Minimax Theory and its Applications
Volume 7 (2022), No. 1, 79–108

Backtracking Gradient Descent Method and some Applications in Large Scale Optimisation Part 1: Theory

Tuyen Trung Truong
Matematikk Institut, Universitetet i Oslo, Blindern, 0851 Oslo, Norway
tuyentt@math.uio.no

Hang-Tuan Nguyen
Axon AI Research, hnguyen@axon.com

Received: May 15, 2020
Accepted: February 26, 2021

Deep Neural Networks (DNN) are essential in many realistic applications, including Data Science. At the core of DNN is numerical optimisation, in particular gradient descent methods (GD). The purpose of this paper is twofold. First, we prove some new results on the backtracking variant of GD under very general situations. Second, we present a comprehensive comparison of our new results to the previously known results in the literature, providing pros and cons of these methods. To illustrate the efficiency of Backtracking line search, we will present some experimental results (on validation accuracy, training time and so on) on CIFAR10, based on implemetations developed in another paper by the authors. Source codes for the experiments are available on GitHub.

Keywords: Backtracking, deep learning, global convergence, gradient descent, line search method, optimisation, random dynamical systems.

2010 Mathematics Subject Classification: 65Kxx, 68Txx, 49Mxx, 68Uxx.

1. Introduction

It is undeniable that Optimisation is important, not just in theory but also in applications. It is known that finding global optima is NP-hard. Therefore, one would be very happy if having an iterative method which will converge to a critical point, and moreover can avoid saddle points. It is true that there are many iterative algorithms out there, and each can be good for some special cost functions (such as convex or functions whose gradients are globally Lipschitz continuous). However, it seems not be emphasised the following important facts. The first fact is that almost every of these iterative algorithms do not guarantee convergence to critical points for general cost functions, and hence strictly speaking does not apply to the complicated cost functions one encounters in huge scale and realistic optimisation problems such as in Deep Learning. The second fact is that a classical but largely ignored iterative method, and to some extend underrated, developed by Armijo (also called Backtracking Gradient Descent) in the 1960s, is actually the best theoretically guaranteed and very easy to use. Indeed, see below, it is classically known that any cluster point in the sequences constructed by methods which guarantee Armijo's

ISSN 2199-1413 (printed), ISSN 2199-1421 (electronic) / $ 2.50

condition (some of these other methods will be reviewed in Section 2) is a critical point of the cost function.

The consequence of a combination of the above two facts is that the community and researchers could waste time trying to invent new iterative methods without realising that a good iterative method already exists. This leads to another consequence, namely, that the users do not use the associated implementation of this method, which is very efficient and automatic, and hence waste time and risk being frustrated with many technical difficulties of using other (not theoretically guaranteed, for a general purpose) iterative methods. Also, since large scale optimisation like in Deep Learning does not satisfy the usual constraints (being convex or gradients being globally Lipschitz continuous), many heuristics and tricks have been devised, which in any event: do not increase the theoretical guarantee and still do not prevent the cases when things go wrong. Moreover, using those algorithms, with a lot of manual fine tuning, requires a lot from the users. For example, one can ask how can a user be sure that a very complicated cost function is convex or having gradients being globally Lipschitz, and in the latter case – what is the Lipschitz constant. In contrast, using implementations of Backtracking Gradient Descent is much more simple. For example, the users do not need to worry too much about tuning learning rates. The users can also determine, by a quick glance, whether a function is C^1 or C^2 (these are conditions which are needed for Armijo's method or its modifications to work).

This paper, which is developed from the more theoretical part of our preprint arXiv: 1808.05160, proves that the sequence in Armijo's method actually *converges* under very natural assumptions on the cost functions (including all Morse functions, those one expect to meet if choosing a function randomly). It seems to us that one reservation which makes people reluctant in using Armijo's method in large scale optimisation is the fact that function evaluation in Armijo's condition is expensive. To this end, the more experimental part of our preprint (on CIFAR10 and CIFAR100, with source codes available), which is published separately, addresses by introducing new modifications of Backtracking Gradient Descent - and still with a lot theoretical guarantee - which allows to lower the need of checking Armijo's condition. We note that more recent work by the first author shows that some modifications of Armijo's method can avoid saddle points, and hence obtain the following very neat result: For a Morse function, there are modifications of Armijo's method which converge to local minimum (if not diverge to infinity). More on this will be gradually presented in this paper.

Deep Neural Networks (DNN) are essential in many realistic applications, including Data Science. Some recent spectacular achievements are Alpha Go (in the Game of Go) and driverless cars. Three pioneers of Deep Neural Networks (DNN) were awarded the 2018 Turing prize, considered as the "Nobel prize of computing". Nowadays, there are many tools in daily life applications in which Deep Learning plays a crucial role. Face recognition in smart phones is one such example, as well as Google Translate or Siri assistant on iPhones. Other examples are security cameras, 3D segmentation of medical images [14] and natural resources detection (such as the TGS salt identification challenge on kaggle.com), and the list keeps increasing with time.

Essentially, for each question to be solved (for example, checking whether there is a cat or not a cat in a picture), a researcher will choose - based on experience and knowledge - a specific DNN which is suitable for the task. Roughly speaking, this DNN will, given a test data set I, provide a family of cost functions

$$f(\alpha) := \frac{1}{\sharp I} \sum_{x \in I} \phi(x, y(x), \alpha),$$

where $\sharp I$ is the number of elements of I and for each data $x \in I$, $y(x)$ is the correct label of x. Here α are some parameters and ϕ is a certain map, both depending on the architecture of the DNN. Then, the researcher will try to find values $\hat{\alpha}$ which minimise this function $f(\alpha)$ (usually called cost function or objective function). The found parameter $\hat{\alpha}$ is then used to make predictions when new data are fed in.

It is known that in general there is no close form solution to such an optimisation problem. It is more so with DNN where the number of parameters may be tens of millions. Therefore, in practice, one needs to utilise numerical algorithms. Gradient descent method (whose details will be provided later) is such a method. Historically, the simplicity of the gradient descent method and its practical effectiveness are reasons for the successful implementation and performance of it in DNN. To scope with constraints in time and memory, the mini-batch practice is usually employed. In this practice, one randomly shuffle the training set and partition it into small chunks I_h (mini-batches), from which one obtains a partial sum

$$f_h(\alpha) = \frac{1}{\sharp I_h} \sum_{x \in I_h} \phi(x, y(x), \alpha).$$

One will apply one step of gradient descent method for each such partial sum, and when finishes with all mini-batches in one partition one says that an epoch is finished. To train a DNN, usually one needs to run about tens to hundreds of epochs. Mini-batch training is now main stream in the Deep Learning community, and the reason for using mini-batch comes from the limitation of resources (especially RAM for GPUs) and/or time reduction to compute gradients for each iteration. Computing gradients for full batch takes much more time in comparison to mini-batch, the cost is nearly linear. Moreover, mini-batch can help to escape saddle points.

However, together with the increase in applications and usage of DNN, comes also the awareness about drawbacks of the current practices in DNN. In the annual meeting of AAAS (publisher of the well-known journal Science) in 2019, G. Allen wonders if one can trust discoveries made by Machine Learning algorithms. Some most important concerns are: The lack of a comprehensive theoretical justification of Deep Learning's practice; and Overload of random heuristics and the number of hyper-parameters to choose. Deep Learning can also be very easily fooled: there are adversarial images (B), which are small modifications (either synthetic or physical) of some other images (A), which the models totally fail to classify [18, 31]. The mystery is that the models already classified images A with very high precision, and to a human's eyes B are from the same category as A. Even just gluing stickers on street signs could confuse self-driving cars [18]. (We can then wonder naturally whether natural phenomena, such as rain or snow on the street signs, could also have the same effects.)

Finally, Deep Learning is still not safe. Adversarial attacks are now also available on medical images [20]. In March 2018, a Uber's autonomous car killed a woman in Arizona and a Tesla's autonomous car crashed killing the driver in California. Similar catastrophic scenes can be very realistic in the future, if we cannot resolve the mentioned concerns effectively and rely on the theoretical foundations of Deep Learning (including numerical optimisation methods such as GD).

One important parameter in GD is learning rate (see below), and hence to resolve the above concerns thoroughly one needs to address the question of how to automatically, adaptively and optimally choose learning rates. The main purpose of this paper is to contribute to this very question, in particular we will show that the Backtracking variant of GD works very well both theoretically and practically.

The other purpose of this paper is to provide a clean brief survey of main results in GD, which is relevant and helpful in the view that the literature in this field is too huge and diverse, so a practicer of GD may get confusion about what theoretical results really are and what are their scope of applications, and so to avoid overuse an algorithm to some cost functions which do not satisfy assumptions required by theoretical results. For one example, we note that in many references in the field (including Stochastic GD) there are many "Convergence theorems" where a sequence $\{z_n\}$ constructed by one of the various variants of GD is called convergent if the sequence of gradient values $\{\nabla f(z_n)\}$ converges to 0. These statements can cause confusion theoretically (the meaning used here is weaker than the common definition of convergence in mathematics) and be problematic practically. For example, they can lead to instability, because if one stops the algorithm after 10^6 iterations and after $10^6 + 10^3$ iterations, then the resulting two trained values z_{10^6} and $z_{10^6+10^3}$ – which one intends to use for making new predictions, see above – can theoretically be very much different (not as one may expect) and consequently lead to different new predictions if the sequence $\{z_n\}$ does not converge. An algorithm with better guaranteed convergence, such as Backtracking GD, is very useful in this regard.

Next, we will briefly review about GD. Let $f : \mathbb{R}^m \to \mathbb{R}$ be a C^1 function, that is continuously differentiable. Denote by ∇f the *gradient* of f, that is for $x \in \mathbb{R}^m$:

$$\nabla f(x) = (\frac{\partial f}{\partial x_1}(x), \ldots, \frac{\partial f}{\partial x_m}(x)).$$

We also denote by $||\nabla f(x)||$ the usual Euclidean norm of the vector $\nabla f(x) \in \mathbb{R}^m$, i.e.

$$||\nabla f(x)|| = \sqrt{[\frac{\partial f}{\partial x_1}(x)]^2 + \ldots + [\frac{\partial f}{\partial x_m}(x)]^2}.$$

Finding critical points (in particular, minima) of such a function is an important problem faced often in science and technology. Most common methods include gradient descent (GD) and Newton's method. While Newton's method works better than GD in the local setting, it requires much stronger assumptions and not many general convergence results in the global setting are proven for it. Moreover, second or higher differential order methods such as Newton's are very costly to implement for DNN.

GD was proposed by Cauchy [13] in 1847 to solve systems of non-linear equations, and has been extensively studied for many years since. It has many applications in science and technology, such as being main stream in DNN (for overview see [9, 28]).

The general scheme for this approximation method GD is as follows. We start from a *random* point $z_0 \in \mathbb{R}^m$, and then construct iteratively

$$z_{n+1} = z_n - \delta_n \nabla f(z_n), \tag{1}$$

where $\delta_n > 0$ are appropriately chosen. In the literature, it is common to call δ_n learning rates or step sizes. The hope is that z_n will converge to a (global) minimum. The intuition is taken from the familiar picture one obtains when f is a convex function. Note that z_0 being random is important here: In [27], one can find a function in 2 variables and a specific choice of the point z_0 for which any sequence as in (1), if converges to a critical point at all, can only converge to a saddle point.

The most known GD method is Standard GD, where we choose $\delta_n = \delta_0$ for all n. Hence, we start with a (randomly chosen) point $z_0 \in \mathbb{R}^m$ and define the sequence

$$z_{n+1} = z_n - \delta_0 \nabla f(z_n). \tag{2}$$

Rigorous results have been proven for Standard GD in case the gradient ∇f is globally Lipschitz [16, 15, 22, 3] and the learning rate δ_0 is small enough ($< 1/(2L)$), plus some additional assumptions. These assumptions are basically the best under which convergence of Standard GD can be proven, see Examples 2.13 and 2.14 below.

For ease of later reference, we denote by $C^{1,1}_L$ the set of C^1 functions f whose gradient is globally Lipschitz with Lipschitz constant L, that is $||\nabla f(x) - \nabla f(y)|| \leq L||x-y||$ for all $x, y \in \mathbb{R}^m$. It is worthy to remark that the class $C^{1,1}_L$ is not preserved under small perturbations. For example, as mentioned in Example 2.11 below, there is a common technique in DNN called L^q regularisation (or compensation), designed to prevent overfitting, which for a function f considers perturbations of the form $\hat{f}(x) = f(x) + \epsilon||x||^q$ for some $\epsilon, q > 0$. However, if $f \in C^{1,1}_L$ and $q \neq 2$, then $\hat{f}$ is not in $C^{1,1}_{L'}$ for every L'!

In reality, when doing actual computation on computers, one can only obtain the values for gradients and other quantities within some errors. The inexact version of GD, which is more suitable to realistic applications such as DNN, is as follows:

$$z_{n+1} = z_n - \delta_n v_n, \tag{3}$$

where we assume that v_n is not too far from the gradient direction $\nabla f(z_n)$. There are many ways to specify the condition of "not too far" (see e.g. [9, 8]), here in this paper we use the following common version: there are $A_1, A_2 > 0$ and $1 \geq \mu > 0$ such that for all n

$$A_1||\nabla f(z_n)|| \leq ||v_n|| \leq A_2||\nabla f(z_n)||, \tag{4}$$

$$\langle \nabla f(z_n), v_n \rangle \geq \mu||\nabla f(z_n)|| \times ||v_n||. \tag{5}$$

When $\mu = 1$ we have that v_n is parallel to $\nabla f(z_n)$ for all n, and thus recover the scheme in (1). The geometric meaning of (5) is that the cosine of the angles between $\nabla f(z_n)$ and v_n are positive and bounded away from 0, uniformly in n.

For justification of GD in Deep Learning, a usually cited result is Stochastic GD, which goes back to the work by Robbins and Monro ([32, 9]). However, as will be seen from the analysis in Section 2.3, at current there is a considerable gap between results which can be proven in the deterministic case (for a single cost function) and in the stochastic case (for example, for the common practice of using mini-batches), which will need to be thoroughly addressed if one wants to have a firm theoretical foundation to justify the use of (mini-batch) GD in Deep Learning. In Section 2.3, there are also descriptions of other common variants of GD such as Wolfe's conditions. In the flow setting (i.e. solutions to ODE), the so-called inertial method [7] has also good theoretical guarantees. On the other hand, its discrete realisation cannot yet achieve competitive performance. We will discuss these, and some other algorithms which also make use of Armijo's condition but are different from Backtracking GD, more in Sections 2 and 4. For now, it suffices to mention that Backtracking GD is probably the simplest, but quite effective, among all methods which use Armijo's condition in one way or another.

The research in this field is very extensive, and by no means the references in this paper fully represent the state-of-the-art. For some surveys and implementations of these methods, see e.g. [8, 9, 10, 28, 33].

The focus in this paper is the variant of GD which is called Backtracking GD, because it is related to the backtracking line search method (see Section 2 for precise definition). We will show that this method behaves very nicely, both theoretically and experimentally. In particular, in contrast to Standard GD, convergence can now be proven for Backtracking GD for most functions (including all Morse functions). This expands the class of cost functions for which GD is guaranteed to work, and hence the models, problems and questions which can be dealt with. Besides, Backtracking line search can be used to boost the performance of other methods. For example, for cost functions which are a difference of two convex functions, this is done in [2]. For boosting of other well known algorithms such as NAG and Momentum, see [40, 39].

Some remarks about this paper. This paper is extracted and developed from the more theoretical part of [40]. The more applied part of [40] is published in [39].

This paper is organised as follows. In the next section, we will state our main theoretical results and generalisations (including an inexact version of Backtracking GD), together with comparisons to previous work in the literature. Proofs of main theoretical results are presented in Section 3. In Section 4 we present some experimental results (see Tables 4.1, 4.2, and 4.3, as well as Figures 4.1 and 4.2) on Deep Neural Networks (Resnet18 and LeNet on CIFAR10 dataset), some taken from [39], to illustrate the efficiency of Backtracking GD. In the last section, we present conclusions and some open questions. We emphasise that our results are for general C^1 functions, without requirements about further smoothness of the functions (such as $C^{1,1}_L$, real analytic or Losjasiewicz gradient inequality). In particular, convergence is guaranteed for C^1 functions with at most countably many critical points, which are (by transversality theorems) encountered with probability 100%.

As far as we know, our paper presents a new proof of showing $\lim_{n\to\infty} ||z_{n+1}-z_n|| = 0$, without relying on special properties of f such as being in $C^{1,1}_L$ or Losjasiewicz gra-

dient inequality like previous work in the literature. This paper is also the first to introduce real projective spaces into uncontrained optimisation on Euclidean spaces. For most of currently used Deep Neural Networks, the cost functions are continuous functions which are C^1 except at a small set of exceptional points. Example 2.10 below shows that our results can also be modified to apply to such cost functions. See also [36, 37].

Some further algorithms and heuristic arguments (such as backtracking versions of Momentum and Nesterov accelerated gradient, aimed to avoid convergence to bad local minima, as well as Two-way Backtracking GD), more applicable to realistic applications such as DNN, and experimental results in DNN, can be found in [39, 40]. Source codes for experimental results are available at the GitHub link [42]. Readers will see in [39, 40], modifications (such as Two-way Backtracking GD, a hybrid use between Backtracking GD and Standard GD) needed to implement in Deep Neural Networks, which reduce the need of checking Armijo's condition, as well as a novel scheme for rescaling of learning rates to deal with the mini-batch setting. Simply put, these algorithms do not check Armijo's condition all the time, but only periodically or when the function value does not decrease. Some experimental results from [39, 40] are reproduced in Section 4 of this paper, where some new experimental results are also presented, on CIFAR dataset with two DNNs: Resnet18 and LeNet.

The first author, developing ideas from this paper and [40], extended the results to various settings: avoidance of saddle points for a modification of Backtracking GD [36], and proving convergent results in the Banach space setting [38], and defining a coordinate-wise Armijo's condition and the corresponding coordinate-wise Backtracking GD to better adapt with the case where partial derivatives of the cost function f can be different in sizes for different directions. All in all, as of current, Backtracking GD and its modifications have the strongest theoretical guarantees, very flexible and stable with respect to its hyperparameters.

Acknowledgments. We are grateful to Geir Dahl, who pointed out relevant references, including [8]. T. T. Truong would like to thank the organisers and participants of the Mathematics in Industry Study Group meeting in 2016 at University of South Australia, in particular the DST group, for exposing him to various real life applications of numerical methods and industrial mathematics (including the gradient descent method). He also appreciates very much the many discussions with Neeraj Kashyap. He thanks also Terje Kvernes and the IT support team (Department of Mathematics, University of Oslo) for helping with technical issues. We thank Torus Actions SAS for collaboration in producing Figures 4.1b and 4.1a. This work is partially supported by Young Research Talents grant number 300814 from Research Council of Norway. Part of the experiments in this paper was performed on resources provided by UNINETT Sigma2 – the national infrastructure for high performance computing and data storage in Norway.

2. Statement of the results and comparison to previous work

In this section, we will first introduce our main theoretical results and generalisations (including an inexact version of Backtracking GD). Then we compare our results to previous work.

2.1. Main theoretical results

We recall first the so-called Armijo's condition, for some $0 < \alpha < 1$ and some $x, y \in \mathbb{R}^m$ (the main case of interest is when $y = x - \sigma\nabla f(x)$):

$$f(y) - f(x) \leq \alpha\langle\nabla f(x), y - x\rangle, \tag{6}$$

here $\langle ., .\rangle$ is the standard inner product in $\mathbb{R}^m$. Given as above $f : \mathbb{R}^m \to \mathbb{R}$ a C^1-function, $\delta_0 > 0$ and $1 > \alpha, \beta > 0$ arbitrary numbers, we define the function $\delta(f, \delta_0, x) : \mathbb{R}^m \to \mathbb{R}$ to be the largest number σ among $\{\beta^n\delta_0 : \ n = 0, 1, 2, \ldots\}$ for which

$$f(x - \sigma\nabla f(x)) - f(x) \leq -\alpha\sigma||\nabla f(x)||^2. \tag{7}$$

By Lemma 3.1 below, the function $\delta(f, \delta_0, x)$ is well-defined and is always positive. Moreover, this positivity is uniform on compact subsets of $\mathbb{R}^m$ if ∇f is locally Lipschitz ([3, 15, 16, 22]). For a given $z_0 \in \mathbb{R}^m$, we define the sequence

$$z_n = z_{n-1} - \delta(f, \delta_0, z_{n-1})\nabla f(z_{n-1}), \tag{8}$$

for $n = 1, 2, 3 \ldots$. This update rule is usually called Backtracking GD or Armijo's rule in the literature.

We recall that a point z^* is a *cluster point* of a sequence $\{z_n\}$ if there is a subsequence $\{z_{n_k}\}$ so that $\lim_{k\to\infty} z_{n_k} = z^*$. The sequence $\{z_n\}$ converges if and only if it has one and only one cluster point. It has been known that any cluster point of the sequence $\{z_n\}$ in the Backtracking GD is a critical point of f, see e.g. Proposition 1.2.1 in [8]. (We remark that [8] uses the terminology "limit point" instead of the more common "cluster point", which may cause some confusion occasionaly.) However, in previous work, starting from [3], convergence for Backtracking GD is proven only under strong assumptions similar to those in Theorem 2.12 below, in particular it is required that the function belongs to $C^{1,1}_L$. Our main theoretical result of this paper, stated next, extends this to a much more broad and realistic class of cost functions.

Theorem 2.1. *Let f be a C^1 function and let $\{z_n\}_{n=0,1,2,\ldots}$ be defined as in* (8).

(1) *Either* $\lim_{n\to\infty} ||z_n|| = \infty$ *or* $\lim_{n\to\infty} ||z_{n+1} - z_n|| = 0$.

(2) *Assume that f has at most countably many critical points. Then either* $\lim_{n\to\infty} ||z_n|| = \infty$ *or $\{z_n\}$ converges to a critical point of f.*

(3) *More generally, assume that all connected components of the set of critical points of f are compact. Then either* $\lim_{n\to\infty} ||z_n|| = \infty$ *or $\{z_n\}$ is bounded. Moreover, in the latter case the set of cluster points of $\{z_n\}$ is connected.*

A generalisation of Theorem 2.1, where we allow the vector v_n in $\delta_n v_n = z_{n+1} - z_n$ to not be parallel with $\nabla f(z_n)$ but only satisfies conditions (4) and (5), which are more relevant to the mini-batch practice, will be provided in Theorem 2.8. The proofs of the two theorems are the same. However, since the statement of Theorem 2.8 is much more complicated than that of Theorem 2.1, we choose to state Theorem 2.1 first for easy understanding.

Remark 2.2. Note that the proof of Theorem 2.1 needs only that $\delta(f, \delta_0, z) \leq \delta_0$ for all z, that any cluster point of $\{z_n\}$ is a critical point of f, and that Armijo's condition is satisfied (and not the specific definition of $\delta(f, \delta_0, z)$). Hence, the con-

clusion of the theorem holds in more general settings, for example under which Wolfe's conditions provide sequences $\{z_n\}$ for which $\{\nabla f(z_n)\}$ converges to 0 – for details see Subsection 2.3.

In the theorem, note that a general sequence $\{z_n\}$ may have both a convergent subsequence and another subsequence diverging to infinity. Hence, it can be regarded as a miracle that Backtracking GD guarantees the conclusions of the theorem. For example, for Standard GD, all these conclusions fail in general, see Subsection 2.3 for detail.

If f is a function without critical points, for example $f(t) = e^t$, then the sequence $\{z_n\}$ cannot have any cluster point inside $\mathbb{R}^n$. Therefore, it must diverge to infinity. The same argument applies more generally to f and z_0 for which the set $\{z \in \mathbb{R}^m : f(z) \leq f(z_0)\}$ contains no critical points of f. See [8] for relevant discussion.

The assumption in part 2 of Theorem 2.1 is satisfied by all Morse functions. We recall that a C^2 function f is *Morse* if all of its critical points are non-degenerate. This condition means that whenever $x^* \in \mathbb{R}^m$ is a critical point of f, then the Hessian matrix $\nabla^2 f(x^*) = (\partial^2 f/\partial x_i \partial x_j)_{i,j=1,\ldots,m}(x^*)$ is invertible. All critical points of a Morse function are isolated, and hence there are at most countably many of them.

Moreover, note that Morse functions are dense. In fact, given any C^2 function g, the function $f(x) = g(x) + \langle a, x\rangle$ is Morse for a outside a set of Lebesugue's measure 0, by Sard's lemma. Hence Morse functions are dense in the class of all functions. For example, $g(x) = x^3$ is not a Morse function, but $f(x) = g(x) + ax$ is Morse for all $a \neq 0$. More strongly, by using transversality results, it can be shown that the set of all Morse functions is preserved under small perturbations.

In case f is real analytic, then without the assumption that f has at most countably many critical points, [1] also showed that the sequence $\{z_n\}$ either diverges to infinity or converges. However, the real analytic assumption is quite restrictive. More generally, for cost functions satisfying the so-called Kurdyka-Lojasiewicz inequality, even non-smooth ones, the same conclusion can be obtained [6]. Our results can be regarded as complementary to these results. To illustrate this, we state here one result which is benefited by the combination of ideas. We refer the readers to [36, 38] for the definition of the algorithm Local Backtracking GD.

Theorem 2.3. *Let $f : \mathbb{R}^m \to \mathbb{R}$ be a C^1 function which satisfies the Kurdyka-Lojasiewicz gradient inequality. Let the hyperparameters in the Local Backtracking GD algorithm be randomly chosen, as well as the initial point x_0. Then for the sequence $\{x_n\}$ constructed by the Local Backtracking GD algorithm, there is a bifurcation: either $\{x_n\}$ diverges to infinity, or $\{x_n\}$ converges to a critical point of f which is moreover not a generalised saddle point.*

Proof. If the sequence $\{x_n\}$ does not diverge to infinity, then by [1, 6] it converges to a point x_∞. Then from [36, 38], it follows that x_∞ is a critical point of f and cannot be a generalised saddle point. □

In practice, knowing that the sequence in (8) converges, and that the limit point is moreover a local minimum point, is usually good enough. In fact, for linear networks, it is known that every local minimum is a global minimum [26]. In contrast to conventional wisdom derived from low dimensional intuition, a working hypothesis

in Deep Learning is that local minima with high error (that is, a minimum point at which the value of the function is too much bigger than the optimal value) are exponentially rare in higher dimensions [17, 35], and hence finding local minima is enough. Perturbation can help to escape from saddle points [21], and hence we expect that backtracking gradient descent for random mini-batches will also help to escape saddle points in training neural networks. See also Theorem 2.4 below.
In practical applications, we would like the sequence $\{z_n\}$ to converge to a minimum point. It has been shown in [17] via experiments that for cost functions appearing in DNN the ratio between minima and other types of critical points becomes exponentially small when the dimension m increases, which illustrates a theoretical result for generic functions [11]. Which leads to the question: Would in most cases GD converge to a minimum? We will see that because it is indeed a *descent* method, Backtracking GD answers the above question in affirmative in a certain sense. Before stating the precise result, we formalise the notion of "a critical point which is not a minimum".

Generalised saddle point. Let f be a C^1 function and let z_∞ be a critical point of f. Assume that f is C^2 near z_∞. We say that z_∞ is a *generalised saddle point* of f if the Hessian $\nabla^2 f(z_\infty)$ has at least one *negative* eigenvalue.

If z_∞ is a non-degenerate critical point of f, then it is a minimum if and only if all eigenvalues of the Hessian $\nabla^2 f(z_\infty)$ are positive. Hence in this case we see that a critical point of f is a minimum if and only if it is not a generalised saddle point. If $U \subset \mathbb{R}^m$ is an open set, we denote by $Vol(U)$ its Lebesgue measure. Also, if $\epsilon > 0$ and $z_\infty \in \mathbb{R}^m$, we denote by $B(z_\infty, \epsilon) = \{x \in \mathbb{R}^m : \ ||x - z_\infty|| < \epsilon\}$ the open ball of radius ϵ and centre z_∞. For any critical point z_∞ of f, we define:

$\mathcal{D}(z_\infty) = \{z_0 \in \mathbb{R}^m$: the sequence $\{z_n\}$ in (8) *does not contain* any subsequence converging to $z_\infty\}$.

Theorem 2.4. *Let f be a C^1 function. Assume that z_∞ is a generalised saddle point of f. For every $\epsilon > 0$, there is a non-empty open set $U(z_\infty, \epsilon) \subset \mathcal{D}(z_\infty) \cap B(z_\infty, \epsilon)$. Moreover, we have the following density 1 property:*

$$\lim_{\epsilon \to 0} \frac{Vol(U(z_\infty, \epsilon))}{Vol(B(z_\infty, \epsilon))} = 1.$$

Remark 2.5. (1) It is known that if the starting point z_0 of Backtracking GD is close enough to an isolated local minimum point, then the sequence $\{z_n\}$ will converge to that local minimum point. See e.g. Proposition 1.2.5 (Capture Theorem) in [8].

(2) Theorem 2.4 asserts roughly that if z_∞ is a cluster point of $\{z_n\}$, then it is very rare that z_∞ is a saddle point. In the proof of Theorem 2.4, the open sets $U(z_\infty, \epsilon)$ are chosen to be cones.

(3) It is observed that in practical problems in artificial neural networks, the cost function f has approximately the same value at all local minimum points. Hence, finding any local minimum is usually good enough for practical purposes. Moreover ([17]), for higher-dimensional functions we expect that saddle points are prevalent, and it is very difficult to escape saddle points. Hence, having a density 1 property as in Theorem 2.4 is good toward this concern.

2.2. Some generalisations of Theorem 2.1

The following is a modification of Theorem 2.1 which is more suitable to realistic applications.

Theorem 2.6. *Let $f : \mathbb{R}^m \to \mathbb{R}$ be a C^1 function which has at most countably many critical points. Let z_0 be a point in $\mathbb{R}^m$, $0 < \alpha, \beta < 1$ and $\delta_0 > 0$.*

(1) *Assume that ∇f is locally Lipschitz near every critical points of f. Let $\{z_n\}_{n\geq 0}$ be the sequence constructed from Backtracking GD update rule* (8)*. Then either $\lim_{n\to\infty} ||z_n|| = \infty$ or $\inf_{n\geq 0} \delta(f, \delta_0, z_n) > 0$.*

(2) *Conversely, let $\{\delta_n\}_{n\geq 0}$ be a sequence of real numbers in $(0, \delta_0)$ with $\inf_n \delta_n > 0$. Define the sequence $\{z_n\}$ by the update rule: $z_n = z_{n-1} - \delta_{n-1}\nabla f(z_{n-1})$ for $n \geq 1$. Assume moreover that for every $n \geq 1$ we have*

$$f(z_n) - f(z_{n-1}) \leq -\alpha\delta_{n-1}||\nabla f(z_{n-1})||^2.$$

Then either $\lim_{n\to\infty} ||z_n|| = \infty$ or $\{z_n\}$ converges to a critical point of f.

Proof. (1) Assume that there is a subsequence of $\{z_n\}$ which is bounded. Then from Theorem 2.1 we have that the sequence $\{z_n\}$ converges to a critical point z_∞ of f. By assumption on f, there is an open neighbourhood $B(z_\infty, r)$ of z_∞ on which ∇f is locally Lipschitz. Then it is well-known that $\inf_{z\in B(z_\infty,r)} \delta(f, \delta_0, z) > 0$, see the previous subsection. Choose N to be an integer such that $z_n \in B(z_\infty, r)$ for all $n \geq N$. We then have

$$\inf_{n=0,1,\dots} \delta(f, \delta_0, z_n) \geq \min\{\inf_{0\leq n\leq N} \delta(f, \delta_0, z_n), \inf_{z\in B(z_\infty,r)} \delta(f, \delta_0, z)\} > 0.$$

(2) It can be checked that under the assumption $\inf_{n=0,1,\dots} \delta_n > 0$, the proof of Proposition 1.2.1 in [8] goes through and shows that any cluster point of the sequence $\{z_n\}$ must be a critical point of f. Then from Remark 2.2 it follows that proofs of all parts of Theorem 2.1 go through and give us the desired result. □

Remark 2.7. Example 2.13 shows that the assumption on ∇f being locally Lipschitz near critical points of f is necessary for part 1 of Theorem 2.6 to hold.

Also, the assumption $\inf_n \delta_n > 0$ is needed in general for part 2 of Theorem 2.6 to hold. In fact, let $f(x) = x^2$ and choose any $z_0 > 0$. If we choose any sequence of positive numbers $\{\delta_n\}$ so that $\sum_{n=0}^{\infty} \delta_n$ is small enough (depending on z_0, α), then the sequence $\{z_n\}$ defined by the update rule $z_n = z_{n-1} - \delta_{n-1}f'(z_{n-1})$ for $n \geq 1$ will satisfy $f(z_n) - f(z_{n-1}) \leq -\alpha\delta_{n-1}|f'(z_{n-1})|^2$ for all $n \geq 1$ and $\lim_{n\to\infty} z_n = z_\infty$ exists, but this point z_∞ is > 0, and hence is not the unique critical point 0 of f.

The following generalisation of Theorem 2.1 is relevant to the practice of using mini-batches in DNN. It uses the inexact version of Backtracking GD which we now introduce.

Inexact Backtracking GD. Fix $A_1, A_2, \delta_0 > 0$, $1 \geq \mu > 0$ and $1 > \alpha, \beta > 0$. We start with a point z_0 and define the sequence $\{z_n\}$ by the following procedure. At step n:

(i) Choose a vector v_n satisfying

$$A_1\,||\nabla f(z_n)|| \leq ||v_n|| \leq A_2\,||\nabla f(z_n)|| \quad \text{and} \quad \langle \nabla f(z_n), v_n\rangle \geq \mu||\nabla f(z_n)|| \times ||v_n||.$$

(These are the same as conditions (4) and (5) for Inexact GD.)

(ii) Choose δ_n to be the largest number σ among $\{\delta_0, \delta_0\beta, \delta_0\beta^2, \ldots\}$ so that

$$f(z_n - \sigma v_n) - f(z_n) \leq -\alpha\sigma\langle f(z_n), v_n\rangle.$$

(This is Armijo's condition (6).)

(iii) Define $z_{n+1} = z_n - \delta_n v_n$.

As in Lemma 3.1, we can choose δ_n to be bounded on any compact set K on which ∇f is nowhere zero.

Theorem 2.8. *Let $f : \mathbb{R}^m \to \mathbb{R}$ be a C^1 function and let z_n be a sequence constructed from the Inexact Backtracking GD procedure.*

(1) *Either $\lim_{n\to\infty} ||z_n|| = \infty$ or $\lim_{n\to\infty} ||z_{n+1} - z_n|| = 0$.*

(2) *Assume that f has at most countably many critical points. Then either $\lim_{n\to\infty} ||z_n|| = \infty$ or $\{z_n\}$ converges to a critical point of f.*

(3) *More generally, assume that all connected components of the set of critical points of f are compact. Then either $\lim_{n\to\infty} ||z_n|| = \infty$ or $\{z_n\}$ is bounded. Moreover, in the latter case the set of cluster points of $\{z_n\}$ is connected.*

The proof of this theorem is identical to that of Theorem 2.1. Below are some other practical modifications.

Example 2.9. (Sequence of cost functions) Another generalisation of Theorem 2.1 is as follows. Let f_n converge uniformly on compact sets to f, so that ∇f_n also converges uniformly on compact sets to ∇f. Now consider the sequence $z_{n+1} = z_n - \delta_n \nabla f_n(z_n)$, where $\delta_n = \delta(f_n, \delta_0, z_n)$. Then all conclusions of Theorem 2.1 are satisfied.

Example 2.10. (Non-smooth functions) Analysing the proofs of the conclusions in Theorem 2.1, we can see that they do not need the assumption that $f(x)$ is C^1. For example, assuming only that f is directionally differentiable and $\nabla f(x)$ is locally bounded is enough. Even we may consider more general notions of derivative and boundedness, such as Lebesgue's integral and derivative, and essential maximum and minimum. The function $f(x) = |x|$ is not differentiable at 0, but we can check that the Fundamental Theorem of Calculus still applies by using Lebesgue's integral, and hence can check that Theorem 2.1 is valid also for this function.

Another direction to generalise Theorem 2.1 is to consider functions on open subsets Ω of $\mathbb{R}^m$. In this case, we have basically the same conclusion, if we either stop the iteration or choose a disturbance v_n of $\nabla f(z_n)$ so that $z_n - \delta_n v_n \in \Omega$ whenever the point z_n is on the boundary of Ω. Theorem 2.8 can be used to justify for this algorithm.

Example 2.11. (Regularisation) There is a common practice of using regularisation in machine learning, that is we consider instead of a cost function f, the compensated version $g(x) = f(x) + \lambda||x||^q$ (Lq regularisation), where $\lambda > 0$ and $q > 0$ are constants. It is observed that usually working with the regularisation cost function $g(x)$ gives better convergence and prevent overfitting (see e.g. [28]). Theoretically, regularisation is also good in light of Theorem 2.1 and the fact mentioned above that Morse functions are dense.

2.3. Comparison to previous work

In the influential paper [3], where condition (6) was introduced into GD, Armijo proved the existence of Standard GD and Backtracking GD for functions in $C^{1,1}_L$, under some further assumptions. The most general result which can be proven with his method is as follows (see e.g. Proposition 12.6.1 in [24]).

Theorem 2.12. *Assume that f is in $C^{1,1}_L$, f has compact sublevels (that is, all the sets $\{x : f(x) \leq b\}$ for $b \in \mathbb{R}$ are compact), and the set of critical points of f is bounded and has at most countably many elements. Let $\delta \leq 1/(2L)$. Then Standard GD converges to one critical point of f.*

An analog of Theorem 2.12 for gradient descent flow (solutions to $x'(t) = -\nabla f(x(t))$) is also known (see e.g. Appendix C.12 in [23]). Both the assumptions that f is in $C^{1,1}_L$ and δ is small enough are necessary for the conclusion of Theorem 2.12, as shown by the next two very simple examples.

Example 2.13. (When $\gamma = 1/2$, a similar example is given as Exercise 1.2.3 in [8], without proof. For the sake of completeness, here we provide a proof.) Let $0 < \gamma < 1$ be a rational number. Let $f : \mathbb{R} \to \mathbb{R}$ be the function $f(x) = |x|^{1+\gamma}$. Then it can be checked that f is in C^1 and $f'(x)$ is Hölder continuous with the Hölder exponent γ, that is $|f'(x) - f'(y)| \leq C|x - y|^\gamma$ for some constant $C > 0$ and for all $x, y \in \mathbb{R}$. Moreover, 0 is the only critical point of f (and is in fact a global minimum) and f has compact sublevels.

Now we show that for each sequence $\{\delta_n\}$ so that $\inf_n \delta_n > 0$, there is a countable set $A \subset \mathbb{R}$ (depending on $\{\delta_n\}$) such that for every $0 \neq z_0 \in \mathbb{R}\backslash A$, the sequence $\{z_n\}$ in (1) does not converge to 0. (This example can be applied to the Standard GD.)

We write $\gamma = p/q$ for relatively prime positive integers p and q. Note that for any $\delta, y \in \mathbb{R}$ there are at most $\max\{2p, 2q\}$ solutions x to $x - \delta f'(x) = y$. Therefore, since $\{\delta_n\}$ is a countable set, it follows that there is a countable set $A \subset \mathbb{R}$ so that: if $z_n = 0$ for some n then $z_0 \in A$. Now choose any $z_0 \in \mathbb{R}\backslash A$ so that z_n converges to 0, we claim that $\delta_1 := \inf_n \delta_n = 0$. Assume otherwise that $\delta_1 > 0$, we will obtain a contradiction as follows. Note that for all $x \neq 0$, we have $x.f'(x) > 0$ and $|f'(x)| = (1 + \gamma)|x|^\gamma$. Therefore, if $x \neq 0$ is close to 0 then $|f'(x)| >> |x|$. Thus, since $\lim_{n\to\infty} z_n = 0$ and $z_n \neq 0$ for all n (by the assumption on z_0), by discarding a finite number of points in the sequence $\{z_n\}$ if necessary, we can assume that

$$|z_{n+1}| = |z_n - \delta_n f'(z_n)| \geq \delta_1 |z_n|^\gamma / 2$$

for all $n = 0, 1, 2, \ldots$. Putting $\delta_2 = \delta_1/2$, by iterating the above inequality we obtain

$$|z_n| \geq \delta_2^{1+\gamma+\gamma^2+\ldots+\gamma^n} . |z_0|^{\gamma^n},$$

for all n. Taking limit when $n \to \infty$, we obtain

$$0 = \lim_{n\to\infty} |z_n| \geq \delta_2^{1/(1-\gamma)} > 0,$$

which is a contradiction. Therefore, $\inf_n \delta_n = 0$, as claimed.

In contrast, for the function in the above example, Backtracking GD converges to the global minimum 0 for every choice of initial point z_0.

Example 2.14. Let ϵ_0 be any positive number. Consider a smooth function $f : \mathbb{R} \to \mathbb{R}$ which on the set $|x| \geq \epsilon_0$ has the value $f(x) = |x|$ (note that the absolute value function $|x|$ is nowadays one of the most popular functions in artificial neural networks). We can easily construct such functions with the additional requirement that $f(x)$ has exactly one critical point $x = 0$ which is a global minimum point. Then the derivative $f'(x)$ satisfies: $f'(x) = 1$ if $x \geq \epsilon_0$ and $f'(x) = -1$ if $x \leq -\epsilon_0$. Let δ_0 be an arbitrary positive number $> 2\epsilon_0$, and z_0 be an arbitrary number satisfying $\epsilon_0 < z_0 < \delta_0 - \epsilon_0$. Then the sequence in (2) is periodic. In fact, since $z_0 > \epsilon_0$ we have $f'(z_0) = 1$, and hence $z_1 = z_0 - \delta_0 f'(z_0) = z_0 - \delta_0$. By the conditions on ϵ_0 and δ_0, we have $z_1 = z_0 - \delta_0 < -\epsilon_0$ and hence $f'(z_1) = -1$. Therefore, $z_2 = z_1 - \delta_0 f'(z_1)$ $= (z_0 - \delta_0) - \delta_0 \times (-1) = z_0$. Thus the sequence $\{z_n\}$ is periodic, and the cluster points of it are z_0 and z_1, neither of them is the unique critical point 0 of f.

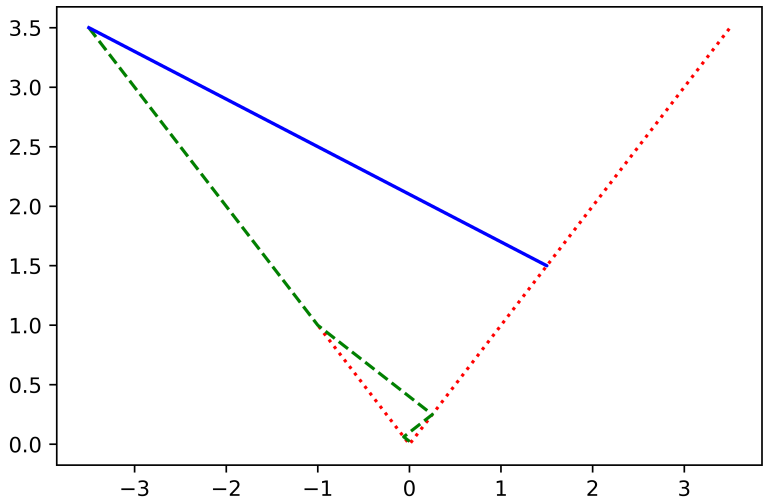


Figure 2.1: The behaviours of Standard GD and Backtracking GD for Example 2.14. Red dotted line: the function $f(x) = |x|$; blue solid line: Standard GD gets stuck when going back and forth between two points; green dashed line: Backtracking GD – which always makes sure $f(x - \delta(f, \delta_0, x)\nabla f(x)) \leq f(x)$ – works properly to reach the local minimum.

Concerning the issue of saddle points, we have the following very strong result for functions in class $C^{1,1}_L$ ([25, 30]).

Theorem 2.15. *Let f be in $C^{1,1}_L$ and $\delta < 1/L$. Then there exists a set $E \subset \mathbb{R}^k$ of Lebesgue measure 0 so that if $z_0 \in \mathbb{R}^k \backslash E$, then $\{z_n\}$ in Standard GD, if converges, will not converge to a saddle point.*

The main idea is that then the map $x \mapsto x - \delta\nabla f(x)$ is a diffeomorphism, and hence we can use the Stable-Center Manifold Theorem in dynamical systems (cited as Theorem 4.4 in [25]). For to deal with the case where the set of critical points of the function is uncountable, the new idea in [30] is to use Lindelöf's lemma that any open cover of an open subset of $\mathbb{R}^m$ has a countable subcover. In the above theorem, convergence of $\{z_n\}$ is important, otherwise one may not be able to apply

the Stable-Center manifold theorem. However, for convergence of $\{z_n\}$, one has to use Theorem 2.12, and needs to assume more, as seen from Example 2.13 above. Therefore, Proposition 4.9 in [25] is not valid as stated.

Theorem 2.4, while having weaker conclusion than that of Theorem 2.15, can be applied to all functions and hence can also be used to justify for the fact that GD in most of the case will contain only minima as cluster points. Also, we do not need to require that the sequence $\{z_n\}$ is convergent, in contrast to what mentioned in the previous paragraph for the results in [25]. The next example shows that the idea of using dynamical systems, as in [25], at current cannot be used for Backtracking GD.

Example 2.16. Let $\delta_0 > 0$ be one solution of the equation $p(t) = 6t^2 - 6t + 1 = 0$ (this equation has one positive solution since we have $p(1/2) = -1/2 < 0$ and $p(+\infty) = +\infty$). Let $f : \mathbb{R} \to \mathbb{R}$ be the function $f(x) = x^3$. In the Backtracking GD procedure, we choose $\alpha = 1/2$. Then the map $g(x) = x - \delta(f, \delta_0, x)f'(x)$ is not continuous at $x_0 = 1$. In fact, the choice of δ_0 implies that

$$f(1 - \delta_0 f'(1)) - f(1) = -\delta_0 |f'(1)|^2/2,$$

hence $\delta(f, \delta_0, 1) = \delta_0$. Since $f'(x)$ is not identically 0 in a neighbourhood of 1 and the function $\delta(f, \delta_0, x)$ takes values in a discrete set, if $g(x)$ were continuous at 1 as a function in x, we would have $\delta(f, \delta_0, x) = \delta_0$ for all x sufficiently close to 1. The latter means that for x sufficiently close to 1, we have

$$(x - 3\delta_0 x^2)^3 - x^3 + 9\delta_0 x^4/2 \leq 0.$$

Dividing by x^3, we see that $h(x) = (1-3\delta_0 x)^3 - 1 + 9\delta_0 x/2 \leq 0$ for all x close to 1. However, it can be seen that for the choice of δ_0, $h'(x) = 9\delta_0(3\delta_0 x - 1)^2 + 9\delta_0/2 > 0$, and hence there must be a sequence $x_n \to 1$ so that $h(x_n) > h(1) = 0$ for all n. Thus, for this sequence, $g(x_n)$ does not converge to $g(1)$, as claimed. □

A key point in the proofs of these papers, as well as of the paper [1], is that some estimates on the convergence rate for functions in $C^{1,1}_L$ can be explicitly obtained. In contrast, our proof of Theorem 2.1 is very indirect, since such estimates are not available for general C^1 functions. However, the proof of part 1 of Theorem 2.1 suggests that a reasonable criterion when working with a general function $f \in C^1$ is to stop the iteration when $\delta_n ||\nabla f(x_n)||$ is small enough. Note also that in practical Deep Learning, knowing the convergence rate is not that important as early stopping: by monitoring how well the model performs on a suitable validation set, to help avoid overfitting. Their method extends straightforwardly to those satisfying KL-condition.

There are other variants of GD which are regarded as state-of-the-art algorithms in DNN such as Momentum and Nesterov accelerated gradient, Adam, Adagrad, Adadelta, and RMSProp (see an overview in [33]). Some of these variants (such as Adagrad and Adadelta) allow choosing learning rates δ_n to decrease to 0 (inspired by Stochastic GD, see next paragraph) in some complicated manners which depend on the values of gradients at the previous points $x_0, \ldots, x_{n-1}$. However, as far as we know, convergence for such methods are not yet available beyond the usual setting such as in Theorem 2.12.

Stochastic GD is the default method used to justify the use of GD in DNN, which goes back to Robbins and Monro, see [9]. The most common version of it is to assume that we have a fixed cost function F (as in the deterministic case), but we replace the gradient $\nabla_\kappa F(\kappa_n)$ by a random vector v_n (here the random variables are points in the dataset, see also Inexact GD), and then show the convergence in probability of the sequence of values $F(\kappa_n)$ and of gradients $\nabla_\kappa F(\kappa_n)$ to 0 (in application the random vector v_n will be $\nabla_\kappa F_{I_n}(\kappa_n)$). However, the assumptions for these convergence results (for $F(\kappa_n)$ and $\nabla_\kappa F(\kappa_n)$) to be valid still require those in the usual setting as in Theorem 2.12. In the case where there is noise, the following additional conditions on the learning rates are needed [32]:

$$\sum_{n\geq 1} \delta_n = \infty, \ \sum_{n\geq 1} \delta_n^2 < \infty. \tag{9}$$

However, in Standard GD, which is a popular version in DNN, all the learning rates are the same and hence condition $\sum_{n\geq 1} \delta_n^2 < \infty$ is violated. Moreover, showing that the gradients $\nabla_\kappa F(\kappa_n)$ converge to 0 is far from proving the convergence of κ_n itself.

When using GD in DNN, even when the underlying function F is in $C_L^{1,1}$, it may be difficult to obtain a good lower bound estimate for the Lipschitz constant L, since these functions can have thousands of variables. Hence it can be difficult to obtain a good choice for the learning rate δ_0. The common practice in DNN is to manually fine-tune learning rates [28]: trial and error, do experiments and then observe and modify learning rates until achieving an acceptable descent of the cost (or loss) function. However, this practice is very time-consuming (especially when working with large datasets and/or complicated architectures) and depending too much on the researcher's experience. In contrast, Backtracking GD is automatic.

Remarks. Recently, there are some work estimating the global Lipschitz constant L for the gradient of cost functions coming from some specific DNN architectures, see [19, 34]. Besides what mentioned in the previous paragraphs, here we note a couple more remarks. First, even if ∇f is globally Lipschitz continuous with Lipschitz constant L, the local Lipschitz constant $L(x)$ for ∇f near a critical point x may be much smaller than L, and hence using the learning rate δ_0 in the order of $1/L$ may be too small compared to what allowed $1/L(x)$. To this end, we note that Backtracking GD can obtain good upper estimates for the local Lipschitz constant $L(x)$. Beyond this, recall from the introduction that a DNN is only an attempt to approximate an unknown function, and since the unknown function may not be in $C_L^{1,1}$, it is wise to work with as most general cost function as possible.

Wolfe's method. Since Wolfe's method is very close to Backtracking GD, we provide a more detailed comparison for it here. There are some abstract conditions on functions with only directional derivatives (see the definition about serious steps on page 228 in [41]), under which convergence results can be proven. Note, however, that the descent process proposed by Wolfe in [41] is more complicated than Backtracking GD, since at each iteration the learning rate δ_n – denoted t_n in his paper – and hence the point z_{n+1}, is to be chosen with respect to some of 5 choices listed in the definition on page 228 in that paper, not the unique one based on $||\nabla f(z_n)||$ alone as usually done in practice and considered in the current paper. We mention here in particular conditions (iii) and (iv) in [41]. Condition (iii) is exactly Armijo's

condition (6). Condition (iv) is that for a fixed $c_2 > 0$ there is ξ_n between z_n and z_{n+1} so that $\langle \nabla f(\xi_n), v_n\rangle \geq c_2\langle \nabla f(z_n), v_n\rangle$ *and* f is non-increasing from z_n to ξ_n.

Extracted from the above two conditions (iii) and (iv) in Wolfe's method are the following two conditions, usually called Wolfe's conditions

$$f(z_n - \delta_n v_n) - f(z_n) \leq -c_1\delta_n\langle \nabla f(z_n), v_n\rangle,$$
$$\langle \nabla f(z_n - \delta_n v_n), v_n\rangle \leq c_2\langle \nabla f(z_n), v_n\rangle,$$

for some fixed constants $1 > c_2 > c_1 > 0$. The first condition is exactly Armijo's theorem (condition (iii) in Wolfe's paper). The second condition is only a half of condition (iv) in Wolfe's method. It has been shown that if f is a C^1 function which is *bounded from below*, then a positive δ_n can be chosen to satisfy these Wolfe's conditions. Moreover, if f is in $C^{1,1}_L$ and v_n satisfies condition (i) in Inexact GD, then a result by G. Zoutendijk shows the convergence of $\nabla f(z_n)$ to 0. For more details, the readers can consult [29]. Therefore, under the assumptions mentioned above (that is f is in $C^{1,1}_L$, bounded from below, and v_n satisfies condition (i) in Inexact GD), by combining with Remark 2.2, we can prove conclusions of Theorem 2.1 with Armijo's rule replaced by Wolfe's conditions. Hence, we can see that the scope of application of Backtracking GD is wider than that of using Wolfe's conditions. Moreover, since Backtracking GD needs only Armijo's condition, while Wolfe's conditions require more, in general the learning rates determined by Backtracking GD will be not smaller than that determined from Wolfe's conditions, and hence intuitively will be better for convergence. In [5], it is shown that Wolfe's method fails for simple functions such as $a|x| + y$ for a large enough. See [37] where we discussed in some further details on this type of functions and other functions.

Other methods using Armijo's condition. Backtracking GD is probably the simplest but quite effective algorithm using Armijo's condition. Here we will discuss about some other methods. One is a proximal modification of Gauss-Seidel algorithm [7], which needs to solve an optimal subproblem that could be non-trivial. In the setting of flows (that is, using solutions to ODE), there is a so-called inertial method [6], which is based on second order ODE and is inspired by some physical laws. This method is shown, for Morse functions with *compact sublevels*, to converge to local minima for generic choices of initial conditions. (As mentioned, the counterpart of this result for Backtracking GD is given in [36, 38], where the compact sublevel assumption is not needed.) The well known gradient flow method (which is based on first order ODE), in contrast, is not guaranteed to avoid saddle points, and can be viewed as a limit of the inertial method. To be applicable to realistic applications, the inertial method needs to be implemented in the discrete setting. Such a discrete implementation was given in [12]. However, it seems that this implementation is quite sensitive to its hyperparameter (and hence needs careful manual fine tuning), and currently is only tested for a simple DNN (LeNet) where it does not yet achieve competitive performance. We will compare with our algorithms ([40, 39]) in Section 4.

3. Proofs of the main results

The next two lemmas are key steps to our main results.

Lemma 3.1. *Let f be a C^1 function. Then*

(1) *For any $x \in \mathbb{R}^m$, there is a positive integer n_0 for which the following is satisfied:*

$$f(x - \beta^{n_0}\delta_0 \nabla f(x)) - f(x) \leq -\beta^{n_0}\delta_0 \alpha ||\nabla f(x)||^2.$$

(2) *For any compact subset K of $\mathbb{R}^m$ with $\inf_{x\in K} ||\nabla f(x)|| > 0$, we have*

$$\inf_{x\in K} \delta(f, \delta_0, x) > 0.$$

Proof. We will give the proof for $\alpha = 1/2$. The other cases can be treated similarly.

(1) This is a simple consequence of Taylor's expansion for a continuously differentiable multivariable function. Below is the detail. We define a function $g : \mathbb{R} \to \mathbb{R}$ by the formula $g(t) = f(x - t\delta_0 \nabla f(x))$. Then g is continuously differentiable, and by the chain rule $g'(t) = -\delta_0 \nabla f(x - t\delta_0 \nabla_0 f(x)).\nabla f(x)$. Here we use the dot product between two vectors $\nabla f(x - t\delta_0 \nabla_0 f(x))$ and $\nabla f(x)$ in $\mathbb{R}^m$: if $u = (u_1, u_2, \ldots, u_n)$ and $v = (v_1, v_2, \ldots, v_n)$ then $u.v = u_1 v_1 + \ldots + u_n v_n$. The Fundamental Theorem of Calculus, $g(1) - g(0) = \int_t^1 g'(s)ds$, can be explicitly written in this case as follows:

$$f(x - t\delta_0 \nabla f(x)) - f(x) = -\delta_0 \int_0^t \nabla f(x - s\delta_0 \nabla_0 f(x)).\nabla f(x) ds. \tag{10}$$

If $\nabla f(x) = 0$, then we can choose simply $n_0 = 0$. Hence, we can assume for the remaining of the proof that $\nabla f(x) \neq 0$, and hence $||\nabla f(x)|| > 0$. Because f is continuously differentiable, for the positive number $\epsilon_0 = 1/2||\nabla f(x)||^2$, there is a $\gamma_0 > 0$ so that if $t_0 > 0$ is such that $t_0||\delta_0 \nabla f(x)|| < \gamma_0$ then

$$||\nabla f(x - s\delta_0 \nabla_0 f(x)) - \nabla f(x)|| \leq \epsilon_0 = 1/2||\nabla f(x)||,$$

for all $0 \leq s \leq t_0$. By the triangular inequality, we have the following simple estimate for the integrand in (10)

$$\begin{aligned}\nabla f(x - s\delta_0 \nabla_0 f(x)).\nabla f(x) &= (\nabla f(x - s\delta_0 \nabla_0 f(x)) - \nabla f(x)).\nabla f(x) + ||\nabla f(x)||^2 \\ &\geq ||\nabla f(x)||^2 - ||\nabla f(x - s\delta_0 \nabla_0 f(x)) - \nabla f(x)|| \times ||\nabla f(x)||.\end{aligned}$$

Hence, with this choice of t_0, for all $0 \leq t \leq t_0$ we have from (8) that

$$f(x - t\delta_0 \nabla f(x)) - f(x) \leq -\frac{t\delta_0}{2}||\nabla f(x)||^2.$$

We can always find a positive integer n_0 so that $t = \beta^{n_0}$ satisfies the needed condition, and for this choice of t we obtain the conclusion of the lemma.

(2) In the proof of (1), we choose $\epsilon_0 = \inf_{x\in K} ||\nabla f(x)||/2$. Then by the assumption in (1), we have that $\epsilon_0 > 0$. Since $\nabla f(x)$ is uniformly continuous on compact subsets of $\mathbb{R}^m$, there is a $\gamma_0 > 0$ for which whenever

$$\sup_{x\in K} t_0||\delta_0 \nabla f(x)|| \leq \gamma_0,$$

then for all $0 \leq s \leq t_0$ we have

$$\sup_{x\in K} ||\nabla f(x - s\delta_0 \nabla_0 f(x)) - \nabla f(x)|| \leq \epsilon_0 = 1/2 \inf_{x\in K} ||\nabla f(x)||.$$

We can then repeat the remaining of the proof of (1). □

The next lemma concerns a simple property of trigonometric functions. We recall that the function arccos : $[0,1] \to [0,\pi/2]$ is defined so that $\arccos(u) = v$ iff $u = \cos(v)$. For $x = (x_1, \ldots, x_m), y = (y_1, \ldots, y_m) \in \mathbb{R}^m$, we define

$$dist(x,y) = \arccos\left(\frac{|1 + \sum_{i=1}^m x_i y_i|}{\sqrt{1 + \sum_{i=1}^m x_i^2}\sqrt{1 + \sum_{i=1}^m y_i^2}}\right). \tag{11}$$

Recall that by the Cauchy-Schwarz inequality we always have

$$0 \leq \frac{|1 + \sum_{i=1}^m x_i y_i|}{\sqrt{1 + \sum_{i=1}^m x_i^2}\sqrt{1 + \sum_{i=1}^m y_i^2}} \leq 1,$$

and hence the function $dist(x,y)$ is well-defined. As used throughout the whole paper, $||x-y|| = \sqrt{\sum_{i=1}^m |x_i - y_i|^2}$ is the usual Euclidean distance on $\mathbb{R}^m$. We have the following result.

Lemma 3.2. *There is a constant $C > 0$ so that for all $x, y \in \mathbb{R}^m$ we have*

$$C||x-y|| \geq dist(x,y).$$

Proof. We will choose $C \geq \pi$. Therefore, if $||x-y|| \geq 1$ then there is nothing to prove, since the range of the arccos function is $[0,\pi/2]$. For $0 \leq \epsilon \leq 1$, we define

$$h(\epsilon) = \inf_{x,y \in \mathbb{R}^m:\ ||x-y|| \leq \epsilon} \frac{|1 + \sum_{i=1}^m x_i y_i|}{\sqrt{1 + \sum_{i=1}^m x_i^2}\sqrt{1 + \sum_{i=1}^m y_i^2}}.$$

By Cauchy-Schwarz' inequality $1 + (a+b)/2 \geq \sqrt{1+a}\sqrt{1+b}$ (which the readers can easily check by squaring the two sides and simplifying), for $a = \sum_{i=1}^m x_i^2$ and $b = \sum_{i=1}^m y_i^2$, we have

$$\frac{1}{\sqrt{1 + \sum_{i=1}^m x_i^2}\sqrt{1 + \sum_{i=1}^m y_i^2}} \geq \frac{1}{1 + (\sum_{i=1}^m x_i^2 + \sum_{i=1}^m y_i^2)/2}.$$

By assumption, $0 \leq \epsilon \leq 1$, and hence if $||x-y|| \leq \epsilon \leq 1$ we have by Cauchy-Schwarz' inequality

$$1 + \sum_{i=1}^m x_i y_i = 1 + \sum_{i=1}^m x_i^2 + \sum_{i=1}^m x_i(y_i - x_i)$$

$$\geq 1 + \sum_{i=1}^m x_i^2 - \sqrt{\sum_{i=1}^m x_i^2}\sqrt{\sum_{i=1}^m (x_i - y_i)^2} \geq 1 + \sum_{i=1}^m x_i^2 - \sqrt{\sum_{i=1}^m x_i^2} \geq 0.$$

Therefore, under the same assumption on x, y and ϵ:

$$\frac{|1 + \sum_{i=1}^m x_i y_i|}{\sqrt{1 + \sum_{i=1}^m x_i^2}\sqrt{1 + \sum_{i=1}^m y_i^2}} = \frac{1 + \sum_{i=1}^m x_i y_i}{\sqrt{1 + \sum_{i=1}^m x_i^2}\sqrt{1 + \sum_{i=1}^m y_i^2}}$$

$$\geq \frac{1 + \sum_{i=1}^m x_i y_i}{1 + (\sum_{i=1}^m x_i^2 + \sum_{i=1}^m y_i^2)/2} = 1 - \frac{1}{2}\frac{||x-y||^2}{1 + (\sum_{i=1}^m x_i^2 + \sum_{i=1^m y_i^2})/2}$$

$$\geq 1 - \frac{1}{2}||x-y||^2 \geq 1 - \frac{1}{2}\epsilon^2.$$

Therefore, $h(\epsilon) \geq 1 - \epsilon^2/2$. Using that the arccos function is decreasing on its domain of definition, we then have for $||x - y|| \leq \epsilon \leq 1$:

$$\begin{aligned} dist(x,y) &= \arccos\Big(\frac{|1 + \sum_{i=1}^m x_i y_i|}{\sqrt{1 + \sum_{i=1}^m x_i^2}\sqrt{1 + \sum_{i=1}^m y_i^2}}\Big) \\ &\leq \arccos(h(\epsilon)) \leq \arccos(1 - \epsilon^2/2). \end{aligned}$$

Now, using the classical inequality that $\cos(\pi\epsilon) \leq 1 - \epsilon^2/2$ if $\epsilon \leq \epsilon_0$ for a small enough $\epsilon_0 > 0$ (explicitly determined), we have that provided $||x - y|| \leq \epsilon_0$ then

$$dist(x,y) \leq \pi||x - y||.$$

Therefore, if we choose $C = \pi/\epsilon_0$, we obtain for all $x, y \in \mathbb{R}^m$:

$$dist(x,y) \leq C||x - y||,$$

as desired. □

We will need the notation of a compact metric space and in particular the real projective space $\mathbb{PR}^m$ which we now briefly recall for the readers' convenience.

Definition 3.3. (Compact metric spaces and real projective spaces) A *metric space* (X, d) is a set X equipped with a distance $d : X \times X \to [0, \infty)$, with the following three properties: (i) $d(x, y) = 0$ iff $x = y$, (ii) (symmetry) $d(x, y) = d(y, x)$, and (iii) (triangle inequality) $d(x, y) + d(y, z) \geq d(x, z)$ for all $x, y, z \in X$. A sequence $\{x_n\}$ is said to *converge* to x in (X, d) if $\lim_{n\to\infty} d(x_n, x) = 0$. A metric space (X, d) is *compact* if any sequence $\{x_n\}$ has a convergent subsequence. Note that the usual Euclidean space $(X = \mathbb{R}^m, d = ||.||)$ is a metric space but is *not* compact. However, we can define a compact metric space $(\mathbb{PR}^m, d)$ – called the *real projective space* of dimension m – with the following three properties: (i) $\mathbb{PR}^m$ contains $\mathbb{R}^m$ as a set, (ii) For $x, y \in \mathbb{R}^m$ the distance $d(x, y)$ is exactly the function $dist(x, y)$ in (11), and (iii) if $\{x_n\} \subset \mathbb{R}^m$ converges in $(\mathbb{PR}^m, d)$ to a point $z \in \mathbb{PR}^m \backslash \mathbb{R}^m$, then $\lim_{n\to\infty} ||x_n|| = \infty$.

We are now ready to prove the main results of this paper.

Proof of Theorem 2.1. For simplicity, we give the proof for $\alpha = 1/2$ only. The other cases can be treated similarly.

1) By construction, we have

$$f(z_{n+1}) - f(z_n) \leq -||\nabla f(z_n)|| \times ||z_{n+1} - z_n||/2,$$

and hence (by multiplying both sides with -1) for all n:

$$f(z_n) - f(z_{n+1}) \geq ||\nabla f(z_n)|| \times ||z_{n+1} - z_n||/2.$$

Since $\{f(z_n)\}$ is a decreasing sequence, we have two cases to consider.

Case 1: $\lim_{n\to\infty} f(z_n) = -\infty$. In this case, it follows easily (since f is a continuous function, and hence is bounded on any compact set) that $\lim_{n\to\infty} ||z_n|| = \infty$.

Case 2: $\{f(z_n)\}$ is bounded. Then $\lim_{n\to\infty}(f(z_n) - f(z_{n+1})) = 0$.

Fix $\epsilon > 0$ a small number. We partition $\{0, 1, 2, 3, \ldots\}$ into 2 sets:

$$A(\epsilon) = \{n : \ \delta(f, \delta_0, z_n)||\nabla f(z_n)|| \leq \epsilon\}; \quad B(\epsilon) \ = \ \{0, 1, 2, \ldots\}\backslash A(\epsilon).$$

For $n \in A(\epsilon)$, we have from (8) that $||z_{n+1} - z_n|| = \delta(f, \delta_0, z_n)||\nabla f(z_n)|| \leq \epsilon$.

For $n \in B(\epsilon)$ we have $||\nabla f(z_n)|| \geq \dfrac{\epsilon}{\delta(f, \delta_0, z_n)} \geq \dfrac{\epsilon}{\delta_0}$.

Now we can choose $n(\epsilon) > 0$ so that for $n \geq n(\epsilon)$ then

$$0 \leq f(z_n) - f(z_{n+1}) \leq \epsilon^2.$$

Combining the above inequalities, we obtain, for $n \geq n(\epsilon)$ and $n \in B(\epsilon)$

$$\epsilon^2 \geq f(z_n) - f(z_{n+1}) \geq \delta(f, \delta_0, z_n)||\nabla f(z_n)||^2/2$$
$$= ||z_{n+1} - z_n|| \times ||\nabla f(z_n)||/2 \geq \frac{\epsilon}{2\delta_0}||z_{n+1} - z_n||.$$

Therefore, $2\delta_0\epsilon \geq ||z_{n+1} - z_n||$ if $n \in B(\epsilon)$ and $n \geq n(\epsilon)$. Therefore, for all $n \geq n(\epsilon)$ we have $||z_{n+1} - z_n|| \leq \max\{\epsilon, 2\delta_0\epsilon\}$. Since $\epsilon > 0$ is arbitrary, it follows that in Case 2 we have $\lim_{n\to\infty} ||z_{n+1} - z_n|| = 0$.

(2) We let $(\mathbb{PR}^m, d)$ be the real projective space of dimension m which we introduced in the front of the proof of Theorem 2.1. Let $\{z_n\}$ be the sequence from (8), with an arbitrary initial point $z_0 \in \mathbb{R}^m$ and an arbitrary choice of δ_0.

We have two cases to consider.

Case 1: $\lim_{n\to\infty} f(z_n) = -\infty$. In this case, we then have $\lim_{n\to\infty} ||z_n|| = \infty$.

Case 2: The remaining case where $\{f(z_n)\}$ is bounded. In this case, by part 1 above we have $\lim_{n\to\infty} ||z_{n+1} - z_n|| = 0$. By Lemma 3.2 and the properties of the real projective space mentioned in the front of the proof of Theorem 2.1, we have $d(z_{n+1}, z_n) \leq C||z_{n+1} - z_n||$ for all n, where $C > 0$ is a constant. In particular, we also have $\lim_{n\to\infty} d(z_{n+1}, z_n) = 0$.

Let D be the cluster set of the sequence $\{z_n\}$ in the usual Euclidean space $(\mathbb{R}^m, ||.||)$, and let D' be the cluster set of the sequence $\{z_n\}$ in the real projective space $(\mathbb{PR}^m, d)$. While the metrics $||.||$ and d are different on $\mathbb{R}^m$, they induce the same topology on $\mathbb{R}^m$. Therefore, from elementary point set topology, we obtain that D' is equal to the closure $\overline{D}$ of D in $(\mathbb{PR}^m, d)$, and $D = D' \cap \mathbb{R}^m$.

Because in Case 2 we showed above that $\lim_{n\to\infty} d(z_{n+1}, z_n) = 0$, we can apply results in [4] to the compact metric space $(\mathbb{PR}^m, d)$ to obtain that D' is connected. Since D must be contained in the set of critical points of f, it is at most countable by the assumption. Then from elementary point set topology we have the following conclusion:

(i) Either $D = \emptyset$, thus all cluster points of $\{z_n\}$ are contained in $\mathbb{PR}^m\backslash\mathbb{R}^m$, and hence $\lim_{n\to\infty} ||z_n|| = \infty$, or

(ii) $D = D' = 1$ point, that is $\{z_n\}$ converges to a unique point $z_\infty \in \mathbb{R}^m$.

(3) The proof is similar to that of part 2 above. □

Next we give the proof of Theorem 2.4.

Proof of Theorem 2.4. As above, we will treat only the case $\alpha = 1/2$ here. We may assume that $z_\infty = 0$ and $f(0) = 0$. Since $z_\infty = 0$ is a critical point of f, we have $\nabla f(0) = 0$. Hence, for $z \in B(0, \epsilon_0)$, where $\epsilon_0 > 0$ small enough, and for $0 < \delta < \delta_0$, we get by Taylor's expansion

$$\begin{aligned}\nabla f(z) &= \nabla f(0).z + \int_0^1 \nabla^2 f(tz).zdt \\ &= \int_0^1 \nabla^2 f(0).zdt + \int_0^1 [\nabla^2 f(tz) - \nabla^2 f(0)].zdt = \nabla^2 f(0).z + \gamma_1(z).||z||,\end{aligned}$$

where $\lim_{z\to 0} \gamma_1(z) = 0$. Using the same argument we obtain

$$\begin{aligned}f(z - \delta\nabla f(z)) &= f(0) + \nabla f(0).(z - \delta\nabla f(z)) \\ &\quad + \int_0^1 \nabla^2 f(t(z - \delta\nabla f(z))).(z - \delta\nabla f(z)).(z - \delta\nabla f(z))dt \\ &= \nabla^2 f(0).(z - \nabla^2 f(0).z).(z - \nabla^2 f(z_0).z) + \gamma_2(z)||z||^2,\end{aligned}$$

where $\lim_{z\to 0} \gamma_2(z) = 0$.

Since $\nabla^2 f(0)$ is a symmetric real square matrix (hence is diagonalisable) and with at least one negative eigenvalue, by Silvester's law of inertia we may assume that in $B(0, \epsilon)$ the quadratic form $\nabla^2 f(0).z.z$, where $z = (z_1, \ldots, z_k)$ has the form (where $z_1, \ldots, z_j$ correspond to eigenvectors of positive eigenvalues, $z_{j+1}, \ldots, z_m$ correspond to eigenvectors of zero eigenvalue, and $z_{m+1}, \ldots, z_k$ correspond to eigenvectors of positive eigenvalues of $\nabla^2 f(0)$, and hence we can – after applying a linear change of coordinates – assume that the eigenvalues of $\nabla^2 f(0)$ are ± 1 and 0)

$$\nabla^2 f(0).z.z = (z_1^2 + \ldots + z_j^2) - (z_{m+1}^2 + \ldots + z_k^2),$$

where $m \leq k - 1$. From the above calculations, we obtain directly, noting that $\nabla^2 f(0).z = (z_1, \ldots, z_m, -z_{m+1}, \ldots, -z_k)$:

$$\begin{aligned}f(z - \delta\nabla f(z)) &= \nabla^2 f(0).(z - \nabla^2 f(0).z).(z - \nabla^2 f(0).z) + \gamma_2(z)||z||^2 \\ &= -4(z_{m+1}^2 + \ldots + z_k^2) + \gamma_2(z)(z_1^2 + \ldots + z_k^2).\end{aligned}$$

Therefore, provided

$$z = (z_1, \ldots, z_k) \in B(0, \epsilon_0) \text{ and } 4(z_{m+1}^2 + \ldots + z_k^2) > |\gamma_2(z)|(z_1^2 + \ldots + z_k^2),$$

then for the sequence $\{z_n\}$ in equation (8) with the initial value $z_0 = z$ we have $f(z_n) \leq f(z_1) < 0$ for all $n \geq 1$. Hence $\{z_n\}$ does not contain any subsequence converging to $z_\infty = 0$. Therefore, such a z belongs to $\mathcal{D}(z_\infty)$.

If we define for any $0 < \epsilon \leq \epsilon_0$ the number $\gamma_\epsilon = \sup_{z\in B(0,\epsilon)} |\gamma_2(z)|$, then $\lim_{\epsilon\to 0} \gamma_\epsilon = 0$. Moreover, the open set

$$U(z_\infty, \epsilon) = \{z = (z_1, \ldots, z_k) \in B(z_\infty, \epsilon) : \ 4(z_{m+1}^2 + \ldots + z_k^2) > \gamma_\epsilon(z_1^2 + \ldots + z_k^2)\}$$

belongs to $\mathcal{D}(z_\infty)$, and

$$\lim_{\epsilon\to 0} \frac{Vol(U(z_\infty, \epsilon))}{Vol(B(z_\infty, \epsilon))} = 1.$$

Therefore the proof of the theorem is complete. □

4. Efficiency of Backtracking GD: Some experimental results

Based on the theoretical results in the previous sections, some practical implementations of Backtracking GD for DNN have been developed in [40, 39]. Here we briefly describe these algorithms, and then present some experimental results for the CIFAR10 dataset on the networks Resnet18 and LeNet.

To satisfy Armijo's condition is expensive, in particular in Large scale optimisation as in DNN. The point is that one needs to run a finite, but generally unknown in advance, while loop of function evaluations to find the right learning rate which satisfies Armijo's condition. To reduce the need of doing function evaluations, [40, 39] proposes a variation of Backtracking GD, named Two-way Backtracking GD which roughly works as follows. At step n, instead of starting the search for learning rate δ_n from δ_0, one starts from $\delta = \delta_{n-1}$. A difference from the usual procedure for Backtracking GD is that in this case if δ *satisfies* Armijo's condition then we *increase* δ to δ/β, and repeat until either $\delta > \delta_0$ or δ does not satisfy Armijo's condition anymore. Only if $\delta = \delta_{n-1}$ does not satisfy Armijo's condition that we will decrease δ to $\beta\delta$ as in the usual scheme. This Two-way Backtracking GD has good theoretical guarantee, under quite general settings. An implementation of this Two-way Backtracking GD on DNN is given in [40, 39] under the name MBT-GD, which is very close to Two-way Backtracking GD and which mixes with Standard GD for further time saving. It is observed that MBT-GD works very stably across various mini-batch sizes (both small and large ones) and stable with the hyperparameters (that is, the starting learning rate δ_0 and the constants α, β in Armijo's conditions).

Besides Two-way Backtracking GD, it is also proposed in [40, 39] combinations between (Two-way) Backtracking GD and NAG or Momentum (called Backtracking NAG and Backtracking MMT), to boost the performance of the latter two methods. Theoretical guarantee of these combinations are given under quite general settings. Implementations of these combinations on DNN are given in [40, 39], under the names MBT-NAG and MBT-MMT. Note that MBT-NAG and MBT Momentum are not as close to Backtracking NAG and Backtracking Momentum as MBT-GD to Two-way Backtracking GD. Roughly speaking, in the original MMT and NAG algorithms, there are two hyperparameters: learning rate (δ in our terminology, but usually is also denoted by η) and momentum γ, both are constants. In MBT-NAG and MBT-MMT, we use the same momentum constant as in these original papers, but then vary learning rate δ adaptively, using Armijo's condition. The main idea is that if the point is close to a critical point, then the contribution of γ (which is multiplied to the gradient) is not decisive, while the learning rate should be chosen by Armijo's condition to assure sufficient descent. We observe that MBT-NAG and MBT-MMT work stably with respect to the hyperparameters coming from (Two-way) Backtracking GD (such as δ_0, α and β), while may be sensitive to the momentum hyperparameter γ. On the other hand, we expect that using the Backtracking NAG and Backtracking MMT algorithms (mentioned above), the obtained results will be less sensitive even with respect to the hyperparameter γ. However, since these latter algorithms are more expensive and challenging to implement, and also since MBT-MMT and MBT-NAG work very well if an appropriate value for momentum is chosen, we defer such an implementation of Backtracking NAG and Backtracking MMT to a future work. A good rule is that we should choose γ small

if the mini-batch size is small, while for a larger mini-batch size the momentum can be chosen larger correspondingly. One possible explanation is that the cost function coming from a larger mini-batch is more resemble of the cost function of the full batch than that coming from a smaller mini-batch.

For more details about MBT-GD, MBT-NAG and MBT-Momentum, please see [39, 40] for description and the GitHub link [42] for source codes. Below, we report experimental results for the dataset CIFAR10 on two DNNs: Resnet18 and LeNet. Experiments are run on a Tesla P-100 GPU or equivalent.

4.1. Experimental results for CIFAR10 on Resnet18

Here we report on experiments with CIFAR10 on Resnet18, with mini-batch size 200. The complexity of this model is appropriate for the dataset CIFAR10. Table 4.1 is taken from [40, 39], while Figures 4.1a and 4.1b are produced in cooperation with Torus Actions SAS. Since the mini-batch size 200 is large, we choose a large momentum $\gamma = 0.9$.

Learning rates	100	10	1	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}
SGD	10.00	89.47	91.14	*92.07*	89.83	84.70	54.41	28.35	10.00
MMT	10.00	10.00	10.00	*92.28*	91.43	90.21	85.00	54.12	28.12
NAG	10.00	10.00	10.00	*92.41*	91.74	89.86	85.03	54.37	28.04
Adagrad	10.01	81.48	90.61	88.68	*91.66*	86.72	54.66	28.64	10.00
Adadelta	91.07	92.05	*92.36*	91.83	87.59	73.05	46.46	22.39	10.00
RMSprop	10.19	10.00	10.22	89.95	91.12	*91.81*	91.47	85.19	65.87
Adam	10.00	10.00	10.00	90.69	90.62	*92.29*	91.33	85.14	66.26
Adamax	10.01	10.01	91.27	91.81	*92.26*	91.99	89.23	79.65	55.48
MBT-GD					*91.64*				
MBT-MMT					*93.70*				
MBT-NAG					**93.85**				

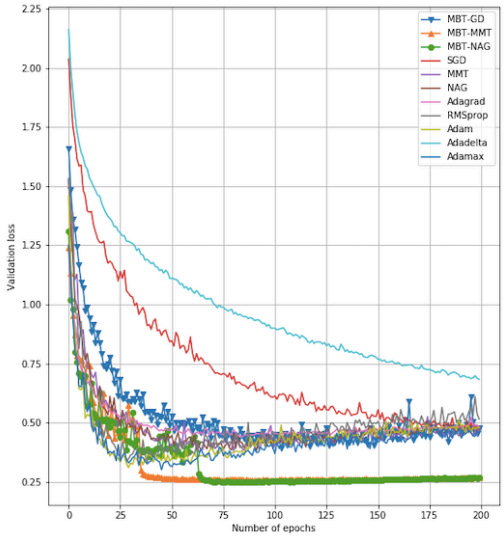
Table 4.1: Best validation accuracy for CIFAR10 on Resnet18 after 200 training epochs (batch size 200) of different optimisers using different starting learning rates (MBT methods, being stable with starting learning rate, only use starting learning rate 10^{-2} as default).

4.2. Experimental results for CIFAR10 on LeNet

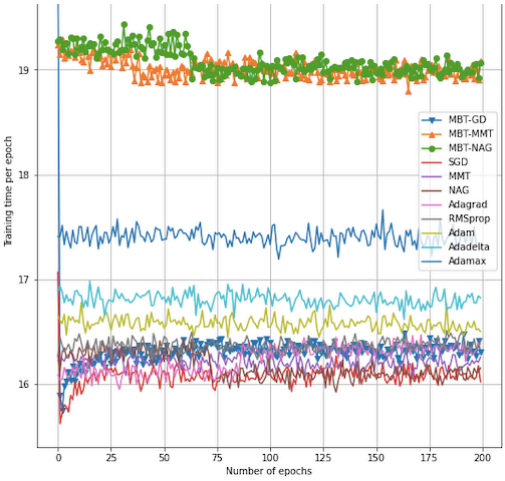
The inertial method [7] has been implemented in [12]. There, experiments on CIFAR10, but on a simpler DNN (LeNet) and with small mini-batch size 32, have been reported. Inertial (Newton) method has been compared against SGD, Adam and Adagrad, with the following best validation accuracy (among 200 epochs): SGD $\sim$ 59%, Adam $\sim$ 61%, Adagrad $\sim$ 63%, and Inertial method $\sim$ 57%. These are obtained using a *learning rate decay schedule*. Hence, it can be seen that this model is not suitable enough for the dataset CIFAR10.

Below we report on the performance of our Backtracking methods MBT-GD, MBT-NAG and MBT-Momentum, for CIFAR10 with LeNet, for the same choice of mini-batch size 32 and the normalisation as in [12]. The performance of all 3 methods are stable when the hyperparameters δ_0 and α in Backtracking GD are changed. In the results reported below, we use the values $\delta_0 = 1$ or 100, and $\alpha = 1e-4$ or $1e-1$. In this case, a larger value of momentum $\gamma = 0.9$ does not provide good and stable results for both MMT, NAG and their MBT- versions, while a smaller value

$\gamma = 0.5$ produces good performance. The reason could be, as mentioned above, that the setting here (LeNet, mini-batch size 32) is not good enough to model CIFAR10. Recall that this hyperparameter γ is not from Armijo's condition, but is special to Momentum and NAG methods.



(a) Validation loss for different algorithms, for CIFAR10 on Resnet18.



(b) Training time (in seconds) per epoch for different algorithms, for CIFAR10 on Resnet18.

Figure 4.1: The actual training time from scratch for SGD, MMT, NAG, Adagrad, RMSProp, Adam, Adadelta and Adamax must be a high multiple of what reported here, in 4.1b, since these methods need manual fine-tune of hyperparameters to achieve good performance.

For each experiment, we will report Learning rate and Training time (seconds/epoch), and the Best validation accuracy among 200 epochs. To compare, we also report the performance of the original non-Backtracking methods. For the non-Backtracking methods, we fix the learning rate throughout 200 epochs, and we choose different learning rates from a grid search to observe. Tables 4.2 and 4.3 are based on average performance of 5 random runs. Figure 4.2 is chosen from the best run among 5 random runs.

Method	ValidBest
MBT-SGD	62.27
MBT-MMT	56.816
MBT-NAG	58.064

Table 4.2: Average values (from 5 random runs) of Best Validation accuracy among 200 epochs (ValidBest, in %) obtained by Backtracking methods, for CIFAR10 on LeNet after 200 training epochs (mini-batch size 32, normalisation setting as in [12]). For MBT-MMT and MBT-NAG, the hyperparameter momentum is chosen to be $\gamma = 0.5$. Training time (seconds/epoch): for MBT-SGD $\sim$ 14.5928, for MBT-MMT $\sim$ 12.806, and for MBT-NAG $\sim$ 12.886. Initial learning rate is 1. Learning rate at epoch 200: for MBT-SGD $\sim$ 0.0007, for MBT-MMT $\sim$ 0.00051, and for MBT-NAG $\sim$ 0.00068.

LR/Method	SGD	MMT	NAG
1, ValidBest	50.37	20.724	19.802
1e-1, ValidBest	62.886	63.436	61.846
1e-2, ValidBest	64.214	64.288	64.196
1e-3, ValidBest	49.034	57.86	57.794
1e-4, ValidBest	12.32	16.912	17.18
1e-5, ValidBest	10.062	10.162	10.162
1e-6, ValidBest	9.996	10.004	10.004

Table 4.3: Average values (from 5 random runs) of Best Validation accuracy among 200 epochs (ValidBest, in %)s, for CIFAR10 on LeNet after 200 training epochs (mini-batch size 32), across different learning rates (LR) in a grid search, for Non-Backtracking methods. Normalisation setting is as in [12]. For MMT and NAG, the hyperparameter momentum is chosen to be $\gamma = 0.5$. Average training time for all methods is about 8.4 seconds/epoch.

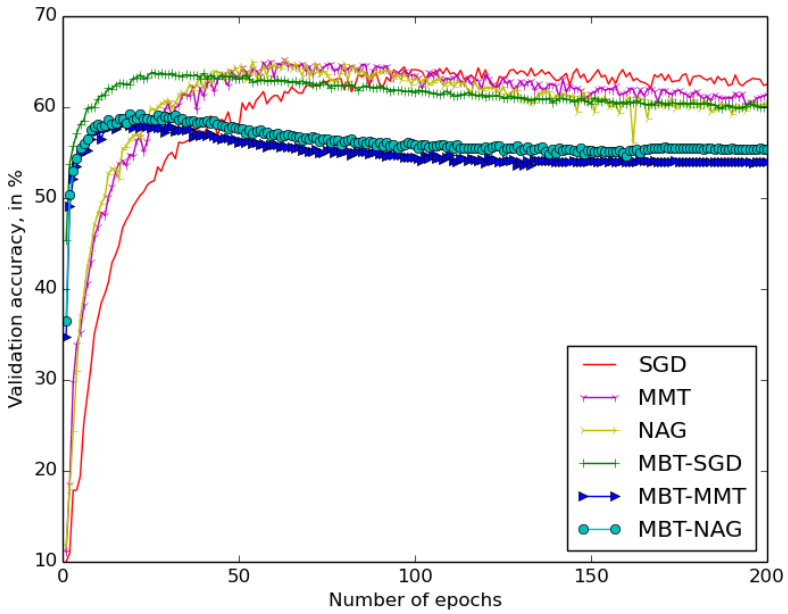


Figure 4.2: The evolution of validation accuracy in a training run, for both Non-Backtracking methods (SGD, MMT and NAG) and the corresponding Backtracking versions (MBT-SGD, MBT-MMT and MBT-NAG). For each method, we choose the best run among 5 random runs to report. The learning rate for Non-Backtracking methods is fixed to be $1e-2$. The initial learning rate for Backtracking methods is 1. Readers can consult Tables 4.2 and 4.3 for more insights. The momentum hyperparameter, for (MBT-)MMT and (MBT-)NAG methods is $\gamma = 0.5$.

5. Conclusions

This paper is extracted and developed from the more theoretical part of our preprint [40]. The more applied part of [40] is separated into another paper [39].

In this paper we showed that Backtracking GD method works very well for general C^1 functions. In particular, if f has at most countably many critical points (for example if f is a Morse function) then the sequence constructed from Backtracking GD either diverges to infinity or converges to a critical point of f. Some modifica-

tions, including an inexact version, are also available. We also proved another result showing that in a certain sense it is very rare for any cluster point of the sequence to be a saddle point. We presented many examples illustrating various aspects of these results. Our method can also be applied to Wolfe's conditions, see Subsection 2.3.

As far as we know, our paper presents a new proof of showing $\lim_{n\to\infty} ||z_{n+1}-z_n|| = 0$, without relying on special properties of f such as being in $C^{1,1}_L$ or Losjasiewicz gradient inequality like previous work in the literature. This paper is also the first to introduce real projective spaces into uncontrained optimisation on Euclidean spaces.

The first author, developing ideas from this paper and [40], extended the results to various settings: avoidance of saddle points for a modification of Backtracking GD [36], proving convergent results in the Banach space setting [38], and defining a coordinate-wise Armijo's condition and the corresponding coordinate-wise Backtracking GD to better adapt with the case where partial derivatives of the cost function f can be different in sizes for different directions. All in all, as of current, methods using Armijo's condition have the strongest theoretical guarantees and are very flexible. Among these, Backtracking GD (and modifications) is probably the easiest one to implement while keeping the best theoretical guarantee, achieving very good experimental results, and being stable with respect to hyperparameters. Backtracking GD can also be used to boost the performance of other methods, such as NAG or Momentum.

We also present experimental results for implementations of Backtracking GD (developed in [40, 39]) for the dataset CIFAR10 on two different DNNs: Resnet18 and LeNet. The experiments illustrate the efficiency of Backtracking GD in large scale optimisation.

Theoretically, the above mentioned heuristic argument also suggests that good properties of Standard GD still hold for Backtracking GD. In particular, we expect that the following conjecture is true concerning saddle points.

Conjecture 5.1. *Assume that $f : \mathbb{R}^m \to \mathbb{R}$ is a C^1 function and is C^2 near its critical points. Then the set of initial points $z_0 \in \mathbb{R}^m$ for which the cluster points of the sequence $\{z_n\}$ in (8) contain a saddle point has Lebesgue measure* 0.

We note that this statement is stronger than the known results for the Standard GD. This is due to the fact that we have part (2) in Theorem 2.1 for Backtracking GD, which is unavailable for the Standard GD.

Theorems 2.4 and 2.3 support this conjecture. Besides these theorems and the above heuristic argument, here is a more realistic approach and evidence towards Conjecture 5.1. Assume for simplicity that f has at most countably many critical points. If the cluster points of the sequence $\{z_n\}$ contains a saddle point, then by Theorem 2.1 the whole sequence $\{z_n\}$ converges to z_∞. As we showed previously in this paper, since f is C^2 around z_∞, the sequence $\delta(f, \delta_0, z_n)$ will then take values in a fixed finite set. Thus, Backtracking GD then should be at most a combination of a finite number of sequences constructed from Standard GD. Then we expect that results and arguments from [25, 30] apply.

In the same vein, we state some other general open questions of great interest, which we hope to address in the near future.

Question 1. Can Theorem 2.1 be extended to all C^1 functions or to functions on infinite dimensions $f : \mathbb{R}^\infty \to \mathbb{R}$? This question can have applications to other fields such as PDE, where to solve a PDE we can try to reduce to the problem of finding extremal points of functionals on a space of functions, which is of infinite dimension.

Question 2. Can we prove a version of Theorem 2.1 for constrained optimisation, where the variables x are constrained in a convex subset D of $\mathbb{R}^m$? In convex optimisation, there is a method called projected gradient descent to deal with this problem, and we speculate that a similar procedure may work for Question 2.

Question 3. Is there a stochastic version of Theorem 2.1, as Stochastic GD is for Standard GD? Note that in the current version of Stochastic GD, Armijo's condition is not considered, and that when considering Backtracking GD, in general the learning rates $\delta(f, \delta_0, x)$ are not continuous as a function in x as shown in Example 2.16.

Question 4. What are the roles of the hyper-parameters α, β, δ_0 in Backtracking GD? This question is a simple case of how to deal with hyper-parameters in optimisation, and itself has important implications for Deep Learning. As a first step, we will find ways to formulate this as an optimisation problem itself. The same question can be asked for MMT and NAG, as well as their backtracking versions as proposed in Section 2.

References

[1] P.-A. Absil, R. Mahony, B. Andrews: *Convergence of the iterates of descent methods for analytic cost functions*, SIAM J. Optimization 16/2 (2005) 531–547.

[2] F. J. Aragon Artacho, P. T. Vuong: *The boosted DC algorithm for nonsmooth functions*, SIAM J. Optimization 30/1 (2020) 980–1006.

[3] L. Armijo: *Minimization of functions having Lipschitz continuous first partial derivatives*, Pacific J. Math. 16/1 (1966) 1–3.

[4] M. D. Asic, D. D. Adamovic: *Limit points of sequences in metric spaces*L, Amer. Math. Monthly 77/6 (1970) 613–616.

[5] A. Asl, M. L. Overton: *Analysis of the gradient method with an Armijo-Wolfe line search on a class of nonsmooth convex functions*, Optimization Methods Software 35/2 (2020) 223–242.

[6] H. Attouch, J. Bolte, B. F. Svaiter: *Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward-backward splitting, and regularized Gauss-Seidel methods*, Math. Programming 137/1 (2013) 91–129.

[7] H. Attouch, X. Goudou, P. Redont: *The heavy ball with friction method. The continuous dynamical system, global exploration of the local minima of a real-valued function by asymptotical analysis of a dissipative dynamical system*, Comm. Contemp. Math. 2/1 (2000) 1–34.

[8] D. P. Bertsekas: *Nonlinear Programming*, 2nd ed., Athena Scientific, Belmont (1999).

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

[10] S. Boyd, L. Vandenberghe: *Convex Optimization*, 7th printing with corrections, Cambridge University Press, Cambridge (2009).

[11] A. J. Bray, D. S. Dean: *Statistics of critical points of gaussian fields on large-dimensional spaces*, Physics Review Letter 98 (2007) 150201.

[12] C. Castera, J. Bolte, C. Févotte, E. Pauwels: *An inertial Newton algorithm for deep learning*, J. Mach. Learn. Res. 22 (2021), art. no. 134, 31 p.

[13] A. Cauchy: *Méthode générale pour la résolution des systèmes d'équations simultanées*, C. R. Acad Sci. Paris 25/2 (1847) 536–538.

[14] O. Cicek, A. Abdulkadir, S. S. Lienkamp, T. Brox, O. Ronneberger: *3D U-Net: Learning dense volumetric segmentation from sparse annotation*, in: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2016*, 19th Int. Conf., Athens 2016, Proceedings Part II, Lecture Notes in Computer Science 9901, Springer, Cham (2016) 424–432.

[15] J. B. Crockett, H. Chernoff: *Gradient methods of maximization*, Pacific J. Math. 5 (1955) 33–50.

[16] H. B. Curry: *The method of steepest descent for non-linear minimization problems*, Quart. J. Appl. Math. 2/3 (1944) 258–261.

[17] Y. N. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, Y. Bengio: *Identifying and attacking the saddle point problem in high-dimensional non-convex optimization*, in: Advances in Neural Information Processing Systems, NIPS' 14, Volume 2 (2014) 2933–2941.

[18] K. Eyholt, I. Evtimov, E. Fernades, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, D. Song: *Robust physical-world attacks on deep learning visual classification*, IEEE Conf. Computer Vision and Pattern Recognition (2018) 1625–1634

[19] M. Fazlyab, A. Robey, H. Hassani, M. Morari, G. J. Pappas: *Efficient and accurate estimation of Lipschitz constants for deep neural networks*, in: NIPS'19, Proc. 33rd Int. Conf. Neural Information Processing Systems (2019) 11427–11438.

[20] S. G. Finlayson, J. D. Bowers, J. Ito, J. L. Zittrain, A. L. Beam, I. S. Kohane: *Adversarial attacks on medical machine learning*, Science 363/6433 (2019) 1287–1289.

[21] R. Ge, F. Huang, C. Jin, Y. Yuan: *Escaping from saddle points – online stochastic gradient for tensor decomposition*, J. Machine Learning Res. 40 (2015) 797–842.

[22] A. A. Goldstein: *Cauchy's method of minimization*, Numer. Math. 4 (1962) 146–150.

[23] U. Helmke, J. B. Moore: *Optimization and Dynamical Systems*, Communications and Control Engineering, Springer, London (1994).

[24] K. Lange: *Optimization*, 2nd edition, Springer Texts in Statistics 95, Springer, New York (2013).

[25] J. D. Lee, M. Simchowitz, M. I. Jordan, B. Recht: *Gradient descent only converges to minimizers*, J. Machine Learning Res. 49 (2016) 1246–1257.

[26] K. Kawaguchi: *Deep learning without poor local minima*, in: *Advances in Neural Information Processing Systems (NIPS 2016)* 29 (2016) 586–594.

[27] Y. Nesterov: *Introductory Lectures on Convex Optimization : a Basic Course*, Kluwer Acad. Publishers, Dordrecht (2004).

[28] M. A. Nielsen: *Neural Networks and Deep Learning*, Determination Press (2015).

[29] J. Nocedal, S. J. Wright: *Numerical Optimization*, Springer Series in Operations Research and Financial Engineering, Springer, Cham (2006).

[30] I. Panageas, G. Piliouras: *Gradient descent only converges to minimizers: Non-isolated critical points and invariant regions*, in: *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*, C. H. Papadimitrou (ed.), Leibniz International Proceedings in Informatics (LIPICS), Dagstuhl Publishing, Dagstuhl (2017) 2:1–2:12.

[31] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, A. Swami: *Practical black-box attacks against machine learning*, in: *Proceedings of the 2017 Asia Conference on Computer and Communications Security*, ACM Digital Library (2017) 506–519.

[32] H. Robbins, S. Monro: *A stochastic approximation method*, Ann. Math. Statistics 22 (1951) 400–407.

[33] S. Ruder: *An overview of gradient descent optimisation algorithms*, arXiv: 1609.04747 (2016).

[34] K. Scaman, A. Virmaux: *Lipschitz regularity of deep neural networks: analysis and efficient estimation*, in: *Proc. of the 32nd International Conference on Neural Information Processing Systems (NIPS 18)*, ACM Digital Library (2018) 3839–3848.

[35] G. Swirszcz, W. M. Czarnecki, R. Pascanu: *Local minima in training of neural networks*, arXiv: 1611.06310 (2016).

[36] T. T. Truong, *Convergence to minima for the continuous version of backtracking gradient descent*, arXiv: 1911.04221 (2019).

[37] T. T. Truong: *Coordinate-wise Armijo's condition: general case*, arXiv: 2003.05252 (2020). (This article includes arXiv: 1911.07820 (2019) as a very special case.)

[38] T. T. Truong, *Some convergent results for Backtracking Gradient Descent method on Banach spaces*, arXiv: 2001.05768 (2020).

[39] T. T. Truong, T. H. Nguyen, *Backtracking gradient descent method and some applications in large scale optimisation. Part 2: Algorithms and experiments*, Applied Math. Optimization 84/3 (2021) 2557–2586. (This article consists of the more experimental part of [40], together with arXiv: 2001.02005 (2020) and arXiv: 2007.03618 (2020).)

[40] T. T. Truong, T. H. Nguyen: *Backtracking gradient descent method for general C^1 functions with applications to deep learning*, arXiv: 1808.05160 (2018).

[41] P. Wolfe: *Convergence conditions for ascent methods*, SIAM Review 11/2 (1969) 226–235.

[42] https://github.com/hank-nguyen/MBT-optimizer.