\psi(t)\rangle\end{aligned} \tag{14}$$

for all $v(t) \in D\mathcal{J}(\theta(t))$. Using (6), we get $\frac{1}{\beta}(\dot\theta(t) - \dot\psi(t)) \in D\mathcal{J}(\theta(t))$ and $-\dot\psi(t) = (\alpha - \frac{1}{\beta})\theta(t) + \frac{1}{\beta}\psi(t)$ a.e. Choosing $v(t) = \frac{1}{\beta}(\dot\theta(t) - \dot\psi(t))$ yields:

$$\frac{\mathrm{d}E_\lambda}{\mathrm{d}t}(\theta(t),\psi(t)) = \lambda\left\langle\dot\theta(t), \frac{\dot\theta(t)-\dot\psi(t)}{\beta}\right\rangle - (\alpha-\frac{1}{\beta})\left\langle\dot\theta(t),\dot\psi(t)\right\rangle - \frac{1}{\beta}\left\langle\dot\psi(t),\dot\psi(t)\right\rangle.$$

Then, expressing everything as a function of $\dot\theta$ and $\frac{1}{\beta}(\psi-\theta)$, one can show that a.e. on $\mathbb{R}_+$:

$$\begin{aligned}\frac{\mathrm{d}E_\lambda}{\mathrm{d}t}(\theta,\psi)(t) &= -\alpha|\dot\theta(t)|^2 - \beta\left|\frac{\dot\theta(t)-\dot\psi(t)}{\beta}\right|^2 + (\lambda-\alpha\beta-1)\,\langle\dot\theta(t), \frac{\dot\theta(t)-\dot\psi(t)}{\beta}\rangle \\ &= -\left(\sqrt{\alpha}\dot\theta(t) + \frac{\alpha\beta+1-\lambda}{2\sqrt{\alpha}}\frac{\dot\theta(t)-\dot\psi(t)}{\beta}\right)^2 - \left(\beta - \frac{(\alpha\beta+1-\lambda)^2}{4\alpha}\right)\left|\frac{\dot\theta(t)-\dot\psi(t)}{\beta}\right|^2.\end{aligned}$$

We aim at choosing λ so that E_λ is decreasing that is $\left(\beta - \frac{(\alpha\beta+1-\lambda)^2}{4\alpha}\right) > 0$. This holds whenever $\lambda \in \left[(1-\sqrt{\alpha\beta})^2, (1+\sqrt{\alpha\beta})^2\right]$. We choose $\lambda_{\min} = (1-\sqrt{\alpha\beta})^2$, and $\lambda_{\max} = (1+\sqrt{\alpha\beta})^2$, for these two values we obtain for almost all $t > 0$:

$$\begin{cases}\dot E_{\lambda_{\min}}(\theta(t),\psi(t)) &= -\left|\sqrt{\alpha}\dot\theta(t) + \frac{1}{\sqrt{\beta}}\left(\dot\theta(t)-\dot\psi(t)\right)\right|^2 \\ \dot E_{\lambda_{\max}}(\theta(t),\psi(t)) &= -\left|\sqrt{\alpha}\dot\theta(t) - \frac{1}{\sqrt{\beta}}\left(\dot\theta(t)-\dot\psi(t)\right)\right|^2\end{cases} \tag{15}$$

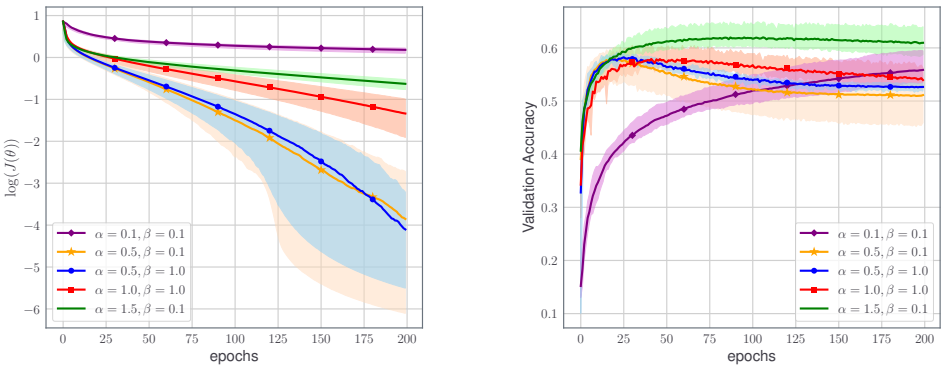
Remark finally that by definition $E_{\min} = E_{\lambda_{\min}}$ and $E_{\max} = E_{\lambda_{\max}}$. □

C Additional experiments

All the results displayed in this section are related to INDIAN applied to the learning problem described in Section 4. The experimental setup is the same and we investigate the influence of hyperparameters and stepsize schedule.

C.1 Hyperparameters

Figure 3 (a) illustrates the influence of the hyperparameters α and β in training. While some choices appears better for training, others provide a better generalization, as shown by validation performance displayed in Figure 3 (b). It seems that tuning these parameters allows to achieve a compromise between training and testing as discussed at the end of Section 4.2.

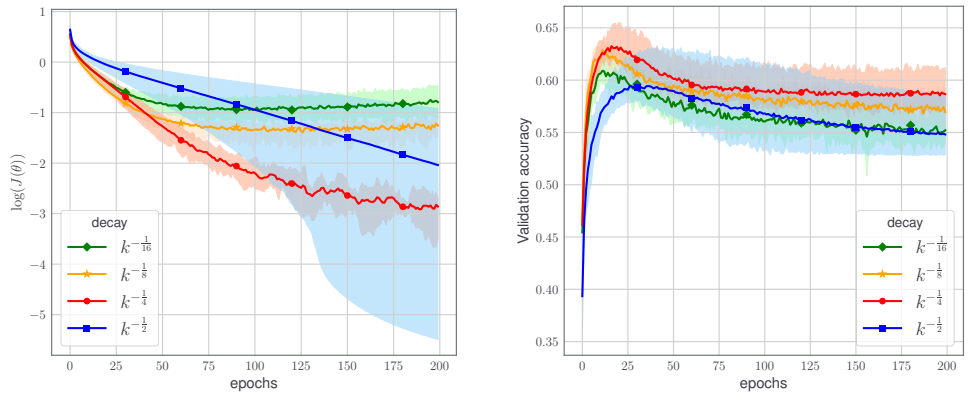


(a) Training loss (b) Test accuracy

Figure 3: INDIAN performance in the experiment of Section 4.2 for five different values of the hyperparameters (α, β). As before, solid line represent mean value over the five runs while pale surfaces represent the gap between the best and the worst run.

C.2 Stepsize decay

Using INDIAN with $(\alpha, \beta) = (0.5, 0.1)$ on the DL problem decribed in Section 4, we now investigate the influence of the stepsize schedule. We use schedules of the form $\gamma_k = \gamma_0(k+1)^{-1/q}$ where γ_0 is found using a grid search over four epochs. First, to link this experiment with the previous ones, notice that the blue curve on Figure 4 (a) corresponds to the exact same experiment as the blue curve on Figure 2. Then the most striking result is the red curve which shows that using $1/4$ for the decreasing power allows to reach the same speed as ADAM for early training and to achieve better training loss in the long term. This illustrates that moderate tuning of INDIAN can outperform state-of-the-art algorithms.



(a) Training loss (b) Test accuracy

Figure 4: INDIAN performance in the experiment of Section 4.2 for four different stepsize schedules.

D Python Codes

The python notebooks that reproduce the experiment of this paper are available at `https://github.com/camcastera/Code-for-an-Inertial-Newton-algorithm-for-DL/`.

The ready-to-use code to train other networks with the INDIAN algorithm, is available here: `https://github.com/camcastera/Indian-for-DeepLearning/`.