Stochastic multilevel methods for deep learning

Vom Fachbereich Mathematik
der Technischen Universität Darmstadt
zur Erlangung des Grades eines

Doktors der Naturwissenschaften
(Dr. rer. nat.)

genehmigte

Dissertation

von

Isabel Jacob, M. Sc.

Referent:	Prof. Dr. Stefan Ulbrich
Korreferent:	Prof. Dr. Marc E. Pfetsch
Tag der Einreichung:	18. Oktober 2024
Tag der mündlichen Prüfung:	2. Dezember 2024

Darmstadt 2024

Stochastic multilevel methods for deep learning

Autor: Isabel Jacob

Vom Fachbereich Mathematik der Technischen Universität Darmstadt zur Erlangung des akademischen Grades eines Doktors der Naturwissenschaften (Dr. rer. nat.) genehmigte Dissertation, Darmstadt.

Jahr der Veröffentlichung der Dissertation auf TUprints: 2025

Tag der mündlichen Prüfung: 02.12.2024

Die Veröffentlichung steht unter folgender Creative Commons Lizenz:
CC BY 4.0 International
https://creativecommons.org/licenses/by/4.0/

Acknowledgments

First and foremost, i would like to thank my supervisor Prof. Dr. Stefan Ulbrich for the opportunity to work on this thesis and contribute to his research group. I am truly grateful for his invaluable guidance, his constructive feedback and his continual support in the last years and especially during the past months. He always kept his door open for discussions and his encouragement helped me grow professionally as well as personally. I also appreciate the many opportunities he provided to attend conferences and broaden my perspectives.

I would also like to thank Prof. Dr. Marc Pfetsch for agreeing to act as a referee of this thesis. Furthermore, I thank Prof. Dr. Tabea Tscherpel and Prof. Dr. Michael Kohler for taking the time to be members of the examination committee.

I would like to thank all my present and former colleagues in our optimization research group at TU Darmstadt for the wonderful time we had both inside and outside the office. The camaraderie, insightful discussions, and shared experiences made this an enriching and truly memorable experience. I am especially grateful to my colleagues Jannik Breitkopf, Elisa Corbean, Erik Jansen, Lea Rehlich, and Marcel Steinhardt for proofreading this thesis and their valuable feedback.

I am deeply thankful to my family, my sister Viktoria Jacob and my parents Jutta Willert-Jacob and Prof. Dr. Ralf Jacob, for their support and love, not only during this academic journey but throughout my entire life.

Last but not least I want to express my deepest thanks to my dearest colleague, Jannik Breitkopf, for his calming support, his patience and for his love.

Zusammenfassung

Da die Anwendungsfälle für neuronale Netze immer komplexer werden, werden zugleich auch moderne neuronale Netze größer und komplexer, um mithalten zu können. Mit zunehmender Tiefe und Komplexität solcher Netze steigt jedoch auch der Rechenaufwand von gängigen Trainingsmethoden wie dem stochastischen Gradientenverfahren (SGD).

Multilevelverfahren, die traditionell zur Lösung von Differentialgleichungen mit Hilfe von Hierarchien an Diskretisierungen verwendet werden, bieten die Möglichkeit, den Rechenaufwand zu reduzieren. Diese Methoden arbeiten mit verschiedenen Stufen an Komplexität. Auf den unteren Stufen kann eine grobe Näherung der Lösung schnell und mit geringem Aufwand berechnet werde. Diese Näherungen werden dann auf höheren Stufen schrittweise verfeinert, um genauere Näherungen zu erhalten. Durch den Wechsel zwischen den Stufen können Multilevelverfahren schneller konvergieren ohne an Genauigkeit einzubüßen.

In dieser Arbeit wird ein Multilevel stochastisches Gradientenverfahren (MLSGD) vorgestellt, das beide Ansätze miteinander kombiniert. Ziel dabei ist es, das Training von neuronalen Netzen durch den Einsatz von Multilevelmethoden zu beschleunigen. Der zentrale Beitrag dieser Arbeit liegt in der Entwicklung und Analyse von MLSGD. Wie üblich bei Multilevelverfahren, werden Prolongations- und Restriktionsoperatoren verwendet, um den Wechsel zwischen verschiedenen Stufen zu ermöglichen. Um Konsistenz erster Ordnung zu gewährleisten, wird ein Gradientenkorrekturterm zur Zielfunktion hinzugefügt. Des Weiteren verwenden wir zusätzliche Bedingungen, wie eine Schrittweitenregularisierung und eine Winkelbedingung. Wir diskutieren die Konvergenzeigenschaften der Methode unter der Annahme, dass konstante Schrittweiten verwendet werden. Hierbei untersuchen wir auch den Einsatz stochastischer Richtungen anstelle des Gradienten in der Gradientenkorrektur und den Einfluss von Varianzreduktion in beiden Fällen.

Anschließend untersuchen wir das Verhalten des Verfahrens in der Praxis. Auch hier untersuchen wir den Einfluss von einer stochastischen Gradientenkorrektur und Varianzreduktion. MLSGD wird dazu auf den Bildklassifizierungsdatensatz CIFAR-10

angewendet. Wir verwenden dabei zwei verschiedene Ansätze um eine Hierarchie zu bilden, basierend entweder auf der Netzwerktiefe oder der Bildauflösung. Für beide Ansätze werden geeignete Prolongations- und Restriktionsoperatoren konstruiert.

Abstract

As the use cases for neural networks become increasingly complex, modern neural networks must also grow deeper and more intricate to keep up. However, with increased depth and complexity, common training methods like stochastic gradient descent methods (SGD) become more computationally expensive. Multilevel methods, traditionally used to solve differential equations through hierarchies of discretizations, offer the potential to reduce computational effort. These methods operate across multiple levels of complexity. On lower levels, a coarse approximation of the solution can be obtained quickly and at low computational cost. These coarse approximations are then refined on higher levels to achieve a more accurate solution. By alternating between levels, multilevel methods can accelerate convergence while maintaining precision.

This thesis introduces a multilevel stochastic gradient descent algorithm (MLSGD) that combines both concepts aiming to accelerate neural network training through multilevel techniques. The core contribution of this thesis is the development and analysis of MLSGD. As in traditional multilevel methods, prolongation and restriction operators enable transitions between levels. To ensure first-order coherence, a gradient correction is added to the objective function as well as additional conditions including step size regularization and an angle condition. We analyze the convergence properties of the method under the assumption of fixed step sizes. Additionally, we investigate the influence of stochastic directions in the gradient correction as a replacement for full gradients as well as the effect of variance reduction in both cases.

Finally, we evaluate the practical performance of the method as well as the effect of stochastic gradient correction and variance reduction. To this end, MLSGD is applied to the image classification dataset CIFAR-10. We explore two different approaches to constructing a hierarchy, based either on network depth or image resolution. For both approaches, we construct suitable prolongation and restriction operators.

Symbols

$\mathbb{R}$	the set of real numbers
$\mathbb{R}^d$	the set of d dimensional vectors over $\mathbb{R}$
$\mathbb{N}$	the set of natural numbers
$\mathbb{1}_M$	indicator function for the set M
$x^\top$	the transpose of a vector $x \in \mathbb{R}^d$
x_i	the ith entry of the vector x
$x_1, \dots, x_m$	a sequence of m vectors
$A \in \mathbb{R}^{m \times n}$	a $m \times n$ matrix with entries in $\mathbb{R}$
$A^\top$	the transpose of A
$A_{i,j}$	the (i, j) element of A
$\langle \cdot, \cdot \rangle$	inner product
$\| \cdot \|$	norm
$\| \cdot \|_F$	Frobenius norm
$\mathcal{P}$	probability
$\mathbb{E}, \mathbb{E}_\xi$	expected value, expected value with respect to ξ
$\mathbb{V}$	variance
$z \sim \mathbb{P}$	sampling according to a distribution $\mathbb{P}$
$[\xi]_{i,j,s}^{l,k,t}$	the set of ξ sampled strict after iteration (i, j, s) up until (including) (l, k, t)
$[\xi]^{l,k,t}$	the set of ξ sampled up until iteration (l, k, t)
$\mathcal{X}$	sample set
$\mathcal{Y}$	labels set
$\mathcal{H}$	hypothesis function set
$\ell \colon \mathcal{Y} \times \mathcal{Y} \longrightarrow \mathbb{R}$	loss function
$\nabla f(x)$	the gradient of a function $f \colon \mathbb{R}^d \longrightarrow \mathbb{R}$ at $x \in \mathbb{R}^d$
$\min, \max$	minimize, maximize
$\min_{x \in C} f(x)$	the minimal value of a function f over a set C if existent
$\log$	the natural logarithm
$\sin, \cos$	sine, cosine

Contents

CHAPTER 1

Introduction

Machine learning is currently subject of increased interest in media and society with its diverse use cases in chat bots, robotics or image creation. Deep learning refers to a special case of machine learning that uses neural networks with multiple layers. Essentially, neural networks are functions that have a high number of parameters. By tuning these parameters, the behavior of the function is changed. The high number of parameters makes neural networks suitable for complex applications but also computationally expensive. Additionally, the datasets used for training can be large in number of samples as well as the size of the samples. To reduce the computational effort, we propose a multilevel approach that uses auxiliary networks with less parameters and smaller input sizes. Because these networks are less computationally expensive, the aim is to find mechanisms and conditions such that the training progress of these networks can contribute to the training of the original network.

1.1 Main contribution

In this thesis we propose a multilevel stochastic gradient descent algorithm. To this end, we use a design similar to that of multilevel trust region methods as introduced in [GST08; Gra+08]. This results in an algorithm with a multilevel structure whose updates are performed using stochastic directions as known from stochastic gradient descent. To enforce first order coherence, we use a gradient correction. There are two versions of the gradient correction. The first version uses the gradient of the finer level in the gradient correction as usually done in multilevel optimization methods. However, since computing the gradient is very expensive in machine learning, the second version uses stochastic approximations of the finer level gradient. For both cases, we also analyze the influence of

a variance reduction term as introduced in [JZ13]. Moreover, we employ a regularization on the length of the steps and an angle condition to ensure the directions from the coarser levels contribute to the finer level. Our theoretical analysis of the algorithm assumes fixed step sizes. We show that using fixed step sizes the bound on the expected mean of the norm of the gradients of the objective function converges to a constant depending on the step size, the number of levels and the quality of the stochastic directions in all four cases. For a deterministic gradient correction with variance reduction, the constant vanishes.

1.2 Comparison to related work

There have been approaches of combining multilevel methods with machine learning problems. The multilevel trust region methods on which we have based our algorithm have also been applied to deep learning problems, see [GKT23; KK23]. However, despite the stochastic setting of the application, [GKT23] assume a deterministic setting in the theoretical analysis of their multilevel objective-function-free trust region (MOFFTR) algorithm. On the other hand, [KK23] proposed a hybrid approach, by adapting the mini batch sizes. Initially, for subsampling the gradient and approximating the Hessian by rank one updates smaller batch sizes are used which are then gradually increased until the full dataset is used and the method operates deterministically. This is why the convergence properties follow then from the deterministic multilevel trust region method in [GST08].

However, as trust region methods need to calculate, respectively approximate, the gradient and the Hessian, they are less common in machine learning problems. The algorithm that is usually used in machine learning context is stochastic gradient descent or a variant of it. Thus we focus on a multilevel stochastic gradient algorithm.

There have been other approaches to combine multilevel methods with stochastic gradient methods. First of all, there are approaches that use the similarity between residual networks and discretizations of ordinary equations to establish a hierarchy by varying the mesh of discretizations like [Hab+18; HR17a; Cha+17; GKK20]. In the case of residual networks, this means to vary the number of layers in a residual block. In [HR17a] and [Cha+17] as well as in the second approach in [Hab+18] this similarity is used to establish a hierarchy of networks with different numbers of layers in a residual block. Then this multilevel hierarchy to initialize the parameters on the finest level by utilizing the coarser levels. With a similar hierarchy [GKK20] employed a multilevel line search algorithm based on MG/OPT by [Nas00] that used V-cycles for each minibatch and then gradient descent as optimization method on a level. This way the V-cycle is deterministic and fastens the computation of a minibatch update. Compared to our

algorithm however, we use several stochastic updates during a V-cycle and the V-cycle is applied to the objective function consisting of all batches.

A different approach of providing a multilevel hierarchy in supervised machine learning that was proposed by [BKK20] is changing the number of training samples from level to level. This means the coarser level uses a subset of the training samples of the finer level. In this way, the gradients of the coarser level are stochastic gradients of the finer level.

Similar to one of our approaches, the first approach in [Hab+18] use the resolution of the images to establish a hierarchy. The coarse level then has the same network architecture but trains on images of coarser resolution. The coarse level parameters are chosen by first prolongating the input, applying the fine level convolutions, and then restricting the result. This is the same approach as the one we use, except that we notice that under certain assumptions, these coarse convolutions coincide with convolutions of a smaller kernel size. In this approach however, the kernel sizes of the networks on all levels stay the same. The idea of using the resolution for establishing a hierarchy has also been extended to the training of video models in [Wu+19]. During training the input is subsampled in frames, height and width and variable minibatch sizes are used. As only one model is trained on input of varying size, the parameters do not need to be transferred and there is no gradient correction.

While these studies provide valuable heuristic and experimental results, they do not go into detail about the convergence properties of the method. Convergence results are based mostly on the convergence of deterministic methods, if they are presented at all, which means they neglect the stochastic aspects of machine learning.

In [MNT19] the authors propose a multilevel stochastic gradient approach, which they apply to optimal control problems where the objective function is the expected value of a strongly convex function. They perform stochastic gradient descent with multilevel Monte-Carlo estimators as stochastic directions independent of previous iterations. To establish a hierarchy they use Finite Element approximation on refined nested meshes. In their framework the number of levels increases with each iteration thus increasing accuracy. They present a convergence result along with optimal choices for parameters to reach a desired tolerance with minimal computational effort.

1.3 Structure of the thesis

This thesis is organized as follows. First, we start with the general background needed to understand the problem. Thus, in Chapter 2 we give an introduction to machine learning. We derive the optimization problem usually considered in machine learning in Section 2.1 and give a definition for neural networks and explore different layer types in

Section 2.2. In Section 2.3 we introduce optimization methods that are well suited for machine learning problems and the assumptions under which these converge.

Chapter 3 deals with the central topic of this thesis as it introduces the algorithm we propose. Starting with an overview over multilevel optimization methods in Section 3.1, we extend the machine learning optimization problem to a multilevel setting. This enables us to motivate and introduce the multilevel stochastic gradient algorithm in Section 3.2. The convergence analysis in Section 3.3 is divided into the four cases for the gradient correction, i.e., the deterministic version and the stochastic version, and the use of a variance reduction.

Lastly, in Chapter 4 we apply the algorithm to a common machine learning problem: image classification. After establishing a base architecture in Section 4.1, we use two approaches to establish a hierarchy. First in Section 4.2, we vary the number of layers in a network to create networks of varying complexity. In the second approach in Section 4.3, we reduce the resolution of the dataset and the size of the convolutional kernels in the networks. For both approaches, we construct the prolongation and restriction operators and present numerical results, paying particular attention to the difference in performance between the two variants of the gradient correction. In Section 4.4 we review and discuss our results.

We summarize our findings in Chapter 5. Furthermore, we give a short outlook on further possible variations or extensions of our algorithm.

CHAPTER 2

Machine Learning Setup

At its core, machine learning is an optimization problem. To understand where we can use optimization to enhance machine learning algorithms, it is crucial to establish a basic understanding of machine learning itself.

Machine learning is part of the field of artificial intelligence. The goal in machine learning is to find algorithms that are able to extract information from given data and make predictions on unseen data. In contrast to classical programming which is based upon explicit instructions, machine learning algorithms have a high number of degrees of freedom and are trained to choose them on their own. Human intervention is only necessary at a few stages in the learning process: in the choice of models and learning algorithms, preprocessing data and interpreting results.

This autonomy is useful for a variety of tasks. First of all, there are tasks that are too complex to be formalized but intuitive to solve for a human being. For example image classification or voice recognition belong to this category. Enabling a computer to perform such tasks allows for modern technology using this information. Second, in times of Big Data, where a large amount of data from different sources and in different contexts is available, processing and analyzing this data becomes impossible for humans. Machine learning methods are able to solve both such tasks and through their adaptive nature are also easily customized to fit different use cases, see [SB14].

Due to its wide range of applications, machine learning has been divided into various subfields. Depending on the goal, according to [GBC16] there are mainly two subfields, supervised and unsupervised learning. In supervised learning the algorithm is trained on a labeled dataset. Each data point has one or more correct labels and the task for the algorithm is to give the correct label, especially when confronted with previously unseen

data. This subfield of machine learning includes classification, where there is a finite number of labels, and regression tasks, where the labels are real values. In unsupervised learning, on the other hand, the algorithm is presented with unlabeled data and tasked with identifying patterns within it or structuring the dataset. Unsupervised learning includes the grouping of data, known as clustering, or the generation of similar data, as in generative models.

This chapter outlines the machine learning setup and the problem to be discussed as well as optimization methods to use on these kinds of problems. It is structured into three parts. The fundamental optimization problem underlying the supervised machine learning problem we consider is introduced in Section 2.1. Section 2.2 explains the structure of the model used in our application, including possible variations, their advantages and disadvantages. In Section 2.3, we present optimization methods commonly used in machine learning problems with a focus on stochastic gradient descent and its variations as this is the basis of the algorithm presented in Chapter 3.

2.1 Mathematical formulation of the machine learning problem

First of all let us introduce the optimization problem that we are typically confronted with in machine learning. After discussing the problem in Section 2.1.1 and its variations by the use of loss functions in Section 2.1.2, we name some example applications to showcase the practical relevance of machine learning problems and how to fit them into our setup.

2.1.1 Supervised machine learning or classification

Let $\mathcal{X}$ be the input space and $\mathcal{Y}$ the label space. In a supervised classification problem the goal is to classify inputs $x \in \mathcal{X}$ with their corresponding labels $y \in \mathcal{Y}$ where we assume that (x, y) can be generated independently and identically distributed from an unknown distribution $\mathbb{P}$. This means we want to construct a function h in the function space $\mathcal{H} = \{h \mid h : \mathcal{X} \longrightarrow \mathcal{Y}\}$ that minimizes the *risk of misclassification*

$$\min_{h \in \mathcal{H}} \mathbb{E}_{(x,y)\sim\mathbb{P}}[\mathbb{1}_{[h(x)\neq y]}]. \tag{2.1}$$

For easier notation assume that $\mathcal{X} \subseteq \mathbb{R}^{n_x}$ and $\mathcal{Y} \subseteq \mathbb{R}^{n_y}$ with $n_x, n_y \in \mathbb{N}$.

Unfortunately, Problem (2.1) is unusable in practice since the distribution $\mathbb{P}$ is unknown. In supervised machine learning we assume to have a *training set* consisting of $N \in \mathbb{N}$

samples $\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$, i.e. we depend on having inputs with already known correct labels. Using this set we can minimize the *empirical risk of misclassification*

$$R_N(h) \coloneqq \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}_{[h(x_i) \neq y_i]}, \tag{2.2}$$

instead.

Usually, a certain difference between the performance on the training set and the performance on unseen data remains. If the difference becomes too large, the training set has been *overfitted*. This corresponds to learning the training set and not generalizing well. To prevent overfitting, there are several approaches and safeguards.

First of all, the minimization should not take place over the whole function space $\mathcal{H}$. Instead, the feasible set is reduced by restricting the function space $\mathcal{H}$ to, for example, linear classifiers or neural networks. This also has the practical advantage of knowing the specific characteristics of the objective function beforehand when implementing an algorithm.

Second, another way to improve generalization is to increase the size of the training set. However, it should be noted that obtaining new data can be difficult and expensive, making this approach impractical. There are compromises, such as data augmentation. This involves making slight changes to the input without affecting the correct label, allowing for the generation of almost new samples, for example by applying geometric distortions of the input [Lec+98] or by randomly perturbing the colors in an image [KSH12].

Third, by using a regularization term in the objective function oftentimes in the form of ℓ_2 *regularization* that penalizes large weights, simple solutions are favored which helps generalization. We discuss the regularization techniques used in our applications in Section 4.1.2.

2.1.2 Loss functions

Because $\mathbb{1}_{[h(x_i) \neq y_i]}$ is a discontinuous function that is difficult to use in standard minimization algorithms, one introduces a continuous approximation of it, $l \colon \mathcal{Y} \times \mathcal{Y} \longrightarrow \mathbb{R}$. Functions that measure the discrepancy between true labels and predictions, including $\mathbb{1}_{[h(x_i) \neq y_i]}$ itself, are referred to as *loss functions*. Eq. (2.2) then becomes

$$\min_{h \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^{N} l(h(x_i), y_i). \tag{2.3}$$

Without further assumptions on h this is a nonconvex optimization problem. A straightforward choice for a loss function would be the well known *square loss* that for $h, y \in \mathbb{R}^n$ is given by

$$l(h, y) = \|h - y\|_2^2 .$$

Although this loss function is easy to implement, it can be sensitive to outliers, i.e. they might contribute disproportionately to the loss. Thus when using an imbalanced data set, the learning of underrepresented classes with many outliers might be neglected. This is why it is primarily used in regression tasks where continuous values are predicted instead of discrete separate classes. Another disadvantage is that it does not take into account how high the probability is that the model assigns to the wrong classes.

A different class of loss functions are *cross entropy loss functions* like

$$l(h, y) = -\sum_{j=1}^{C} y_j \log(h_j),$$

where C is the number of classes and $y_j, h_j \in [0, 1]$. To achieve h_j taking values between 0 and 1, usually a softmax function is used, see Section 2.2.1. Incorrect labels correspond to $y_j = 0$, thus the cross entropy loss function disregards these labels even when they are falsely classified. However, for correct labels, when the prediction is far from 1 and closer to 0, the negative logarithm takes very large values. In this way, cross-entropy loss functions penalize false negatives more heavily than they reward correct classifications. This encourages the model to improve its predictions for the misclassified class and thus helps with unbalanced datasets. A standard in modern day classification problems is the categorical cross entropy loss function which is a cross entropy function in combination with a softmax activation function, see Section 2.2.1.

For good performance, it is essential to choose an appropriate loss function. It depends on the given classification task, which loss function works best.

2.1.3 Applications

In Chapter 4 we consider image classification as an application of the proposed algorithm. However, this subsection briefly outlines other potential areas where classification can be applied in modern society to emphasize its meaning.

Image classification and object detection

As an example we consider the task of image classification. Nowadays, there are plenty of applications where image classification is used to simplify and facilitate our lives. Face

recognition is a built-in feature of our cell phones, either used to unlock our phones [Tea17] or to search the gallery application for pictures of people, animals or sights [Tea21]. It is even possible to monitor plants to identify diseases early and improve efficiency in agriculture [Sho+23].

To formulate the problem of finding such a classification algorithm the set of inputs are images and the labels their motive. In the case of recognizing people in images this is also called facial recognition [Tai+14; Sun+14; HR17b].

When the task is not only to classify the object but also to pinpoint its exact location in an image, then we are talking about object detection [Red+16]. This is particularly important in the context of autonomous vehicles for detecting obstacles. In this case, the inputs are still images, but the labels include not only the correct class for the object but also its location. For 3D object detection, for instance, boxes containing the object are used [SZT19].

Medical data

A very sensitive field of application for machine learning algorithms is medicine. Medical imaging is an obvious application for image recognition because of the amount of data involved and the inherently classification-driven nature of the tasks within this domain. This technology is used across a wide range of procedures, from X-rays, like mammography, to CT scans or MRIs to detect different types of tumors [GK18; SA19] or neurological disorders [Can+15]. To train such algorithms, the inputs are the images and the labels a diagnosis or the location of a tumor for example. But also with the history of a patients data, machine learning algorithms can also predict outcomes, like heart failures [WRS10] or a patients reaction to a certain medication, for example in cancer treatment [Men+13; Bor+18].For an extensive overview on different applications of machine learning in medicine, we refer the reader to [She+22].

Natural language processing

Similarly to image classification, natural language processing is increasingly more important in our modern life. It is used, for example, in voice recognition [Dah+12; GMH13; Abd+14], which is needed for virtual assistants. These assistants transcribe spoken language and react to it, enabling users to make calls from their vehicle, set cooking alarms, and operate their smart home quickly and hands-free. A training set in voice recognition could consist of voice records and their corresponding transcribed text. However, because manually transcribed training data is expensive, modern approaches

use semi-supervised [Kap+14; Zha+22] approaches, where the training data consists of labeled and unlabelled samples.

Another application are translation tools where the integration of machine learning techniques has improved and refined the translation quality. Here the input is text in one language and the label a translated text [Cho+14]. Text based classification can be used to classify emails as spam or not spam, or sort them into folders [CKP03]. The algorithm takes the text in an email as input and outputs a label of either "spam" or "not spam" or the corresponding folder.

Finance and marketing

Moreover, machine learning classification is applied in the finance sector to analyze data quickly. For example, in credit scoring, when a person or company applies for a loan, their trustworthiness can be assessed by analyzing various factors. For individuals this includes their credit history, income statement, demographic factors and history of payment, whereas for companies financial statements, balance sheets, income statements and cash flows are relevant. Additionally, this can also be used for loan default prediction. In these cases the label is binary meaning "trustworthy" or "not trustworthy", respectively "default" or "no default" [AGH18; KKL10]. Furthermore, classification can also be used for fraud detection where the input is a current transaction and the customer transaction history and the label "fraud" or "not fraud" [Sin+12].

In marketing, classification is a valuable tool for categorizing products and enhancing search functionality. It is frequently employed in recommendation systems to classify customers into different segments [Wan22], allowing personalized product recommendations based on interests and preferences of customers.

2.2 Residual neural networks

In this section we examine possible forms of the hypothesis function $h \in \mathcal{H}$ from Section 2.1. Throughout the history of machine learning several approaches have been made like support vector machines [CV95], random forests [Ho95; Bre01] or k-nearest-neighbors [FH89]. In this thesis though we focus on *neural networks*, more specifically residual convolutional neural networks. For more information on standard machine or deep learning concepts, we refer the reader to the monographies [GBC16; SB14].

2.2.1 Neural networks for image classification

Although neural networks are currently experiencing a surge in interest, the basic ideas date back to the 1940s and interest in this field has varied greatly over time. The first model that could be interpreted as an early neural network was introduced in 1943 by McCullough and Pitts in [MP43]. It was a simple one-layered model that could perform logical operations.

These networks were inspired by neurons in the human brain. However, they were greatly simplified compared to their biological counterpart. Nonetheless, there exist several parallels, for example their structure and learning ability, between biological neurons and neural networks thus coining the term "neural".

In its simplest form, a neural network consists of an input layer, an output layer and in between hidden layers. A layer takes the entries in the previous layers, weighs them individually and thus computes new entries. This way each layer is connected to the layer before and after. Let for $l \in \mathbb{N}$ denote the lth layer in a network with $F^l \colon \mathbb{R}^{n_x^l} \times \mathbb{R}^{n_W^l} \longrightarrow \mathbb{R}^{n_x^{l+1}}$. This means a neural network h with $L \in \mathbb{N}$ layers and parameters $W^l \in \mathbb{R}^{n_W^l}$ is a function

$$h(x, W^1, \dots, W^L) = F^L(F^{L-1}(\dots(F^1(x, W^1)\dots), W^{L-1}), W^L).$$

In general such a layer may look like

$$F^l(x^l, W^l) = C_2^l\left(w_3^l\right)\sigma^l\left(N^l\left(C_1^l\left(w_1^l\right)x^l, w_2^l\right)\right), \tag{2.4}$$

where $l \in \{0, \dots, L\}$, x^l is the input to this layer, also known as *features*, N^l is a *normalization layer*, σ^l an *activation function*, C_1^l, C_2^l are linear operators (usually *convolutions*) and $W^l = (w_1^\top, w_2^\top, w_3^\top)^\top \in \mathbb{R}^{n_W^l}$ are parameters stacked as a vector. We now define these terms.

Activation functions

In the first neural networks, the goal was to simulate the neurons firing in the brain. Thus activation functions were used to mimic the binary behavior of a biological neuron, i.e., the neuron either fires or does not fire. One example of such an early activation function is a simple *Heaviside step function* $\sigma(x) = \mathbb{1}_{\mathbb{R}_+}$, that takes only the values 0 or 1 [MP43].

An improvement over the Heaviside step function is the *sigmoid function*

$$\sigma(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1}.$$

In contrast to the Heaviside step function, the sigmoid function is smooth and also mimics the behavior of saturating at the values 0 or 1. However, this saturating behavior can also be a disadvantage because the gradients become small. This leads to vanishing gradients and prevents learning. Another disadvantage is that the values are not centred around 0. An easy way to solve this problem is to use the *hyperbolic tangent* with

$$\sigma(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

instead, which are centered around 0 but otherwise similar to the sigmoid function.

The most popular activation function in modern networks is *Rectified Linear Unit (ReLU)* [Jar+09; NH10; GBB11] defined as

$$\sigma(x) = \max\{0, x\}.$$

The ReLU function improves the vanishing gradient problem and shows faster convergence than sigmoid or tanh functions, see [GBB11]. It is important to acknowledge that ReLU is not differentiable in 0 and thus does not have gradients at that point, let alone Lipschitz continuous gradients. Unfortunately, differentiability and Lipschitz continuous gradients is a common assumption for the convergence of optimization methods in machine learning such as the one we discuss in Section 2.3. Following the success of ReLU several offspring variants from ReLU have emerged to improve its flaws. The vanishing gradients problem is not entirely solved for ReLU as it can still happen that neurons stop learning because of the constant 0 term for negative numbers. *Leaky ReLU* introduced in [MHN13], for example, is defined as

$$\sigma(x) = \max\{0.01x, x\}$$

and compensates the vanishing neurons problem by using a slight slope.

As x is most likely a vector, all the activation functions are applied entry wise. These activation functions all have in common that they introduce nonlinearity to the network allowing it to learn more complex relations.

A special kind of activation function that is only used in the last layer is the *softmax activation function*

$$\sigma(x)_i = \frac{e^{x^i}}{\sum_{j=1}^{K} e^{x^j}} \text{ for } x \in \mathbb{R}^K.$$

It normalizes the output to a probability distribution that assigns higher probabilities to classes with higher entries. This is helpful in classification tasks where the model shall give probabilities for different classes and an essential part of the categorical cross entropy loss function, see Section 2.1.2.

Normalization layers

Covariate shift describes the phenomenon that during training the statistical distribution, i.e. the mean and the variance, of the features changes from layer to layer. To help with internal covariate shift and improve general training stability of neural networks, normalization layers are often used. By normalizing, the value range of the features and that of their gradients is balanced. This allows for a larger learning rate and thus faster progress. Furthermore, the gradients do not become too small, leading to vanishing gradients, or too large, leading to exploding gradients.

Even though there are several different normalization layers, *batch normalization* is the most known and widely used normalization layer. It was introduced in 2015 by Ioffe and Szegedy [IS15]. It centers and scales over the whole batch to the mean and standard deviation thus normalizing the input. This is done by subtracting the mini-batch mean and dividing by the mini-batch standard deviation during training. For N inputs $\{x_i\}_{i=1}^N$ to the layer in a mini batch, the mean and variance are computed as

$$m = \frac{1}{N}\sum_{i=1}^{N} x_i \quad \text{and} \quad v^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - m)^2.$$

Two new parameters are also introduced, a scaling factor $\gamma \in \mathbb{R}$ and a shifting factor $\beta \in \mathbb{R}$ such that the distribution can be adjusted during training. The layer is then defined as

$$N(x, \gamma, \beta) = \gamma \frac{x - m}{\sqrt{v^2 + \varepsilon}} + \beta,$$

using a small constant $\varepsilon > 0$ for numerical stability. During training, the appearing batch means and variances are tracked such that when after training, the estimated mean and variance can be used. This is important as, later on, the trained model will presumably be called with single input samples to classify and not whole batches.

Additionally to the other advantages, the authors report in [IS15] that batch normalization is particularly effective in reducing the sensitivity to the parameter initialization.

Fully connected layers

In the early development of neural networks for image classification, most networks used *fully connected layers*, also referred to as *dense layers*. When visualising the input to a network as flattened vectors, the entries of a vector are considered nodes in a network and in a fully connected layer every node is connected to every node in the previous layer, see for example Figure 2.1. This is where the name 'fully connected' derives from.

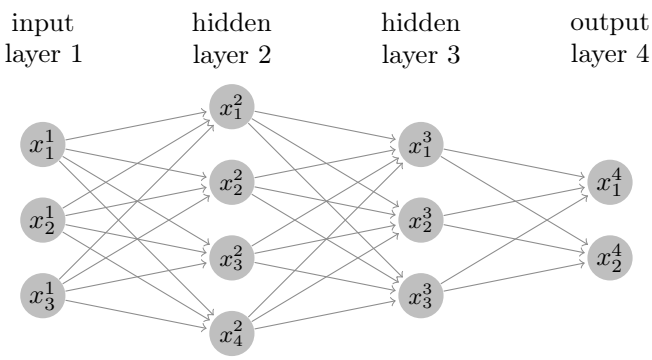


Figure 2.1 An example of a simple fully connected neural network with one input layer, two fully connected hidden layers and one fully connected output layer.

Suppose the input is a vector $x \in \mathbb{R}^{n_x}$. Then the parameters of a fully connected layer include a *weight matrix* and a *bias*. The fully connected layer then performs a matrix-vector multiplication and adds the bias to the output.

This means before applying a fully connected layer in a network one has to flatten the input to vector form. In the case of image classification this means an image of a certain height h, width w and a certain number of channels c would first be flattened to a vector of shape hwc. Let k be the desired dimension of the output vector. The weight matrix is $W \in \mathbb{R}^{k \times hwc}$ while the bias $b \in \mathbb{R}^k$ is a vector. Then the output is computed by

$$y = Wx + b.$$

Fully connected layers are frequently used as later layers of modern networks. Such layers are able to learn arbitrarily complex relationships between features as they are universal approximators, see [Hor91; Les+93]. However, this comes at the cost of a higher number of parameters, which makes them more expensive to use. Additionally, the improved expressiveness allows them to better fit the training data and thus increases the risk of overfitting.

Convolutional layers

Introduced in 1989 by Yann LeCun et al. [LB98] *convolutional layers* are used in most state-of-the-art neural networks for image classification. Their popularity surged after a GPU based implementation of a convolutional neural network (CNN) called AlexNet [KSH12] won the ImageNet Large Scale Visual Recognition Challenge in 2012.

The novelty of convolutional layers is that in contrast to fully connected layers they do not view the whole image at once but rather small patches of the images. This means they can focus on local patterns and by stacking them consecutively, earlier layers can learn simple features like edges and textures whereas the later layers learn more complex and abstract shapes.

This is accomplished by using *discrete convolutions*. The discrete convolution between two functions $f, g\colon \mathbb{Z} \longrightarrow \mathbb{R}$ is defined as

$$(f \star g)(n) = \sum_{m=-\infty}^{\infty} f(m)g(n-m).$$

In a convolutional layer, the input is discretely convolved with parameters. The parameters are also called a *kernel* in this context. In image classification this corresponds to moving the kernel over the image, element wise multiplying the kernel with the image and summing it up over all elements and all channels. See Figure 2.2 for a simple example. Note that the numbers in the kernel are only integers to simplify the example. Typically, the values in kernels and input are floating point numbers. These kernels are usually not that big, a common size is 3×3. Even kernel sizes are usually avoided because they create an imbalance and off-centre the image. There is, however, a reason for the use of even kernel sizes in our case which we discuss in Chapter 4. From now on, we refer to discrete convolutions as convolutions to ease the reading.

In image processing an image typically has more than one channel (e.g. values for red, green and blue in an RGB image). Each channel has its own kernel and all kernels together make up the *filter*, see Figure 2.3. To create more than one output channel, one then has to use several filters. One such output channel is called *feature map*, as it contains several features.

This means a convolution C with k output channels and weights $W \in \mathbb{R}^{m\times n\times c\times k}$ operates on an image $x \in \mathbb{R}^{h\times w\times c}$ with height h, width w and c channels like

$$(C(x))_{r,s,t} = \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{q=1}^{c} x_{r+i-1,s+j-1,q} \cdot W_{i,j,q,t}$$

where $r \in \{1, \dots, h-m+1\}, s \in \{1, \dots, w-n+1\}$ and $t \in \{1, \dots, k\}$.

From a different point of view, the convolution operation on one channel with a kernel $W \in \mathbb{R}^{m\times n}$ can also be written as multiplication with a sparse matrix C when the input

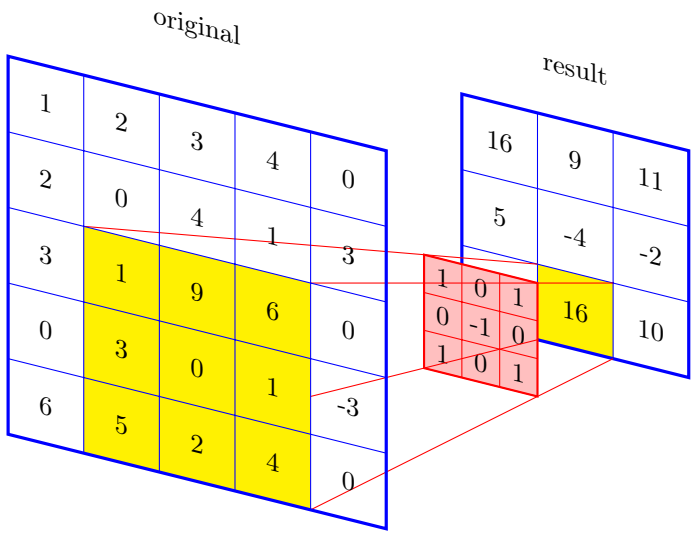


Figure 2.2 Operation of a convolution kernel on a matrix.

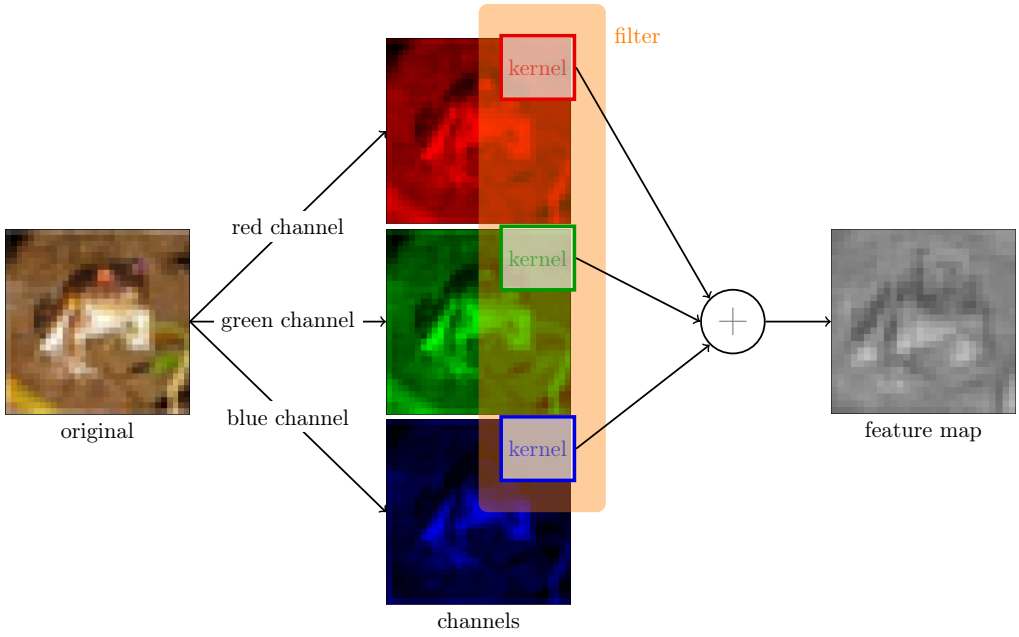


Figure 2.3 Illustration of the interplay of kernel, filter, channels and feature map in a convolutional layer using an image of a frog from CIFAR10 [Kri09] as an example.

has been flattened into a vector (instead of matrix form). This means an image with h rows and w columns

$$\begin{pmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,w} \\ x_{2,1} & x_{2,2} & \dots & x_{2,w} \\ & \ddots & & \\ x_{h,1} & x_{h,2} & \dots & x_{h,w} \end{pmatrix}$$

gets reshaped into a vector

$$\begin{pmatrix} x_{1,1} \\ x_{1,2} \\ \vdots \\ x_{1,w} \\ x_{2,1} \\ x_{2,2} \\ \vdots \\ x_{2,w} \\ \vdots \\ x_{h,1} \\ x_{h,2} \\ \dots \\ x_{h,w} \end{pmatrix} \in \mathbb{R}^{hw}.$$

Then $C(x) = Cx$. Further, assume $c_{1,1}, \dots, c_{m,n}$ to be a basis of the convolution matrices, where $c_{i,j}$ is the convolution matrix corresponding to convolution with the kernel that contains 0 except for in the ith row and jth column that contains a 1. Then we have the following representation of C, which will be useful in Section 4.3.2:

$$C(x) = \sum_{i=1}^{m} \sum_{j=1}^{n} W_{i,j} c_{i,j} x. \tag{2.5}$$

As an example, we discuss the form of a canonical basis of the 3×3 convolution matrices in detail. To ease the notation let us assume that there is only one channel. The case for more than one channel can be obtained by repeating the procedure for each channel. Because of the recurring patterns, we define smaller $(w-2) \times w$ matrices

$$X_1 = \begin{pmatrix} I & 0 & 0 \end{pmatrix}, \quad X_2 = \begin{pmatrix} 0 & I & 0 \end{pmatrix}, \quad X_3 = \begin{pmatrix} 0 & 0 & I \end{pmatrix} \in \mathbb{R}^{(w-2)\times w},$$

where I is the $(w-2) \times (w-2)$ identity matrix. Then for $j = 1, \dots 3$ we write

$$c_{1,j} = \begin{pmatrix} X_j & 0 & 0 & 0 & 0 \\ 0 & \ddots & 0 & 0 & 0 \\ 0 & 0 & X_j & 0 & 0 \end{pmatrix}, \qquad c_{2,j} = \begin{pmatrix} 0 & X_j & 0 & 0 & 0 \\ 0 & 0 & \ddots & 0 & 0 \\ 0 & 0 & 0 & X_j & 0 \end{pmatrix},$$

$$c_{3,j} = \begin{pmatrix} 0 & 0 & X_j & 0 & 0 \\ 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & X_j \end{pmatrix}$$

as $(h-2)(w-2) \times hw$ matrices.

As can be seen from the dimensions above, a convolution with kernel size greater than 1 always reduces the dimensions of the input. If the convolution kernel has the size $m \times n$ then the resulting image will have a height of $h - m + 1$ and a width of $w - n + 1$. To counteract this effect and get an output of the same size as the input, *padding* can be used. This means before the convolution is applied, $m - 1$ additional rows and $n - 1$ additional columns are added. The additional rows and columns are evenly distributed to the left and right borders, as well as the top and bottom borders of the image. If $m - 1$ is odd, it is common to add more rows to the bottom, respectively if $n - 1$ is odd, more columns to the right. See Figure 2.4 for an illustration of the concept.

There are different choices for the additional rows and columns. We use zero padding in our convolution layers. This means that $m - 1$ zero rows and $n - 1$ zero columns are

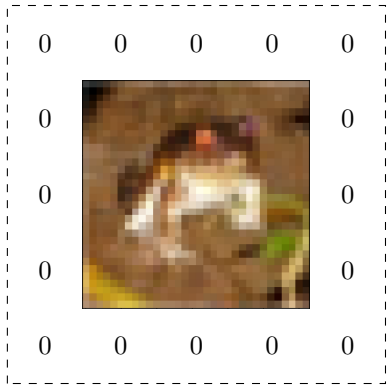


Figure 2.4 An image of a frog from [Kri09] padded with zeros. The output of a 3×3 convolution on the image inside the dashed line would be the same size as the original image of the frog.

added. In the 3×3 kernel example from above this would correspond to increasing the size of the X_i to

$$\tilde{X}_1 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & \ddots & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \qquad \tilde{X}_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

$$\tilde{X}_3 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \qquad 0 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

that are now quadratic $w \times w$ matrices. Then for $j = 1, \dots 3$ we write

$$c_{1,j}^{\text{zero}} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ \tilde{X}_j & 0 & 0 & 0 \\ 0 & \ddots & 0 & 0 \\ 0 & 0 & \tilde{X}_j & 0 \end{pmatrix}, \qquad c_{2,j}^{\text{zero}} = \begin{pmatrix} \tilde{X}_j & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \tilde{X}_j \end{pmatrix},$$

$$c_{3,j}^{\text{zero}} = \begin{pmatrix} 0 & \tilde{X}_j & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \tilde{X}_j \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

as $hw \times hw$ matrices.

Pooling layers

To reduce the dimensionality of the features but also preserve critical information, pooling layers are used in neural networks. Throughout a CNN usually the number of channels increases by the use of convolutional layers. To compensate for this, the size of the images is decreased by using pooling layers. This contributes to computational efficiency.

The most used pooling types are *average pooling* and *max pooling*. Similar to a convolutional layer both pooling layers use a non-overlapping sliding window over which the pooling operation is taken. As the names suggests, in max pooling the maximum value over the window is taken and in average pooling the average. This means for a window of size $s \times s$, and an input x of size $h \times w$, the output of max pooling is

$$\text{maxpool}(I)(i,j) = \max\{x(m,n) \mid m \in ((i-1)s, is] \cap \mathbb{Z}, n \in ((j-1)s, js] \cap \mathbb{Z}\}$$

and the output of average pooling is

$$\operatorname{avgpool}(I)(i,j) = \frac{1}{s^2} \sum_{m \in ((i-1)s, is] \cap \mathbb{Z}, n \in ((j-1)s, js] \cap \mathbb{Z}} x(m,n).$$

Average pooling has a smoothing effect thus reducing noise, in contrast to max pooling that emphasizes the most prominent features. Both kinds of pooling layers improve the spacial translation invariance of the network, as they can capture the presence of a feature in a certain region of the image. As with other aspects of the architecture, the best choice of the pooling layer depends on the given task and might differ from application to application.

2.2.2 Residual networks and their stability

In Section 2.2.1 we have already seen the evolution of several layer types to prevent problems during training like exploding or vanishing gradients. As it turned out, the layer type (2.4) could be improved. He et al. proposed in [He+16] a new layer type

$$R^l(x^l, W^l) = x^l + F^l\left(x^l, W^l\right) = x^l + C_2^l\left(w_3^l\right)\sigma^l\left(N^l\left(C_1^l\left(w_1^l\right)x^l, w_2^l\right)\right) \tag{2.6}$$

called *residual layers*, with the same layer $F^l\left(x^l, W^l\right)$ as in (2.4). These layers introduce a skipping connection by adding the input to the output. The authors found that this facilitates learning the identity mapping for the network. Residual networks, i.e. networks that employ residual layers, are very successful because they reduce vanishing gradients thus allowing deeper more complex networks. Further, [HR17a] have found that residual networks are stable under suitable assumptions.

In neural networks stability is relevant and desirable. Stability here means, that small changes in the inputs do not lead to large changes in the output. This is especially desirable when dealing with deliberately altered data, like adversarial examples. Here the input has been slightly perturbed such that it is unnoticeable to human observation but with the purpose of misleading a network to misclassify. Moreover, stability makes networks not only robust against noisy data but also helps with vanishing or exploding gradients, thus improving the networks overall performance.

To explain why residual networks are more stable, we interpret them as nonlinear differential equations. This approach allows us to establish criteria that determine the stability of forward propagation within a network by applying our knowledge of the stability of difference schemes, see [HR17a; Hab+18].

First we relate the forward propagation to differential equations. We now consider a network with several residual layers all using the same convolutions C_1, C_2, normalization

layer N and activation functions σ. Note that the weights of each layer still depend on l. According to Eq. (2.6), for $l = 0, \dots, L-1$ the forward propagation of the lth residual layer is

$$x_{l+1} = x_l + hC_2(w_3)\sigma(N(C_1(w_1)x_i, w_2)),$$

where x_l is the output of the previous layer i and we have introduced a step size h. This can be interpreted as an explicit Euler discretization of the nonlinear ordinary differential equation (ODE)

$$\begin{aligned} X'(t) &= C_2(W(t))\sigma(N(C_1(W(t))X(t), W(t))) \text{ for } t \in [0,T], \\ X(0) &= X_0, \end{aligned} \tag{2.7}$$

where the time t corresponds to moving through the different layers.

Before discussing the stability of the explicit Euler method, we discuss the stability of the ODE (2.7) itself. We give the definition of stability from [RH20].

Definition 2.5 ([RH20])**.** *The forward propagation in* (2.7) *is called* stable *if there is a constant $M > 0$ independent of T such that*

$$\left\| X(T) - \tilde{X}(T) \right\|_F \leq M \left\| X(0) - \tilde{X}(0) \right\|_F,$$

where $X, \tilde{X}$ are solutions of (2.7) *for different initial values $X_0, \tilde{X}_0$ and $\|\cdot\|_F$ is the Frobenius norm.*

One possible approach to a stable forward propagation is by using a slightly different layer

$$F^l(x^l, W^l) = -C_1(w_1^l)^\top \sigma(N(C_1(w_1^l)x^l, w_2^l)),$$

where $C_2(w_3^l)$ has been replaced by $-C_1(w_1^l)$. This symmetric layer type is introduced in [Hab+18]. The authors also prove that with this layer type the ODE is stable. We state their results here and disregard the normalization layer for the analysis of the stability for notational simplicity.

Theorem 2.6 ([RH20])**.** *Let σ be monotonically increasing, $C_1 : \mathbb{R}^{n_W} \longrightarrow \mathbb{R}^{n_{x^{l+1}} \times n_{x^l}}$, $W \colon \mathbb{R} \longrightarrow \mathbb{R}^{n_W}$ and $X_0 \in \mathbb{R}^{n_0}$. Then the ODE*

$$\begin{aligned} X'(t) &= -C_1(W(t))^\top \sigma(C_1(W(t))X(t)) =: \mathcal{F}(X(t), W(t)) \text{ for } t \in [0,T], \\ X(0) &= X_0, \end{aligned} \tag{2.8}$$

is stable.

Proof. Because σ is monotonically increasing, we have

$$(C_1(W(t))X(t) - C_1(W(t))\tilde{X}(t))^\top(\sigma(C_1(W(t))X(t)) - \sigma(C_1(W(t))\tilde{X}(t))) \geq 0.$$

Thus

$$\begin{aligned}
\partial_t \left\| X(t) - \tilde{X}(t) \right\|_F^2 &= 2(X(t) - \tilde{X}(t))^\top(\partial_t X(t) - \partial_t \tilde{X}(t)) \\
&= 2(X(t) - \tilde{X}(t))^\top(-C_1(W(t))^\top \sigma(C_1(W(t))X(t))) \\
&\quad + C_1(W(t))^\top \sigma(C_1(W(t))\tilde{X}(t)) \\
&= -2(X(t) - \tilde{X}(t))^\top C_1(W(t))^\top(\sigma(C_1(W(t))X(t)) \\
&\quad - \sigma(C_1(W(t))\tilde{X}(t))) \\
&= -2(C_1(W(t))X(t) - C_1(W(t))\tilde{X}(t))^\top(\sigma(C_1(W(t))X(t)) \\
&\quad - \sigma(C_1(W(t))\tilde{X}(t))) \\
&\leq 0.
\end{aligned}$$

Then we integrate

$$0 \geq \int_0^T \partial_t \left\| X(t) - \tilde{X}(t) \right\|_F^2 \mathrm{d}t = X(T) - \tilde{X}(T) - (X(0) - \tilde{X}(0))$$

which shows the stability of the function. □

Now we discuss the stability of the discretization resulting from the explicit Euler method.

Theorem 2.7 ([RH20])**.** *Let σ be monotonically increasing and differentiable, $C_1 : \mathbb{R}^{n_W} \longrightarrow \mathbb{R}^{n_{x^{l+1}} \times n_{x^l}}$ and $w_1 \in \mathbb{R}^{n_W}$. Then the explicit Euler discretization of* (2.8)

$$x_{l+1} = x_l - hC_1(w_1(t_l))^\top \sigma(C_1(w_1(t_l))x_l)$$

is stable for sufficiently small values of $h > 0$.

Proof. The stability of the explicit Euler method is well established and a standard numerics result. It is stable if

$$\max_{i=1,2,\ldots,n} |1 + h\lambda_i(J_X(t_l))| \leq 1 \tag{2.9}$$

for all $l = 0, 1, 2, \dots L$, where $\lambda_i(J_X(t_l))$ is the ith eigenvalue of the Jacobian matrix of $\mathcal{F}(X(t), W(t)$ with respect to X at time $t_l = lh$. For (2.9) to hold, the eigenvalues of $J_X(t_l)$ need to be non positive. It holds

$$J_X(t_l) = -C_1(w_1(t_l))^\top \operatorname{diag}\left(\sigma'(C_1(w_1(t_l))x)\right) C_1(w_1(t_l)).$$

Because σ is monotonically increasing, $J_X(t_l)$ is negative semi definite and symmetric. Thus all eigenvalues of $J_X(t_l)$ are real and bounded above by zero, i.e. we have

$$\max_{i=1,\dots,n} \operatorname{Re}(\lambda_i(J_X(t_l))) \leq 0.$$

Therefore there exists $h > 0$ sufficiently small such that (2.9) holds. □

Due to the results from the stability discussion in this section, from now on we assume convolutional layers to be of the form

$$F^l(x^l, W^l) = -C_1\left(w_1^l\right)^\top \sigma\left(N\left(C_1\left(w_1^l\right)x^l, w_2\right)\right),$$

and, analogously, residual layers of the form

$$R^l(x^l, W^l) = x^l + F^l(x^l, W^l). \tag{2.10}$$

2.3 Optimization methods for machine learning

Now that we have stated our optimization problem and discussed the structure of our model, the next step is to discuss the tools we have to train our model. In this section, we take a look at different optimization methods ranging from basic to more sophisticated methods.

To approach these methods in the most general way as possible, we consider the optimization problem

$$\min_{w \in \mathbb{R}^n} F(w) = \mathbb{E}_{\xi \sim \mathbb{P}}\left[f(w, \xi)\right], \tag{2.11}$$

where $F\colon \mathbb{R}^n \longrightarrow \mathbb{R}$ with $n \in \mathbb{N}$, a random vector $\xi \in \mathbb{R}^d$ having probability distribution $\mathbb{P}$ supported in a set $\Omega \subseteq \mathbb{R}^d$ and $f\colon \mathbb{R}^n \times \Omega \longrightarrow \mathbb{R}$. From now on we use the shorter notation

$$\mathbb{E}_\xi\left[f(w, \xi)\right] \coloneqq \mathbb{E}_{\xi \sim \mathbb{P}}\left[f(w, \xi)\right].$$

Note that problems of the form (2.3) fall under this generalization when assuming $\xi = (x, y)$ to be distributed according to a discrete probability distribution on the

samples (x_i, y_i) in the training set, where each sample has probability $\frac{1}{N}$, and choosing $f(w, (x, y)) = l(h(x; w), y)$.

As the following methods rely on the gradients of the objective function at the iterates, we assume the objective function to be differentiable. This assumption may not always hold, such as when using a ReLU activation function. However, the methods can remain applicable in these cases by employing subgradients instead. For an insight into nonsmooth optimization methods refer to [Cla90].

2.3.1 Gradient descent

The first most basic algorithm that comes to mind when trying to solve an optimization problem with a differentiable objective function is *gradient descent*. Here the gradient of the objective function is used as a search direction which is a guaranteed direction of descent. Therefore the update using a step size α is

$$x_{k+1} = x_k - \alpha \nabla F(x_k).$$

Gradient descent, or batch gradient descent as it is often called in the machine learning context, converges under certain conditions on α, like the Armijo condition, to a global minimum if the objective function is convex and to a local minimum if the objective function is nonconvex [UU12].

However, in supervised machine learning the objective function is usually based upon a dataset as established in Section 2.1.2, so the objective function looks like (2.3). This means that before updating the current iterate the gradient over the whole dataset has to be calculated. There are several disadvantages to this approach. First, it makes the method intractable for datasets that are too large to fit into memory. Second, new samples cannot be added during training, hence it cannot be used as an online algorithm. Third, and perhaps most significantly, it is slow and prone to getting stuck in local minima. Consequently it is common in the context of machine learning not to use the whole dataset for making an update, i.e. to use only a few samples from the dataset or even only one. This approach may be at the expense of accuracy for the sake of speed, as it only considers a subset of the dataset. In the following section, we examine this method and the necessary conditions for convergence.

2.3.2 Stochastic gradient descent (SGD)

To reduce the time before taking an update when addressing a machine learning problem like (2.11), it is common not to compute the (whole) gradient of the objective function but a stochastic gradient $g(w, \xi)$, as defined below, instead.

Definition 2.8. *A* stochastic gradient *of* $F\colon \mathbb{R}^n \longrightarrow \mathbb{R}$ *is a random vector* $g(w, \xi) \in \mathbb{R}^n$ *such that* $\mathbb{E}_\xi [g(w, \xi)] = \nabla F(w)$ *holds for all* $w \in \mathbb{R}^n$.

In machine learning such a stochastic gradient usually is derived using only a subset of the dataset, whether it consists of a single sample or multiple samples as explained in Section 2.3.1. This results in *stochastic gradient descent (SGD)* introduced by [RM51] which is the most prominent algorithm in machine learning and is shown in Algorithm 1.

Algorithm 1: Stochastic gradient descent (SGD)

Input: Learning rates α_k, oracle for stochastic vector g, number of maximum iterations M, initial point w_0
Output: w_M
for $k = 0, 1, 2, \ldots, M-1$ **do**
  Draw independently a sample ξ_k;
  Obtain $g(w_k, \xi_k)$;
  $d_k \leftarrow g(w_k, \xi_k)$;
  $w_{k+1} \leftarrow w_k - \alpha_k d_k$;
end
return w_M

Here it is also possible to include new samples later on, allowing online learning.

We follow the convergence analysis of [BCN18]. The first assumption is about the smoothness of the objective function, expressed by the Lipschitz continuity of its gradients.

Assumption 2.9. *The objective function* $F\colon \mathbb{R}^n \longrightarrow \mathbb{R}$ *is continuously differentiable and its gradient* $\nabla F\colon \mathbb{R}^n \longrightarrow \mathbb{R}^n$ *is Lipschitz continuous with Lipschitz constant* $L > 0$. *This means*

$$\|\nabla F(v) - \nabla F(w)\|_2 \leq L \|v - w\|_2 \text{ for all } v, w \in \mathbb{R}^n.$$

Assumption 2.9 ensures that the gradient of F does not change arbitrarily fast. As [BCN18] mention, this is relevant in most gradient based methods. Otherwise the step size must be controlled more carefully as the gradient may change unpredictably.

Here it might be necessary to make a few comments on the Lipschitz continuity of the gradients of (residual) neural networks. As they are implemented today, most of them use ReLU activation functions which are not differentiable in zero and whose gradients, where they exist, are not globally Lipschitz. This means, the gradients of the network are also not globally Lipschitz. However, it is worth noting, that the gradients of ReLU are locally Lipschitz if bounded away from zero.

Lemma 2.10. *Under Assumption 2.9 we have*

$$F(v) \leq F(w) + \nabla F(w)^\top (v - w) + \frac{1}{2} L \left\| v - w \right\|_2^2 .$$

Proof. It holds

$$\begin{aligned}
F(v) &= F(w) + \int_0^1 \frac{\partial F(w + t(v-w))}{\partial t} \mathrm{d}t \\
&= F(w) + \int_0^1 \nabla F(w + t(v-w))^\top (v - w) \mathrm{d}t \\
&= F(w) + \nabla F(w)^\top (v - w) + \int_0^1 \left(\nabla F(w + t(v-w)) - \nabla F(w) \right)^\top (v - w) \mathrm{d}t \\
&\leq F(w) + \nabla F(w)^\top (v - w) + \int_0^1 L \left\| t(v-w) \right\|_2 \left\| v - w \right\|_2 \mathrm{d}t \\
&\leq F(w) + \nabla F(w)^\top (v - w) + \frac{1}{2} L \left\| v - w \right\|_2^2 ,
\end{aligned}$$

which finishes the proof. □

As it turns out, it is not even necessary to use a stochastic gradient in stochastic gradient descent. Instead it is sufficient to use a stochastic vector $g(w, \xi)$ that fulfills the following assumptions:

Assumption 2.11. *The objective function and the SGD algorithm satisfy the following:*

a) The sequence of iterates $\{w_k\}$ is contained in an open set over which F is bounded below by a scalar F_{inf}.

b) There exist scalars $\mu_G \geq \mu > 0$ such that for all $k \in \mathbb{N}$,

$$\nabla F(w_k)^\top \mathbb{E}_\xi [g(w_k, \xi)] \geq \mu \| \nabla F(w_k) \|_2^2 \quad \text{and} \tag{2.12}$$

$$\| \mathbb{E}_\xi [g(w_k, \xi)] \|_2 \leq \mu_G \| \nabla F(w_k) \|_2 . \tag{2.13}$$

c) There exist scalars $M \geq 0$ *and* $M_V \geq 0$ *such that, for all* $k \in \mathbb{N}$,

$$\mathbb{V}_\xi[g(w_k, \xi)] \leq M + M_V \|\nabla F(w_k)\|_2^2, \tag{2.14}$$

where the variance of the random vector $g(w_k, \xi)$ *is*

$$\mathbb{V}_\xi[g(w_k, \xi)] \coloneqq \mathbb{E}_\xi[\|g(w_k, \xi)\|_2^2] - \|\mathbb{E}_\xi[g(w_k, \xi)]\|_2^2 .$$

In the previous assumption, Eq. (2.12) states that $-g(w_k, \xi)$ should be a descent direction in expectation to $F(w_k)$ or rather that the expected angle between $g(w_k, \xi)$ and $\nabla F(w_k)$ should be large. This is clearly fulfilled if $g(w_k, \xi)$ is a stochastic gradient of $\nabla F(w_k)$ meaning $\mathbb{E}_\xi[g(w_k, \xi] = \nabla F(w_k)$. Eq. (2.13) assumes that the norm of the first moment of $g(w_k, \xi)$ behaves as the norm of $\nabla F(w_k)$.

The variance of search directions can significantly vary depending on the quality of the stochastic direction, such as the number of samples used to compute it. Eq. (2.14) sets a condition on this variance, bounding it from above. Using more samples for example might reduce the constant M thus giving better search directions. On the other hand, the inherent noise and randomness in SGD can help to escape local minima [Rud16].

From Assumption 2.11 we can derive:

$$\mathbb{E}_{\xi_k}[\|g(w_k, \xi_k)\|_2^2] \leq M + M_G \|\nabla F(w_k)\|_2^2 \quad \text{with } M_G \coloneqq M_V + \mu_G^2 \geq \mu^2 > 0. \tag{2.15}$$

Depending on the step size choice, there are different convergence results for SGD. Unlike gradient descent, SGD fluctuates more. Without decreasing the step size, SGD may not converge to a minimum but rather oscillate around it. Therefore, it is necessary for almost sure convergence to a stationary point to decrease the step size slowly. Each w_k is determined by the independent realizations of all previous random variables, i.e. $\xi_1, \ldots, \xi_{k-1}$. Thus, the total expectation of $F(w_k)$ must be taken with respect to the joint distribution of all random variables, this means in our case

$$\mathbb{E}_{\xi_1, \ldots, \xi_{k-1}}[F(w_k)] = \mathbb{E}_{\xi_1} \mathbb{E}_{\xi_2} \ldots \mathbb{E}_{\xi_{k-1}}[F(w_k)].$$

Now we can state the convergence result for classic SGD when using fixed step sizes.

Theorem 2.12 (Convergence for classic SGD with fixed step size)**.** *Under Assumption 2.11, suppose that SGD (Algorithm 1) is run with fixed step size* $0 \leq \alpha_k = \alpha \leq \frac{\mu}{LM_G}$*. Then the following holds:*

$$\mathbb{E}_{\xi_1,\ldots,\xi_{K-1}}\left[\frac{1}{K}\sum_{k=1}^{K}\|\nabla F(w_k)\|_2^2\right] \leq \frac{\alpha LM}{\mu} + \frac{2(F(w_0)-F_{\inf})}{K\mu\alpha} \xrightarrow[K\to\infty]{} \frac{\alpha LM}{\mu}.$$

Theorem 2.13 (Convergence for classic SGD with diminishing step sizes)**.** *Under Assumption 2.11, suppose that SGD (Algorithm 1) is run with diminishing step sizes that satisfy*

$$\sum_{k=1}^{\infty}\alpha_k = \infty \text{ and } \sum_{k=1}^{\infty}\alpha_k^2 < \infty.$$

Define $A_K \coloneqq \sum_{k=1}^{K}\alpha_k$*. Then the following holds:*

$$\mathbb{E}_{\xi_1,\ldots,\xi_{K-1}}\left[\frac{1}{A_K}\sum_{k=1}^{K}\alpha_k\|\nabla F(w_k)\|_2^2\right] \xrightarrow[K\to\infty]{} 0.$$

Proof. The proof for both theorems can be found in Bottou, Curtis, Nocedal, 2016 [BCN18]. □

The result in Theorem 2.12 shows that for fixed step sizes the convergence of SGD depends on the noise in the data, the step sizes chosen and the initial point. When there is no noise or the noise is proportional to $\|F(w_k)\|_2^2$ then $M = 0$ and the expected value of the mean of $\|\nabla F(w_k)\|_2^2$ converges to zero. When $M > 0$ then a constant term remains that can be reduced by decreasing the step size. However, in that case it will take SGD longer to approach a minimum.

In the case of diminishing step sizes Theorem 2.13 implies that in that case $\mathbb{E}_{\xi_1,\ldots,\xi_{K-1}}[\|F(w_k)\|_2^2]$ cannot be bounded away from zero, i.e.

$$\liminf_{k\to\infty}\mathbb{E}_{\xi_1,\ldots,\xi_{K-1}}\left[\|F(w_k)\|_2^2\right] = 0.$$

In contrast to the fixed step size case, when using the diminishing step sizes from Theorem 2.13 the infimum limit of the expected value of the gradient norms at the iterates is zero even if $M > 0$.

In the special case of a strongly convex objective, [BCN18] give an improved result. This result can also be used to understand the behavior of SGD near a strong local minimizer. This is true even if F is not strongly convex but smooth. Thus we introduce a new assumption.

Assumption 2.14. *The objective function $F\colon \mathbb{R}^d \longrightarrow \mathbb{R}$ is strongly convex, i.e., there exists a constant $\gamma > 0$ such that for all $v, w \in \mathbb{R}^d$ it holds*

$$F(v) \geq F(w) + \nabla F(w)^\top (v - w) + \frac{1}{2}\gamma \|v - w\|_2^2 .$$

In that case, F has a unique minimizer which we denote by w^*.

With that assumption, [BCN18] state the following convergence result.

Theorem 2.15 (Convergence for classic SGD on a strongly convex objective with fixed step sizes)**.** *Under Assumptions 2.11 and 2.14, suppose that SGD (Algorithm 1) is run with fixed step size $0 \leq \alpha_k = \alpha \leq \frac{\mu}{LM_G}$. Then the following holds for all $k \in \mathbb{N}$:*

$$\mathbb{E}_{\xi_1,\ldots,\xi_{K-1}} \left[F(w_k) - F(w^*)\right] \leq \frac{\alpha LM}{2\gamma\mu} + (1 - \alpha\gamma\mu)^{k-1} \left(F(w_0) - F(w^*) - \frac{\alpha LM}{2\gamma\mu}\right).$$

Proof. The proof for the theorem can be found in Bottou, Curtis, Nocedal, 2016 [BCN18]. □

Note here that even in this strongly convex case, Theorem 2.15 only states convergence to a neighborhood of the solution, not the solution itself.

2.3.3 Adaptive moment estimation (Adam)

Even though standard stochastic gradient descent, as discussed in the section before, proves to be more effective in machine learning scenarios than gradient descent, it still struggles with some challenges. In this subsection we discuss advanced versions of SGD that address some of its shortcomings leading up to one of the currently most popular and widely used optimization methods in deep learning: Adam.

Momentum based gradient methods

In some cases, when the slope in some directions is much steeper than in others, SGD can oscillate [Sut86] thus making only slow progress. Adding momentum like introduced in [Pol64] helps SGD to traverse such areas [Qia99]. Here a fraction of the previous step is also added to the current step resulting in

$$\begin{aligned} s_k &= \gamma s_{k-1} + \alpha \nabla F(x_k) \\ w_{k+1} &= w_k - s_k, \end{aligned}$$

where $s_0 = 0$. This way, when subsequent gradients are consistently pointing in the same direction then bigger steps are taken in that direction. Otherwise, the steps in the directions are reduced [Rud16]. This aligns with the idea that a ball rolling down a hill also possesses momentum resulting in less sharp turns and decreased oscillations. Additionally, it also helps to still make progress even when the gradients become small.

Adaptive methods

As stated above, the convergence of SGD is highly dependent on the selection of an appropriate step size. However, choosing an appropriate step size can be a challenging task, given the high dimensional and nonlinear optimization landscape of deep networks [Li+18]. For that reason adaptive step size methods like *Adagrad*, introduced in [DHS11], have been proposed. The idea is to differentiate between the updates for each parameter allowing for individualized step sizes based on the importance of each parameter. For instance, Adagrad achieves this by dividing the step size for each parameter by a moving average of squared past gradient norms.

When combining this approach with momentum, this yields the currently popular and widely used *Adaptive Moment Estimation (Adam)* introduced in [KB14]. Adam uses a moving average of the gradients as well as of the squared gradients thus offering the benefits of both approaches. As the original convergence proof of Adam contains an error as pointed out in [RKK18], we use the slightly more general notation from [Déf+22] that also covers Adagrad.

Assuming, we have $0 \leq \beta_1 < \beta_2 \leq 1$ and non-negative step sizes $(\alpha_k)_{k\in\mathbb{N}}$. Then for each component, we define

$$m_{k+1,i} = \beta_1 m_{k,i} + (g(w_k, \xi_k))_i, \tag{2.16}$$

$$v_{k+1,i} = \beta_2 v_{k,i} + (g(w_k, \xi_k))_i^2, \tag{2.17}$$

$$w_{k+1,i} = w_{k,i} - \alpha_k \frac{m_{k+1,i}}{\sqrt{v_{k+1,i} + \varepsilon}}, \tag{2.18}$$

where m_0, v_0 are initialized as zero vectors. This initialization leads to a bias towards 0 which is why the original Adam algorithm from [KB14] uses corrective terms instead.

We can recover the original definition of Adam with the corrective terms by using the step size

$$\overline{\alpha}_n = \alpha \cdot \frac{1-\beta_1}{\sqrt{1-\beta_2}} \cdot \frac{1}{1-\beta_1^n} \cdot \sqrt{1-\beta_2^n}. \tag{2.19}$$

Here, $\frac{1}{1-\beta_1^n}$ corresponds to the corrective term for m_k and $\sqrt{1-\beta_2^n}$ to the one for v_k.

If one chooses $\beta_1 = 0, \beta_2 = 1$ and $\alpha_k = \alpha$ this yields Adagrad.

In [Déf+22] the authors use the step size

$$\alpha_k = \alpha(1-\beta_1)\sqrt{\frac{1-\beta_2^k}{1-\beta_2}}$$

in their convergence analysis, where the corrective term for m_k from (2.19) is neglected, simplifying their proof. Both cases for Adagrad and Adam are covered in Algorithm 2.

Algorithm 2: Adaptive method with optional momentum

Input: Learning rates α_k, oracle for stochastic vector g, number of maximum iterations M, initial point w_0
Output: w_M
$m_0 \leftarrow 0$;
$v_0 \leftarrow 0$;
for $k = 0, 1, 2, \ldots, M-1$ **do**
  Draw independently a sample ξ_k;
  Obtain $g(w_k, \xi_k)$;
  for $i = 0, 1, 2, \ldots, n$ **do**
    $m_{k+1,i} \leftarrow \beta_1 m_{k,i} + (g(w_k, \xi_k))_i$;
    $v_{k+1,i} \leftarrow \beta_2 v_{k,i} + (g(w_k, \xi_k))_i^2$;
    $w_{k+1,i} \leftarrow w_{k,i} - \alpha_k \frac{m_{k+1,i}}{\sqrt{v_{k+1,i}+\varepsilon}}$;
  end
end
return w_M

To denote the iteration at which we examine the expected value of the squared norm of the gradients at the iterates, we define an index τ_N in $\{0, \ldots, N-1\}$.

Definition 2.16. *Denote with $\tau_N \in \mathbb{N}$ an index in $\{0, \ldots, N-1\}$ such that for all $j \in \mathbb{N}, j < N$ the probability $\mathcal{P}[\tau_N = j]$ is proportional to $1 - \beta_1^{N-j}$.*

When $\beta_1 = 0$ then τ_N is sampled uniformly. As the value of $\beta_1 > 0$ increases, smaller indices for τ_N are favored.

Similar assumptions to Assumptions 2.9 and 2.11 for SGD are needed. Again we require Lipschitz continuity of the gradients and lower boundedness of the objective function. Additionally, the stochastic gradients must be uniformly almost surely bounded.

Assumption 2.17. *The objective function and the stochastic gradients satisfy the following:*

1. *F is bounded below by a scalar F_{inf}.*

2. *The objective function $F\colon \mathbb{R}^n \longrightarrow \mathbb{R}$ is continuously differentiable and its gradient $\nabla F\colon \mathbb{R}^n \longrightarrow \mathbb{R}^n$ is Lipschitz continuous with Lipschitz constant $L > 0$.*

3. *There is a scalar $R \geq 0$ such that the stochastic gradients $g(w, \xi)$ are bounded in the following sense:*

$$\|g(w,\xi)\|_\infty \leq R - \sqrt{\epsilon} \quad \text{a.s. for all } w \in \mathbb{R}^n$$

for an $\epsilon > 0$.

Under these assumptions the authors state the following results:

Theorem 2.18 (Adagrad with momentum)**.** *Let Assumption 2.17 hold. Let $\beta_2 = 1, \alpha_k = \alpha > 0$ and $0 \leq \beta_1 < 1$. Let the iterates w_k be chosen as in* (2.18)*. Then for any natural number $N > \frac{\beta_1}{1-\beta_1}$ we have*

$$\mathbb{E}\left[\|\nabla F(w_{\tau_N})\|_2^2\right] \leq 2R\sqrt{N}\frac{F(x_0) - F_{\inf}}{\alpha\tilde{N}} + \frac{\sqrt{N}}{\tilde{N}} E \ln\left(1 + \frac{NR^2}{\epsilon}\right),$$

where

$$\tilde{N} = N - \frac{\beta_1}{1-\beta_1} \quad \text{and} \quad E = \alpha n R L + \frac{12nR^2}{1-\beta_1} + \frac{2\alpha^2 n L^2 \beta_1}{1-\beta_1}.$$

Theorem 2.19 (Adam with momentum)**.** *Let Assumption 2.17 hold. Let $0 \leq \beta_1 < \beta_2 \leq 1$ and $\alpha_k = \alpha(1-\beta_1)\sqrt{\frac{1-\beta_2^n}{1-\beta_2}}$ with $\alpha > 0$. Let the iterates w_k be chosen as in* (2.18)*. Then for any natural number $N > \frac{\beta_1}{1-\beta_1}$ we have*

$$\mathbb{E}\left[\|\nabla F(w_{\tau_N})\|_2^2\right] \leq 2R\frac{F(x_0) - F_{\inf}}{\alpha\tilde{N}} + E\left(\frac{1}{\tilde{N}} \ln\left(1 + \frac{R^2}{(1-\beta_2)\epsilon}\right) - \frac{N}{\tilde{N}} \ln(\beta_2)\right),$$

where

$$\tilde{N} = N - \frac{\beta_1}{1-\beta_1}$$

and

$$E = \frac{\alpha n R L (1-\beta_1)}{(1-\beta_1/\beta_2)(1-\beta_2)} + \frac{12nR^2\sqrt{1-\beta_1}}{(1-\beta_1/\beta_2)^{3/2}\sqrt{1-\beta_2}} + \frac{2\alpha^2 n L^2 \beta_1}{(1-\beta_1/\beta_2)(1-\beta_2)^{3/2}}.$$

Proof. The proof of both theorems can be found in Défossez et al., 2022 [Déf+22]. □

As N goes to infinity, the terms involving N become dominant. For Adam this means the terms containing $\frac{1}{\tilde{N}}$ determine the convergence rate. It should be noted that the last

term $\frac{N}{\tilde{N}} \ln(\beta_2)$ does only go to zero if β_2 goes to one. However, far from a solution the first term dominates the convergence which decreases at a rate of $\mathcal{O}(1/N)$.

On the other hand, for Adagrad, as N goes to infinity $\frac{\sqrt{N}}{\tilde{N}}$ goes to zero but only at a rate of $\mathcal{O}(1/\sqrt{N})$. That difference between $\mathcal{O}(1/N)$ and $\mathcal{O}(1/\sqrt{N})$ might be why Adam is usually the better choice when comparing Adam and Adagrad in practice. However, their performance depends on the given optimization problem, the data and the specific task.

One advantage of adaptive methods is that the step size, or learning rate, does not have to be tuned manually anymore. In comparison to SGD, Adam may have a faster convergence in addition to better training performance in practice. Then again, SGD is easier to implement, less computationally expensive because of its simplicity and also more memory efficient as it does not have to save past gradients. In both cases one has hyperparameters that must be tuned for best performance. Nonetheless, providing a definitive answer to the question of which optimization method to use is hard. Instead, one must rather select the optimization method best suited for the problem at hand. In most modern implementations Adam is the default choice that provides a satisfactory performance.

2.3.4 Stochastic variance reduced gradient (SVRG)

Because of the variance of the stochastic directions used in SGD, as established in Section 2.3.2, SGD can oscillate around the optimal solution. For the case of fixed step sizes, this is results in the remaining constant in Theorem 2.12 or the need for reducing the step sizes over time as in Theorem 2.13. Adjusting the method, such as using a larger batch or adding momentum, reduces the variance in the updates which allows the use of larger step sizes, thus accelerates convergence. However, one can also reduce the variance explicitly by using a variation reduction term. This results in a variant of SGD called *stochastic variance reduced gradient (SVRG)* that was established and examined by Johnson and Zhang in [JZ13].

Instead of using $g(w_k, \xi_k)$ for an update, every $m \in \mathbb{N}$ iterations a reference point $\tilde{w} \in \mathbb{R}^n$ is chosen for which $\nabla F(v)$ is computed and the update is performed with

$$\tilde{g}(w_k, \xi_k) = g(w_k, \xi_k) + (\nabla F(\tilde{w}) - g(\tilde{w}, \xi_k)).$$

The full algorithm is shown in Algorithm 3.

The authors of [JZ13] showed that when the objective function is strongly convex, SVRG can have a linear convergence rate.

Algorithm 3: Stochastic variance reduced gradient

Input: Learning rates α, oracle for stochastic vector g, number of maximum iterations M, initial point $\tilde{w}_0$
Output: $\tilde{w}_M$
for $s = 0, 1, 2, \ldots, M-1$ **do**
  $\tilde{w} \leftarrow \tilde{w}_{s-1}$;
  $G \leftarrow \nabla F(\tilde{w})$;
  $w_0 \leftarrow \tilde{w}$;
  for $k = 0, 1, 2, \ldots, m-1$ **do**
    Draw independently a sample ξ_k;
    Obtain $g(w_k, \xi_k)$;
    $w_{k+1} \leftarrow w_k - \alpha \left(g(w_k, \xi_k) - g(\tilde{w}, \xi_k) + G\right)$;
  end
  $\tilde{w}_s \leftarrow w_k$ for randomly chosen $k \in \{0, 1, \ldots, m\}$;
end
return $\tilde{w}_M$

Theorem 2.20 (SVRG for strongly convex functions)**.** *Let f be Lipschitz continuous with constant L and convex and let F be strongly convex with constant γ, i.e., satisfying Assumption 2.14. Let $w^* = \operatorname{argmin}_w F(w)$. Assume that the step size α and the length of the inner cycle m fulfill*

$$\eta = \frac{1}{\gamma\alpha(1 - 2L\alpha)m} + \frac{2L\alpha}{1 - 2L\alpha} < 1.$$

Then after s outer iterations we have

$$\mathbb{E}[F(\tilde{w}_s)] \leq \mathbb{E}[F(w^*)] + \eta^s [F(\tilde{w}_0) - F(w^*)].$$

Proof. The proof of this theorem can be found in Johnson, Zhang, 2013 [JZ13]. □

Comparing Theorem 2.20 to Theorem 2.15, we notice that the improvement introduced by variance reduction is the ability to achieve true convergence to the solution. In contrast, Theorem 2.15 shows that without reducing the step size, the iterates produced by SGD may not converge to the solution but instead remain in a neighborhood of the solution with a residual error of $\frac{\alpha L M}{2\gamma\mu}$. This is a crucial limitation of SGD. Furthermore, SVRG allows for a greater flexibility in choosing the step size by controlling the step size with the length of the inner cycle. This allows for larger step sizes and thus faster convergence making SVRG a significant improvement of SGD for strongly convex problems. On the downside however, the computation of the full gradient is very costly. This is especially true for large-scale problems where frequent full gradient computations can negate the benefits of faster convergence. In these cases, the simpler and less expensive SGD may be preferred despite its inaccurate convergence.

CHAPTER 3

A Stochastic Multilevel Optimization Approach

In some of the applications seen in Chapter 2, it is possible to establish a hierarchy of decreasing complexity for the inputs. Inputs with lower complexity are computationally cheaper to work with in optimization algorithms. We can use this to our advantage by applying a multilevel optimization method, which can speed up the training process. However, other multilevel optimization algorithms require the computation of the gradients or the Hessian, which can be very expensive. Therefore, a novel stochastic multilevel approach is needed.

This chapter introduces the algorithm that we consider. First we give an overview of multilevel methods in general in Section 3.1. Our main contribution is a novel multilevel stochastic gradient algorithm which we discuss in Section 3.2. In Section 3.3 we analyse the convergence of the algorithm where we differentiate between using the gradient in the gradient correction and using stochastic directions instead.

3.1 Multilevel methods

Before we introduce our algorithm, we give a quick history of multigrid methods and multilevel optimization so far.

3.1.1 Multigrid methods

Multigrid methods are iterative numerical methods used to solve partial differential equations (PDEs) or linear algebraic systems arising from discretizing PDEs. They use a hierarchy of grids with different resolutions to accelerate the convergence of iterative solvers. This makes them especially useful in large scale problems.

Multigrid methods date back to the middle of the 20th century when the idea to solve problems on differently sized grids emerged. Fedorenko introduced the first twogrid [Fed62] and multigrid methods [Fed64]. Brandt [Bra73; Bra77] and Hackbusch [Hac78; Hac76] popularized the geometric multigrid approach by showcasing its effieciency in solving elliptic PDEs. Later, algebraic multigrid methods were introduced in [BMR85] as a way to solve linear systems using multigrid principles without explicit knowledge of the geometry of the problem.

For a general introduction to multigrid methods and their historical development we refer the reader to the references [Hac85; Bra93].

When discussing multilevel optimization, it is helpful to distinguish the term multilevel from multigrid and multiscale. This distinction allows for a more precise understanding of the subject matter. Here we follow the distinction made in [Sha08]. Multigrid describes a numerical method used to solve PDEs using a hierarchy of grids. Multiscale emphasizes the existence of several scales in one problem whereas multilevel is a broad term referring to a system of abstract hierarchies. In our algorithm there is no notion of grid hence we do not use the term multigrid. Instead, we use the term multilevel since the algorithm covers hierarchies of different types expressed by different objective functions as we see later on.

V-Cycles, F-Cycles and W-Cycles

Depending on the sequence of operations performed at each level, multilevel methods use different cycle patterns. The most common ones are V-cycles, F-Cycles and W-cycles depicted schematically in Figure 3.1. The following overview is based on the insights presented in [TOS01].

The V-cycle pictured in Figure 3.1a is one of the earliest structures in multigrid. Due to its simple and straightforward approach, the V-cycle remains fundamental in modern multilevel techniques yielding a fast convergence for a wide range of problems. It is characterized by two phases. First a downward pass, where a relaxation is applied at every level and then an upward phase where the coarse grid step is corrected.

The W-cycle is an extension of the V-cycle with additional recursive calls of the relaxed problem, where the resulting shape is more complex and resembles that of the letter 'W'. They were introduced to improve the convergence of multigrid methods for certain types of problems. It is depicted in Figure 3.1b.

The F-cycle, or full multigrid cycle, is also an extension of the V-cycle and introduced for even faster convergence and more robustness. It has less recursive calls than a W-cycle as one can see in Figure 3.1c.

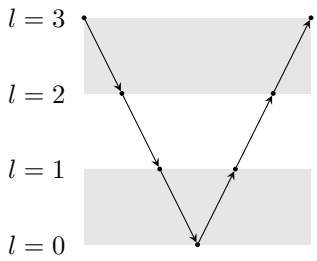
With regards to computational expenses, a V-cycle structure is the cheapest followed by the F-cycle. The W-cycle is the most expensive one as it has the most recursive calls. The easy-to-implement structure and the low costs of the V-cycle make it well suited for problems that do not require a very sophisticated approach. The algorithm we propose uses a V-cycle structure because it is easier to prove convergence. The reason for this is that, for convergence, the steps taken at the lower level should be small, as we will see in the next sections, and this is easier to control with a V-cycle structure.

3.1.2 Multilevel optimization

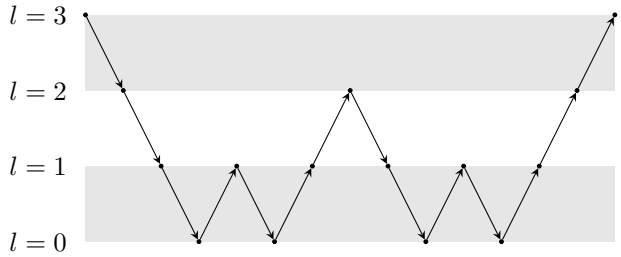
Following the success of multigrid methods, several multilevel optimization techniques have been developed to improve the convergence of different optimization methods. Even though similar in name, these multilevel methods are not to be confused with bilevel optimization problems where one problem is nested within another.

An intuitive way to apply multigrid methods in optimization is to use it to solve the Newton equation. However, as Newton's method is less effective when the initial point is far from the solution, Nash introduced in [Nas00] a multilevel line search method, MG/OPT, that is designed for global convergence on unconstrained convex optimization problems. Instead of using multigrid methods to solve linear systems, it uses a hierarchy of objective functions to find the solution. For first order consistency in the neighborhood of the initial point, a gradient correction is applied to the coarse level. This together with the convexity of the functions guarantees descent in the fine level when using a line search approach.

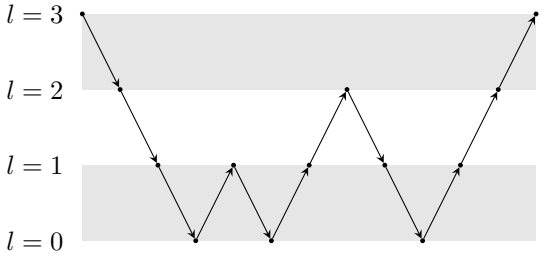
In [Bor05], Borzì proved the convergence of MG/OPT for the case that during the multigrid cycle at least one iteration of the optimization method was performed, either before or after descending to a lower level. This way these intermediate steps can be related to the pre-smoothing, respectively post-smoothing, in multigrid methods. To extend MG/OPT to non convex problems, Wen and Goldfarb used in [WG10] a new condition on the step size using backtracking if the condition is not fulfilled. For the case of non smooth but convex multilevel optimization, Parpas et al. used a multilevel iterative shrinkage thresholding algorithm (ISTA) in [Par17]. They applied it to a large



(a) V-Cycle.



(b) W-Cycle.



(c) F-Cycle.

Figure 3.1 Cycle types on three levels $l = 0, 1, 2, 3$.

scale image restoration problem using the image resolution to establish a hierarchy. This use case is similar to ours which we discuss in Chapter 4.

In contrast to line search methods, trust region methods take a different approach at finding a good step size. There have also been applications of multilevel approaches to trust region methods for unconstrained optimization problems in [GST08] and later on for constrained problems in [Gra+08]. These multilevel trust region methods are based on the MG/OPT method which is why many ideas coincide, for example first order coherence is again ensured by a gradient correction term. In the trust region context however, the size of the step is regulated by trust regions on each level and the coarse level step must fulfill a quality control before being accepted. These methods are shown to converge to first order critical points if the objective function is twice continuously

differentiable [GST08] or continuously differentiable [GK09]. Variants of these multilevel trust region methods have also been applied in deep learning settings, see for example [GKT23; KK23].

Our algorithm is also based on the techniques presented in [GST08] and thus on the MG/OPT method.

3.2 The multilevel stochastic gradient method

We are still considering the Problem (2.11) from Section 2.3, i.e.

$$\min_{w\in\mathbb{R}^n} F(w) = \mathbb{E}_\xi\left[f(w,\xi)\right]$$

but we now assume that there is a hierarchy of approximations $f_l : \mathbb{R}^{n_l} \times \Omega \longrightarrow \mathbb{R}$ to f for $l = 0, 1, \ldots, L$ and $n_0 \leq n_1 \leq \ldots \leq n_{L-1} \leq n_L = n \in \mathbb{N}$, where on the finest level the approximation is the function itself, i.e., $f_L = f$. This way we also have a hierarchy of approximations F_l to F by $F_l(w_l) \coloneqq \mathbb{E}_\xi\left[f_l(w_l,\xi)\right]$.

Let $\langle\cdot,\cdot\rangle$ denote the standard inner product in $\mathbb{R}^{n_l}$ and $\|\cdot\|$ denote the Euclidean norm. Then we assume to have level dependent vector norms $\|\cdot\|_l$ and inner products $\langle\cdot,\cdot\rangle_l$ for $l = 0, \ldots, L$ for which there exists h_l such that $1 = h_L \leq h_{L-1} \leq \ldots \leq h_0$ and

$$\|\cdot\|_l = \sqrt{h_l}\,\|\cdot\| \text{ and } \langle\cdot,\cdot\rangle_l = h_l\langle\cdot,\cdot\rangle.$$

Thus, denote with $V_l = (\mathbb{R}^{n_l}, \langle\cdot,\cdot\rangle_l)$ the Hilbert space $\mathbb{R}^{n_l}$ equipped with the inner product $\langle\cdot,\cdot\rangle_l$.

Further, we denote the gradient of F with respect to the inner product $\langle\cdot,\cdot\rangle_l$ of $F_l : V_l \longrightarrow \mathbb{R}$ by $\nabla_l F_l(w)$ which is the uniquely given vector, that satisfies

$$\langle\nabla_l F_l(w), u\rangle_l = F_l'(w)u \quad \text{for all } u \in V_l,$$

where $F_l'(w)$ denotes the Jacobian matrix of F_l.

3.2.1 Prolongation and restriction

Given a problem which is defined in varying complexity and accuracy on several levels, operators that transfer information between these different levels are necessary. In the context of multilevel methods the downward operators that reduce complexity are called *restriction* while the upward operators that increase complexity are known as

prolongations. In our case we define these operators for the optimization variables, which are the so-called weights in machine learning.

We assume to have variables w_l on ordered levels of complexity $l = 0, 1, \ldots, L$. Then $P_{l+1} \colon V_l \longrightarrow V_{l+1}$ is the prolongation taking input data to a higher level, i.e.,

$$P_{l+1}(w_l) = w_{l+1} \in \mathbb{R}^{n_{l+1}}.$$

Similarly, the restriction $R_l \colon V_l \longrightarrow V_{l-1}$ decreases the complexity of input data and variables, i.e.

$$R_l(w_l) = w_{l-1} \in \mathbb{R}^{n_{l-1}}.$$

In our application, the restriction and prolongation are linear operators. Therefore, to reduce the amount of parentheses used, we drop the brackets and we write $P_{l+1}w_l$ and $R_l w_l$.

Further, we define the operator norm of P_{l+1} by

$$\begin{aligned}
\|P_{l+1}\|_{l,l+1} &\coloneqq \sup_{\|w_l\|_l \leq 1} \|P_{l+1}w_l\|_{l+1} \\
&= \sup_{\|w_l\| \leq \frac{1}{\sqrt{h_l}}} \sqrt{h_{l+1}}\, \|P_{l+1}w_l\| \\
&= \frac{\sqrt{h_{l+1}}}{\sqrt{h_l}}\, \|P_{l+1}\|\,,
\end{aligned}$$

where $\|P_{l+1}\| \coloneqq \sup_{\|w_l\| \leq 1} \|P_{l+1}w_l\|$ is the operator norm induced by the euclidean norm. Moreover, we let P^* denote the adjoint operator $P_l^* \colon V_l \longrightarrow V_{l-1}$ defined by

$$\langle P_l w_{l-1}, w_l \rangle_l = \langle w_{l-1}, P_l^* w_l \rangle_{l-1},$$

where $w_{l-1} \in V_{l-1}$ and $w_l \in V_l$. Note that since

$$\begin{aligned}
\langle P_l w_{l-1}, w_l \rangle_l &= h_l \langle P_l w_{l-1}, w_l \rangle \\
&= h_l \langle w_{l-1}, P_l^\top w_l \rangle \\
&= \frac{h_l}{h_{l-1}} \langle w_{l-1}, P_l^\top w_l \rangle_{l-1} \\
&= \langle w_{l-1}, \frac{h_l}{h_{l-1}} P_l^\top w_l \rangle_{l-1},
\end{aligned}$$

we have

$$P_l^* = \frac{h_l}{h_{l-1}} P_l^\top.$$

For the adjoint, we define the operator norm similarly

$$\begin{aligned}\left\|P_{l+1}^*\right\|_{l+1,l} &:= \sup_{\|w_{l+1}\|_{l+1}\leq 1} \left\|P_{l+1}^* w_{l+1}\right\|_l \\ &= \sup_{\|w_{l+1}\|\leq \frac{1}{\sqrt{h_{l+1}}}} \sqrt{h_l}\left\|P_{l+1}^* w_{l+1}\right\| \\ &= \frac{\sqrt{h_l}}{\sqrt{h_{l+1}}}\left\|P_{l+1}^*\right\|.\end{aligned}$$

In some works, like [GK09; Kop20], there are three operators involved: $\bar{P}$ which is a projection operator, $\bar{R}$ which is a restriction operator and $\bar{I}$ which is the prolongation operator. Thus, in this case there are two operators that go from the upper level to the lower. The projection operator is used to restrict the iterates, i.e., give an initial point on the lower level, and the restriction operator is used to restrict the gradients. This is to provide a more general approach. Using $\bar{P} = R$, $\bar{R} = P^*$ and $\bar{I} = P$, this is coherent with our approach and notation.

3.2.2 Level-dependent objective functions

As we optimize, it is also essential to guarantee that the objective functions of our problems align in a certain manner. The idea is to use an additive term, a *gradient correction*, such that the objective functions at the different levels are first order coherent to the objective functions at the corresponding level above. This means that the gradients at the lower level at the starting point coincide with the restricted gradients from the upper level. The necessity for such a correction was already established in [Nas00]. We follow the derivation of the correction from [Kop20].

Derivation of the level dependent objective functions

Recall F_l is the objective function on level l for all $l = 0, \ldots, L$ and let $w_{l+1} \in V_{l+1}$ be a variable on the level $l+1$. Ideally, when starting from a point $\overline{w}_l := R_{l+1} w_{l+1}$ the step on the lower level l would correspond to the prolongated step on the higher level $l+1$ meaning $F_l(\overline{w}_l + s_l) = F_{l+1}(w_{l+1} + P_{l+1} s_l)$ for all $s_l \in V_l$. This can be achieved by using an additive correction term $v_l \colon V_l \longrightarrow \mathbb{R}$, such that

$$v_l(s_l) := F_{l+1}(w_{l+1} + P_{l+1} s_l) - F_l(\overline{w}_l + s_l)$$

and a new objective function

$$\begin{aligned}\overline{H}_l(\overline{w}_l + s_l) &\coloneqq F_l(\overline{w}_l + s_l) + v_l(s_l) \\ &= F_l(\overline{w}_l + s_l) + F_{l+1}(w_{l+1} + P_{l+1}s_l) - F_l(\overline{w}_l + s_l) \\ &= F_{l+1}(w_{l+1} + P_{l+1}s_l).\end{aligned}$$

However, this approach is not tractable as the computation of $v_l(s_l)$ involves evaluating the finer level objective function $F_{l+1}(w_{l+1} + P_{l+1}s_l)$ making it more expensive than just using the fine level. Hence, usually a first order Taylor approximation of v_l at the point $0 \in V_l$ is used. This means

$$\tilde{v}_l(s_l) = v_l(0) + \langle \nabla_l v_l(0), s_l \rangle_l,$$

is used instead. The advantage here is that v_l only has to be evaluated once for $0 \in V_l$. With

$$\nabla_l v_l(s_l) = P_{l+1}^* \nabla_{l+1} F_{l+1}(w_{l+1} + P_{l+1}s_l) - \nabla_l F_l(\overline{w} + s_l),$$

we get

$$\begin{aligned}\tilde{H}_l(\overline{w}_l + s_l) &\coloneqq F_l(\overline{w}_l + s_l) + \tilde{v}_l(s_l) \\ &= F_l(\overline{w}_l + s_l) + (F_{l+1}(w_{l+1}) - F_l(\overline{w}_l)) \\ &\quad + \langle P_{l+1}^* \nabla_{l+1} F_{l+1}(w_{l+1}) - \nabla_l F_l(\overline{w}_l), s_l \rangle_l.\end{aligned}$$

The second term $F_{l+1}(w_{l+1}) - F_l(\overline{w}_l)$ is a zeroth order coupling enforcing zeroth order coherence. However, as it is constant, it is usually neglected when optimizing. The last term containing the gradients enforces the first order coherence. This means that $\nabla_l \tilde{H}_l(\overline{w}_l)$ coincides with $P_{l+1}^* \nabla_{l+1} F_{l+1}(w_{l+1})$. This can be seen by

$$\nabla_l \tilde{H}_l(\overline{w}_l) = \nabla_l F_l(\overline{w}_l) + P_{l+1}^* \nabla_{l+1} F_{l+1}(w_{l+1}) - \nabla_l F_l(\overline{w}_l) = P_{l+1}^* \nabla_{l+1} F_{l+1}(w_{l+1}).$$

Definition of level dependent functions in our case

Following this derivation, we define *level dependent objective functions* H_l. On the finest level L there is no correction necessary. Therefore,

$$H_L(w_L) \coloneqq F_L(w_L).$$

On the coarser layers we define

$$H_l(w_l) \coloneqq F_l(w_l) + \langle P_{l+1}^* \nabla_{l+1} H_{l+1}(w_{l+1}) - \nabla F_l(R_{l+1} w_{l+1}), w_l \rangle_l.$$

Comparing H_l to the previous derivation of $\tilde{H}$, we have neglected the term $\langle P_{l+1}^* \nabla_{l+1} H_{l+1}(w_{l+1}) - \nabla_l F_l(R_{l+1} w_{l+1}), R_{l+1} w_{l+1} \rangle_l$ here, since it is constant and therefore does not affect the minimum. Note, that the gradient of H_l is

$$\nabla_l H_l(w_l) = \nabla_l F_l(w_l) + (P_{l+1}^* \nabla_{l+1} H_{l+1}(w_{l+1}) - \nabla_l F_l(R_{l+1} w_{l+1})).$$

The gradient correction is also often defined using R_l instead of P_{l+1}^*. To this end, one needs to assume that $P_{l+1}^* = cR_l$ with a constant scalar c. In doing so, $R_l \nabla_{l+1} H_{l+1}(w_{l+1})$ can be interpreted as the restricted gradients of the upper level. However, using P_{l+1}^* instead decouples the prolongation and restriction operators thus allowing for greater flexibility in their use. Therefore we do not assume $P_{l+1}^* = cR_l$ and define the gradient correction using $P_{l+1}^* \nabla_{l+1} H_{l+1}(w_{l+1})$ as above.

Note that if the points at which the gradient correction is computed are related by restriction, meaning $w_{k-1} = R_k \dots R_l w_l$ for levels $k < l$, this also means by recursion that the objective functions for all levels $k < l$ are first order coherent to the one at level l. This can be seen easily. To this end, assume that level $k < l$ is coherent to level l then we obtain inductively

$$\begin{aligned}
&\nabla_{k-1} H_{k-1}(R_k \dots R_l w_l) \\
&= \nabla_{k-1} F_{k-1}(R_k \dots R_l w_l) + (P_k^* \nabla_k H_k(w_k) - \nabla_{k-1} F_{k-1}(R_k w_k)) \\
&= \nabla_{k-1} F_{k-1}(R_k \dots R_l w_l) + (P_k^* \nabla_k H_k(w_k) - \nabla_{k-1} F_{k-1}(R_k \dots R_l w_l)) \\
&= P_k^* \nabla_k H_k(w_k) = P_k^* \nabla_k H_k(R_{k+1} \dots R_l w_l) \\
&= P_k^* \dots P_l^* \nabla_l H_l(w_l).
\end{aligned} \tag{3.1}$$

This shows that level $k-1$ is also coherent to level l. Since we already showed that level $l-1$ is first order coherent to level l by definition of the level dependent objective function, by induction all lower levels are first order coherent to level l. Therefore, it is also possible to write the objective function as

$$\begin{aligned}
H_l(w_l) &= F_l(w_l) + \langle P_{l+1}^* \nabla_{l+1} H_{l+1}(w_{l+1}) - \nabla_l F_l(R_{l+1} w_{l+1}), w_l \rangle_l \\
&= F_l(w_l) + \langle P_{l+1}^* \dots P_L^* \nabla_L F_L(w_L) - \nabla_l F_l(R_{l+1} \dots R_L w_L), w_l \rangle_l.
\end{aligned}$$

On the use of stochastic gradients instead of full gradients

When considering SGD, the question arises as to whether it is necessary to compute the full gradient of the higher level and use it for the gradient correction term. After all, the gradient computation is very expensive and it might be more efficient to use it to perform an update on the higher level if the coarse level step is not as helpful. Or perhaps it might be even more efficient to compute several smaller batch updates which would have the same computational effort as one full gradient computation.

Since using a stochastic direction or a stochastic gradient instead of the full gradient in the gradient correction lightens the computational burden, this may be a viable option. In Section 3.3.4 we examine the impact of this change on the convergence in greater detail.

3.2.3 Ensuring the quality of the low level step

To ensure the quality of the low level step, we take several measures. Firstly, we employ a step size regularization to guarantee the significance of the low-level step for the higher level. Secondly, an angle condition is checked to verify that the step is useful to the higher level. Finally, we exclude cases where the lower level would not provide meaningful directions before even descending to the lower level.

We examine these measures in detail and compare them to measures taken in multilevel trust region methods like [GST08; Gra+08].

Restricting the step size

By design, the gradient correction established in Section 3.2.2 only yields first order coherence to the higher level objective function in a sufficiently small neighborhood of the starting point. Thus, on each level we should not move too far from the given initial point in order for the lower level to yield descent directions to the upper level. To encourage this, a regularization term is added to the objective function such that

$$\begin{aligned} H_l(w_l) = F_l(w_l) + \langle P_{l+1}^* \nabla_{l+1} H_{l+1}(w_{l+1}) - \nabla_l F_l(R_{l+1} w_{l+1}), w_l \rangle_l \\ + \lambda \frac{1}{2} \|w_l - R_{l+1} w_{l+1}\|_l^2. \end{aligned} \tag{3.2}$$

At first glance this is a measure that the multilevel trust region methods in [GST08; Gra+08] do not have. However, similar to the lower level model, the quadratic model in trust region methods is only 'trustworthy' in a neighborhood of the initial point. For this reason, trust region methods use the trust region parameter δ to bound the step size. The parameter λ here can now be seen as an inverse to δ. Increasing it, reduces the step size and vice versa. Hence, it is not necessary for trust region methods to employ a step size regularization when the regularization comes already built in the trust region.

At least for the case of a deterministic algorithm the advantages of this regularization are easily understood. Following the argumentation in [Nas00], we repeat how using such a step size regularization enforces the prolongated lower level step to be a descent direction to the upper level.

To this end, suppose for this subsection that all F_l for $l = 0, \dots, L$ are twice differentiable and suppose we use the objective function (3.2). First of all we notice that when we solve the minimization problem

$$\min_{w_l \in \mathbb{R}^{n_l}} H_l(w_l)$$

exactly for an optimal w_l^*, then

$$0 = \nabla_l H_l(w_l^*) = \nabla_l F_l(w_l^*) + P_{l+1}^* \nabla_{l+1} H_{l+1}(w_{l+1}) - \nabla_l F_l(\overline{w}_l) + \lambda(w_l^* - \overline{w}_l),$$

where we have denoted $\overline{w}_l \coloneqq R_{l+1} w_{l+1}$ for readability. Hence,

$$P_{l+1}^* \nabla_{l+1} H_{l+1}(w_{l+1}) = \nabla_l F_l(\overline{w}_l) - \nabla_l F_l(w_l^*) - \lambda(w_l^* - \overline{w}_l)$$

holds. Now we check whether the prolongated lower level step $P_{l+1}(w_l^* - \overline{w}_l)$ is a descent direction for the higher level objective function H_{l+1}. Therefore, we consider

$$\begin{aligned}
&\langle \nabla_{l+1} H_{l+1}(w_{l+1}), P_{l+1}(w_l^* - \overline{w}_l) \rangle_{l+1} \\
&= \langle P_{l+1}^* \nabla_{l+1} H_{l+1}(w_{l+1}), w_l^* - \overline{w}_l \rangle_l \\
&= \langle \nabla_l F_l(\overline{w}_l) - \nabla_l F_l(w_l^*) - \lambda(w_l^* - \overline{w}_l), w_l^* - \overline{w}_l \rangle_l \\
&= \langle \nabla_l F_l(\overline{w}_l) - \nabla_l F_l(w_l^*), w_l^* - \overline{w}_l \rangle_l \\
&\quad - \langle w_l^* - \overline{w}_l, \lambda I(w_l^* - \overline{w}_l) \rangle_l \\
&= \frac{1}{h_l} \langle \nabla F_l(\overline{w}_l) - \nabla F_l(w_l^*), w_l^* - \overline{w}_l \rangle \\
&\quad - \frac{1}{h_l} \langle w_l^* - \overline{w}_l, \lambda I(w_l^* - \overline{w}_l) \rangle \\
&= \frac{1}{h_l} \langle \int_0^1 \nabla^2 F_l(t\overline{w}_l + (1-t)w_l^*)(\overline{w}_l - w_l^*) \mathrm{d}t, w_l^* - \overline{w}_l \rangle \\
&\quad - \frac{1}{h_l} (w_l^* - \overline{w}_l)^\top \lambda I (w_l^* - \overline{w}_l) \\
&= -\frac{1}{h_l} (w_l^* - \overline{w}_l)^\top \left(\int_0^1 \nabla^2 F_l(t\overline{w}_l + (1-t)w_l^*) \mathrm{d}t + \lambda I \right)^\top (w_l^* - \overline{w}_l).
\end{aligned}$$

Now we see that if the mean of the Hessian of F_l is not positive definite, we can use λ to counteract non positive eigenvalues of the Hessian. By this, we mean that by taking for example $\lambda > |\min_{t \in [0,1]} \lambda_{\min}(\nabla^2 F_l(t\overline{w}_l + (1-t)w_l^*))|$ the matrix $\nabla^2 F_l(t\overline{w}_l + (1-t)w_l^*) + \lambda I$ is positive definite for $t \in [0,1]$. In this case, $P_{l+1}(w_l^* - \overline{w}_l)$ is a descent direction.

Acceptance of the step

In trust region methods verifying the step before accepting is essential. This is to ensure that the step leads to a sufficient decrease in the objective function. Continuing the analogy to trust region methods, in addition to restricting the size of the step taken, we further ensure that the step taken is a 'good' direction in our sense. As previously noted, the lower level model is only a reliable approximation in the neighborhood of the initial point. We denote the iterates now with $w_{l,k,t}$, where $l = 0, \dots, L$ denotes the level of the iterate, k denotes the current multilevel cycle and t denotes the iteration on that level in that cycle. Further, the points at which the gradient correction is computed are related by restriction meaning $w_{l,k,0} = R_{l+1} \dots R_L w_{L,k,0}$. This results in a V-cycle structure for the multilevel algorithm. Note that in this case, because of the first order coherence over all levels discussed in Section 3.2.2, the gradient correction can also be written as

$$\begin{aligned}&\langle P_{l+1}^* \nabla_{l+1} H_{l+1}(w_{l+1,k,0}) - \nabla_l F_l(w_{l,k,0}), w_l \rangle_l \\ &= \langle P_{l+1}^* \dots P_L^* \nabla_L F_L(w_{L,k,0}) - \nabla_l F_l(w_{l,k,0}), w_l \rangle_l.\end{aligned}$$

To shorten the notation, we now let $w_{l,k,t}^L$ be the iterate after prolongation of the step from level l, iteration k and step t to level L. This means

$$\begin{aligned}w_{l,k,t}^L &:= w_{L,k,0} + P_L \dots P_{l+1}(s_{l,k,t}) \\ &= w_{L,k,0} + P_L \dots P_{l+1}(w_{l,k,t} - w_{l,k,0}) \\ &= w_{L,k,0} + P_L \dots P_{l+1}(w_{l,k,t} - R_{l+1} \dots R_L w_{L,k,0}).\end{aligned}$$

Note that in this setting it is also possible for l to be L, in which case a prolongation is not necessary, which can be expressed by using the identity as prolongation.

Hence, to ensure that the step is a sufficient descent direction, we check for level l and cycle k whether the proposed $t+1$-th iterate $\tilde{w}_{l,k,t+1}$ fulfills

$$-\langle \nabla F_L(w_{L,k,0}), \tilde{w}_{l,k,t+1}^L - w_{0,k,0}^L \rangle \geq \mu \left\| \nabla F_L(w_{L,k,0}) \right\| \left\| \tilde{w}_{l,k,t+1}^L - w_{0,k,0}^L \right\|. \tag{3.3}$$

If (3.3) is not fulfilled, the latest step on the lower level is rejected and the step from before is prolongated to the higher level.

Eq. (3.3) can be seen as an angle condition because

$$\cos(\theta_{l,k}) = \frac{-\langle \nabla F_L(w_{L,k,0}), \tilde{w}_{l,k,t+1}^L - w_{0,k,0}^L \rangle}{\left\| \nabla F_L(w_{L,k,0}) \right\| \left\| \tilde{w}_{l,k,t+1}^L - w_{0,k,0}^L \right\|},$$

where $\theta_{l,k}$ is the angle between $-\nabla F_L(w_{L,k,0})$ and $\tilde{w}_{l,k,t+1} - w^L_{0,k,0}$. Now if (3.3) holds, then

$$\cos(\theta_{l,k}) \geq \mu.$$

As $\cos(\theta_{l,k})$ increases, the angle between the prolongated step and the negative gradient of the objective function, i.e., $\theta_{l,k}$, decreases. Hence, the update from level l is closer to the steepest descent. Therefore, checking the angle condition (3.3) ensures the quality of the step.

Early termination of the recursion

As already established in [GST08; Gra+08], in some cases the low level cannot give helpful steps. To reduce computational effort by preventing unnecessary steps, we exclude these cases before proceeding to the lower level.

First, it is possible for the current gradient to lie in or close to the kernel of P^*. In this case, the norm of $P_l^* \nabla_l H_l(w_{l,k,t})$ is significantly smaller than the norm of $\nabla_l H_l(w_{l,k,t})$. Therefore, iterates on the low level are first order critical for the lower level model but the resulting iterates on the higher level are are not first order critical. Hence, we only descend onto the lower level if

$$\|P_l^* \nabla_l H_l(w_{l,k,t})\|_{l-1} \geq \eta \|\nabla_l H_l(w_{l,k,t})\|_l \tag{3.4}$$

for a constant $\eta > 0$.

Second, if the level has already reached the desired accuracy, it would also be pointless to descend onto the lower level. As the gradient in the initial point of level $l-1$ coincides with $P_l^* \nabla_l H_l(w_{l,k,t})$, the low level appears to be close to a stationary point when $\|P_l^* \nabla_l H_l(w_{l,k,t})\|_{l-1}$ is small. Therefore, we can exclude this even before calling the low level by checking

$$\|P_l^* \nabla_l H_l(w_{l,k,t})\|_{l-1} \geq \epsilon, \tag{3.5}$$

where ϵ is the tolerance.

3.2.4 The MLSGD algorithm

Our considerations so far result in Algorithm 4. We call the algorithm multilevel stochastic gradient descent and shorten it with MLSGD.

The inputs needed for the algorithm are a maximum number of iterations of MLSGD K_l and an initial point $w_{l,k,0}$. The number of epochs for training T_l which is the number of iterations the solver on a given level should perform and the tolerance $\epsilon > 0$ are parameters of the method. In each iteration it needs to be decided whether the algorithm should be called recursively or whether the recursion is stopped. If the lowest level is reached, or if (3.4) and (3.5) are not fulfilled, then the recursion is stopped and instead a standard step is performed. Otherwise, the algorithm is called recursively. This then gives a solution for the lower level. Using this solution, the step taken on the lower level is reconstructed and then prolongated and used as an update on the higher level. Afterwards, a few steps of the solver are performed on the current level. After the maximum number of iterations is reached, the current iterate is returned.

Algorithm 4: Multilevel stochastic gradient descent MLSGD($w_{l,k,0}, K_l$)

Parameter : Number of epochs for training T_l, tolerance ϵ, current level l
Output: Iterate $w_{l,K_l,T_l}, \{T_{i,k}\}_{i \leq k}$
for $k = 0, 1, 2, \ldots, K_l - 1$ **do**
  if $\|P_l^* \nabla H_l(w_{l,k,0})\|_{l-1} < \eta \|\nabla H_l(w_{l,k,0})\|_l$
    or
    $\|P_l^* \nabla H_l(w_{l,k,0})\|_{l-1} < \epsilon$
    or
    $l = 0$
  then
    $w_{l,k,1} = w_{l,k,0}$
    $T_{i,k} = 0$ for all $i < l$
    $(w_{l,k,T_{l,k}}, T_{l,k}) = \text{aSGD}(w_{l,k,1}, T_l)$
  else
    $(w_{l-1,k,T_{l-1,k}}, \{T_{i,k}\}_{i<l}) = \text{MLSGD}(R_l w_{l,k,0}, 1)$
    $s_{l-1,k,T_{l-1}} = w_{l-1,k,T_{l-1}} - R_l w_{l,k,0}$
    $w_{l,k,1} = w_{l,k,0} + P_l(s_{l-1,k,T_{l-1}})$
    $(w_{l,k,T_l+1}, T_{l,k}) = \text{aSGD}(w_{l,k,1}, T_l)$
  end
  $w_{l,k+1,0} = w_{l,k,T_l+1}$
end
return $w_{l,K_l,T_l+1}, \{T_{i,k}\}_{i \leq k}$

Instead of the stochastic gradient descent introduced in Algorithm 1, the MLSGD shown in Algorithm 4 uses Algorithm 5 which is an adapted version of the SGD Algorithm 1 that checks the angle condition. Moreover, note that Algorithm 4 enforces the use of V-cycles rather than W-cycles by setting $K_l = 1$ in recursive calls of itself. The schematic overview of the V-cycle structure in the case of MLSGD is depicted in Figure 3.2. This means in our case, that in each cycle the point $w_{L,k,0}$ is restricted all the way down before any updates occur.

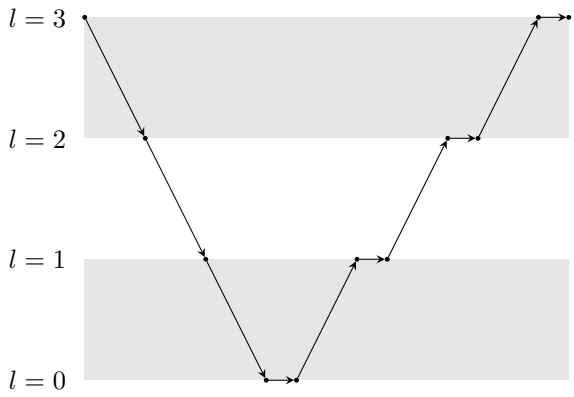


Figure 3.2 The resulting V-cycle in a three level scenario for MLSGD.

Algorithm 5: Adapted stochastic gradient descent for MLSGD aSGD($w_{l,k,1}, T_l$)

Parameter : Learning rates $\alpha_{l,k,t}$, current level l, full gradient $\nabla F(w_{L,k,0})$, the restricted point $w_{l,k,0}$, prolongation operators $P_L, \ldots, P_{l+1}$

Output: Iterate $w_{l,k,T_{l,k}+1}, T_{l,k}$

for $t = 1, 2, \ldots, T_l$ **do**

  Draw independently a sample $\xi_{l,k,t}$

  $d_{l,k,t} = g_l^H(w_{l,k,t}, \xi_{l,k,t})$

  $\tilde{w}_{l,k,t+1} = w_{l,k,t} - \alpha_{l,k,t} d_{l,k,t}$

  if $-\langle \nabla F_L(w_{L,k,0}), \tilde{w}^L_{l,k,t+1} - w^L_{0,k,0} \rangle \geq \mu \left\| \nabla F_L(w_{L,k,0}) \right\| \left\| \tilde{w}^L_{l,k,t+1} - w^L_{0,k,0} \right\|$ **then**

    $w_{l,k,t+1} = \tilde{w}_{l,k,t+1}$

  else

    $T_{l,k} = t - 1$

    return $w_{l,k,t}, T_{l,k}$

  end

end

$T_{l,k} = T_l$

return $w_{l,k,T_{l,k}+1}, T_{l,k}$

3.3 Convergence analysis

This section discusses the convergence of the algorithm. The proof idea is similar to that of the convergence of the stochastic gradient method in Theorem 2.12. Therefore, the required assumptions that are discussed in Section 3.3.1 share some similarity. In addition to similar assumptions, the proof is also based on an angle condition on the multilevel steps and a bound on the variance of the multilevel steps in relation to the norm of the gradient of the highest level objective function. We consider fixed step sizes.

We examine four different cases when discussing convergence. In Section 3.3.2 we use the full gradient in the gradient correction. This makes computing it very expensive but gives an exact correction for the lower level. Thus, to fully utilize this computational effort, one should perform a sufficient number of steps at the lower level. In Section 3.3.3 we introduce variance reduction and study its effect on convergence. In Section 3.3.4 we reduce computational cost by using a stochastic direction in the gradient correction instead of the full gradient. Finally, in Section 3.3.5 we analyze the impact of variance reduction in combination with a stochastic gradient correction on the convergence.

To simplify the notation, we only specify the random part ξ when it is necessary. In the other cases it is omitted. With $[\xi]^{l,k,t} = \{\xi_{i,j,s}\}_{i\leq l, j\leq k, s\leq t}$ we denote the set of all $\xi_{i,j,s}$ up to iteration l,k,t and with $[\xi]^{l,k,t}_{i,j,s}$ the ones that are drawn after iteration (i,j,s) up to iteration (l,k,t). With $\xi = \{\xi_{i,j,s}\}_{i,j,s}$ we denote from now on the set of all $\xi_{i,j,s}$.

The expected value with respect to a set $[\xi]^{l,k,t}_{i,j,s}$ is denoted by $\mathbb{E}_{\xi^{l,k,t}_{i,j,s}}$.

Note that one possibility to compute a stochastic direction $g_l^H(w_l,\xi_l)$ for the objective function H_l is to use a stochastic direction $g_l^F(w_l,\xi_l)$ for F_l. This can be

$$g_l^H(w_l,\xi_l) = g_l^F(w_l,\xi_l) + (P_{l+1}^*\nabla_{l+1}H_{l+1}(w_{l+1}) - \nabla_l F_l(R_{l+1}w_{l+1})).$$

3.3.1 Necessary assumptions

We make similar assumptions as for the stochastic gradient descent method. For SGD, Assumption 2.9 assumed Lipschitz continuity of the gradients of the objective function. Additionally, we now also need an assumption on the smoothness of the objective functions on the different levels. Therefore, we assume Lipschitz continuity of their gradients on all levels.

Assumption 3.3. *The objective functions $F_l \colon V_l \longrightarrow \mathbb{R}$ for $l = 0, 1, \dots L$ are continuously differentiable and their gradients $\nabla_l F_l \colon V_l \longrightarrow V_l$ are Lipschitz continuous with Lipschitz constants $L_l > 0$. This means*

$$\|\nabla_l F_l(v) - \nabla_l F_l(w)\|_l \leq L_l \|v - w\|_l \text{ for all } v, w \in V_l, l = 0, 1, \dots L.$$

It follows that

$$F_l(v) \leq F_l(w) + \langle \nabla_l F_l(w), v - w \rangle + \frac{1}{2} L_l \|v - w\|_l^2 . \tag{3.6}$$

Under Assumption 3.3 we have for each level l a Lipschitz constant L_l. As there is a finite number of levels we denote the maximum Lipschitz constant with $\mathcal{L} \coloneqq \max_{0 \leq l \leq L} L_l$.

In some cases, we also need the assumption of the Lipschitz continuity for the stochastic directions.

Assumption 3.4. *Let $l \in \{0, \dots, L\}$. The stochastic directions chosen in Algorithm 4 are Lipschitz continuous with constant $\tilde{L}_l$ in the sense that for all $\xi \in \Omega, v, w \in \mathbb{R}^{n_l}$ it holds*

$$\left\| g_l^F(v, \xi) - g_l^F(w, \xi) \right\|_l \leq \tilde{L}_l \|v - w\|_l .$$

For notational ease, we further assume $L_l = \max\{L_l, \tilde{L}_l\}$ where L_l is the Lipschitz constant of $\nabla_l F_l$.

Additionally, we assume that all stochastic directions $g_l^H(w_{l,k,t}, \xi_{l,k,t})$ and $g_l^F(w_{l,k,t}, \xi_{l,k,t})$ fulfill Assumption 2.11. We state them again to introduce the notation.

Assumption 3.5. *The stochastic directions $g_l^H(w_{l,k,t}, \xi_{l,k,t})$ and $g_l^F(w_{l,k,t}, \xi_{l,k,t})$ satisfy for all $w_{l,k,t}$ the following:*

a) There exist scalars $\overline{\mu}_l^F \geq \mu_l^F > 0, \overline{\mu}_l^H \geq \mu_l^H > 0$ such that for all $l = 0, \dots, L$ and $k, t \in \mathbb{N}$,

$$\begin{aligned} \langle \nabla_l F_l(w_{l,k,t}), \mathbb{E}_{\xi_{l,k,t}}[g_l^F(w_{l,k,t}, \xi_{l,k,t})] \rangle_l &\geq \mu_l^F \|\nabla_l F_l(w_{l,k,t})\|_l^2 \quad \text{and} \\ \|\mathbb{E}_{\xi_{l,k,t}}[g_l^F(w_{l,k,t}, \xi_{l,k,t})]\|_l &\leq \overline{\mu}_l^F \|\nabla_l F_l(w_{l,k,t})\|_l \end{aligned}$$

as well as

$$\begin{aligned} \langle \nabla_l H_l(w_{l,k,t}), \mathbb{E}_{\xi_{l,k,t}}[g_l^H(w_{l,k,t}, \xi_{l,k,t})] \rangle_l &\geq \mu_l^H \|\nabla_l H_l(w_{l,k,t})\|_l^2 \quad \text{and} \\ \|\mathbb{E}_{\xi_{l,k,t}}[g_l^H(w_{l,k,t}, \xi_{l,k,t})]\|_l &\leq \overline{\mu}_l^H \|\nabla_l H_l(w_{l,k,t})\|_l. \end{aligned}$$

b) There exist scalars $M_l^F, M_l^H \geq 0$ *and* $\tilde{M}_l^F, \tilde{M}_l^H \geq 0$ *such that, for all* $l = 0, \ldots, L$ *and* $k, t \in \mathbb{N}$,

$$\mathbb{V}_{\xi_{l,k,t}}[g_l^F(w_{l,k,t}, \xi_{l,k,t})] \leq M_l^F + \tilde{M}_l^F \|\nabla_l F_l(w_{l,k,t})\|_l^2$$

and

$$\mathbb{V}_{\xi_{l,k,t}}[g_l^H(w_{l,k,t}, \xi_{l,k,t})] \leq M_l^H + \tilde{M}_l^H \|\nabla_l H_l(w_{l,k,t})\|_l^2.$$

Again, from Assumption 3.5 we can derive:

$$\mathbb{E}_{\xi_{l,k,t}}\left[\left\|g_l^F(w_{l,k,t}, \xi_{l,k,t})\right\|_l^2\right] \leq M_l^F + \overline{M}_l^F \|\nabla_l F_l(w_{l,k,t})\|_l^2$$

and

$$\mathbb{E}_{\xi_{l,k,t}}\left[\left\|g_l^H(w_{l,k,t}, \xi_{l,k,t})\right\|_l^2\right] \leq M_l^H + \overline{M}_l^H \|\nabla_l H_l(w_{l,k,t})\|_l^2.$$

We define $\overline{M}_l^F := \tilde{M}_l^F + (\overline{\mu}_l^F)^2 > 0$ and $\overline{M}_l^H := \tilde{M}_l^H + (\overline{\mu}_l^H)^2 > 0$. Furthermore, the maximum values of M_l^H and $\overline{M}_l^H$ over all levels are denoted by $\mathcal{M} := \max_{0 \leq l \leq L} M_l^H$ and $\overline{\mathcal{M}} := \max_{0 \leq l \leq L} \overline{M}_l^H$.

Moreover, we need assumptions on the prolongation operators and the behavior of the multilevel steps in Algorithm 4.

Assumption 3.6. *The objective function and the iterates of the MLSGD Algorithm 4 satisfy the following:*

a) There is $\rho \geq 0$ *such that for all* $l = 0, 1, \ldots, L$ *we have*

$$\|P_l\|_{l-1,l}, \|P_l^*\|_{l,l-1} \leq \rho.$$

Further, we denote $\mathcal{R}_{l,L} = \|R_l\|_{l,l-1} \cdots \|R_L\|_{L,L-1}$, $\overline{\mathcal{R}} = \max_{0 \leq i < j \leq L} \mathcal{R}_{i,j}$ *as well as* $\overline{\rho} = \max\{1, \rho^{2L}\}$ *for notational ease.*

b) There is $\mu > 0$ *such that for all* $l = 0, \ldots, L$ *and* $k, t \in \mathbb{N}$ *with* $t \leq T_{l,k} + 1$ *the iterates fulfill the angle condition*

$$-\langle \nabla F_L(w_{L,k,0}), w_{l,k,t}^L - w_{L,k,0} \rangle \geq \mu \|\nabla F_L(w_{L,k,0})\| \left\|w_{l,k,t}^L - w_{L,k,0}\right\|.$$

c) There is $\nu > 0$ such that for all $i, l = 0, \dots, L$ with $i \leq l$ and $k, t \in \mathbb{N}$ with $t \leq T_{i,k} + 1$ we have

$$\left\| w^L_{l,k,T_{l,k}+1} - w_{L,k,0} \right\| \geq \nu \left\| w_{i,k,t} - w_{i,k,0} \right\|_i .$$

d) There is $\Theta > 0$ such that for all $l = 0, \dots, L$ and $k \in \mathbb{N}$ we have

$$\mathbb{E}_{[\xi]^{l,k,T_{l,k}+1}_{l,k,0}} \left[\left\| w^L_{l,k,T_{l,k}+1} - w_{L,k,0} \right\| \right] \geq \frac{1}{\Theta} \sum_{i=0}^{l} \sum_{t=1}^{T_{i,k}} \alpha_{i,k,t} \left\| \nabla F_L(w_{L,k,0}) \right\| .$$

Let $\mathcal{T}_{l,k} \coloneqq \sum_{i=0}^{l} T_{i,k}$ be the number of steps up until level l for $l = 0, \dots, L$ in a cycle $k \in \mathbb{N}$. When using a fixed step size $\alpha_{l,k,t} = \overline{\alpha}$ for all l, k, t, Assumption 3.6d) yields

$$- \left\| \nabla F_L(w_{L,k,0}) \right\| \mathbb{E}_{[\xi]^{l,k+1,0}_{l,k,0}} \left[\left\| w^L_{l,k,T_{l,k}+1} - w_{L,k,0} \right\| \right] \leq -\frac{1}{\Theta} \overline{\alpha} \mathcal{T}_{l,k} \left\| \nabla F_L(w_{L,k,0}) \right\|^2 . \quad (3.7)$$

As we are considering the operator norm of linear operators in a finite dimensional space, they are alway bounded and we choose $\rho \coloneqq \max_{l=0,\dots,L} \{ \|P_l\|_{l-1,l} , \|P_l^*\|_{l,l-1} \}$. With this choice, Assumption 3.6a) is fulfilled. Additionally, in most applications the prolongation operators have a norm of 1, as is also the case in our application.

Assumption 3.6b) enforces the angle condition (3.3) and is fulfilled when the angle condition is checked before accepting steps on the lower levels. Of course, this is only practically feasible when a deterministic gradient correction is used. Assumption 3.6d) relates the expected value of the norm of the steps taken across multiple levels to the magnitude of the gradient and ensures that the expected progress is proportional to the steepness of the highest level objective function.

Assumptions for the first step on the lowest level

We now discuss the assumptions for the first step on the lowest level l_0. As already mentioned, Assumption 3.6a) is always fulfilled. Assumption 3.6c) holds trivially for the first lowest level step, when $T_{l,k} = 1$. We now reason why the angle condition holds in expectation for the first step, even without checking it in the algorithm. We denote the lowest level with l_0. Since we only descend onto the lower levels if (3.4) is fulfilled, we have that

$$\begin{aligned} \left\| P^*_{l_0+1} \nabla_{l_0+1} H_{l_0+1}(w_{l_0+1,k,0}) \right\|_{l_0} &\geq \eta \left\| \nabla_{l_0+1} H_{l_0+1}(w_{l_0+1,k,0}) \right\|_{l_0+1} \\ &\geq \eta \left\| P^*_{l_0+2} \nabla_{l_0+2} H_{l_0+2}(w_{l_0+2,k,0}) \right\|_{l_0+1} . \end{aligned} \quad (3.8)$$

Repeating this until reaching the highest level yields

$$\begin{aligned}\left\|P_{l_0+1}^* \dots P_L^* \nabla F(w_{L,k,0})\right\|_{l_0} &= \left\|P_{l_0+1}^* \nabla_{l_0+1} H_{l_0+1}(w_{l_0+1,k,0})\right\|_{l_0} \\ &\geq \eta^{L-l_0} \left\|\nabla F(w_{L,k,0})\right\|.\end{aligned}$$

For the starting point $w_{l_0,k,1} = w_{l_0,k,0}$ holds and thus the gradient of the objective function is

$$\begin{aligned}\nabla_{l_0} H_{l_0}(w_{l_0,k,1}) &= \nabla_{l_0} H_{l_0}(w_{l_0,k,0}) \\ &= \nabla F_{l_0}(w_{l_0,k,0}) + (P_{l_0+1}^* \dots P_L^* \nabla F(w_{L,k,0}) - \nabla_{l_0} F_{l_0}(w_{l_0,k,0})) \\ &= P_{l_0+1}^* \dots P_L^* \nabla F(w_{L,k,0}).\end{aligned}$$

By Assumption 3.5, a stochastic direction $g_{l_0}^H(w_{l_0,k,1}, \xi_{l_0,k,1})$ for $H_{l_0}(w_{l_0,k,1})$ fulfills

$$\begin{aligned}&\langle \nabla F(w_{L,k,0}), P_L \dots P_{l_0+1} \mathbb{E}_{\xi_{l_0,k,1}} \left[g_{l_0}^H(w_{l_0,k,1}, \xi_{l_0,k,1})\right]\rangle \\ &= \langle P_{l_0+1}^* \dots P_L^* \nabla F(w_{L,k,0}), \mathbb{E}_{\xi_{l_0,k,1}} \left[g_{l_0}^H(w_{l_0,k,1}, \xi_{l_0,k,1})\right]\rangle_{l_0} \\ &= \langle \nabla_{l_0} H_{l_0}(w_{l_0,k,1}), \mathbb{E}_{\xi_{l_0,k,1}} \left[g_{l_0}^H(w_{l_0,k,1}, \xi_{l_0,k,1})\right]\rangle_{l_0} \\ &\geq \mu_{l_0}^H \left\|\nabla_{l_0} H_{l_0}(w_{l_0,k,1})\right\|_{l_0}^2 \\ &= \mu_{l_0}^H \left\|P_{l_0+1}^* \dots P_L^* \nabla F(w_{L,k,0})\right\|_{l_0}^2 \\ &\geq \mu_{l_0}^H \eta^{2(L-l_0)} \left\|\nabla F(w_{L,k,0})\right\|^2.\end{aligned}$$

Therefore, there is $\mu = \mu_{l_0}^H \eta^{2(L-l_0)}$ such that Assumption 3.6b) is fulfilled.

We now show that Assumption 3.6d) holds as well. To this end, we consider the product of all prolongation operators from level l_0 to L, i.e., $P_{l_0,L} \coloneqq P_L \dots P_{l_0+1}$. Then $P_{l_0,L}$ is a linear operator from level l_0 to level L for which we have

$$\begin{aligned}\left\|P_{l_0,L} P_{l_0,L}^* \nabla F(w_{L,k,0})\right\| &= \sup_{\|x\|=1} \langle x, P_{l_0,L} P_{l_0,L}^* \nabla F(w_{L,k,0})\rangle \\ &\geq \langle \frac{\nabla F(w_{L,k,0})}{\|\nabla F(w_{L,k,0})\|}, P_{l_0,L} P_{l_0,L}^* \nabla F(w_{L,k,0})\rangle \\ &= \langle \frac{P_{l_0,L}^* \nabla F(w_{L,k,0})}{\|\nabla F(w_{L,k,0})\|}, P_{l_0,L}^* \nabla F(w_{L,k,0})\rangle_{l_0} \\ &= \frac{\left\|P_{l_0,L}^* \nabla F(w_{L,k,0})\right\|_{l_0}^2}{\|\nabla F(w_{L,k,0})\|}.\end{aligned}$$

As reasoned above in (3.8), it holds

$$\left\|P_{l_0,L}^* \nabla F(w_{L,k,0})\right\|_{l_0} = \left\|P_{l_0+1}^* \dots P_L^* \nabla F(w_{L,k,0})\right\|_{l_0} \geq \eta^{L-l_0} \left\|\nabla F(w_{L,k,0})\right\|$$

and thus

$$\begin{aligned}\left\|P_{l_0,L}P_{l_0,L}^*\nabla F(w_{L,k,0})\right\| &\geq \frac{\eta^{2(L-l_0)}\left\|\nabla F(w_{L,k,0})\right\|^2}{\left\|\nabla F(w_{L,k,0})\right\|}\\ &= \eta^{2(L-l_0)}\left\|\nabla F(w_{L,k,0})\right\|.\end{aligned}$$

Now, we use this estimate when considering

$$\begin{aligned}\mathbb{E}_{[\xi]_{l_0,k,0}^{l_0,k,1}}\left[\left\|w_{l_0,k,2}^L - w_{0,k,0}^L\right\|\right] &\geq \left\|\mathbb{E}_{[\xi]_{l_0,k,0}^{l_0,k,1}}\left[w_{l_0,k,2}^L - w_{0,k,0}^L\right]\right\|\\ &= \left\|\mathbb{E}_{[\xi]_{l_0,k,0}^{l_0,k,1}}\left[-\alpha_{l_0,k,0}P_L\dots P_{l_0+1}g_{l_0}^H(w_{l_0,k,0},\xi_{l_0,k,0})\right]\right\|\\ &\geq \left\|-\alpha_{l_0,k,0}P_L\dots P_{l_0+1}\mathbb{E}_{[\xi]_{l_0,k,0}^{l_0,k,1}}\left[g_{l_0}^H(w_{l_0,k,0},\xi_{l_0,k,0})\right]\right\|\end{aligned}$$

and since $\nabla_{l_0}H_{l_0}(w_{l_0,k,1}) = P_{l_0+1}^*\dots P_L^*\nabla F(w_{L,k,0})$ as seen in (3.1), this gives for an unbiased estimator $g_{l_0}^H(w_{l_0,k,0},\xi_{l_0,k,0})$ that

$$\begin{aligned}\mathbb{E}_{[\xi]_{l_0,k,0}^{l_0,k,1}}\left[\left\|w_{l_0,k,2}^L - w_{0,k,0}^L\right\|\right] &\geq \alpha_{l_0,k,0}\left\|P_L\dots P_{l_0+1}P_{l_0+1}^*\dots P_L^*\nabla F(w_{L,k,0})\right\|\\ &\geq \alpha_{l_0,k,0}\left\|P_{l_0,L}P_{l_0,L}^*\nabla F(w_{L,k,0})\right\|.\end{aligned}$$

Similar to (3.8), this is

$$\mathbb{E}_{[\xi]_{l_0,k,0}^{l_0,k,1}}\left[\left\|w_{l_0,k,2}^L - w_{0,k,0}^L\right\|\right] \geq \eta^{2(L-l_0)}\alpha_{l_0,k,0}\left\|\nabla F(w_{L,k,0})\right\|.$$

Thus, Assumption 3.6d) is fulfilled with $\Theta \coloneqq \eta^{2(L-l_0)}$.

3.3.2 Convergence with deterministic gradient correction

We start with a lemma where we examine the descent on the highest level after several steps on the lower levels.

Lemma 3.7. *Let Assumptions 3.3, 3.5 and 3.6 hold. Let the iterates $\{w_{l,k,t}\}_{l,k,t}$ with $l = 0,1,\dots,L$, $k = 0,\dots K_l - 1$ and $t = 0,\dots,T_{l,k}+1$ be generated by Algorithm 4 after a recursive call at $w_{L,k,0}$. Then*

$$\begin{aligned}&F_L(w_{l,k,T_{l,k}+1}^L) - F_L(w_{L,k,0})\\ &\leq -\mu\left\|\nabla F_L(w_{L,k,0})\right\|\left\|w_{l,k,T_{l,k}+1}^L - w_{L,k,0}\right\| + \frac{1}{2}L_L\left\|w_{l,k,T_{l,k}+1}^L - w_{L,k,0}\right\|^2.\end{aligned}$$

Proof. From Assumption 3.3 and (3.6) we know

$$\begin{aligned} &F_L(w^L_{l,k,T_{l,k}+1}) - F_L(w_{L,k,0}) \\ &\leq \langle \nabla F_L(w_{L,k,0}), w^L_{l,k,T_{l,k}+1} - w_{L,k,0} \rangle + \frac{1}{2} L_L \left\| w^L_{l,k,T_{l,k}+1} - w_{L,k,0} \right\|^2. \end{aligned}$$

By Assumption 3.6b), this is

$$\begin{aligned} &F_L(w^L_{l,k,T_{l,k}+1}) - F_L(w_{L,k,0}) \\ &\leq -\mu \left\| \nabla F_L(w_{L,k,0}) \right\| \left\| w^L_{l,k,T_{l,k}+1} - w_{L,k,0} \right\| + \frac{1}{2} L_L \left\| w^L_{l,k,T_{l,k}+1} - w_{L,k,0} \right\|^2. \end{aligned}$$ □

In the following lemma, we provide an estimate for the expected value of the last term on the right-hand side.

Lemma 3.8. *Let Assumptions 3.3, 3.5 and 3.6 hold. Let the iterates $\{w_{l,k,t}\}_{l,k,t}$ with $l = 0, 1, \ldots, L$, $k = 0, \ldots K_l - 1$ and $t = 0, \ldots, T_{l,k} + 1$ be generated by Algorithm 4 after a recursive call at $w_{L,k,0}$. Assume that the step size is sufficiently small, i.e., fulfills*

$$\left(\sum_{i=0}^{l} \sum_{t=1}^{T_{i,k}+1} \alpha_{i,k,t} \right)^2 \leq \frac{\nu^2}{4\overline{\rho}\overline{\mathcal{M}}\mathcal{L}}.$$

Then for all $l = 0, \ldots, L$ and $k \in \mathbb{N}$ we have

$$\begin{aligned} &\mathbb{E}_{[\xi]^{l,k,T_{l,k}+1}_{l,k,0}} \left[\left\| w^L_{l,k,T_{l,k}+1} - w_{L,k,0} \right\|^2 \right] \\ &\leq \left(\sum_{i=0}^{l} \sum_{t=1}^{T_{i,k}+1} \alpha_{i,k,t} \right)^2 \left(a + A \left\| \nabla F(w_{L,k,0}) \right\|^2 \right) \end{aligned}$$

with $a \coloneqq 2\overline{\rho}\mathcal{M}$ and $A \coloneqq 4\overline{\rho}^2\overline{\mathcal{M}}$. If the step size is constant $\overline{\alpha}$, then

$$\mathbb{E}_{[\xi]^{l,k+1,0}_{l,k,0}} \left[\left\| w^L_{l,k,T_{l,k}+1} - w_{L,k,0} \right\|^2 \right] \leq a\overline{\alpha}^2 \mathcal{T}^2_{l,k} + A\overline{\alpha}^2 \mathcal{T}^2_{l,k} \left\| \nabla F_L(w_{L,k,0}) \right\|^2. \tag{3.9}$$

Proof. We define $\mathcal{A}_k \coloneqq \sum_{i=0}^{l} \sum_{t=1}^{T_{i,k}+1} \alpha_{i,k,t}$. Per definition we have

$$w^L_{l,k,T_{l,k}+1} - w_{L,k,0} = \sum_{i=0}^{l} \sum_{t=1}^{T_{i,k}+1} \alpha_{i,k,t} P_L \ldots P_{i+1} g^H_i(w_{i,k,t}, \xi_{i,k,t}).$$

Then with Jensen's inequality, we get

$$
\begin{aligned}
&\left\| w^L_{l,k,T_{l,k}+1} - w_{L,k,0} \right\|^2 \\
&= \left\| \sum_{i=0}^{l} \sum_{t=1}^{T_{i,k}+1} \alpha_{i,k,t} P_L \dots P_{i+1} g_i^H(w_{i,k,t}, \xi_{i,k,t}) \right\|^2 \\
&= \mathcal{A}_k^2 \left\| \sum_{i=0}^{l} \sum_{t=1}^{T_{i,k}+1} \frac{\alpha_{i,k,t}}{\mathcal{A}_k} P_L \dots P_{i+1} g_i^H(w_{i,k,t}, \xi_{i,k,t}) \right\|^2 \\
&\leq \mathcal{A}_k \sum_{i=0}^{l} \sum_{t=1}^{T_{i,k}+1} \alpha_{i,k,t} \left\| P_L \dots P_{i+1} g_i^H(w_{i,k,t}, \xi_{i,k,t}) \right\|^2 \\
&\leq \mathcal{A}_k \sum_{i=0}^{l} \sum_{t=1}^{T_{i,k}+1} \alpha_{i,k,t} \left\| P_L \right\|^2_{L-1,L} \dots \left\| P_{i+1} \right\|^2_{i,i+1} \left\| g_i^H(w_{i,k,t}, \xi_{i,k,t}) \right\|_i^2 \\
&\leq \mathcal{A}_k \sum_{i=0}^{l} \sum_{t=1}^{T_{i,k}+1} \alpha_{i,k,t} \rho^{2(L-i)} \left\| g_i^H(w_{i,k,t}, \xi_{i,k,t}) \right\|_i^2 .
\end{aligned}
$$

Now applying the expected value yields

$$
\begin{aligned}
&\mathbb{E}_{[\xi]_{l,k,0}^{l,k,T_{l,k}+1}} \left[\left\| w^L_{l,k,T_{l,k}+1} - w_{L,k,0} \right\|^2 \right] \\
&\leq \mathcal{A}_k \sum_{i=0}^{l} \sum_{t=1}^{T_{i,k}+1} \alpha_{i,k,t} \rho^{2(L-i)} \mathbb{E}_{[\xi]_{l,k,0}^{l,k,T_{l,k}+1}} \left[\left\| g_i^H(w_{i,k,t}, \xi_{i,k,t}) \right\|_i^2 \right] \qquad (3.10) \\
&\leq \mathcal{A}_k \sum_{i=0}^{l} \sum_{t=1}^{T_{i,k}+1} \alpha_{i,k,t} \rho^{2(L-i)} \mathbb{E}_{[\xi]_{l,k,0}^{l,k,T_{l,k}+1}} \left[M_i^H + \overline{M}_i^H \left\| \nabla_i H_i(w_{i,k,t}) \right\|_i^2 \right] .
\end{aligned}
$$

For $\left\| \nabla_i H_i(w_{i,k,t}) \right\|_i^2$ we have that

$$
\begin{aligned}
\left\| \nabla_i H_i(w_{i,k,t}) \right\|_i^2 &= \left\| P_{i+1}^* \dots P_L^* \nabla F_L(w_{L,k,0}) + \nabla_i F_i(w_{i,k,t}) - \nabla_i F_i(w_{i,k,0}) \right\|_i^2 \\
&= \left\| P_{i+1}^* \dots P_L^* \nabla F_L(w_{L,k,0}) \right\|_i^2 + \left\| \nabla_i F_i(w_{i,k,t}) - \nabla_i F_i(w_{i,k,0}) \right\|_i^2 \\
&\quad + 2 \langle P_{i+1}^* \dots P_L^* \nabla F_L(w_{L,k,0}), \nabla_i F_i(w_{i,k,t}) - \nabla_i F_i(w_{i,k,0}) \rangle_i .
\end{aligned}
$$

Since $\langle a, b \rangle_i \leq \frac{1}{2} \left\| a \right\|_i^2 + \frac{1}{2} \left\| b \right\|_i^2$ holds, this implies

$$
\begin{aligned}
\left\| \nabla_i H_i(w_{i,k,t}) \right\|_i^2 &\leq 2 \left\| P_{i+1}^* \dots P_L^* \nabla F_L(w_{L,k,0}) \right\|_i^2 + 2 \left\| \nabla_i F_i(w_{i,k,t}) - \nabla_i F_i(w_{i,k,0}) \right\|_i^2 \\
&\leq 2 \left\| P_{i+1}^* \right\|^2_{i+1,i} \dots \left\| P_L^* \right\|^2_{L,L-1} \left\| \nabla F_L(w_{L,k,0}) \right\|^2 + 2 L_i \left\| w_{i,k,t} - w_{i,k,0} \right\|_i^2 .
\end{aligned}
$$

Then with Assumption 3.6c), this shows

$$\begin{aligned}\left\|\nabla_i H_i(w_{i,k,t})\right\|_i^2 \leq 2\rho^{2(L-i)} \left\|\nabla F_L(w_{L,k,0})\right\|^2 \\ + \frac{2L_i}{\nu^2} \left\| w_{l,k,T_{l,k}+1}^L - w_{l,k,0} \right\|^2 .\end{aligned}$$

Reinserting this into (3.10) yields

$$\begin{aligned}&\mathbb{E}_{[\xi]_{l,k,0}^{l,k,T_{l,k}+1}} \left[\left\| w_{l,k,T_{l,k}+1}^L - w_{L,k,0} \right\|^2 \right] \\ &\leq \mathcal{A}_k \sum_{i=0}^{l} \sum_{t=1}^{T_{i,k}+1} \alpha_{i,k,t} \rho^{2(L-i)} \mathbb{E}_{[\xi]_{l,k,0}^{l,k,T_{l,k}+1}} \left[M_i^H \right. \\ &\quad \left. + \overline{M}_i^H \left(2\rho^{2(L-i)} \left\|\nabla F_L(w_{L,k,0})\right\|^2 + \frac{2L_i}{\nu^2} \left\| w_{l,k,T_{l,k}+1}^L - w_{L,k,0} \right\|^2 \right) \right] \\ &\leq \mathcal{A}_k \sum_{i=0}^{l} \sum_{t=1}^{T_{i,k}+1} \alpha_{i,k,t} \rho^{2(L-i)} M_i^H \\ &\quad + \mathcal{A}_k \sum_{i=0}^{l} \sum_{t=1}^{T_{i,k}+1} \alpha_{i,k,t} \rho^{2(L-i)} \overline{M}_i^H 2\rho^{2(L-i)} \left\|\nabla F_L(w_{L,k,0})\right\|^2 \\ &\quad + \mathcal{A}_k \sum_{i=0}^{l} \sum_{t=1}^{T_{i,k}+1} \alpha_{i,k,t} \rho^{2(L-i)} \overline{M}_i^H \frac{2L_i}{\nu^2} \mathbb{E}_{[\xi]_{l,k,0}^{l,k,T_{l,k}+1}} \left[\left\| w_{l,k,T_{l,k}+1}^L - w_{L,k,0} \right\|^2 \right] .\end{aligned}$$

With the bounds $\mathcal{M}, \overline{\mathcal{M}}, \mathcal{L}$ and $\overline{\rho}$ this is

$$\begin{aligned}&\mathbb{E}_{[\xi]_{l,k,0}^{l,k,T_{l,k}+1}} \left[\left\| w_{l,k,T_{l,k}+1}^L - w_{L,k,0} \right\|^2 \right] \\ &\leq \mathcal{A}_k^2 \overline{\rho} \mathcal{M} + \mathcal{A}_k^2 2\overline{\rho}^2 \overline{\mathcal{M}} \left\|\nabla F_L(w_{L,k,0})\right\|^2 \\ &\quad + \mathcal{A}_k^2 \overline{\rho}\, \overline{\mathcal{M}} \frac{2\mathcal{L}}{\nu^2} \mathbb{E}_{[\xi]_{l,k,0}^{l,k,T_{l,k}+1}} \left[\left\| w_{l,k,T_{l,k}+1}^L - w_{L,k,0} \right\|^2 \right] .\end{aligned}$$

Since by assumption $\left(\sum_{i=0}^{l} \sum_{t=1}^{T_{i,k}+1} \alpha_{i,k,t} \right)^2 \leq \frac{\nu^2}{4\overline{\rho}\,\overline{\mathcal{M}}\mathcal{L}}$, we have $\mathcal{A}_k^2 \overline{\rho}\, \overline{\mathcal{M}} \frac{2\mathcal{L}}{\nu^2} \leq \frac{1}{2}$. Therefore, after rearranging the terms, we get

$$\begin{aligned}&\frac{1}{2} \mathbb{E}_{[\xi]_{l,k,0}^{l,k,T_{l,k}+1}} \left[\left\| w_{l,k,T_{l,k}+1}^L - w_{L,k,0} \right\|^2 \right] \\ &\leq \mathcal{A}_k^2 \overline{\rho} \mathcal{M} + \mathcal{A}_k^2 2\overline{\rho}^2 \overline{\mathcal{M}} \left\|\nabla F_L(w_{L,k,0})\right\|^2\end{aligned}$$

and thus

$$\begin{aligned}&\mathbb{E}_{[\xi]_{l,k,0}^{l,k,T_{l,k}+1}}\left[\left\|w_{l,k,T_{l,k}+1}^{L}-w_{L,k,0}\right\|^2\right]\\&\leq \mathcal{A}_k^2 2\overline{\rho}\mathcal{M}+\mathcal{A}_k^2 4\overline{\rho}^2\overline{\mathcal{M}}\left\|\nabla F_L(w_{L,k,0})\right\|^2.\end{aligned}$$

Therefore, defining $a \coloneqq 2\overline{\rho}\mathcal{M}$ and $A \coloneqq 4\overline{\rho}^2\overline{\mathcal{M}}$, the first statement follows. The second statement follows from

$$\overline{\alpha}\mathcal{T}_{l,k}=\sum_{i=0}^{l}\sum_{t=1}^{T_{i,k}+1}\alpha_{i,k,t}$$

with $\alpha_{i,k,t}=\overline{\alpha}$. □

We now show that under additional assumptions the bound on $\mathbb{E}_{[\xi]_{l,k,0}^{l,k,T_{l,k}+1}}\left[\left\|w_{l,k,T_{l,k}+1}^{L}-w_{L,k,0}\right\|^2\right]$ can be improved.

Lemma 3.9. *Let Assumptions 3.3 to 3.6 hold. Let the iterates $\{w_{l,k,t}\}_{l,k,t}$ with $l = 0,1,\dots,L$, $k=0,\dots K_l-1$ and $t=0,\dots,T_{l,k}+1$ be generated by Algorithm 4 after a recursive call at $w_{L,k,0}$ using a constant step size $\alpha_{l,k,t}=\overline{\alpha}$ and stochastic gradients for the updates, i.e., $\mathbb{E}_{\xi_{l,k,t}}[g_l^H(w_{l,k,t},\xi_{l,k,t})]=\nabla_l H_l(w_{l,k,t})$. We show that under reasonable assumptions (see (3.12)) there is $L_w>0$ such that for $\overline{\alpha}>0$ satisfying*

$$2\overline{\alpha}^2\mathcal{T}_{L,k}\rho^{2L}\left(\mathcal{T}_{L,k}+\overline{\mathcal{M}}(1+\overline{\alpha}L_w\mathcal{L}\mathcal{T}_{L,k})\right)\frac{\mathcal{L}^2}{\nu^2}\leq\frac{1}{2}$$

the estimate holds

$$\begin{aligned}&\mathbb{E}_{[\xi]_{L,k,0}^{L,k+1,0}}[\|w_{L,k+1,0}-w_{L,k,0}\|^2]\\&\leq 4\overline{\alpha}^2\mathcal{T}_{L,k}\rho^{4L}\left(\mathcal{T}_{L,k}+\overline{\mathcal{M}}(1+\overline{\alpha}L_w\mathcal{L}\mathcal{T}_{L,k})\right)\|\nabla F_L(w_{L,k,0})\|^2\\&+2\overline{\alpha}^2\mathcal{T}_{L,k}\rho^{2L}\mathcal{M}(1+\overline{\alpha}L_w\mathcal{L}\mathcal{T}_{L,k})\\&=(\overline{\alpha}^2\mathcal{T}_{L,k}^2)\tilde{A}_k\|\nabla F_L(w_{L,k,0})\|^2+(\overline{\alpha}^2\mathcal{T}_{L,k}^2)\tilde{a}_k\end{aligned}$$

with $\tilde{a}_k=2\overline{\rho}\mathcal{M}(1/\mathcal{T}_{L,k}+\overline{\alpha}L_w\mathcal{L})$ and $\tilde{A}_k=4\overline{\rho}^2\left(1+\overline{\mathcal{M}}(1/\mathcal{T}_{L,k}+\overline{\alpha}L_w\mathcal{L})\right)$.

Proof. By definition, we have

$$\begin{aligned}&w_{L,k+1,0}([\xi]^{L,k+1,0})-w_{L,k,0}([\xi]^{L,k,0})\\&=-\sum_{l=0}^{L}\sum_{t=1}^{T_{l,k}}\alpha_{l,k,t}P_L\cdots P_{l+1}g_l^H(w_{l,k,t}([\xi]^{L,k,t-1}),\xi_{l,k,t}).\end{aligned}$$

Hence, for $\alpha_{l,k,t} = \overline{\alpha}$, we obtain with the notation $(l,t) < (l',t')$ if $l < l'$ or $(l = l'$ and $t < t')$ and the abbreviations $P_{l,L} = P_L \cdots P_{l+1}$ and $w_{l,k,t} = w_{l,k,t}([\xi]^{l,k,t-1})$ that

$$\begin{aligned}
&\mathbb{E}_{[\xi]_{L,k,0}^{L,k+1,0}}[\|w_{L,k+1,0} - w_{L,k,0}\|^2] \\
&= \overline{\alpha}^2 \sum_{l,l'=0}^{L} \sum_{t=1}^{T_{l,k}} \sum_{t'=1}^{T_{l',k}} \mathbb{E}_{[\xi]_{L,k,0}^{L,k+1,0}}[g_l^H(w_{l,k,t}, \xi_{l,k,t})^\top P_{l,L}^\top P_{l',L} g_{l'}^H(w_{l',k,t'}, \xi_{l',k,t'})] \\
&= \overline{\alpha}^2 \sum_{l=0}^{L} \sum_{t=1}^{T_{l,k}} \mathbb{E}_{[\xi]_{l,k,0}^{l,k,t}}[\|P_{l,L} g_l^H(w_{l,k,t}, \xi_{l,k,t})\|^2] \\
&\quad + 2\overline{\alpha}^2 \sum_{(l,t)<(l',t')} \mathbb{E}_{[\xi]_{l,k,0}^{l',k,t'}}[g_l^H(w_{l,k,t}, \xi_{l,k,t})^\top P_{l,L}^\top P_{l',L} \nabla H_{l'}(w_{l',k,t'})].
\end{aligned} \tag{3.11}$$

Here, we have used that $w_{l,k,t}$ and $w_{l',k,t'}$ do not depend on $\xi_{l',k,t'}$ for $(l,t) < (l',t')$. However, $w_{l',k,t'}$ depends on $\xi_{l,k,t}$. To estimate further, we introduce for (l,t) the auxiliary sequence

$$\begin{aligned}
w_{l,k,t+1}^{l,t} &= w_{l,k,t}, \\
w_{l',k,t'+1}^{l,t} &= w_{l',k,t'}^{l,t} - \overline{\alpha} g_{l'}^H(w_{l',k,t'}^{l,t}, \xi_{l',k,t'}) \quad \text{for } (l',t') > (l,t).
\end{aligned}$$

Then $w_{l',k,t'}^{l,t}$ is independent of $\xi_{l,k,t}$, since the corresponding stochastic gradient step is dropped. After that, the updates $-\overline{\alpha} g_{l'}^H(w_{l',k,t'}^{l,t}, \xi_{l',k,t'})$ are used instead of $-\overline{\alpha} g_{l'}^H(w_{l',k,t'}, \xi_{l',k,t'})$. This yields

$$\begin{aligned}
w_{l,k,t+1}^{l,t} - w_{l,k,t+1} &= \overline{\alpha} g_l^H(w_{l,k,t}, \xi_{l,k,t}), \\
w_{l',k,t'+1}^{l,t} - w_{l',k,t'+1} &= w_{l',k,t'}^{l,t} - w_{l',k,t'} - \overline{\alpha}(g_{l'}^H(w_{l',k,t'}^{l,t}, \xi_{l',k,t'}) - g_{l'}^H(w_{l',k,t'}, \xi_{l',k,t'})).
\end{aligned}$$

Hence, the same stochastic gradient path is followed from the slightly different points $w_{l,k,t+1}^{l,t}$ and $w_{l,k,t+1}$.

Let the Lipschitz constant of $g_{l'}^H(\cdot, \xi_{l',k,t'})$ be $L_{l'}$. If $g_{l'}^H(\cdot, \xi_{l',k,t'})$ are gradients of a convex function then for $\bar{\alpha} L_{l'} \leq 2$ it is easy to see that

$$(\mathrm{id} - \overline{\alpha} g_{l'}^H(\cdot, \xi_{l',k,t'}))$$

is non-expansive, i.e.,

$$\begin{aligned}
&\left\| w_{l',k,t'}^{l,t} - w_{l',k,t'} \right\|_{l'} \\
&= \left\| w_{l',k,t'-1}^{l,t} - w_{l',k,t'-1} - \overline{\alpha} \left(g_{l'}^H(w_{l',k,t'-1}^{l,t}, \xi_{l',k,t'-1}) - g_{l'}^H(w_{l',k,t'-1}, \xi_{l',k,t'-1}) \right) \right\|_{l'} \\
&\leq \left\| w_{l',k,t'-1}^{l,t} - w_{l',k,t'-1} \right\|_{l'}.
\end{aligned}$$

Repeating this recursively yields

$$\left\|w^{l,t}_{l',k,t'} - w_{l',k,t'}\right\|_{l'} \leq \left\|w^{l,t}_{l,k,t+1} - w_{l,k,t+1}\right\|_{l} = L_w\overline{\alpha}\left\|g^H_l(w_{l,k,t},\xi_{l,k,t})\right\|_l. \tag{3.12}$$

with $L_w = 1$. Also generally it is likely that the concatenation of the steps has a Lipschitz constant.

We assume now that for $\overline{\alpha}$ small enough, (3.12) holds for some $L_w > 0$. Then we obtain for $(l,t) < (l',t')$ that

$$\begin{aligned}
&\mathbb{E}_{[\xi]^{l',k,t'}_{l,k,0}}[\langle P_{l,L}g^H_l(w_{l,k,t},\xi_{l,k,t}), P_{l',L}\nabla_{l'}H_{l'}(w_{l',k,t'})\rangle] \\
&= \mathbb{E}_{[\xi]^{l',k,t'}_{l,k,0}}[\langle P_{l,L}\nabla_{l}H_{l}(w_{l,k,t}), P_{l',L}\nabla_{l'}H_{l'}(w^{l,t}_{l',k,t'})\rangle] \\
&\quad + \mathbb{E}_{[\xi]^{l',k,t'}_{l,k,0}}[\langle P_{l,L}g^H_l(w_{l,k,t},\xi_{l,k,t}), P_{l',L}(\nabla_{l'}H_{l'}(w_{l',k,t'}) - \nabla_{l'}H_{l'}(w^{l,t}_{l',k,t'})\rangle] \\
&= \mathbb{E}_{[\xi]^{l',k,t'}_{l,k,0}}[\langle P_{l,L}\nabla_{l}H_{l}(w_{l,k,t}), P_{l',L}\nabla_{l'}H_{l'}(w_{l',k,t'})\rangle] \\
&\quad + \mathbb{E}_{[\xi]^{l',k,t'}_{l,k,0}}[\langle P_{l,L}(g^H_l(w_{l,k,t},\xi_{l,k,t}) - \nabla_l H_l(w_{l,k,t})), \\
&\qquad\qquad P_{l',L}(\nabla_{l'}H_{l'}(w_{l',k,t'}) - \nabla_{l'}H_{l'}(w^{l,t}_{l',k,t'}))\rangle] \\
&\leq \mathbb{E}_{[\xi]^{l',k,t'}_{l,k,0}}[\langle P_{l,L}\nabla_{l}H_{l}(w_{l,k,t}), P_{l',L}\nabla_{l'}H_{l'}(w_{l',k,t'})\rangle] \\
&\quad + \mathbb{E}_{[\xi]^{l',k,t'}_{l,k,0}}[\|g^H_l(w_{l,k,t},\xi_{l,k,t}) - \nabla H_l(w_{l,k,t})\|_l\|P_{l,L}\|_{l,L} \\
&\qquad\qquad \|P_{l',L}\|_{l',L}L_{l'}\|w_{l',k,t'} - w^{l,t}_{l',k,t'}\|_{l'}] \\
&\leq \mathbb{E}_{[\xi]^{l',k,t'}_{l,k,0}}[\langle P_{l,L}\nabla_{l}H_{l}(w_{l,k,t}), P_{l',L}\nabla_{l'}H_{l'}(w_{l',k,t'})\rangle] \\
&\quad + \mathbb{E}_{[\xi]^{l',k,t'}_{l,k,0}}[\|g^H_l(w_{l,k,t},\xi_{l,k,t}) - \nabla H_l(w_{l,k,t})\|_l\|P_{l,L}\|_{l,L} \\
&\qquad\qquad \|P_{l',L}\|_{l',L}L_{l'}L_w\overline{\alpha}\|g^H_l(w_{l,k,t},\xi_{l,k,t})\|_l] \\
&\leq \mathbb{E}_{[\xi]^{l',k,t'}_{l,k,0}}[\langle P_{l,L}\nabla_{l}H_{l}(w_{l,k,t}), P_{l',L}\nabla_{l'}H_{l'}(w_{l',k,t'})\rangle] \\
&\quad + \sqrt{\mathbb{E}_{[\xi]^{l',k,t'}_{l,k,0}}[\|g^H_l(w_{l,k,t},\xi_{l,k,t}) - \nabla H_l(w_{l,k,t})\|^2_l]} \\
&\quad \|P_{l,L}\|_{l,L}\|P_{l',L}\|_{l',L}L_{l'}L_w\overline{\alpha}\sqrt{\mathbb{E}_{[\xi]^{l,k,t}_{l,k,0}}[\|g^H_l(w_{l,k,t},\xi_{l,k,t})\|^2_l]}.
\end{aligned}$$

Using that

$$\begin{aligned}
&\mathbb{E}_{[\xi]^{l',k,t'}_{l,k,0}}[\|g^H_l(w_{l,k,t},\xi_{l,k,t}) - \nabla H_l(w_{l,k,t})\|^2_l] \\
&= \mathbb{V}_{[\xi]^{l',k,t'}_{l,k,0}}[g^H_l(w_{l,k,t},\xi_{l,k,t})] \\
&= \mathbb{E}_{[\xi]^{l',k,t'}_{l,k,0}}[\|g^H_l(w_{l,k,t},\xi_{l,k,t})\|^2_l] - \mathbb{E}_{[\xi]^{l',k,t'}_{l,k,0}}[\|g^H_l(w_{l,k,t},\xi_{l,k,t})\|_l]^2 \\
&\leq \mathbb{E}_{[\xi]^{l',k,t'}_{l,k,0}}[\|g^H_l(w_{l,k,t},\xi_{l,k,t})\|^2_l],
\end{aligned}$$

we get

$$\begin{aligned}
&\mathbb{E}_{[\xi]_{l,k,0}^{l',k,t'}}[\langle P_{l,L} g_l^H(w_{l,k,t}, \xi_{l,k,t}), P_{l',L} \nabla_{l'} H_{l'}(w_{l',k,t'})\rangle] \\
&\leq \mathbb{E}_{[\xi]_{l,k,0}^{l',k,t'}}[\langle P_{l,L} \nabla_l H_l(w_{l,k,t}), P_{l',L} \nabla_{l'} H_{l'}(w_{l',k,t'})\rangle] \\
&\quad + \rho^{2L} L_{l'} L_w \overline{\alpha} \mathbb{E}_{[\xi]_{l,k,0}^{l,k,t}}[\|g_l^H(w_{l,k,t}, \xi_{l,k,t})\|_l^2].
\end{aligned}$$

Inserting this in (3.11) yields with $\mathcal{L} := \max_{0 \leq l \leq L} L_l$ that

$$\begin{aligned}
&\mathbb{E}_{[\xi]_{L,k,0}^{L,k+1,0}}[\|w_{L,k+1,0} - w_{L,k,0}\|^2] \\
&\leq \overline{\alpha}^2 \left\| \mathbb{E}_{[\xi]_{L,k,0}^{L,k+1,0}} \left[\sum_{l=0}^{L} \sum_{t=1}^{T_{l,k}} P_{l,L} \nabla_l H_l(w_{l,k,t}) \right] \right\|^2 \\
&\quad + \overline{\alpha}^2 \sum_{l=0}^{L} \sum_{t=1}^{T_{l,k}} \mathbb{E}_{[\xi]_{l,k,0}^{l,k,t}}[\|P_{l,L} g_l^H(w_{l,k,t}, \xi_{l,k,t})\|^2 - \|P_{l,L} \nabla H_l(w_{l,k,t}\|^2] \\
&\quad + \overline{\alpha}^3 \rho^{2L} L_w \mathcal{L} \mathcal{T}_{L,k} \sum_{l=0}^{L} \sum_{t=1}^{T_{l,k}} \mathbb{E}_{[\xi]_{l,k,0}^{l,k,t}}[\|g_l^H(w_{l,k,t}, \xi_{l,k,t})\|_l^2] \\
&\leq \overline{\alpha}^2 \left\| \mathbb{E}_{[\xi]_{L,k,0}^{L,k+1,0}} \left[\sum_{l=0}^{L} \sum_{t=1}^{T_{l,k}} P_{l,L} \nabla_l H_l(w_{l,k,t}) \right] \right\|^2 \\
&\quad + \overline{\alpha}^2 \rho^{2L} (1 + \overline{\alpha} L_w \mathcal{L} \mathcal{T}_{L,k}) \sum_{l=0}^{L} \sum_{t=1}^{T_{l,k}} \mathbb{E}_{[\xi]_{l,k,0}^{l,k,t}}[\|g_l^H(w_{l,k,t}, \xi_{l,k,t})\|_l^2] \\
&\leq \overline{\alpha}^2 \left\| \mathbb{E}_{[\xi]_{L,k,0}^{L,k+1,0}} \left[\sum_{l=0}^{L} \sum_{t=1}^{T_{l,k}} P_{l,L} \nabla_l H_l(w_{l,k,t}) \right] \right\|^2 \\
&\quad + \overline{\alpha}^2 \rho^{2L} (1 + \overline{\alpha} L_w \mathcal{L} \mathcal{T}_{L,k}) \sum_{l=0}^{L} \sum_{t=1}^{T_{l,k}} \left(M_l^H + \overline{M}_l^H \mathbb{E}_{[\xi]_{l,k,0}^{l,k,t}}[\|\nabla H_l(w_{l,k,t})\|_l^2] \right).
\end{aligned}$$

Applying Jensen's inequality to the first term leads to

$$\begin{aligned}
&\overline{\alpha}^2 \left\| \mathbb{E}_{[\xi]_{L,k,0}^{L,k+1,0}} \left[\sum_{l=0}^{L} \sum_{t=1}^{T_{l,k}} P_{l,L} \nabla_l H_l(w_{l,k,t}) \right] \right\|^2 \\
&= \overline{\alpha}^2 \mathcal{T}_{l,k}^2 \left\| \mathbb{E}_{[\xi]_{L,k,0}^{L,k+1,0}} \left[\sum_{l=0}^{L} \sum_{t=1}^{T_{l,k}} \frac{1}{\mathcal{T}_{l,k}} P_{l,L} \nabla_l H_l(w_{l,k,t}) \right] \right\|^2 \\
&\leq \overline{\alpha}^2 \mathcal{T}_{L,k}^2 \sum_{l=0}^{L} \sum_{t=1}^{T_{l,k}} \frac{1}{\mathcal{T}_{L,k}} \mathbb{E}_{[\xi]_{l,k,0}^{l,k,t}} [\| P_{l,L} \nabla_l H_l(w_{l,k,t}) \|^2] \\
&= \overline{\alpha}^2 \mathcal{T}_{L,k} \sum_{l=0}^{L} \sum_{t=1}^{T_{l,k}} \mathbb{E}_{[\xi]_{l,k,0}^{l,k,t}} [\| P_{l,L} \nabla_l H_l(w_{l,k,t}) \|^2].
\end{aligned}$$

Hence, we arrive with the bounds $\mathcal{M} := \max_{0 \leq l \leq L} M_l^H$, $\overline{\mathcal{M}} := \max_{0 \leq l \leq L} \overline{M}_l^H$ at

$$\begin{aligned}
&\mathbb{E}_{[\xi]^{L,k+1,0}} [\| w_{L,k+1,0} - w_{L,k,0} \|^2] \\
&\leq \overline{\alpha}^2 \rho^{2L} \left(\mathcal{T}_{L,k} + \overline{\mathcal{M}} (1 + \overline{\alpha} L_w \mathcal{L} \mathcal{T}_{L,k}) \right) \sum_{l=0}^{L} \sum_{t=1}^{T_{l,k}} \mathbb{E}_{[\xi]^{l,k,t}} [\| \nabla_l H_l(w_{l,k,t}) \|_l^2] \\
&\quad + \overline{\alpha}^2 \mathcal{T}_{L,k} \rho^{2L} \mathcal{M} (1 + \overline{\alpha} L_w \mathcal{L} \mathcal{T}_{L,k}).
\end{aligned}$$

Using the definition of $\nabla_l H_l$, we obtain with Assumption 3.6c)

$$\begin{aligned}
\| \nabla_l H_l(w_{l,k,t}) \|_l &\leq \| P_{l,L}^\top \nabla F_L(w_{L,k,0}) \|_l + \| \nabla F_l(w_{l,k,t}) - \nabla F_l(w_{l,k,0}) \|_l \\
&\leq \rho^L \| \nabla F_L(w_{L,k,0}) \| + L_l \| w_{l,k,t} - w_{l,k,0} \|_l \\
&\leq \rho^L \| \nabla F_L(w_{L,k,0}) \| + \frac{L_l}{\nu} \| w_{L,k+1,0} - w_{L,k,0} \|
\end{aligned}$$

and consequently

$$\| \nabla_l H_l(w_{l,k,t}) \|_l^2 \leq 2\rho^{2L} \| \nabla F_L(w_{L,k,0}) \|^2 + 2 \frac{\mathcal{L}^2}{\nu^2} \| w_{L,k+1,0} - w_{L,k,0} \|^2.$$

Hence, if $\overline{\alpha} > 0$ is small enough such that

$$2\overline{\alpha}^2 \mathcal{T}_{L,k} \rho^{2L} \left(\mathcal{T}_{L,k} + \overline{\mathcal{M}} (1 + \overline{\alpha} L_w \mathcal{L} \mathcal{T}_{L,k}) \right) \frac{\mathcal{L}^2}{\nu^2} \leq \frac{1}{2}$$

then

$$\begin{aligned}
&\mathbb{E}_{[\xi]_{L,k,0}^{L,k+1,0}}[\|w_{L,k+1,0} - w_{L,k,0}\|^2] \\
&\leq 4\overline{\alpha}^2\mathcal{T}_{L,k}\rho^{4L}\left(\mathcal{T}_{L,k} + \overline{\mathcal{M}}(1+\overline{\alpha}L_w\mathcal{L}\mathcal{T}_{L,k})\right)\|\nabla F_L(w_{L,k,0})\|^2 \\
&\quad + 2\overline{\alpha}^2\mathcal{T}_{L,k}\rho^{2L}\mathcal{M}(1+\overline{\alpha}L_w\mathcal{L}\mathcal{T}_{L,k}).
\end{aligned}$$

□

Now applying the results from the lemmata before, we state a convergence result for Algorithm 4 for the case of fixed step size selection and a deterministic gradient correction.

Theorem 3.10 (Convergence rate for MLSGD)**.** *Let Assumptions 3.3, 3.5 and 3.6 hold. Suppose that the MLSGD Algorithm 4 is run with a fixed step size $\overline{\alpha}$. Assume the step size is sufficiently small, i.e., fulfills*

$$\overline{\alpha}\mathcal{T}_{L,k} \leq \min\left\{\frac{\mu}{\Theta L_L 4\overline{\rho}^2\overline{\mathcal{M}}}, \frac{\nu}{2\sqrt{\overline{\rho}\overline{\mathcal{M}}\mathcal{L}}}\right\}.$$

Then the following inequality holds

$$\begin{aligned}
&\frac{1}{\sum_{k=0}^{K-1}\mathcal{T}_{L,k}}\sum_{k=0}^{K-1}\mathcal{T}_{L,k}\mathbb{E}_\xi\left[\left\|\nabla F_L\left(w_{L,k,0}\left([\xi]^{L,k,0}\right)\right)\right\|^2\right] \\
&\leq \frac{2\Theta L_L\overline{\rho}\mathcal{M}\overline{\alpha}\sum_{k=0}^{K-1}\mathcal{T}_{L,k}^2}{\mu\sum_{k=0}^{K-1}\mathcal{T}_{L,k}} + \frac{2\Theta(F_L(w_{L,0,0}) - F_{\inf})}{\mu\overline{\alpha}\sum_{k=0}^{K-1}\mathcal{T}_{L,k}}
\end{aligned}$$

for all $K \in \mathbb{N}$. The last term converges to zero as K approaches infinity.

Proof. Lemma 3.7 yields

$$\begin{aligned}
&F_L(w_{L,k+1,0}) - F_L(w_{L,k,0}) \\
&= F_L(w_{L,k,\mathcal{T}_{L,k}+1}) - F_L(w_{L,k,0}) \\
&\leq -\mu\|\nabla F_L(w_{L,k,0})\|\left\|w_{L,k,\mathcal{T}_{L,k}+1} - w_{L,k,0}\right\| + \frac{1}{2}L_L\left\|w_{L,k,\mathcal{T}_{L,k}+1}^L - w_{L,k,0}\right\|^2.
\end{aligned}$$

Now we apply the expected value to get

$$\begin{aligned}
&\mathbb{E}_{[\xi]_{L,k,0}^{L,k+1,0}}\left[F_L(w_{L,k+1,0}) - F_L(w_{L,k,0})\right] \\
&\leq -\mu\|\nabla F_L(w_{L,k,0})\|\,\mathbb{E}_{[\xi]_{L,k,0}^{L,k+1,0}}\left[\left\|w_{L,k,\mathcal{T}_{L,k}+1} - w_{L,k,0}\right\|\right] \\
&\quad + \frac{1}{2}L_L\mathbb{E}_{[\xi]_{L,k,0}^{L,k+1,0}}\left[\left\|w_{L,k,\mathcal{T}_{L,k}+1} - w_{L,k,0}\right\|^2\right].
\end{aligned}$$

By using (3.7), we obtain

$$\begin{aligned}
&\mathbb{E}_{[\xi]_{L,k,0}^{L,k+1,0}}\left[F_L(w_{L,k+1,0}) - F_L(w_{L,k,0})\right] \\
&\leq -\frac{\mu}{\Theta}\overline{\alpha}\mathcal{T}_{L,k}\left\|\nabla F_L(w_{L,k,0})\right\|^2 \\
&\quad + \frac{1}{2}L_L\mathbb{E}_{[\xi]_{L,k,0}^{L,k+1,0}}\left[\left\|w_{L,k,T_{L,k}+1} - w_{L,k,0}\right\|^2\right].
\end{aligned}$$

Using the bound on $\overline{\alpha}$ together with Lemma 3.8 yields

$$\begin{aligned}
&\mathbb{E}_{[\xi]_{L,k,0}^{L,k+1,0}}\left[F_L(w_{L,k+1,0}) - F_L(w_{L,k,0})\right] \\
&\leq -\frac{\mu}{\Theta}\overline{\alpha}\mathcal{T}_{L,k}\left\|\nabla F_L(w_{L,k,0})\right\|^2 \\
&\quad + \frac{1}{2}L_L\left(a\overline{\alpha}^2\mathcal{T}_{L,k}^2 + A\overline{\alpha}^2\mathcal{T}_{L,k}^2\right)\left\|\nabla F_L(w_{L,k,0})\right\|^2 \\
&= \left(-\frac{\mu}{\Theta} + \frac{1}{2}L_L A\overline{\alpha}\mathcal{T}_{L,k}\right)\overline{\alpha}\mathcal{T}_{L,k}\left\|\nabla F_L(w_{L,k,0})\right\|^2 \\
&\quad + \frac{1}{2}L_L a\overline{\alpha}^2\mathcal{T}_{L,k}^2.
\end{aligned}$$

From this we can use the other bound on $\overline{\alpha}$, that

$$\overline{\alpha}\mathcal{T}_{L,k} \leq \frac{\mu}{\Theta L_L A}.$$

Hence

$$\begin{aligned}
&\mathbb{E}_{[\xi]_{L,k,0}^{L,k+1,0}}\left[F_L(w_{L,k+1,0}) - F_L(w_{L,k,0})\right] \\
&\leq -\frac{1}{2}\frac{\mu}{\Theta}\mathcal{T}_{L,k}\overline{\alpha}\left\|\nabla F_L(w_{L,k,0})\right\|^2 + \frac{1}{2}L_L a\overline{\alpha}^2\mathcal{T}_{L,k}^2.
\end{aligned}$$

Taking the total expectation with respect to $\xi = \{\xi_{i,j,s}\}_{i,j,s\in\mathbb{N}}$ gives

$$\begin{aligned}
&\mathbb{E}_{\xi}\left[F_L(w_{L,k+1,0}([\xi]^{L,k+1,0})) - F_L(w_{L,k,0}([\xi]^{L,k,0}))\right] \\
&\leq -\frac{1}{2}\frac{\mu}{\Theta}\mathcal{T}_{L,k}\overline{\alpha}\mathbb{E}_{\xi}\left[\left\|\nabla F_L\left(w_{L,k,0}\left([\xi]^{L,k,0}\right)\right)\right\|^2\right] + \frac{1}{2}L_L a\overline{\alpha}^2\mathcal{T}_{L,k}^2.
\end{aligned}$$

Finally, for $K \in \mathbb{N}$ it follows

$$\begin{aligned}
&F_{\inf} - F_L(w_{L,0,0}) \\
&\leq \mathbb{E}_\xi \left[F_L(w_{L,K,0}([\xi]^{L,K,0})) \right] - F_L(w_{L,0,0}) \\
&= \sum_{k=0}^{K-1} \mathbb{E}_\xi [F_L(w_{L,k+1,0}([\xi]^{L,k+1,0}))] - \mathbb{E}_\xi [F_L(w_{L,k,0}([\xi]^{L,k,0}))] \\
&\leq -\frac{1}{2}\frac{\mu}{\Theta}\overline{\alpha} \sum_{k=0}^{K-1} \mathcal{T}_{L,k} \mathbb{E}_\xi \left[\left\| \nabla F_L \left(w_{L,k,0} \left([\xi]^{L,k,0} \right) \right) \right\|^2 \right] \\
&\quad + \frac{1}{2} L_L a \overline{\alpha}^2 \sum_{k=0}^{K-1} \mathcal{T}_{L,k}^2.
\end{aligned}$$

Therefore, after rearrangement we have

$$\begin{aligned}
&\sum_{k=0}^{K-1} \mathcal{T}_{L,k} \mathbb{E}_\xi \left[\left\| \nabla F_L \left(w_{L,k,0} \left([\xi]^{L,k,0} \right) \right) \right\|^2 \right] \\
&\leq \frac{\frac{1}{2} L_L a \overline{\alpha}^2 \sum_{k=0}^{K-1} \mathcal{T}_{L,k}^2}{\frac{1}{2}\frac{\mu}{\Theta}\overline{\alpha}} + \frac{F_L(w_{L,0,0}) - F_{\inf}}{\frac{1}{2}\frac{\mu}{\Theta}\overline{\alpha}} \\
&= \frac{\Theta L_L a \overline{\alpha} \sum_{k=0}^{K-1} \mathcal{T}_{L,k}^2}{\mu} + \frac{2\Theta(F_L(w_{L,0,0}) - F_{\inf})}{\mu\overline{\alpha}},
\end{aligned}$$

which yields

$$\begin{aligned}
&\frac{1}{\sum_{k=0}^{K-1} \mathcal{T}_{L,k}} \sum_{k=0}^{K-1} \mathcal{T}_{L,k} \mathbb{E}_\xi \left[\left\| \nabla F_L \left(w_{L,k,0} \left([\xi]^{L,k,0} \right) \right) \right\|^2 \right] \\
&\leq \frac{\Theta L_L a \overline{\alpha} \sum_{k=0}^{K-1} \mathcal{T}_{L,k}^2}{\mu \sum_{k=0}^{K-1} \mathcal{T}_{L,k}} + \frac{2\Theta(F_L(w_{L,0,0}) - F_{\inf})}{\mu\overline{\alpha} \sum_{k=0}^{K-1} \mathcal{T}_{L,k}} \\
&= \frac{2\Theta L_L \overline{\rho} \mathcal{M} \overline{\alpha} \sum_{k=0}^{K-1} \mathcal{T}_{L,k}^2}{\mu \sum_{k=0}^{K-1} \mathcal{T}_{L,k}} + \frac{2\Theta(F_L(w_{L,0,0}) - F_{\inf})}{\mu\overline{\alpha} \sum_{k=0}^{K-1} \mathcal{T}_{L,k}}
\end{aligned}$$

after division by $\sum_{k=0}^{K-1} \mathcal{T}_{L,k}$ and inserting the definition of a and A. As in every cycle we are doing at least T_L steps, we have that

$$\sum_{k=0}^{K-1} \sum_{l=0}^{L} T_{l,k} \geq K T_L \quad \xrightarrow{K \to \infty} \quad \infty$$

and thus

$$\frac{2\Theta(F_L(w_{L,0,0}) - F_{\inf})}{\sum_{k=0}^{K-1}\sum_{l=0}^{L} T_{l,k}\mu\overline{\alpha}} \quad \xrightarrow{K\to\infty} \quad 0.$$

This finishes the proof. □

Similar to the result for SGD in Theorem 2.12, in Theorem 3.10 the convergence depends on the step size, the noise in the steps and the initial point. When the initial point is close to a minimum then the last term is already small and decreases further as the number of iterations K increases. Under additional assumptions, using the bounds from Lemma 3.9, the bound in Theorem 3.10 can be improved.

Theorem 3.11. *Let Assumptions 3.3 to 3.6 hold. Further, let the assumptions in Lemma 3.9 hold. Suppose that the MLSGD Algorithm 4 is run with a fixed step size $\overline{\alpha}$. Assume the step size is sufficiently small, i.e., fulfills*

$$\overline{\alpha}\mathcal{T}_{L,k} \leq \min\left\{\frac{\mu}{\Theta L_L 4\overline{\rho}^2\overline{\mathcal{M}}(1/\mathcal{T}_{L,k} + \overline{\alpha}L_w\mathcal{L})}, \frac{\nu}{2\mathcal{L}\rho^L\sqrt{1+\overline{\mathcal{M}}(1/\mathcal{T}_{L,k}+\overline{\alpha}L_w\mathcal{L})}}\right\}.$$

Then the following inequality holds

$$\begin{aligned}&\frac{1}{\sum_{k=0}^{K-1}\mathcal{T}_{L,k}}\sum_{k=0}^{K-1}\mathcal{T}_{L,k}\mathbb{E}_\xi\left[\left\|\nabla F_L\left(w_{L,k,0}\left([\xi]^{L,k,0}\right)\right)\right\|^2\right]\\ &\leq \frac{2\Theta L_L\rho^{2L}\mathcal{M}\overline{\alpha}\sum_{k=0}^{K-1}\left((1+\overline{\alpha}L_w\mathcal{L}\mathcal{T}_{L,k})\mathcal{T}_{L,k}\right)}{\mu\sum_{k=0}^{K-1}\mathcal{T}_{L,k}} + \frac{2\Theta(F_L(w_{L,0,0})-F_{\inf})}{\mu\overline{\alpha}\sum_{k=0}^{K-1}\mathcal{T}_{L,k}}\end{aligned}$$

for all $K \in \mathbb{N}$. The last term converges to zero as K approaches infinity.

Proof. The proof is analogous to that of Theorem 3.10 except for using the improved bounds $\tilde{a}_k$ and $\tilde{A}_k$ of Lemma 3.9 instead of a, A from Lemma 3.8. □

Compared to Theorem 3.10, in this case the first term contains instead of $\frac{\sum_{k=0}^{K-1}\mathcal{T}_{L,k}^2}{\sum_{k=0}^{K-1}\mathcal{T}_{L,k}}$ the smaller factor $\frac{\sum_{k=0}^{K-1}((1+\overline{\alpha}L_w\mathcal{L}\mathcal{T}_{L,k})\mathcal{T}_{L,k})}{\sum_{k=0}^{K-1}\mathcal{T}_{L,k}}$.

If the stochastic directions have no variance or variance proportional to $\|\nabla F_L(w_{L,k,0})\|^2$, i.e., $\mathcal{M} = 0$, then the first term is zero. Otherwise, the size of the remaining constant can be controlled through the size of the step size and the number of steps on the lower levels. Thus, when using a smaller step size and less steps on the lower levels, the first term is small but then the second term is larger indicating that the algorithm makes slower progress.

3.3.3 Inserting a variance reduction term

Reducing the variance of the updates by using a variance reduction term can improve the convergence of SGD, as discussed in Section 2.3.4. Therefore, it seems appropriate to apply it in a multilevel setting as well and investigate its interaction with a gradient correction. For the objective function F_l with $l = 0, \dots, L$ a variance reduced update is

$$\tilde{g}_l^F(w_{l,k,t}, \xi_{l,k,t}) = g_l^F(w_{l,k,t}, \xi_{l,k,t}) + (\nabla_l F_l(w_{l,k,0}) - g_l^F(w_{l,k,0}, \xi_{l,k,t}))$$

instead of $g_l^F(w_{l,k,t}, \xi_{l,k,t})$. However, since we have adjusted the objective function to use a gradient correction term, we need to examine the update with respect to the objective function H_l. Remember that because of first order coherence, the objective function is

$$H_l(w_l) = F_l(w_l) + \langle P_{l+1}^* \dots P_L^* \nabla F_L(w_{L,k,0}) - \nabla_l F_l(w_{l,k,0}), w_l \rangle_l.$$

With this in mind, applying variance reduction to the objective function H_l for $l = 0, \dots, L$ results in the update

$$\begin{aligned}\tilde{g}_l^H(w_{l,k,t}, \xi_{l,k,t}) &= g_l^F(w_{l,k,t}, \xi_{l,k,t}) + (P_{l+1}^* \dots P_L^* \nabla F_L(w_{L,k,0}) - \nabla_l F_l(w_{l,k,0})) \\ &\quad + (\nabla_l F_l(w_{l,k,0}) - g_l^F(w_{l,k,0}, \xi_{l,k,t})) \\ &= g_l^F(w_{l,k,t}, \xi_{l,k,t}) + (P_{l+1}^* \dots P_L^* \nabla F_L(w_{L,k,0}) - g_l^F(w_{l,k,0}, \xi_{l,k,t})).\end{aligned}$$

Using the new variance reduced update rule, we can give a different estimate of the expected value of the SGD steps than the one in Lemma 3.8.

Lemma 3.12. *Let Assumptions 3.3 to 3.6 hold. Let the iterates $\{w_{l,k,t}\}_{l,k,t}$ with $l = 0, 1, \dots, L$, $k = 0, \dots K_l - 1$ and $t = 0, \dots, T_{l,k} + 1$ be generated by Algorithm 4 after a recursive call at $w_{L,k,0}$. Assume that the step size is sufficiently small, i.e., fulfills*

$$\left(\sum_{i=0}^{l} \sum_{t=1}^{T_{i,k}+1} \alpha_{i,k,t} \right)^2 \leq \frac{\nu^2}{4\overline{\rho}\mathcal{L}}.$$

Then for all $l = 0, \dots, L$ and $k \in \mathbb{N}$, we have

$$\begin{aligned}&\mathbb{E}_{[\xi]_{l,k,0}^{l,k,T_{l,k}+1}} \left[\left\| w_{l,k,T_{l,k}+1}^L - w_{L,k,0} \right\|^2 \right] \\ &\leq \left(\sum_{i=0}^{l} \sum_{t=1}^{T_{i,k}+1} \alpha_{i,k,t} \right)^2 B \left\| \nabla F(w_{L,k,0}) \right\|^2\end{aligned}$$

with $B \coloneqq 4\overline{\rho}^2$*. If the step size is constant* $\overline{\alpha}$*, then*

$$\mathbb{E}_{[\xi]_{l,k,0}^{l,k+1,0}}\left[\left\|w_{l,k,T_{l,k}+1}^{L} - w_{L,k,0}\right\|^2\right] \leq B\overline{\alpha}^2\mathcal{T}_{l,k}^2 \left\|\nabla F_L(w_{L,k,0})\right\|^2. \tag{3.13}$$

Proof. The proof is very similar to that of Lemma 3.8. We define $\mathcal{A}_k \coloneqq \sum_{i=0}^{l}\sum_{t=1}^{T_{i,k}+1}\alpha_{i,k,t}$. Then with Jensen's inequality, we get

$$\begin{aligned}
&\left\|w_{l,k,T_{l,k}+1}^{L} - w_{L,k,0}\right\|^2 \\
&= \left\|\sum_{i=0}^{l}\sum_{t=1}^{T_{i,k}+1}\alpha_{i,k,t}P_L\dots P_{i+1}\tilde{g}_i^H(w_{i,k,t},\xi_{i,k,t})\right\|^2 \\
&= \mathcal{A}_k^2\left\|\sum_{i=0}^{l}\sum_{t=1}^{T_{i,k}+1}\frac{\alpha_{i,k,t}}{\mathcal{A}_k}P_L\dots P_{i+1}\tilde{g}_i^H(w_{i,k,t},\xi_{i,k,t})\right\|^2 \\
&\leq \mathcal{A}_k\sum_{i=0}^{l}\sum_{t=1}^{T_{i,k}+1}\alpha_{i,k,t}\left\|P_L\dots P_{i+1}\tilde{g}_i^H(w_{i,k,t},\xi_{i,k,t})\right\|^2 \\
&\leq \mathcal{A}_k\sum_{i=0}^{l}\sum_{t=1}^{T_{i,k}+1}\alpha_{i,k,t}\left\|P_L\right\|_{L-1,L}^2\dots\left\|P_{i+1}\right\|_{i,i+1}^2\left\|\tilde{g}_i^H(w_{i,k,t},\xi_{i,k,t})\right\|_i^2 \\
&\leq \mathcal{A}_k\sum_{i=0}^{l}\sum_{t=1}^{T_{i,k}+1}\alpha_{i,k,t}\rho^{2(L-i)}\left\|\tilde{g}_i^H(w_{i,k,t},\xi_{i,k,t})\right\|_i^2.
\end{aligned}$$

Now applying the expected value yields

$$\begin{aligned}
&\mathbb{E}_{[\xi]_{l,k,0}^{l,k,T_{l,k}+1}}\left[\left\|w_{l,k,T_{l,k}+1}^{L} - w_{L,k,0}\right\|^2\right] \\
&\leq \mathcal{A}_k\sum_{i=0}^{l}\sum_{t=1}^{T_{i,k}+1}\alpha_{i,k,t}\rho^{2(L-i)}\mathbb{E}_{[\xi]_{l,k,0}^{l,k,T_{l,k}+1}}\left[\left\|\tilde{g}_i^H(w_{i,k,t},\xi_{i,k,t})\right\|_i^2\right].
\end{aligned} \tag{3.14}$$

For $\left\|\tilde{g}_i^H(w_{i,k,t},\xi_{i,k,t})\right\|_i^2$ we have that

$$\begin{aligned}
&\left\|\tilde{g}_i^H(w_{i,k,t},\xi_{i,k,t})\right\|_i^2 \\
&= \left\|P_{i+1}^*\dots P_L^*\nabla F_L(w_{L,k,0}) + g_i^F(w_{i,k,t},\xi_{i,k,t}) - g_i^F(w_{i,k,0},\xi_{i,k,t})\right\|_i^2 \\
&= \left\|P_{i+1}^*\dots P_L^*\nabla F_L(w_{L,k,0})\right\|_i^2 \\
&\quad + \left\|g_i^F(w_{i,k,t},\xi_{i,k,t}) - g_i^F(w_{i,k,0},\xi_{i,k,t})\right\|_i^2 \\
&\quad + 2\langle P_{i+1}^*\dots P_L^*\nabla F_L(w_{L,k,0}), g_i^F(w_{i,k,t},\xi_{i,k,t}) - g_i^F(w_{i,k,0},\xi_{i,k,t})\rangle_i.
\end{aligned}$$

And since $\langle a, b\rangle_i \leq \frac{1}{2}\|a\|_i^2 + \frac{1}{2}\|b\|_i^2$ holds, we obtain

$$\begin{aligned}\left\|\tilde{g}_i^H(w_{i,k,t}, \xi_{i,k,t})\right\|_i^2 &\leq 2\left\|P_{i+1}^* \dots P_L^* \nabla F_L(w_{L,k,0})\right\|_i^2 \\ &\quad + 2\left\|g_i^F(w_{i,k,t}, \xi_{i,k,t}) - g_i^F(w_{i,k,0}, \xi_{i,k,t})\right\|_i^2 \\ &\leq 2\left\|P_{i+1}^*\right\|_{i+1,i}^2 \dots \|P_L^*\|_{L,L-1}^2 \|\nabla F_L(w_{L,k,0})\|^2 \\ &\quad + 2L_i \|w_{i,k,t} - w_{i,k,0}\|_i^2 .\end{aligned}$$

Then with Assumption 3.6c) this yields

$$\begin{aligned}\left\|\tilde{g}_i^H(w_{i,k,t}, \xi_{i,k,t})\right\|_i^2 &\leq 2\rho^{2(L-i)} \|\nabla F_L(w_{L,k,0})\|^2 \\ &\quad + \frac{2L_i}{\nu^2}\left\|w_{l,k,T_{l,k}+1}^L - w_{l,k,0}\right\|^2 .\end{aligned}$$

Reinserting this into (3.14) yields

$$\begin{aligned}&\mathbb{E}_{[\xi]_{l,k,0}^{l,k,T_{l,k}+1}}\left[\left\|w_{l,k,T_{l,k}+1}^L - w_{L,k,0}\right\|^2\right] \\ &\leq \mathcal{A}_k \sum_{i=0}^{l}\sum_{t=1}^{T_{i,k}+1} \alpha_{i,k,t}\rho^{2(L-i)} 2\rho^{2(L-i)} \|\nabla F_L(w_{L,k,0})\|^2 \\ &\quad + \mathcal{A}_k \sum_{i=0}^{l}\sum_{t=1}^{T_{i,k}+1} \alpha_{i,k,t}\rho^{2(L-i)} \frac{2L_i}{\nu^2} \mathbb{E}_{[\xi]_{l,k,0}^{l,k,T_{l,k}+1}}\left[\left\|w_{l,k,T_{l,k}+1}^L - w_{L,k,0}\right\|^2\right].\end{aligned}$$

With the bounds $\mathcal{M}, \overline{\mathcal{M}}, \mathcal{L}$ and $\overline{\rho}$ we obtain

$$\begin{aligned}&\mathbb{E}_{[\xi]_{l,k,0}^{l,k,T_{l,k}+1}}\left[\left\|w_{l,k,T_{l,k}+1}^L - w_{L,k,0}\right\|^2\right] \\ &\leq \mathcal{A}_k^2 2\overline{\rho}^2 \|\nabla F_L(w_{L,k,0})\|^2 \\ &\quad + \mathcal{A}_k^2 \overline{\rho}\frac{2\mathcal{L}}{\nu^2} \mathbb{E}_{[\xi]_{l,k,0}^{l,k,T_{l,k}+1}}\left[\left\|w_{l,k,T_{l,k}+1}^L - w_{L,k,0}\right\|^2\right].\end{aligned}$$

Since by assumption $\mathcal{A}_k^2 \leq \frac{\nu^2}{4\overline{\rho}\mathcal{L}}$, we have $\mathcal{A}_k^2\overline{\rho}\frac{2\mathcal{L}}{\nu^2} \leq \frac{1}{2}$. Therefore, after rearranging the terms, we get

$$\frac{1}{2}\mathbb{E}_{[\xi]_{l,k,0}^{l,k,T_{l,k}+1}}\left[\left\|w_{l,k,T_{l,k}+1}^L - w_{L,k,0}\right\|^2\right] \leq \mathcal{A}_k^2 2\overline{\rho}^2 \|\nabla F_L(w_{L,k,0})\|^2$$

and thus

$$\mathbb{E}_{[\xi]_{l,k,0}^{l,k,T_{l,k}+1}}\left[\left\|w_{l,k,T_{l,k}+1}^L - w_{L,k,0}\right\|^2\right] \leq \mathcal{A}_k^2 4\overline{\rho}^2 \|\nabla F_L(w_{L,k,0})\|^2 .$$

Therefore, defining $B \coloneqq 4\overline{\rho}^2$, the first statement follows. The second statement follows from

$$\overline{\alpha}\mathcal{T}_{l,k} = \sum_{i=0}^{l} \sum_{t=1}^{T_{i,k}+1} \alpha_{i,k,t}$$

with $\alpha_{i,k,t} = \overline{\alpha}$. □

In this case, there is no additional value in giving a bound similar to Lemma 3.9.

Comparing Lemma 3.8 with Lemma 3.12, the advantage of variance reduction is that it eliminates the constant term in the variance bound. As we will see in the proof of the next theorem, this allows the expected value of the mean of the norm of the gradients to approach zero if the step size is sufficiently small.

Theorem 3.13 (Convergence rate for MLSGD with variance reduction)**.** *Let Assumptions 3.3 to 3.6 hold. Suppose that the MLSGD Algorithm 4 is run with a fixed step size $\overline{\alpha}$. Assume the step size is sufficiently small, i.e., fulfills*

$$\overline{\alpha}\mathcal{T}_{L,k} \leq \min\left\{\frac{\mu}{\Theta L_L 4\overline{\rho}^2}, \frac{\nu}{2\sqrt{\overline{\rho}\mathcal{L}}}\right\}.$$

Then the following inequality holds

$$\frac{1}{\sum_{k=0}^{K-1}\mathcal{T}_{L,k}} \sum_{k=0}^{K-1} \mathcal{T}_{L,k}\mathbb{E}_{\xi}\left[\left\|\nabla F_L\left(w_{L,k,0}\left([\xi]^{L,k,0}\right)\right)\right\|^2\right] \leq \frac{2\Theta(F_L(w_{L,0,0}) - F_{\inf})}{\mu\overline{\alpha}\sum_{k=0}^{K-1}\mathcal{T}_{L,k}}$$

for all $K \in \mathbb{N}$. This converges to zero as K approaches infinity.

Proof. The proof is analogous to that of Theorem 3.10 only instead of using a and A, we use 0 and B. We can follow the proof of Theorem 3.10 until we arrive at

$$\begin{aligned}
&\mathbb{E}_{[\xi]_{L,k,0}^{L,k+1,0}}\left[F_L(w_{L,k+1,0}) - F_L(w_{L,k,0})\right] \\
&\leq -\frac{\mu}{\Theta}\overline{\alpha}\mathcal{T}_{L,k}\left\|\nabla F_L(w_{L,k,0})\right\|^2 + \frac{1}{2}L_L\mathbb{E}_{[\xi]_{L,k,0}^{L,k+1,0}}\left[\left\|w_{L,k,T_{L,k}+1} - w_{L,k,0}\right\|^2\right],
\end{aligned}$$

where we use Lemma 3.12 with the bound on $\overline{\alpha}$ to get

$$\begin{aligned}
&\mathbb{E}_{[\xi]_{L,k,0}^{L,k+1,0}}\left[F_L(w_{L,k+1,0}) - F_L(w_{L,k,0})\right] \\
&\leq -\frac{\mu}{\Theta}\overline{\alpha}\mathcal{T}_{L,k}\left\|\nabla F_L(w_{L,k,0})\right\|^2 + \frac{1}{2}L_L\left(B\overline{\alpha}^2\mathcal{T}_{L,k}^2\right)\left\|\nabla F_L(w_{L,k,0})\right\|^2 \\
&= \left(-\frac{\mu}{\Theta} + \frac{1}{2}L_L B\overline{\alpha}\mathcal{T}_{L,k}\right)\overline{\alpha}\mathcal{T}_{L,k}\left\|\nabla F_L(w_{L,k,0})\right\|^2.
\end{aligned}$$

With the following bound

$$\overline{\alpha}\mathcal{T}_{L,k} \leq \frac{\mu}{\Theta L_L B},$$

we obtain

$$\mathbb{E}_{[\xi]_{L,k,0}^{L,k+1,0}}\left[F_L(w_{L,k+1,0}) - F_L(w_{L,k,0})\right] \leq -\frac{1}{2}\frac{\mu}{\Theta}\mathcal{T}_{L,k}\overline{\alpha}\left\|\nabla F_L(w_{L,k,0})\right\|^2.$$

Then similar as before, taking the total expectation with respect to $\xi = \{\xi_{i,j,s}\}_{i,j,s\in\mathbb{N}}$ gives

$$\begin{aligned}&\mathbb{E}_\xi\left[F_L(w_{L,k+1,0}([\xi]^{L,k+1,0})) - F_L(w_{L,k,0}([\xi]^{L,k,0}))\right]\\ &\leq -\frac{1}{2}\frac{\mu}{\Theta}\mathcal{T}_{L,k}\overline{\alpha}\mathbb{E}_\xi\left[\left\|\nabla F_L\left(w_{L,k,0}\left([\xi]^{L,k,0}\right)\right)\right\|^2\right].\end{aligned}$$

Finally, for $K \in \mathbb{N}$ it follows

$$\begin{aligned}&F_{\inf} - F_L(w_{L,0,0})\\ &\leq \mathbb{E}_\xi\left[F_L(w_{L,K,0}([\xi]^{L,K,0}))\right] - F_L(w_{L,0,0})\\ &= \sum_{k=0}^{K-1}\mathbb{E}_\xi[F_L(w_{L,k+1,0}([\xi]^{L,k+1,0}))] - \mathbb{E}_\xi[F_L(w_{L,k,0}([\xi]^{L,k,0}))]\\ &\leq -\frac{1}{2}\frac{\mu}{\Theta}\overline{\alpha}\sum_{k=0}^{K-1}\mathcal{T}_{L,k}\mathbb{E}_\xi\left[\left\|\nabla F_L\left(w_{L,k,0}\left([\xi]^{L,k,0}\right)\right)\right\|^2\right].\end{aligned}$$

Thus, after rearrangement we have

$$\begin{aligned}\sum_{k=0}^{K-1}\mathcal{T}_{L,k}\mathbb{E}_\xi\left[\left\|\nabla F_L\left(w_{L,k,0}\left([\xi]^{L,k,0}\right)\right)\right\|^2\right] &\leq \frac{F_L(w_{L,0,0}) - F_{\inf}}{\frac{1}{2}\frac{\mu}{\Theta}\overline{\alpha}}\\ &= \frac{2\Theta(F_L(w_{L,0,0}) - F_{\inf})}{\mu\overline{\alpha}},\end{aligned}$$

which yields

$$\frac{1}{\sum_{k=0}^{K-1}\mathcal{T}_{L,k}}\sum_{k=0}^{K-1}\mathcal{T}_{L,k}\mathbb{E}_\xi\left[\left\|\nabla F_L\left(w_{L,k,0}\left([\xi]^{L,k,0}\right)\right)\right\|^2\right] \leq \frac{2\Theta(F_L(w_{L,0,0}) - F_{\inf})}{\mu\overline{\alpha}\sum_{k=0}^{K-1}\mathcal{T}_{L,k}}$$

after division by $\sum_{k=0}^{K-1}\mathcal{T}_{L,k}$. As before, we have

$$\frac{2\Theta(F_L(w_{L,0,0}) - F_{\inf})}{\sum_{k=0}^{K-1}\sum_{l=0}^{L}T_{l,k}\mu\overline{\alpha}} \quad \xrightarrow{K\to\infty} \quad 0.$$

This finishes the proof. □

As mentioned before, we showed that using variance reduction resulted in the constant term missing compared to the result in Theorem 3.10. This improves convergence significantly. At the same time, the updates are cheaper to perform, since it is not necessary to compute a full gradient at the lower level.

3.3.4 Convergence with stochastic gradient correction

Now, instead of using the gradient in the gradient correction which makes the use of the lower level at least as expensive as one full epoch on the higher level and thus computationally expensive, we use a stochastic direction. This means we are changing the level dependent objective function from our previous choice

$$\bar{H}_l(w_l) = F_l(w_l) + \langle P_{l+1}^* \dots P_L^* \nabla F_L(w_{L,k,0}) - \nabla_l F_l(w_{l,k,0}), w_l \rangle_l \tag{3.15}$$

(note that for further reference, we denote our old choice now by $\bar{H}_l$) to

$$H_l(w_l) = F_l(w_l) + \langle P_{l+1}^* \dots P_L^* g_L^F(w_{L,k,0}, \zeta_{L,k}) - \nabla_l F_l(w_{l,k,0}), w_l \rangle_l, \tag{3.16}$$

where we denote with g_l^H a stochastic direction for the objective function H_l that should fulfill Assumptions 3.3, 3.5 and 3.6 like the stochastic direction used for updating. Compared to Section 3.3.3, we compute the full gradient of the lower level at the lower level starting point. From a practical point of view, this is counterintuitive, as one would prefer to use a stochastic direction instead to reduce computational effort. We consider this case in Section 3.3.5 and use this section as preparation. The proofs in both cases are very similar.

Note that H_l, $\nabla_l H_l$ and thus also $g_l^H(w_{l,k,t}, \xi_{l,k,t})$ for $l = 0, \dots, L, k = 0, \dots, K_l, t = 0, \dots, T_{l,k}$ now depend on the random vector $\zeta_{l+1,k} \in \Omega$, which we have labeled separately from ξ to emphasize that these directions are only used for correction and not for updating. Further, assume that the ζ are independent from the ξ. As with ξ, by $[\zeta]^{l,k} = \{\zeta_{i,j}\}_{i \le l, j \le k}$ we denote the set of all $\zeta_{i,j}$ up to cycle l, k and with $[\zeta]_{i,j}^{l,k}$ that happened right after cycle (i, j) up to cycle (l, k). And with $\zeta = \{\zeta_{i,j}\}_{i,j}$ we denote the set of all $\zeta_{i,j}$.

In this case, we change Assumption 3.6b) to use the expected value of the step. To counteract the weakening of the angle condition, we also need to strengthen Assumption 3.6d).

3.6b′) There is $\mu > 0$ such that for all $l = 0, \dots, L$ and $k, t \in \mathbb{N}$ with $t \leq T_{l,k} + 1$ the iterates fulfill the angle condition

$$\begin{aligned} &- \langle \nabla F_L(w_{L,k,0}), \mathbb{E}_{[\zeta]^{l,k},[\xi]_{l,k,0}^{l,k,t}} \left[w_{l,k,t}^L - w_{L,k,0} \right] \rangle \\ &\geq \mu \left\| \nabla F_L(w_{L,k,0}) \right\| \left\| \mathbb{E}_{[\zeta]^{l,k},[\xi]_{l,k,0}^{l,k,t}} \left[w_{l,k,t}^L - w_{L,k,0} \right] \right\|. \end{aligned}$$

3.6d′) There is $\Theta \in \mathbb{R}$ such that for all $l = 0, \dots, L$ and $k \in \mathbb{N}$ we have

$$\left\| \mathbb{E}_{[\zeta]^{l,k},[\xi]_{l,k,0}^{l,k,T_{l,k}+1}} \left[w_{l,k,T_{l,k}+1}^L - w_{L,k,0} \right] \right\| \geq \frac{1}{\Theta} \sum_{i=0}^{l} \sum_{t=1}^{T_{l,k}} \alpha_{l-i,k,t} \left\| \nabla F_L(w_{L,k,0}) \right\|.$$

When $\alpha_{l,k,t} = \overline{\alpha}$ it follows from Assumption 3.6d′) that

$$- \left\| \nabla F_L(w_{L,k,0}) \right\| \left\| \mathbb{E}_{[\zeta]^{l,k},[\xi]_{l,k,0}^{l,k,T_{l,k}+1}} \left[w_{l,k,T_{l,k}+1}^L - w_{L,k,0} \right] \right\| \leq - \frac{1}{\Theta} \overline{\alpha} \mathcal{T}_{l,k} \left\| \nabla F_L(w_{L,k,0}) \right\|^2. \tag{3.17}$$

Note that as Lemma 3.7 uses Assumption 3.6b), we need to state a lemma that uses Assumption 3.6b′). For the proof of the theorem later on we will need Assumption 3.6d′).

Lemma 3.14. *Let Assumptions 3.3, 3.5 and 3.6 hold. Let the iterates $\{w_{l,k,t}\}_{l,k,t}$ with $l = 0, 1, \dots, L$, $k = 0, \dots K_l - 1$ and $t = 0, \dots, T_{l,k} + 1$ be generated by Algorithm 4 after a recursive call at $w_{L,k,0}$. Then*

$$\begin{aligned} &\mathbb{E}_{[\zeta]^{l,k},[\xi]_{l,k,0}^{l,k,T_{l,k}+1}} \left[F_L(w_{l,k,T_{l,k}+1}^L) - F_L(w_{L,k,0}) \right] \\ &\leq -\mu \left\| \nabla F_L(w_{L,k,0}) \right\| \left\| \mathbb{E}_{[\zeta]^{l,k},[\xi]_{l,k,0}^{l,k,T_{l,k}+1}} \left[w_{l,k,T_{l,k}+1}^L - w_{L,k,0} \right] \right\| \\ &\quad + \frac{1}{2} L_L \mathbb{E}_{[\zeta]^{l,k},[\xi]_{l,k,0}^{l,k,T_{l,k}+1}} \left[\left\| w_{l,k,T_{l,k}+1}^L - w_{L,k,0} \right\|^2 \right]. \end{aligned}$$

Proof. From Assumption 3.3 and (3.6) we know

$$\begin{aligned} &F_L(w_{l,k,T_{l,k}+1}^L) - F_L(w_{L,k,0}) \\ &\leq \langle \nabla F_L(w_{L,k,0}), w_{l,k,T_{l,k}+1}^L - w_{L,k,0} \rangle + \frac{1}{2} L_L \left\| w_{l,k,T_{l,k}+1}^L - w_{L,k,0} \right\|^2. \end{aligned}$$

Applying the expected value gives

$$\begin{aligned}
&\mathbb{E}_{[\zeta]^{l,k},[\xi]_{l,k,0}^{l,k,T_{l,k}+1}}\left[F_L(w_{l,k,T_{l,k}+1}^L) - F_L(w_{L,k,0})\right] \\
&\leq \langle \nabla F_L(w_{L,k,0}), \mathbb{E}_{[\zeta]^{l,k},[\xi]_{l,k,0}^{l,k,T_{l,k}+1}}\left[w_{l,k,T_{l,k}+1}^L - w_{L,k,0}\right]\rangle \\
&\quad + \frac{1}{2}L_L \mathbb{E}_{[\zeta]^{l,k},[\xi]_{l,k,0}^{l,k,T_{l,k}+1}}\left[\left\|w_{l,k,T_{l,k}+1}^L - w_{L,k,0}\right\|^2\right].
\end{aligned}$$

By Assumption 3.6b′), this leads to

$$\begin{aligned}
&\mathbb{E}_{[\zeta]^{l,k},[\xi]_{l,k,0}^{l,k,T_{l,k}+1}}\left[F_L(w_{l,k,T_{l,k}+1}^L) - F_L(w_{L,k,0})\right] \\
&\leq -\mu \left\|\nabla F_L(w_{L,k,0})\right\| \left\|\mathbb{E}_{[\zeta]^{l,k},[\xi]_{l,k,0}^{l,k,T_{l,k}+1}}\left[w_{l,k,T_{l,k}+1}^L - w_{L,k,0}\right]\right\| \\
&\quad + \frac{1}{2}L_L \mathbb{E}_{[\zeta]^{l,k},[\xi]_{l,k,0}^{l,k,T_{l,k}+1}}\left[\left\|w_{l,k,T_{l,k}+1}^L - w_{L,k,0}\right\|^2\right]. \qquad \square
\end{aligned}$$

Also in this case, we get a different bound on the variance of the multilevel steps compared to Lemma 3.8.

Lemma 3.15. *Let Assumptions 3.3, 3.5 and 3.6 hold. Let the iterates $\{w_{l,k,t}\}_{l,k,t}$ with $l = 0, 1, \ldots, L$, $k = 0, \ldots K_l - 1$ and $t = 0, \ldots, T_{l,k} + 1$ be generated by Algorithm 4 after a recursive call at $w_{L,k,0}$ using a stochastic gradient correction. Assume that the step size is sufficiently small, i.e., fulfills*

$$\left(\sum_{i=0}^{l}\sum_{t=1}^{T_{i,k}+1}\alpha_{i,k,t}\right)^2 \leq \frac{\nu^2}{8\overline{\rho}\overline{\mathcal{M}}\mathcal{L}}.$$

Then for all $l = 0, \ldots, L$ and $k \in \mathbb{N}$, we have

$$\begin{aligned}
&\mathbb{E}_{[\zeta]^{l,k},[\xi]_{l,k,0}^{l,k,T_{l,k}+1}}\left[\left\|w_{l,k,T_{l,k}+1}^L - w_{L,k,0}\right\|^2\right] \\
&\leq \left(\sum_{i=0}^{l}\sum_{t=1}^{T_{i,k}+1}\alpha_{i,k,t}\right)^2 \left(c + C\left\|\nabla F_L(w_{L,k,0})\right\|^2\right)
\end{aligned}$$

with $c \coloneqq 4\overline{\rho}(\mathcal{M} + \overline{\rho}M_L^F)$ and $C \coloneqq 4\overline{\rho}^2(2\overline{\mathcal{M}} + \overline{M}_L^F + 1)$. If the step size is constant $\overline{\alpha}$, then

$$\mathbb{E}_{[\xi]_{l,k,0}^{l,k+1,0}}\left[\left\|w_{l,k,T_{l,k}+1}^L - w_{L,k,0}\right\|^2\right] \leq c\overline{\alpha}^2\mathcal{T}_{l,k}^2 + C\overline{\alpha}^2\mathcal{T}_{l,k}^2\left\|\nabla F_L(w_{L,k,0})\right\|^2. \tag{3.18}$$

Proof. We define $\mathcal{A}_k := \sum_{i=0}^{l} \sum_{t=1}^{T_{i,k}+1} \alpha_{i,k,t}$. As in the proof of Lemma 3.8, using Jensen's inequality we arrive at

$$\begin{aligned}
&\mathbb{E}_{[\zeta]^{L,k},[\xi]_{l,k,0}^{l,k,T_{l,k}+1}} \left[\left\| w_{l,k,T_{l,k}+1}^{L} - w_{L,k,0} \right\|^2 \right] \\
&\leq \mathcal{A}_k \sum_{i=0}^{l} \sum_{t=1}^{T_{i,k}+1} \alpha_{i,k,t} \rho^{2(L-i)} \mathbb{E}_{[\zeta]^{L,k},[\xi]_{l,k,0}^{l,k,T_{l,k}+1}} \left[\left\| g_i^H(w_{i,k,t}, \xi_{i,k,t}) \right\|_i^2 \right]. \qquad (3.19)
\end{aligned}$$

Now the stochastic gradients of $\bar{H}_l$ with deterministic gradient correction (3.15) and our current choice H_l by (3.15) are related by

$$g_i^H(w_{i,k,t}, \xi_{i,k,t}) = g_i^{\bar{H}}(w_{i,k,t}, \xi_{i,k,t}) + P_{i+1}^* \dots P_L^*(g_L^F(w_{L,k,0}, \zeta_{L,k}) - \nabla F_L(w_{L,k,0})).$$

Hence,

$$\begin{aligned}
\left\| g_i^H(w_{i,k,t}, \xi_{i,k,t}) \right\|_i^2 &\leq 2 \left\| g_i^{\bar{H}}(w_{i,k,t}, \xi_{i,k,t}) \right\|_i^2 \\
&\quad + 2\rho^{2(L-i)} \left\| g_L^F(w_{L,k,0}, \zeta_{L,k}) - \nabla F_L(w_{L,k,0}) \right\|^2
\end{aligned}$$

and inserting in (3.19) yields

$$\begin{aligned}
&\mathbb{E}_{[\zeta]^{L,k},[\xi]_{l,k,0}^{l,k,T_{l,k}+1}} \left[\left\| w_{l,k,T_{l,k}+1}^{L} - w_{L,k,0} \right\|^2 \right] \\
&\leq \mathcal{A}_k \sum_{i=0}^{l} \sum_{t=1}^{T_{i,k}+1} \alpha_{i,k,t} \rho^{2(L-i)} 2\, \mathbb{E}_{[\xi]_{l,k,0}^{l,k,T_{l,k}+1}} \left[\left\| g_i^{\bar{H}}(w_{i,k,t}, \xi_{i,k,t}) \right\|_i^2 \right] \qquad (3.20) \\
&+ \mathcal{A}_k \sum_{i=0}^{l} \sum_{t=1}^{T_{i,k}+1} \alpha_{i,k,t} \rho^{4(L-i)} 2\, \mathbb{E}_{\zeta_{L,k}} \left[\left\| g_L^F(w_{L,k,0}, \zeta_{L,k}) - \nabla F_L(w_{L,k,0}) \right\|^2 \right].
\end{aligned}$$

The first sum in (3.20) coincides up to a factor 2 with (3.10) in Lemma 3.8 and can be estimated exactly as in the proof of Lemma 3.8 (note that $g_i^{\bar{H}}(w_{i,k,t}, \xi_{i,k,t})$ is denoted there by $g_i^H(w_{i,k,t}, \xi_{i,k,t})$). To estimate the second sum in (3.20), we note that with Assumption 3.5

$$\begin{aligned}
&\mathbb{E}_{\zeta_{L,k}} \left[\left\| g_L^F(w_{L,k,0}, \zeta_{L,k}) - \nabla F_L(w_{L,k,0}) \right\|^2 \right] \\
&\leq \mathbb{E}_{\zeta_{L,k}} \left[\left\| g_L^F(w_{L,k,0}, \zeta_{L,k}) \right\|^2 \right] + \left\| F_L(w_{L,k,0}) \right\|^2 \\
&\leq \left(M_L^F + (\overline{M}_L^F + 1) \left\| \nabla F_L(w_{L,k,0}) \right\|^2 \right).
\end{aligned}$$

Hence, estimating the first sum in (3.20) as in Lemma 3.8 and the second as above, we arrive under the assumption (note that we have to deal with the additional factor 2)

$$\left(\sum_{i=0}^{l}\sum_{t=1}^{T_{i,k}+1}\alpha_{i,k,t}\right)^2 \leq \frac{\nu^2}{8\overline{\rho}\overline{\mathcal{M}}\mathcal{L}}.$$

as in the proof of Lemma 3.8 at

$$\begin{aligned}
&\mathbb{E}_{[\zeta]^{L,k},[\xi]_{l,k,0}^{l,k,T_{l,k}+1}}\left[\left\|w_{l,k,T_{l,k}+1}^{L}-w_{L,k,0}\right\|^2\right]\\
&\leq \mathcal{A}_k^2 4\overline{\rho}\mathcal{M}+\mathcal{A}_k^2 8\overline{\rho}^2\overline{\mathcal{M}}\left\|\nabla F_L(w_{L,k,0})\right\|^2\\
&+\mathcal{A}_k^2 4\overline{\rho}^2 M_L^F+\mathcal{A}_k^2 4\overline{\rho}^2(\overline{M}_L^F+1)\left\|\nabla F_L(w_{L,k,0})\right\|^2\\
&=\mathcal{A}_k^2 4\overline{\rho}(\mathcal{M}+\overline{\rho}M_L^F)+\mathcal{A}_k^2 4\overline{\rho}^2(2\overline{\mathcal{M}}+\overline{M}_L^F+1)\left\|\nabla F_L(w_{L,k,0})\right\|^2.
\end{aligned}$$

Therefore, defining $c \coloneqq 4\overline{\rho}(\mathcal{M}+\overline{\rho}M_L^F)$ and $C \coloneqq 4\overline{\rho}^2(2\overline{\mathcal{M}}+\overline{M}_L^F+1)$ the first statement follows. The second statement follows from

$$\overline{\alpha}\mathcal{T}_{l,k}=\sum_{i=0}^{l}\sum_{t=1}^{T_{i,k}+1}\alpha_{i,k,t}$$

with $\alpha_{i,k,t}=\overline{\alpha}$. □

When using a stochastic gradient correction, an approach as in Lemma 3.9 yields a similar bound compared to Lemma 3.15 as the additional dependence on the variance regarding the stochastic gradient correction gives a constant term similar to $\overline{\alpha}^2\mathcal{T}_{L,k}^2 c$.

The proof of convergence in this case works similar to the one in Theorem 3.10 in the deterministic case but it utilizes the estimate of the variance of the multilevel steps from Lemma 3.15 instead of the bound from Lemma 3.8. Furthermore, we also need to take the expected value with respect to ζ.

Theorem 3.16 (Convergence rate for MLSGD with stochastic gradient correction)**.** *Let Assumptions 3.3, 3.5 and 3.6 hold. Suppose that the MLSGD Algorithm 4 is run with a fixed step size $\overline{\alpha}$. Assume the step size is sufficiently small, i.e., fulfills*

$$\overline{\alpha}\mathcal{T}_{L,k} \leq \min\left\{\frac{\mu}{4\Theta L_L\overline{\rho}^2(2\overline{\mathcal{M}}+\overline{M}_L^F+1)}, \frac{\nu}{2\sqrt{2\overline{\rho}\overline{\mathcal{M}}\mathcal{L}}}\right\}.$$

Then the following inequality holds

$$\frac{1}{\sum_{k=0}^{K-1}\mathcal{T}_{L,k}}\sum_{k=0}^{K-1}\mathcal{T}_{L,k}\mathbb{E}_{\zeta,\xi}\left[\left\|\nabla F_L\left(w_{L,k,0}\left([\xi]^{L,k,0},[\zeta]^{L,k}\right)\right)\right\|^2\right]$$
$$\leq \frac{4\Theta L_L\overline{\rho}(\mathcal{M}+\overline{\rho}M_L^F)\overline{\alpha}\sum_{k=0}^{K-1}\mathcal{T}_{L,k}^2}{\mu\sum_{k=0}^{K-1}\mathcal{T}_{L,k}}+\frac{2\Theta(F_L(w_{L,0,0})-F_{\inf})}{\mu\overline{\alpha}\sum_{k=0}^{K-1}\mathcal{T}_{L,k}}$$

for all $K\in\mathbb{N}$. The last term converges to zero as K approaches infinity.

Proof. We use Lemma 3.14 instead of Lemma 3.7 and get

$$\mathbb{E}_{[\zeta]^{L,k},[\xi]_{L,k,0}^{L,k,T_{L,k+1}}}\left[F_L(w_{l,k,T_{l,k}+1}^L)-F_L(w_{L,k,0})\right]$$
$$\leq -\mu\left\|\nabla F_L(w_{L,k,0})\right\|\left\|\mathbb{E}_{[\zeta]^{L,k},[\xi]_{L,k,0}^{L,k,T_{L,k+1}}}\left[w_{l,k,T_{l,k}+1}^L-w_{L,k,0}\right]\right\|$$
$$+\frac{1}{2}L_L\mathbb{E}_{[\zeta]^{L,k},[\xi]_{L,k,0}^{L,k,T_{L,k+1}}}\left[\left\|w_{l,k,T_{l,k}+1}^L-w_{L,k,0}\right\|^2\right].$$

Using now Assumption 3.6d′) gives

$$\mathbb{E}_{[\zeta]^{L,k},[\xi]_{L,k,0}^{L,k,T_{L,k+1}}}\left[F_L(w_{l,k,T_{l,k}+1}^L)-F_L(w_{L,k,0})\right]$$
$$\leq -\frac{\mu}{\Theta}\overline{\alpha}\mathcal{T}_{L,k}\left\|\nabla F_L(w_{L,k,0})\right\|^2$$
$$+\frac{1}{2}L_L\mathbb{E}_{[\zeta]^{L,k},[\xi]_{L,k,0}^{L,k,T_{L,k+1}}}\left[\left\|w_{l,k,T_{l,k}+1}^L-w_{L,k,0}\right\|^2\right].$$

The second bound on $\overline{\alpha}$ together with Lemma 3.15 yields

$$\mathbb{E}_{[\zeta]^{L,k},[\xi]_{L,k,0}^{L,k,T_{L,k+1}}}\left[F_L(w_{L,k+1,0})-F_L(w_{L,k,0})\right]$$
$$\leq -\frac{\mu}{\Theta}\overline{\alpha}\mathcal{T}_{L,k}\left\|\nabla F_L(w_{L,k,0})\right\|^2$$
$$+\frac{1}{2}L_L\left(c\overline{\alpha}^2\mathcal{T}_{L,k}^2+C\overline{\alpha}^2\mathcal{T}_{L,k}^2\right)\left\|\nabla F_L(w_{L,k,0})\right\|^2$$
$$=\left(-\frac{\mu}{\Theta}+\frac{1}{2}L_LC\overline{\alpha}\mathcal{T}_{L,k}\right)\overline{\alpha}\mathcal{T}_{L,k}\left\|\nabla F_L(w_{L,k,0})\right\|^2+\frac{1}{2}L_Lc\overline{\alpha}^2\mathcal{T}_{L,k}^2.$$

Now we can use the first bound on $\overline{\alpha}$, that

$$\overline{\alpha}\mathcal{T}_{L,k}\leq\frac{\mu}{\Theta L_LC}.$$

Therefore,

$$\begin{aligned}&\mathbb{E}_{[\zeta]^{L,k},[\xi]_{L,k,0}^{L,k,T_{L,k+1}}}\left[F_L(w_{L,k+1,0}) - F_L(w_{L,k,0})\right]\\&\leq -\frac{1}{2}\frac{\mu}{\Theta}\mathcal{T}_{L,k}\overline{\alpha}\left\|\nabla F_L(w_{L,k,0})\right\|^2 + \frac{1}{2}L_L c\overline{\alpha}^2\mathcal{T}_{L,k}^2.\end{aligned}$$

After taking the total expectation with respect to all ξ and ζ, we get

$$\begin{aligned}&\mathbb{E}_{\zeta,\xi}\left[F_L(w_{L,k+1,0}([\xi]^{L,k+1,0},[\zeta]^{L,k+1})) - F_L(w_{L,k,0}([\xi]^{L,k,0},[\zeta]^{L,k}))\right]\\&\leq -\frac{1}{2}\frac{\mu}{\Theta}\mathcal{T}_{L,k}\overline{\alpha}\mathbb{E}_{\xi}\left[\left\|\nabla F_L\left(w_{L,k,0}\left([\xi]^{L,k,0},[\zeta]^{L,k}\right)\right)\right\|^2\right] + \frac{1}{2}L_L c\overline{\alpha}^2\mathcal{T}_{L,k}^2.\end{aligned}$$

From this, it follows for $K \in \mathbb{N}$ that

$$\begin{aligned}&F_{\inf} - F_L(w_{L,0,0})\\&\leq \mathbb{E}_{\zeta,\xi}\left[F_L(w_{L,K,0}([\xi]^{L,K,0},[\zeta]^{L,k}))\right] - F(w_{L,0,0})\\&= \sum_{k=0}^{K-1}\mathbb{E}_{\zeta,\xi}[F_L(w_{L,k+1,0}([\xi]^{L,k+1,0},[\zeta]^{L,k+1}))] - \mathbb{E}_{\zeta,\xi}[F_L(w_{L,k,0}([\xi]^{L,k,0},[\zeta]^{L,k}))]\\&\leq -\frac{1}{2}\frac{\mu}{\Theta}\overline{\alpha}\sum_{k=0}^{K-1}\mathcal{T}_{L,k}\mathbb{E}_{\xi}\left[\left\|\nabla F_L\left(w_{L,k,0}\left([\xi]^{L,k,0},[\zeta]^{L,k}\right)\right)\right\|^2\right]\\&\quad + \frac{1}{2}L_L c\overline{\alpha}^2\sum_{k=0}^{K-1}\mathcal{T}_{L,k}^2.\end{aligned}$$

Rearranging the terms yields

$$\begin{aligned}&\sum_{k=0}^{K-1}\mathcal{T}_{L,k}\mathbb{E}_{\zeta,\xi}\left[\left\|\nabla F_L\left(w_{L,k,0}\left([\xi]^{L,k,0},[\zeta]^{L,k}\right)\right)\right\|^2\right]\\&\leq \frac{\frac{1}{2}L_L c\overline{\alpha}^2\sum_{k=0}^{K-1}\mathcal{T}_{L,k}^2}{\frac{1}{2}\frac{\mu}{\Theta}\overline{\alpha}} + \frac{F_L(w_{L,0,0}) - F_{\inf}}{\frac{1}{2}\frac{\mu}{\Theta}\overline{\alpha}}\\&= \frac{\Theta L_L c\overline{\alpha}\sum_{k=0}^{K-1}\mathcal{T}_{L,k}^2}{\mu} + \frac{2\Theta(F_L(w_{L,0,0}) - F_{\inf})}{\mu\overline{\alpha}}.\end{aligned}$$

Therefore, by dividing by $\sum_{k=0}^{K-1} \mathcal{T}_{L,k}$ and inserting the definition of c, we have

$$\begin{aligned}
&\frac{1}{\sum_{k=0}^{K-1} \mathcal{T}_{L,k}} \sum_{k=0}^{K-1} \mathcal{T}_{L,k} \mathbb{E}_{\zeta,\xi} \left[\left\| \nabla F_L \left(w_{L,k,0} \left([\xi]^{L,k,0}, [\zeta]^{L,k} \right) \right) \right\|^2 \right] \\
&\leq \frac{\Theta L_L c \overline{\alpha} \sum_{k=0}^{K-1} \mathcal{T}_{L,k}^2}{\mu \sum_{k=0}^{K-1} \mathcal{T}_{L,k}} + \frac{2\Theta(F_L(w_{L,0,0}) - F_{\inf})}{\mu \overline{\alpha} \sum_{k=0}^{K-1} \mathcal{T}_{L,k}} \\
&= \frac{4\Theta L_L \overline{\rho}(\mathcal{M} + \overline{\rho} M_L^F) \overline{\alpha} \sum_{k=0}^{K-1} \mathcal{T}_{L,k}^2}{\mu \sum_{k=0}^{K-1} \mathcal{T}_{L,k}} + \frac{2\Theta(F_L(w_{L,0,0}) - F_{\inf})}{\mu \overline{\alpha} \sum_{k=0}^{K-1} \mathcal{T}_{L,k}}.
\end{aligned}$$

For the same reasons as before, it holds

$$\frac{2\Theta(F_L(w_{L,0,0}) - F_{\inf})}{\sum_{k=0}^{K-1} \sum_{l=0}^{L} T_{l,k} \mu \overline{\alpha}} \quad \xrightarrow{K \to \infty} \quad 0.$$

This finishes the proof. □

In comparison to Theorem 3.10, in the stochastic case the remaining constant now also depends on the variance of the stochastic directions that are used to compute the gradient correction. This is to be expected, since when using a stochastic direction that is not representative of the gradient, the lower level can be mislead and not yield helpful directions to the upper level.

3.3.5 Inserting a variance reduction term in the stochastic setting

After using stochastic directions instead of the higher level gradient in the gradient correction in the previous section, the next natural step is to similarly replace the lower level gradient. One way to achieve this is by combining the stochastic gradient correction case as in Section 3.3.4 with a variance reduction as established in Section 3.3.3. This is expected to reduce the time needed to compute an update compared to both the stochastic gradient correction and the variance reduction version, since only stochastic directions are used in the gradient correction, thereby reducing the need for expensive full gradient computations at either the higher or lower level.

Remember that the objective function with stochastic gradient correction is

$$H_l(w_l) = F_l(w_l) + \langle P_{l+1}^* \dots P_L^* g_L^F(w_{L,k,0}, \zeta_{L,k}) - \nabla_l F_l(w_{l,k,0}), w_l \rangle_l.$$

Therefore, applying variance reduction to the objective function F results in the update

$$\begin{aligned}
&\tilde{g}_l^H(w_{l,k,t}, \xi_{l,k,t}, \zeta_{l,k}) \\
&= g_l^F(w_{l,k,t}, \xi_{l,k,t}) + (P_{l+1}^* \dots P_L^* g_L^F(w_{L,k,0}, \zeta_{L,k}) - \nabla_l F_l(w_{l,k,0})) \\
&\quad + (\nabla_l F_l(w_{l,k,0}) - g_l^F(w_{l,k,0}, \xi_{l,k,t})) \\
&= g_l^F(w_{l,k,t}, \xi_{l,k,t}) + (P_{l+1}^* \dots P_L^* g_L^F(w_{L,k,0}, \zeta_{L,k}) - g_l^F(w_{l,k,0}, \xi_{l,k,t})).
\end{aligned} \tag{3.21}$$

Again, we give a bound on the variance of the multilevel steps.

Lemma 3.17. *Let Assumptions 3.3 to 3.6 hold. Let the iterates $\{w_{l,k,t}\}_{l,k,t}$ with $l = 0, 1, \dots, L$, $k = 0, \dots K_l - 1$ and $t = 0, \dots, T_{l,k} + 1$ be generated by Algorithm 4 after a recursive call at $w_{L,k,0}$ using stochastic gradient correction and variance reduction. Assume that the step size is sufficiently small, i.e., fulfills*

$$\left(\sum_{i=0}^{l} \sum_{t=1}^{T_{i,k}+1} \alpha_{i,k,t} \right)^2 \leq \frac{\nu^2}{4\overline{\rho}\mathcal{L}}.$$

Then for all $l = 0, \dots, L$ and $k \in \mathbb{N}$, we have

$$\begin{aligned}
&\mathbb{E}_{[\zeta]^{l,k}, [\xi]_{l,k,0}^{l,k,T_{l,k}+1}} \left[\left\| w_{l,k,T_{l,k}+1}^L - w_{L,k,0} \right\|^2 \right] \\
&\leq \left(\sum_{i=0}^{l} \sum_{t=1}^{T_{i,k}+1} \alpha_{i,k,t} \right)^2 \left(d + D \left\| \nabla F_L(w_{L,k,0}) \right\|^2 \right),
\end{aligned}$$

with $d \coloneqq 4\overline{\rho}^2 M_L^F$ and $D \coloneqq 4\overline{\rho}^2 \overline{M}_L^F$. If the step size is constant $\overline{\alpha}$, then

$$\mathbb{E}_{[\xi]_{l,k,0}^{l,k+1,0}} \left[\left\| w_{l,k,T_{l,k}+1}^L - w_{L,k,0} \right\|^2 \right] \leq d\overline{\alpha}^2 \mathcal{T}_{l,k}^2 + D\overline{\alpha}^2 \mathcal{T}_{l,k}^2 \left\| \nabla F_L(w_{L,k,0}) \right\|^2. \tag{3.22}$$

Proof. We define $\mathcal{A}_k \coloneqq \sum_{i=0}^{l} \sum_{t=1}^{T_{i,k}+1} \alpha_{i,k,t}$. As in the proof of Lemma 3.12, using Jensen's inequality we arrive at

$$\begin{aligned}
&\mathbb{E}_{[\zeta]^{l,k}, [\xi]_{l,k,0}^{l,k,T_{l,k}+1}} \left[\left\| w_{l,k,T_{l,k}+1}^L - w_{L,k,0} \right\|^2 \right] \\
&\leq \mathcal{A}_k \sum_{i=0}^{l} \sum_{t=1}^{T_{i,k}+1} \alpha_{i,k,t} \rho^{2(L-i)} \mathbb{E}_{[\zeta]^{l,k}, [\xi]_{l,k,0}^{l,k,T_{l,k}+1}} \left[\left\| \tilde{g}_i^H(w_{i,k,t}, \xi_{i,k,t}) \right\|_i^2 \right].
\end{aligned} \tag{3.23}$$

For $\left\|\tilde{g}_i^H(w_{i,k,t}, \xi_{i,k,t})\right\|_i^2$ we have, analogously to the proof of Lemma 3.12 but with $g_L^F(w_{L,k,0}, \zeta_{L,k})$ instead of $\nabla F_L(w_{L,k,0})$, that

$$\begin{aligned}
&\left\|\tilde{g}_i^H(w_{i,k,t}, \xi_{i,k,t})\right\|_i^2 \\
&= \left\|P_{i+1}^* \dots P_L^* g_L^F(w_{L,k,0}, \zeta_{L,k}) + g_i^F(w_{i,k,t}, \xi_{i,k,t}) - g_i^F(w_{i,k,0}, \xi_{i,k,t})\right\|_i^2 \\
&\leq 2\rho^{2(L-i)} \left\|g_L^F(w_{L,k,0}, \zeta_{L,k})\right\|^2 + \frac{2L_i}{\nu^2} \left\|w_{l,k,T_{l,k}+1}^L - w_{l,k,0}\right\|^2 .
\end{aligned}$$

Now, as in the proof of Lemma 3.15, we also have to account for variance in the computation of $g_L^F(w_{L,k,0}, \zeta_{L,k})$. In the same manner, we start by considering the expected value with respect to ζ of $\left\|\tilde{g}_i^H(w_{i,k,t}, \xi_{i,k,t})\right\|_i^2$. With Assumption 3.5, we have

$$\begin{aligned}
\mathbb{E}_{\zeta_{L,k}}\left[\left\|\tilde{g}_i^H(w_{i,k,t}, \xi_{i,k,t})\right\|_i^2\right] \leq 2\rho^{2(L-i)} \left(M_L^F + \overline{M}_L^F \left\|\nabla F_L(w_{L,k,0})\right\|^2\right) \\
+ \frac{2L_i}{\nu^2} \mathbb{E}_{\zeta_{L,k}}\left[\left\|w_{l,k,T_{l,k}+1}^L - w_{l,k,0}\right\|^2\right].
\end{aligned}$$

As the first term on the right hand side is now independent of $[\xi]_{l,k,0}^{l,k,T_{l,k}+1}$, going back to (3.23), this yields

$$\begin{aligned}
&\mathbb{E}_{[\zeta]^{l,k},[\xi]_{l,k,0}^{l,k,T_{l,k}+1}}\left[\left\|w_{l,k,T_{l,k}+1}^L - w_{L,k,0}\right\|^2\right] \\
&\leq \mathcal{A}_k \sum_{i=0}^{l} \sum_{t=1}^{T_{i,k}+1} \alpha_{i,k,t} \rho^{2(L-i)} \left(2\rho^{2(L-i)} \left(M_L^F + \overline{M}_L^F \left\|\nabla F_L(w_{L,k,0})\right\|^2\right)\right. \\
&\quad \left. + \frac{2L_i}{\nu^2} \mathbb{E}_{[\zeta]^{l,k},[\xi]_{l,k,0}^{l,k,T_{l,k}+1}}\left[\left\|w_{l,k,T_{l,k}+1}^L - w_{l,k,0}\right\|^2\right]\right),
\end{aligned}$$

which implies

$$\begin{aligned}
&\mathbb{E}_{[\zeta]^{l,k},[\xi]_{l,k,0}^{l,k,T_{l,k}+1}}\left[\left\|w_{l,k,T_{l,k}+1}^L - w_{L,k,0}\right\|^2\right] \\
&\leq \mathcal{A}_k \sum_{i=0}^{l} \sum_{t=1}^{T_{i,k}+1} \alpha_{i,k,t} 2\rho^{4(L-i)} M_L^F \\
&\quad + \mathcal{A}_k \sum_{i=0}^{l} \sum_{t=1}^{T_{i,k}+1} \alpha_{i,k,t} 2\rho^{4(L-i)} \overline{M}_L^F \left\|\nabla F_L(w_{L,k,0})\right\|^2 \\
&\quad + \mathcal{A}_k \sum_{i=0}^{l} \sum_{t=1}^{T_{i,k}+1} \alpha_{i,k,t} \rho^{2(L-i)} \frac{2L_i}{\nu^2} \mathbb{E}_{[\zeta]^{l,k},[\xi]_{l,k,0}^{l,k,T_{l,k}+1}}\left[\left\|w_{l,k,T_{l,k}+1}^L - w_{l,k,0}\right\|^2\right].
\end{aligned}$$

With the bounds $\mathcal{L}$ and $\overline{\rho}$, we get

$$\begin{aligned}
&\mathbb{E}_{[\zeta]^{l,k},[\xi]_{l,k,0}^{l,k,T_{l,k}+1}}\left[\left\|w_{l,k,T_{l,k}+1}^{L}-w_{L,k,0}\right\|^2\right]\\
&\leq \mathcal{A}_k^2 2\overline{\rho}^2 M_L^F + \mathcal{A}_k^2 2\overline{\rho}^2 \overline{M}_L^F \left\|\nabla F_L(w_{L,k,0})\right\|^2\\
&\quad + \mathcal{A}_k^2 \overline{\rho}\frac{2\mathcal{L}}{\nu^2}\mathbb{E}_{[\zeta]^{l,k},[\xi]_{l,k,0}^{l,k,T_{l,k}+1}}\left[\left\|w_{l,k,T_{l,k}+1}^{L}-w_{l,k,0}\right\|^2\right].
\end{aligned}$$

We use the bound $\left(\sum_{i=0}^{l}\sum_{t=1}^{T_{l,k}}\alpha_{l-i,k,t}\right)^2 \leq \frac{\nu^2}{4\overline{\rho}\mathcal{L}}$, therefore, $\mathcal{A}_k^2\overline{\rho}\frac{2\mathcal{L}}{\nu^2}\leq\frac{1}{2}$. Hence,

$$\frac{1}{2}\mathbb{E}_{[\zeta]^{l,k},[\xi]_{l,k,0}^{l,k,T_{l,k}+1}}\left[\left\|w_{l,k,T_{l,k}+1}^{L}-w_{L,k,0}\right\|^2\right] \leq \mathcal{A}_k^2 2\overline{\rho}^2 M_L^F + \mathcal{A}_k^2 2\overline{\rho}^2 \overline{M}_L^F \left\|\nabla F_L(w_{L,k,0})\right\|^2,$$

which is

$$\begin{aligned}
&\mathbb{E}_{[\xi]_{l,k,0}^{l,k,T_{l,k}+1}}\left[\left\|w_{l,k,T_{l,k}+1}^{L}-w_{L,k,0}\right\|^2\right]\\
&\leq \mathcal{A}_k^2 4\overline{\rho}^2 M_L^F + \mathcal{A}_k^2 4\overline{\rho}^2 \overline{M}_L^F \left\|\nabla F_L(w_{L,k,0})\right\|^2.
\end{aligned}$$

Defining $d \coloneqq 4\overline{\rho}^2 M_L^F$ and $D \coloneqq 4\overline{\rho}^2\overline{M}_L^F$ the first statement follows. As before, the second statement follows from

$$\overline{\alpha}\mathcal{T}_{l,k} = \sum_{i=0}^{l}\sum_{t=1}^{T_{i,k}+1}\alpha_{i,k,t}$$

with $\alpha_{i,k,t}=\overline{\alpha}$. □

Note that in this case, despite the variance reduction, the result still depends on the variance bounds M_l^F and $\overline{M}_l^F$ of the stochastic directions for F.

Theorem 3.18 (Convergence rate for MLSGD with variance reduction and stochastic gradient correction)**.** *Let Assumptions 3.3 to 3.6 hold and let the SGD updates be computed like in* (3.21)*. Suppose that the MLSGD Algorithm 4 is run with a fixed step size $\overline{\alpha}$. Further, assume the step size is sufficiently small, i.e., fulfills*

$$\overline{\alpha}\mathcal{T}_{L,k} \leq \min\left\{\frac{\mu}{4\Theta L_L\overline{\rho}^2\overline{M}_L^F}, \frac{\nu}{2\sqrt{\overline{\rho}\mathcal{L}}}\right\}.$$

Then the following inequality holds

$$\frac{1}{\sum_{k=0}^{K-1}\mathcal{T}_{L,k}}\sum_{k=0}^{K-1}\mathcal{T}_{L,k}\mathbb{E}_{\zeta,\xi}\left[\left\|\nabla F_L\left(w_{L,k,0}\left([\xi]^{L,k,0},[\zeta]^{L,k}\right)\right)\right\|^2\right]$$
$$\leq \frac{4\Theta L_L\overline{\rho}^2 M_L^F\overline{\alpha}\sum_{k=0}^{K-1}\mathcal{T}_{L,k}^2}{\mu\sum_{k=0}^{K-1}\mathcal{T}_{L,k}}+\frac{2\Theta(F_L(w_{L,0,0})-F_{\inf})}{\mu\overline{\alpha}\sum_{k=0}^{K-1}\mathcal{T}_{L,k}}$$

for all $K\in\mathbb{N}$. As K approaches infinity, the last term approaches zero.

Proof. The proof is analogous to the proof of Theorem 3.16, except for using the bound in Lemma 3.17 and thus replacing

$$c \text{ with } d$$

and

$$C \text{ with } D.$$

Following the arguments in the proof of Theorem 3.16, we arrive at

$$\frac{1}{\sum_{k=0}^{K-1}\mathcal{T}_{L,k}}\sum_{k=0}^{K-1}\mathcal{T}_{L,k}\mathbb{E}_{\zeta,\xi}\left[\left\|\nabla F_L\left(w_{L,k,0}\left([\xi]^{L,k,0},[\zeta]^{L,k}\right)\right)\right\|^2\right]$$
$$\leq \frac{\Theta L_L d\overline{\alpha}\sum_{k=0}^{K-1}\mathcal{T}_{L,k}^2}{\mu\sum_{k=0}^{K-1}\mathcal{T}_{L,k}}+\frac{2\Theta(F_L(w_{L,0,0})-F_{\inf})}{\mu\overline{\alpha}\sum_{k=0}^{K-1}\mathcal{T}_{L,k}}$$
$$= \frac{\Theta L_L 4\overline{\rho}^2 M_L^F\overline{\alpha}\sum_{k=0}^{K-1}\mathcal{T}_{L,k}^2}{\mu\sum_{k=0}^{K-1}\mathcal{T}_{L,k}}+\frac{2\Theta(F_L(w_{L,0,0})-F_{\inf})}{\mu\overline{\alpha}\sum_{k=0}^{K-1}\mathcal{T}_{L,k}},$$

where

$$\frac{2\Theta(F_L(w_{L,0,0})-F_{\inf})}{\sum_{k=0}^{K-1}\sum_{l=0}^{L}T_{l,k}\mu\overline{\alpha}} \quad \xrightarrow{K\to\infty} \quad 0. \qquad \square$$

Similar to Theorem 3.13, using variance reduction eliminates the dependence on the variance bounds $\mathcal{M}$ and $\overline{\mathcal{M}}$ of the stochastic directions.

For a better overview, all results on the bound of the expected value of the mean of the norm of the gradients are displayed in Table 3.1.

When the variance of the stochastic directions is high, the most promising approach is to use a deterministic gradient correction combined with variance reduction. In this case, the constant term vanishes. However, the downside is the higher computational cost. This is due to the need to compute the full gradient for the deterministic gradient

Grad. corr.	Var. red.	Bound
Deterministic	No	$\frac{2\Theta L_L \overline{\rho}\mathcal{M}\overline{\alpha}\sum_{k=0}^{K-1}\mathcal{T}_{L,k}^2}{\mu\sum_{k=0}^{K-1}\mathcal{T}_{L,k}} + \frac{2\Theta(F_L(w_{L,0,0})-F_{\inf})}{\mu\overline{\alpha}\sum_{k=0}^{K-1}\mathcal{T}_{L,k}}$
	with Lem. 3.9	$\frac{2\Theta L_L \overline{\rho}\mathcal{M}\overline{\alpha}\sum_{k=0}^{K-1}((1+\overline{\alpha}L_w\mathcal{L}\mathcal{T}_{L,k})\mathcal{T}_{L,k})}{\mu\sum_{k=0}^{K-1}\mathcal{T}_{L,k}} + \frac{2\Theta(F_L(w_{L,0,0})-F_{\inf})}{\mu\overline{\alpha}\sum_{k=0}^{K-1}\mathcal{T}_{L,k}}$
	Yes	$\frac{2\Theta(F_L(w_{L,0,0})-F_{\inf})}{\mu\overline{\alpha}\sum_{k=0}^{K-1}\mathcal{T}_{L,k}}$
Stochastic	No	$\frac{4\Theta L_L \overline{\rho}(\mathcal{M}+\overline{\rho}M_L^F)\overline{\alpha}\sum_{k=0}^{K-1}\mathcal{T}_{L,k}^2}{\mu\sum_{k=0}^{K-1}\mathcal{T}_{L,k}} + \frac{2\Theta(F_L(w_{L,0,0})-F_{\inf})}{\mu\overline{\alpha}\sum_{k=0}^{K-1}\mathcal{T}_{L,k}}$
	Yes	$\frac{4\Theta L_L \overline{\rho}^2 M_L^F\overline{\alpha}\sum_{k=0}^{K-1}\mathcal{T}_{L,k}^2}{\mu\sum_{k=0}^{K-1}\mathcal{T}_{L,k}} + \frac{2\Theta(F_L(w_{L,0,0})-F_{\inf})}{\mu\overline{\alpha}\sum_{k=0}^{K-1}\mathcal{T}_{L,k}}$

Table 3.1 The results for the bound on the expected value of the mean of the squared norm of the gradients with varying gradient correction and variance reduction for comparison.

correction. Therefore, when computational resources are limited, using a stochastic gradient correction may be preferred. When using a stochastic gradient correction, incorporating variance reduction is strongly recommended, as it reduces the dependency on the variance of the stochastic directions compared to using no variance reduction. This benefit can outweigh the drawback of a more complex implementation.

In the cases, where a stochastic gradient correction is used or there is no variance reduction, a constant residual term remains, which is a non zero upper bound for the expected value of the mean of the norm of the gradients of the objective function at the iterates. Decreasing the step size and the number of steps on each level, reduces the remainder term. However, since this also slows down the convergence of the second term, it is preferable to start with larger step sizes and a higher number of iterations at the lower levels so that the second term becomes small, and then gradually reduce them to reduce the first term.

CHAPTER 4

Image Classification

Following the theoretical discussion in the previous chapter, in this chapter we examine the performance of multilevel stochastic gradient in image classification. We discuss details of the implementation and numerical results.

Image classification is a common application of machine learning. By image classification, we mean categorizing an input image into one of several predefined classes. These training tasks usually involve a dataset consisting of images and their corresponding classes or labels. We use two different approaches to create a multilevel hierarchy.

First, we vary the number of layers and thus generate networks of different complexity. On one hand, networks with less layers have less optimization variables, making them computationally cheaper and faster to train. On the other hand, deeper networks with a larger number of layers are capable of learning more complex patterns, but have increased computational costs and training times.

For the second approach, we create datasets of decreasing complexity by changing the resolution of the images. By reducing the resolution, the dataset becomes cheaper to use during training. However, due to the overparametrization of deep neural networks, we adjust the architecture of the network by using smaller kernels in the convolutional layers. This not only balances the complexity of the model and the dataset, but further reduces the computational cost of training.

In Section 4.1 we share the details of the implementation, such as the used base network architecture and regularization. For both approaches we discuss the multilevel hierarchy, the choice of restriction and prolongation operators and numerical results separately in Sections 4.2 and 4.3. In Section 4.4 we summarize and discuss the results.

4.1 Implementation and architecture

The data set that we use is the CIFAR-10 data set [Kri09]. The dataset consists of 60,000 images, evenly distributed across 10 classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. Each image has three RGB color channels and a resolution of 32x32 pixels. Of these images, 50,000 are used for training, while 10,000 are set aside for testing. Additionally, we extract 10,000 images from the training set to create a validation set, which is used to monitor performance during training. For a small sample of images in this dataset see the left hand side in Figure 4.5.

4.1.1 Architecture

Even though the specifics of each architecture in the different versions may vary, all of them are based on the same basic architecture that we introduce in the following. We implement the architecture in PyTorch similar to the one of Ruthotto and Haber in [RH20]. An illustration is shown in Figure 4.1.

It starts with an opening convolution layer that increases the number of channels followed by a batch normalization layer and a ReLU activation function. The main part consists of consecutive residual blocks where each block contains several residual convolutional layers as defined in (2.10). The activation function σ^l in (2.10) is chosen as a ReLU activation function. The normalization layer N_l in (2.10) is implemented as a total variation normalization with $\epsilon = 10^{-3}$. The number of such residual layers in a residual block can vary and is specified separately with each network architecture. There are connecting layers between the residual blocks consisting of 1×1 convolutional layers to increase the channel size, a batch normalization layer, a ReLU activation function and an average pooling layer with a 2×2 stencil. While the 1×1 convolution increases the depth of the feature maps, the average pooling layer then reduces their spatial dimensions. Finally, the output of the last residual block is flattened. This means, that all the feature maps are concatenated and reshaped into a single high-dimensional vector. This vector is then passed through a fully connected layer that maps into the final output space of ten classes. A softmax activation function then maps the output to a probability distribution over ten classes.

We initialize biases with zero and weights using *Xavier initialization*. This initialization is also known as *Glorot initialization* and was proposed in [GB10] to prevent vanishing or exploding gradients. For each layer the input size is computed from the kernel height h, the kernel width w and the number of channels of the input c_{in} and of the output c_{out} as

$$f_{\text{in}} = hwc_{\text{in}} \quad \text{and} \quad f_{\text{out}} = hwc_{\text{out}}$$

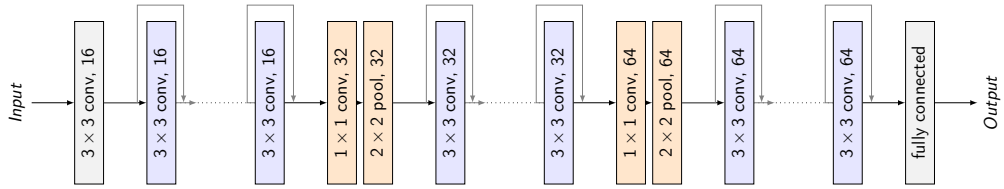


Figure 4.1 The base architecture used. Residual layers are blue and connecting layers are orange. The number of channels in the output is indicated.

and then the non bias weights in that layer are initialized with a normal distribution $\mathcal{N}(0, \sqrt{2/(f_{\text{in}} + f_{\text{out}})})$.

4.1.2 Regularization

As in [RH20], we use ℓ_2-regularization with parameter $2 \cdot 10^{-3}$ on non-bias weights, i.e.,

$$2 \cdot 10^{-3} \sum_{\text{Layer } i} \|w_1^i\|_2$$

to mitigate overfitting. Moreover we use a smoothness regularizer, as mentioned in Section 2.2.2, that penalizes changes in the weights of the convolutional layers from layer to layer in the residual blocks by

$$2 \cdot 10^{-3} \sum_{i \in \mathcal{L}} \|w_1^{i+1} - w_1^i + 2 \cdot 10^{-3}\|_F,$$

where $\mathcal{L} = \{i \in \mathbb{N}\text{: Layer } i \text{ and } i+1 \text{ are in the same residual block}\}$, $\|\cdot\|_F$ is the Frobenius matrix norm and w_1^i are the weights associated to the convolution operator in the ith-layer. This regularization penalizes the change of the weights from one residual layer to the next. Further, as mentioned in Section 3.2.3, whenever we use gradient correction in the coarse network, we also use a step size regularization as defined in (3.2) with parameter $\lambda = 5$.

4.1.3 Training

For the objective function we use a cross entropy loss function together with the regularizations from Section 4.1.2. For training we use SGD with a constant learning rate that is chosen as the best performing out of $\{0.1, 0.05, 0.01, 0.005, 0.001, 0.0005\}$. In all our tests we use a batch size of 250. Further, we also test reducing the learning rate by a fixed factor at regular intervals. Moreover, we use a simple data augmentation of

randomly flipping the images horizontally and randomly cropping by 6.25% where the borders have been filled with zeros as in [RH20].

All our computations are performed on a cluster featuring two Intel Xeon Gold 6130 CPUs, each running at 2.10 GHz. The system consists of 64 virtual processors (32 physical cores across two CPUs, with hyper-threading enabled), with 22.5 MB of L3 cache per CPU. The cluster runs Ubuntu 22 and is equipped with 768 GB RAM. We use PyTorch version 1.13.1.

4.2 Hierarchy based on network depth

As multigrid methods derive from solving differential equations and the hierarchy in these cases is established by using different sized grids in the discretization, one can use the similarity of residual networks and forward discretizations of differential equations to establish a hierarchy in a similar way. In multigrid methods the differential equations are discretized on different mesh sizes. According to the analogy, this corresponds to varying the number of layers in a network as discussed in Section 2.2.2.

As previously mentioned in Section 1.2, several approaches have been proposed using this hierarchy in different multilevel algorithms such as [HR17a; GKK20; GKT23].

4.2.1 Multilevel hierarchy

We use three levels, i.e. $L = 2$. We denote the time step in the residual blocks on level l by h_l. Starting with eight residual layers in one residual block, the time step on the highest level is $h_2 = 1$. For $l = 1$ we double the time step $h_1 = 2$ and halve the number of layers in one residual block. On the lowest level $l = 0$ we double the step size again, resulting in a step size of $h_0 = 4$ and two layers in a residual block. The resulting architectures are shown in Figure 4.2. It is based on the architecture described in Section 4.1.1. In this case, the opening layer increases the number of channels from 3 to 16 and the connecting layers increase them to 32, respectively 64. The fully connected layer flattens and classifies into the ten classes. The number of weights in the different layers is shown in Table 4.1. The CIFAR-10 dataset is not changed.

4.2.2 Restriction and prolongation operators

Suppose we have a trained model at level l, then we want to use that knowledge for a model at a higher level $l + 1$. The same holds for the other way round. As the

Table 4.1 The number of weights in the different layers for levels $l = 2, 1$ and 0 with the hierarchy established through the number of layers in a network.

	level $l = 2$	**level** $l = 1$	**level** $l = 0$
3×3 conv, 16	480	480	480
3×3 conv, 16	2336	2336	2336
3×3 conv, 16	2336	2336	2336
3×3 conv, 16	2336	2336	–
3×3 conv, 16	2336	2336	–
3×3 conv, 16	2336	–	–
3×3 conv, 16	2336	–	–
3×3 conv, 16	2336	–	–
3×3 conv, 16	2336	–	–
1×1 conv, 32	608	608	608
2×2 pool, 32	0	0	0
3×3 conv, 32	9280	9280	9280
3×3 conv, 32	9280	9280	9280
3×3 conv, 32	9280	9280	–
3×3 conv, 32	9280	9280	–
3×3 conv, 32	9280	–	–
3×3 conv, 32	9280	–	–
3×3 conv, 32	9280	–	–
3×3 conv, 32	9280	–	–
1×1 conv, 64	2240	2240	2240
2×2 pool, 64	0	0	0
3×3 conv, 64	36992	36992	36992
3×3 conv, 64	36992	36992	36992
3×3 conv, 64	36992	36992	–
3×3 conv, 64	36992	36992	–
3×3 conv, 64	36992	–	–
3×3 conv, 64	36992	–	–
3×3 conv, 64	36992	–	–
3×3 conv, 64	36992	–	–
fully connected	40970	40970	40970
total number of weights	433162	238730	141514

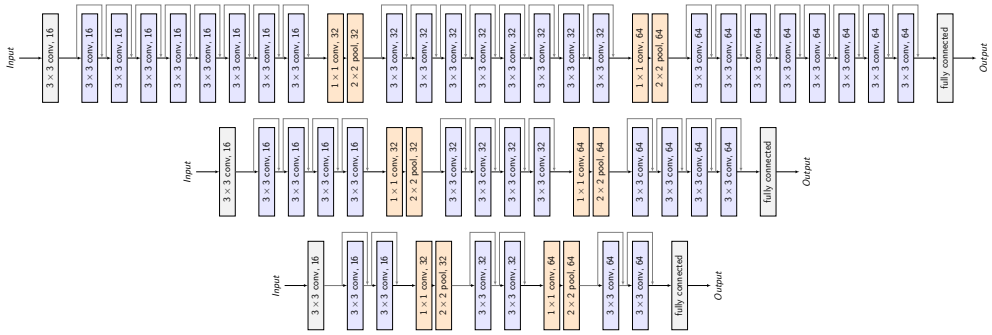


Figure 4.2 Architectures of the networks on level $l = 2, 1$ and 0 with 8, respectively 4, respectively 2 layers in a residual block. Residual layers are blue and connecting layers are orange. The number of channels in the output is indicated.

optimization problem consists of the weights of the network, the goal is to find a way to transfer the weights of the model on one level to weights for the model one level higher, respectively lower. In this case, the kernels of the connecting convolutional layers, the weights of the batch, the total variation normalization layers and of the fully connected layer as well as all of the biases have the same dimensions on all levels and can thus be transferred directly. However, since the number of residual layers in the residual blocks is different for different levels, we cannot transfer these weights directly.

Restriction and prolongation for weights in the residual layers

For a level l in this setting, the number of residual layer weights in the corresponding network is half as many as in $l + 1$. For the prolongation it is common to use a kind of interpolation, either constant or linear.

First, we consider the prolongation of weights from level $l = 1$ to the highest level, i.e., level $l = 2$. In this case, level $l = 2$ has eight layers in a residual block and level $l = 1$ has four. Assume $w_1^1, w_2^1, w_3^1, w_4^1$ to be the weights from the four layers on level $l = 1$, and $w_1^2, w_2^2, w_3^2, w_4^2, w_5^2, w_6^2, w_7^2, w_8^2$ the weights for the eight layers on level $l = 2$. To prolongate these weights, constant interpolation yields

$$\begin{aligned} w_1^2 &= w_1^1, & w_2^2 &= w_1^1, \\ w_3^2 &= w_2^1, & w_4^2 &= w_2^1, \\ w_5^2 &= w_3^1, & w_6^2 &= w_3^1, \\ w_7^2 &= w_4^1, & w_8^2 &= w_4^1 \end{aligned}$$

and linear interpolation yields

$$\begin{aligned} w_1^2 &= w_1^1, & w_2^2 &= \frac{1}{2}(w_1^1 + w_2^1), \\ w_3^2 &= w_2^1, & w_4^2 &= \frac{1}{2}(w_2^1 + w_3^1), \\ w_5^2 &= w_3^1, & w_6^2 &= \frac{1}{2}(w_3^1 + w_4^1), \\ w_7^2 &= w_4^1, & w_8^2 &= w_4^1. \end{aligned}$$

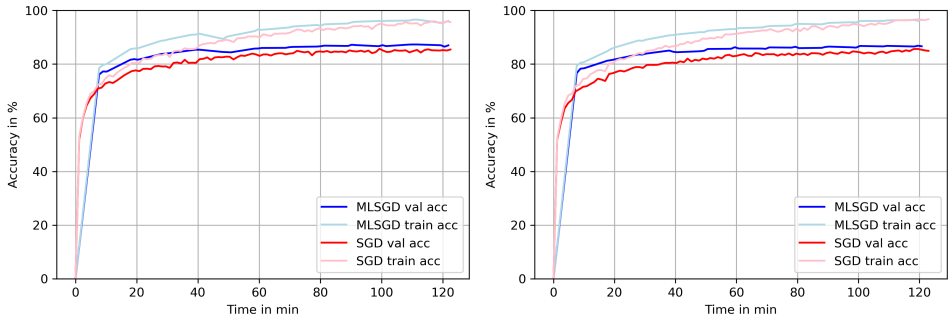
For simplicity, we use constant interpolation as we did not notice a difference between both methods. When writing all weights into one high-dimensional vector, constant interpolation can also be written as

$$P \begin{pmatrix} w_1^1 \\ w_2^1 \\ w_3^1 \\ w_4^1 \end{pmatrix} = \begin{pmatrix} I & 0 & 0 & 0 \\ I & 0 & 0 & 0 \\ 0 & I & 0 & 0 \\ 0 & I & 0 & 0 \\ 0 & 0 & I & 0 \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & I \\ 0 & 0 & 0 & I \end{pmatrix} \begin{pmatrix} w_1^1 \\ w_2^1 \\ w_3^1 \\ w_4^1 \end{pmatrix} = \begin{pmatrix} w_1^2 \\ w_2^2 \\ w_3^2 \\ w_4^2 \\ w_5^2 \\ w_6^2 \\ w_7^2 \\ w_8^2 \end{pmatrix},$$

where matrix entries of I, respectively 0, correspond to appropriately sized identity matrices, respectively zero matrices. For the restriction we then take the transposed prolongation and scale it, this means $R = \frac{1}{2}P$, i.e.,

$$R \begin{pmatrix} w_1^2 \\ w_2^2 \\ w_3^2 \\ w_4^2 \\ w_5^2 \\ w_6^2 \\ w_7^2 \\ w_8^2 \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} w_1^2 \\ w_2^2 \\ w_3^2 \\ w_4^2 \\ w_5^2 \\ w_6^2 \\ w_7^2 \\ w_8^2 \end{pmatrix} = \begin{pmatrix} w_1^1 \\ w_2^1 \\ w_3^1 \\ w_4^1 \end{pmatrix}.$$

For prolongating the weights from level $l = 0$, we also choose constant interpolation and the restriction as its transpose scaled by $\frac{1}{2}$.



(a) Constant step size

(b) Reducing step size

Figure 4.3 Plot of the results for the multilevel hierarchy using the number of layers of layers for MLSGD without gradient correction with different step size regimen. Training and validation accuracy are plotted against the time.

4.2.3 Numerical results

We compare the performance of MLSGD with that of SGD. For this purpose, we train two models. In both cases, we train a network with eight layers in a residual block, but one is trained using the established multilevel hierarchy and one is trained using SGD. For MLSGD, we use for the maximum number of iterations at each level $T_2 = 3, T_1 = 5, T_0 = 7$ and we do 12 V-cycles. For SGD, we use 100 epochs of SGD. This epoch number is chosen such as to take the same time training.

To start, we present results using no gradient correction at all. The best performing learning rate out of the tested learning rates for SGD and MLSGD is 0.05. We present results for using the constant step size 0.05 as well as reducing the step size 0.05 after every 45-th epochs, where we also count epochs on the lower levels for MLSGD, by a factor of 0.7. The training accuracy and the validation accuracy over time in minutes is shown in Figure 4.3. The final performance and training times are shown in Table 4.2. Note, that the first time the MLSGD models are evaluated is after the first epoch on the highest level because that is when the weights are updated the first time. As a result, the initial portion of the plot appears to show linear increase, although in reality the accuracy increases non-linearly.

We notice that MLSGD demonstrates a significant improvement in training speed compared to SGD. Specifically, in reaching a validation performance of 80%, 84% or 85%, MLSGD takes about half the time of SGD to reach these thresholds. MLSGD reaches the 80% validation threshold after about 17 minutes, whereas SGD takes more than 30 minutes. The gap widens for the higher validation thresholds with MLSGD reaching 85% after about 38 minutes and SGD needing 96 to 113 minutes.

Table 4.2 Results for the multilevel hierarchy using the number of layers for MLSGD without gradient correction with different step size regimen.

method	**MLSGD**	**SGD**	**MLSGD**	**SGD**
start step size	0.05		0.05	
step size regimen	–		reduce by 0.7	
train acc	96.03%	95.65%	96.52%	96.46%
val acc	86.74%	85.27%	86.84%	85.29%
test acc	86.64%	85.05%	86.14%	85.06%
training time	2:01:49	2:02:30	2:00:46	2:03:00
time to 80% val acc	0:17:53	0:32:54	0:17:39	0:35:28
time to 84% val acc	0:30:28	1:09:20	0:37:58	1:06:12
time to 85% val acc	0:38:03	1:26:58	0:37:58	1:53:09

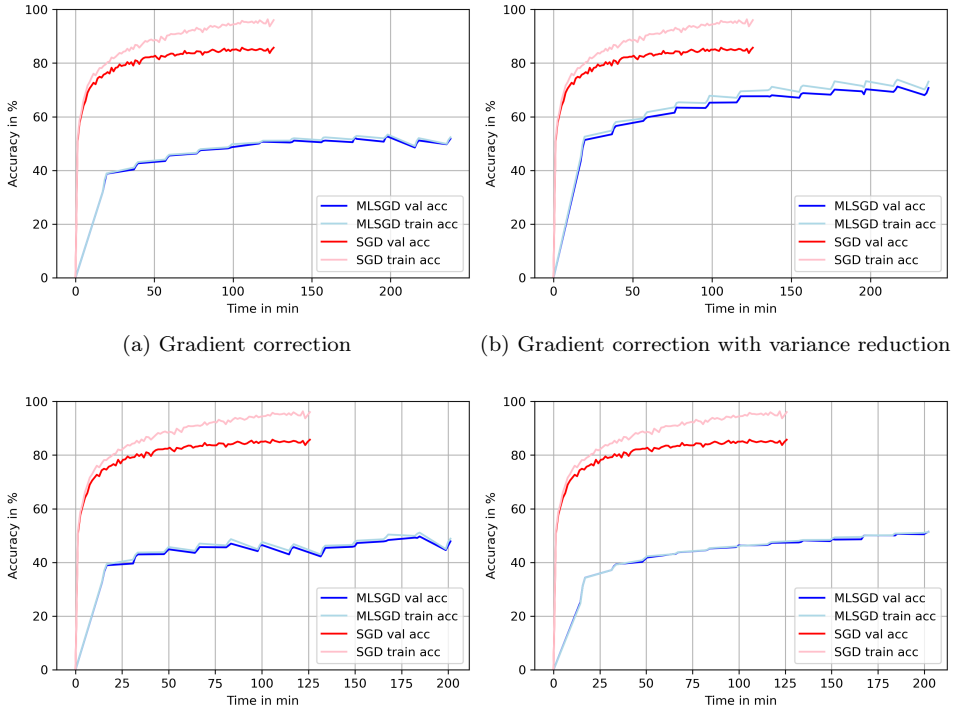
Despite the similar total training times, the final training, testing and validation accuracy of MLSGD is roughly 1% higher than that of SGD. This is also represented in the plots where the validation and training accuracy of MLSGD at equivalent time points are consistently higher than the ones of SGD.

Additionally, we present results using gradient correction. As analyzed in Section 3.3, we distinguish between the four cases of using a deterministic gradient correction with and without variance reduction and a stochastic gradient correction with and without variance reduction. The first V-cycle never uses a gradient correction to provide a warm start for the highest level. When using a stochastic gradient correction, we use 3.125% of the training samples to compute a stochastic approximation of the gradient. Again, we do 12 V-cycles for MLSGD. When using no variance reduction, a learning rate of 0.001 performs best with deterministic and stochastic gradient correction. When using variance reduction with a deterministic gradient correction, a learning rate of 0.01 is best and when using a stochastic gradient correction with variance reduction, we need to use a much smaller learning rate of 0.0005 to reach a stable training.

The training accuracy and the validation accuracy over time in minutes is shown in Figure 4.4. For comparison, the results for SGD with a step size of 0.05 after 100 epochs are plotted as well. The final performance and training times are shown in Table 4.3.

First, we notice that 12 V-cycles with gradient correction are much more time consuming than without. Without using gradient correction, we need about 2 hours for training whereas gradient correction takes between 1.6 and 2 times as long. Here, using a stochastic gradient correction instead of a deterministic one reduces the time for 12 V-cycles from almost four hours to three hours and twenty minutes.

However, when comparing the training progress, we observe that the best training and validation performance of the four settings is achieved when a deterministic gradient



(a) Gradient correction

(b) Gradient correction with variance reduction

(c) Stochastic gradient correction

(d) Stochastic gradient correction with variance reduction

Figure 4.4 Plot for the results for the multilevel hierarchy using the number of layers for different variants of MLSGD with a fixed stepsize. Training and validation accuracy are plotted against the time.

correction with variance reduction is used. Nevertheless, with a test accuracy of about 70% after almost four hours, it is not a competitive alternative to neither SGD, which reaches a test accuracy of 85% after less than two hours, nor MLSGD without gradient correction, which has a test accuracy of 85% after 38 minutes.

4.3 Hierarchy based on image resolution

Another natural approach for the multilevel structure is to use the resolution of the images, as also done in [RH20]. We generate new data sets from the original data sets by restricting the images using average pooling. Moreover, we use differently sized convolution kernels on each level.

Table 4.3 Results for the multilevel hierarchy using the number of layers for different variants of MLSGD with a fixed stepsize.

	MLSGD			
gradient correction	deterministic		stochastic	
variance reduction	yes	no	yes	no
Figure 4.4	a)	b)	c)	d)
train acc	73.23%	52.38%	51.35%	48.60%
val acc	70.61%	51.15%	51.16%	47.99%
test acc	70.80%	51.71%	51.27%	48.54%
training time	3:56:37	3:58:02	3:22:14	3:21:13

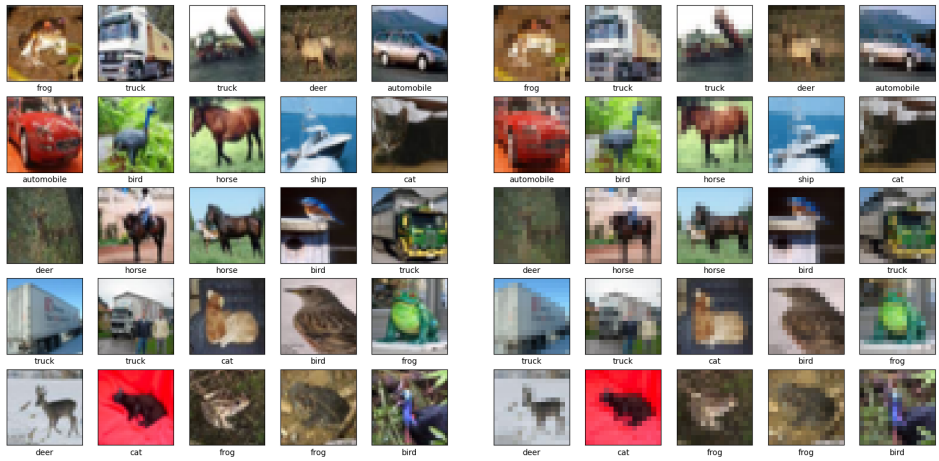


Figure 4.5 Example images for the CIFAR-10 data sets. On the left are images from the original data set used at level $l = 1$. On the right are the reduced resolution images for level $l = 0$.

4.3.1 Multilevel hierarchy

In this approach, we use two levels as opposed to the previous three level approach. The CIFAR-10 data set contains images with a resolution of 32×32 pixels. For level $l = 1$, the original data set is used. For the level $l = 0$, we create a new dataset by applying average pooling with a kernel size of 2×2. This way the dimensions of the images are halved and the number of pixels is reduced by $\frac{1}{4}$ resulting in 16×16. Examples from these datasets are shown in Figure 4.5.

The networks we implement are based on the architecture introduced in Section 4.1.1 but vary in the size of convolution kernels. For the network on level $l = 1$, we use convolution kernels of size 3×3 in the residual blocks and the opening convolutional layer. For $l = 0$ we use convolution kernels of size 2×2. Since the input size is different at

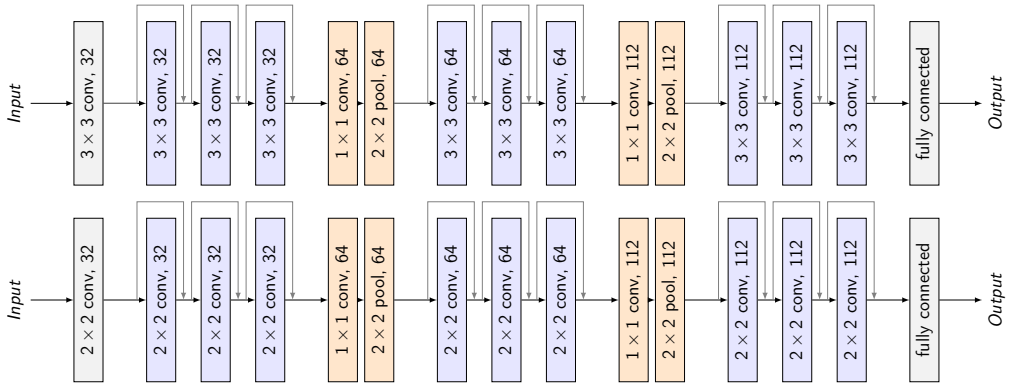


Figure 4.6 Architectures of the networks on level $l = 1$ and 0 with 3×3, respectively 2×2 kernels. Residual layers are blue and connecting layers are orange. The number of channels in the output is indicated.

both levels, the feature maps also have different dimensions. For this reason, the number of weights in the fully connected layer also differs in between the two architectures. The opening layer increases the number of channels to 32 and the connecting layers to 64, respectively 112. We use three consecutive residual layers in the residual blocks. The time step in the residual layers is chosen as $h_l = 1$ in both cases. The resulting architectures are shown in Figure 4.6. The number of weights in each layer can be seen in Table 4.4.

4.3.2 Restriction and prolongation operators

Again, we need to give restriction and prolongation operators to transfer the weights for a network at one level to a network at another level. Also in this case, the dimensions of the weights of the opening and connecting convolution layers, of the batch or the total variation normalization layers as well as all of the biases are the same across both levels. Therefore, these weights can be transferred directly. However, weights associated to the convolution layers in the residual blocks and the fully connected layers are different sized on both levels and thus cannot be transferred directly.

Restriction for 3×3 convolution kernel

Let the input feature maps to a convolutional layer have dimensions $h \times w \times c$ where h is the height, w is the width and c is the number of channels with $h, w, c \in \mathbb{N}$. For the following derivation, we assume the input has been reshaped into a vector in $\mathbb{R}^{hw \times c}$. The prolongation and the restriction then operate on the spatial dimensions, changing

Table 4.4 The number of weights in the different layers for both levels $l = 1$ and 0 with the hierarchy established through the resolution of the dataset.

	level $l = 1$	**level** $l = 0$
3×3 conv, 16	960	960
3×3 conv, 16	9280	4160
3×3 conv, 16	9280	4160
3×3 conv, 16	9280	4160
1×1 conv, 32	2240	2240
2×2 pool, 32	0	0
3×3 conv, 32	36992	16512
3×3 conv, 32	36992	16512
3×3 conv, 32	36992	16512
1×1 conv, 64	7504	7504
2×2 pool, 64	0	0
3×3 conv, 64	113120	50400
3×3 conv, 64	113120	50400
3×3 conv, 64	113120	50400
fully connected	71690	17930
total number of weights	560570	241850

the height and width but keeping the number of channels the same. We define R as the restriction operator on the input, which maps the higher resolution images to lower resolution. As already mentioned in Section 4.3.1, we use average pooling for restriction with a 2×2 kernel. This means, the average over four adjacent pixels is computed to create one output pixel. We define P to be the prolongation operator, which maps lower-resolution images to a higher resolution. So far, we have not explicitly needed to define a prolongation operator on the input. However, we now choose the prolongation to be constant interpolation, meaning that each pixel in the lower resolution image is replicated four times to reach the desired dimensions. We represent the restriction and prolongation operators on the input images as matrices acting on the reshaped feature maps

$$P = \begin{pmatrix} p & 0 & 0 & \cdots & 0 \\ p & 0 & 0 & \cdots & 0 \\ 0 & p & 0 & \cdots & 0 \\ 0 & p & 0 & \cdots & 0 \\ & & \ddots & & \\ 0 & 0 & 0 & \cdots & p \\ 0 & 0 & 0 & \cdots & p \end{pmatrix} \in \mathbb{R}^{(hw)\times(hw/4)},$$

$$R = \begin{pmatrix} r & r & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & r & r & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \ddots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & r & r \end{pmatrix} \in \mathbb{R}^{(hw/4)\times(hw)},$$

where

$$p = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ & & \ddots & & \\ 0 & 0 & 0 & \cdots & 1 \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix} \in \mathbb{R}^{w\times(w/2)}$$

and

$$r = \frac{1}{4} \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \ddots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \in \mathbb{R}^{(w/2)\times w}.$$

These matrices allow us to examine the interplay of restriction, convolution and prolongation in detail.

The goal is to build a convolutional layer in such a way that progress from one level can exactly be transferred to another level. To that end, assume that $k_{i,j}$ is a 2×2 convolution with 1 in the (i,j)-th entry and 0 in all others. Then it follows from (2.5) that $\{k_{i,j}\}_{i,j=1}^{2}$ can be seen as a basis for the 2×2 convolutions. Let $c_{i,j}$ denote 3×3 convolution with a kernel that has a 1 in the (i,j)-th entry and 0 in all others. By Eq. (2.5), any 3×3 convolution can then be written as

$$\sum_{i,j=1}^{3} w_{i,j} c_{i,j}.$$

Remember from Section 2.2.1 that a convolution with $c_{i,j}$ with $i,j \in \{1,2,3\}$ can be written as matrix multiplication with the following matrices

$$X_1 = \begin{pmatrix} I & 0 & 0 \end{pmatrix}, \quad X_2 = \begin{pmatrix} 0 & I & 0 \end{pmatrix}, \quad X_3 = \begin{pmatrix} 0 & 0 & I \end{pmatrix} \in \mathbb{R}^{(w-2)\times w},$$

where I is the $(w-2) \times (w-2)$ identity matrix that is expanded through additional zero columns. Then for $j = 1, \dots 3$ we write

$$c_{1,j} = \begin{pmatrix} X_j & 0 & 0 & 0 & 0 \\ 0 & \ddots & 0 & 0 & 0 \\ 0 & 0 & X_j & 0 & 0 \end{pmatrix}, \qquad c_{2,j} = \begin{pmatrix} 0 & X_j & 0 & 0 & 0 \\ 0 & 0 & \ddots & 0 & 0 \\ 0 & 0 & 0 & X_j & 0 \end{pmatrix},$$

$$c_{3,j} = \begin{pmatrix} 0 & 0 & X_j & 0 & 0 \\ 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & X_j \end{pmatrix}.$$

Similarly, convolution with $k_{i,j}$, for $i,j \in \{1,2\}$, can be seen as matrix multiplication with the matrices

$$k_{1,1} = \begin{pmatrix} Y_1 & 0 & 0 & 0 \\ 0 & \ddots & 0 & 0 \\ 0 & 0 & Y_1 & 0 \end{pmatrix}, \qquad k_{1,2} = \begin{pmatrix} Y_2 & 0 & 0 & 0 \\ 0 & \ddots & 0 & 0 \\ 0 & 0 & Y_2 & 0 \end{pmatrix},$$

$$k_{2,1} = \begin{pmatrix} 0 & Y_1 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & Y_1 \end{pmatrix}, \qquad k_{2,2} = \begin{pmatrix} 0 & Y_2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & Y_2 \end{pmatrix},$$

where

$$Y_1 = \begin{pmatrix} I & 0 \end{pmatrix}, \quad Y_2 = \begin{pmatrix} 0 & I \end{pmatrix} \in \mathbb{R}^{(w-1)\times w}$$

are identity matrices with an additional zero column on the right, respectively left.

However, we do not simply want to adopt a standard 3×3 convolution on the higher level. Instead, the convolution on the higher level is constructed by reusing

the exact weights from the lower level. To reach the same degrees of freedom as in a 3×3 convolution, additional weights are added. Hence, the general structure of the convolution at the higher level is given by

$$\sum_{(i,j)\in\mathcal{I}} w_{i,j} c_{i,j} + \sum_{i,j=1}^{2} \overline{w}_{i,j} P k_{i,j} R$$

where R and P are the restriction and prolongation on the feature maps in the corresponding dimensions. The first term represents the additional degrees of freedom for the higher level with yet to be determined index sets $\mathcal{I} \subseteq \{1,2,3\}^2$. The second term reuses the weights from the lower level. Here, the low level convolution kernel $k_{i,j}$ operates on the restricted feature maps. After applying the low level convolution, the result is prolongated to match the higher level dimensions. This way, the information from the lower level is integrated while at the same time providing flexibility for the higher level through additional degrees of freedom.

To determine the set $\mathcal{I}$, we analyze the behavior of $k_{i,j}$ when applied to the restricted images. Ideally, we would replicate this operation exactly using 3×3 convolutions on the original images. However, as we will see, this is not possible.

When combining the convolutions $\{k_{i,j}\}_{i,j=1}^{2}$ with the restriction and prolongation, this is the same as multiplying with the matrices

$$Pk_{1,1}R = \frac{1}{4}\begin{pmatrix} A_1 & A_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ A_1 & A_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & A_1 & A_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & A_1 & A_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \ddots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & A_1 & A_1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & A_1 & A_1 & 0 & 0 \end{pmatrix},$$

$$Pk_{1,2}R = \frac{1}{4}\begin{pmatrix} A_2 & A_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ A_2 & A_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & A_2 & A_2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & A_2 & A_2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \ddots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & A_2 & A_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & A_2 & A_2 & 0 & 0 \end{pmatrix},$$

$$Pk_{2,1}R = \frac{1}{4}\begin{pmatrix} 0 & 0 & A_1 & A_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & A_1 & A_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & A_1 & A_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & A_1 & A_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \ddots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & A_1 & A_1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & A_1 & A_1 \end{pmatrix},$$

$$Pk_{2,2}R = \frac{1}{4}\begin{pmatrix} 0 & 0 & A_2 & A_2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & A_2 & A_2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & A_2 & A_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & A_2 & A_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \ddots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & A_2 & A_2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & A_2 & A_2 \end{pmatrix},$$

where 0 are matrices filled with zeros of the same size as A_1 and A_2 and these are

$$A_1 = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \ddots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{pmatrix}, \quad A_2 = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \ddots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}.$$

However, it is not possible to replicate these exact matrices by using a linear combination of $\{c_{i,j}\}_{i,j=1}^{3}$. For $k_{1,1}$, we can use for example $c_{1,1}, c_{1,2}, c_{2,1}$ and $c_{2,2}$, i.e.,

$$k_{1,1} \approx \frac{1}{4}(c_{1,1} + c_{1,2} + c_{2,1} + c_{2,2}) = \frac{1}{4}\begin{pmatrix} B_1 & B_1 & 0 & 0 & 0 & 0 \\ 0 & B_1 & B_1 & 0 & 0 & 0 \\ 0 & 0 & \ddots & \ddots & 0 & 0 \\ 0 & 0 & 0 & B_1 & B_1 & 0 \end{pmatrix}$$

with the matrix

$$B_1 = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & \ddots & \ddots & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}.$$

This way, every second row of $k_{1,1}$ and $\frac{1}{4}(c_{1,1}+c_{1,2}+c_{2,1}+c_{2,2})$ coincide. Similarly, choosing for the other kernels

$$k_{2,1} \approx \frac{1}{4}(c_{1,2}+c_{1,3}+c_{2,2}+c_{2,3}) = \frac{1}{4}\begin{pmatrix} B_2 & B_2 & 0 & 0 & 0 & 0 \\ 0 & B_2 & B_2 & 0 & 0 & 0 \\ 0 & 0 & \ddots & \ddots & 0 & 0 \\ 0 & 0 & 0 & B_2 & B_2 & 0 \end{pmatrix}$$

$$k_{1,2} \approx \frac{1}{4}(c_{2,1}+c_{2,2}+c_{3,1}+c_{3,2}) = \frac{1}{4}\begin{pmatrix} 0 & B_1 & B_1 & 0 & 0 & 0 \\ 0 & 0 & B_1 & B_1 & 0 & 0 \\ 0 & 0 & 0 & \ddots & \ddots & 0 \\ 0 & 0 & 0 & 0 & B_1 & B_1 \end{pmatrix}$$

$$k_{2,2} \approx \frac{1}{4}(c_{2,2}+c_{2,3}+c_{3,2}+c_{3,3}) = \frac{1}{4}\begin{pmatrix} 0 & B_2 & B_2 & 0 & 0 & 0 \\ 0 & 0 & B_2 & B_2 & 0 & 0 \\ 0 & 0 & 0 & \ddots & \ddots & 0 \\ 0 & 0 & 0 & 0 & B_2 & B_2 \end{pmatrix}$$

with matrix

$$B_2 = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & \ddots & \ddots & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

also gives representations where every second row is correct. Hence, we assume that the lower level approximately learns the weights for the following convolution kernels

$$\frac{1}{4}\begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad \frac{1}{4}\begin{pmatrix} 0 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix}, \quad \frac{1}{4}\begin{pmatrix} 0 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \end{pmatrix}, \quad \frac{1}{4}\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}.$$

For the higher level convolution, we now want to complement these kernels with a subset of the convolution basis $\{c_{i,j}\}_{i,j=1}^{3}$ such that we form a new basis for 3×3 convolutions. This allows us to represent any 3×3 convolution. Specifically, we achieve this by selecting $c_{1,2}, c_{2,1}, c_{2,2}, c_{2,3}, c_{3,2}$. Hence the convolution at the higher level is defined as

$$\sum_{(i,j)\in\mathcal{I}} w_{i,j} c_{i,j} + \sum_{i,j=1}^{2} \overline{w}_{i,j} P k_{i,j} R$$

with $\mathcal{I} = \{(1,2),(2,1),(2,2),(2,3),(3,2)\}$. The weights for the higher-level convolution are

$$w_{1,2}, w_{2,1}, w_{2,2}, w_{2,3}, w_{3,2}, \overline{w}_{1,1}, \overline{w}_{1,2}, \overline{w}_{2,1}, \overline{w}_{2,2}$$

while the weights for the lower-level convolution are

$$\overline{w}_{1,1}, \overline{w}_{1,2}, \overline{w}_{2,1}, \overline{w}_{2,2}.$$

Therefore, the prolongation of the weights P_1 and the restriction R_1 are

$$P_1 \begin{pmatrix} \overline{w}_{1,1} \\ \overline{w}_{1,2} \\ \overline{w}_{2,1} \\ \overline{w}_{2,2} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \overline{w}_{1,1} \\ \overline{w}_{1,2} \\ \overline{w}_{2,1} \\ \overline{w}_{2,2} \end{pmatrix} \quad \text{and} \quad R_1 \begin{pmatrix} w_{1,2} \\ w_{2,1} \\ w_{2,2} \\ w_{2,3} \\ w_{3,2} \\ \overline{w}_{1,1} \\ \overline{w}_{1,2} \\ \overline{w}_{2,1} \\ \overline{w}_{2,2} \end{pmatrix} = \begin{pmatrix} \overline{w}_{1,1} \\ \overline{w}_{1,2} \\ \overline{w}_{2,1} \\ \overline{w}_{2,2} \end{pmatrix}.$$

Restriction and prolongation for the fully connected layer

For the fully connected layer weights we use the same restriction and prolongation as for the initial images in the data set meaning average pooling and linear interpolation. However, before applying the prolongation or restriction, the weight matrix must be reshaped such that its dimensions resemble the dimensions of an input image.

As established in Section 2.2.1, the weight matrix W in a fully connected layer has the shape $k \times hwc$, where k is the desired output dimension, h the original height of the input, w the width of the input and c the number of channels. Suppose the input in unflattened shape is $x \in \mathbb{R}^{h \times w \times c}$.

To find an appropriate restriction and prolongation, for a level $l \in \{0, 1, \dots, L\}$ we reshape the weight matrix keeping the first dimension fixed, thus getting a tensor $\mathcal{W}^l$ with dimension $k \times h \times w \times c$. This way the entry $\mathcal{W}^l_{i,m,n,j}$ for $i = 1, \dots, k, m = 1, \dots, h, n = 1, \dots, w, j = 1, \dots, c$ describes the scalar weight associated to the (i, m, n, j)-th entry in the input.

For each $i = 1, \dots, k$ and $j = 1, \dots, c$ we get a matrix $\mathcal{W}^l_{i,\cdot,\cdot,j}$. Now this matrix corresponds to the weights applied to $x_{i,\cdot,\cdot,j}$ which can be seen as an image. Thus we apply the same prolongation and restriction as to the input images meaning average pooling R and linear interpolation P and then reshape the resulting matrices to the desired shape, i.e.

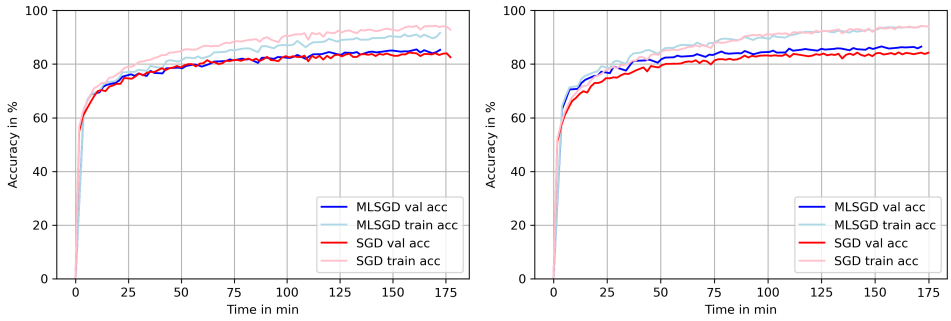
$$(R_1(\mathcal{W}^1))_{i,\cdot,\cdot,j} = R(\mathcal{W}^1_{i,\cdot,\cdot,j}) \quad \text{and} \quad (P_1(\mathcal{W}^0))_{i,\cdot,\cdot,j} = P(\mathcal{W}^0_{i,\cdot,\cdot,j})$$

for $i = 1, \dots, k, j = 1, \dots, c$.

4.3.3 Numerical results

Again, we compare the performance of MLSGD to that of SGD by training two models. Both models are networks with 3×3 convolution kernels and three layers in a residual block, but one is trained using the established multilevel hierarchy and the other using SGD. For SGD, we use 100 epochs. To approximately take the same training time as SGD, for MLSGD without gradient correction we use as maximum iterations on the different levels $T_1 = 7, T_0 = 3$ and we do 23 two-level V-cycles. Again we test two step size regimen. First, we use a constant step size of 0.05 for MLSGD and 0.1 for SGD. In a second test, we reduce the step sizes over time. We start with a step size of 0.1 which we reduce every 50 epochs by a factor of 0.7 for both, SGD as well as MLSGD. The training accuracy and validation accuracy over time in minutes is plotted in Figure 4.7. The results are presented in Table 4.5.

Similar to the results in Section 4.2.3, we notice that MLSGD outperforms SGD in validation and test performance although they still fall short of the results obtained in Section 4.2.3. This difference is likely due to differences in the architectures, as the architectures in this section are less deep than the ones in Section 4.2. In the case without step size reduction, MLSGD and SGD perform comparable in validation and test accuracy, where MLSGD has a minor advantage. However, the final training accuracy of SGD is higher than that of MLSGD, which is also consistent with the plots in Figure 4.7, indicating that SGD overfits more.



(a) Constant step size

(b) Reducing step size

Figure 4.7 Plot of the results for the multilevel hierarchy using the resolution of layers for MLSGD without gradient correction with different step size regimen. Training and validation accuracy are plotted against the time.

Table 4.5 Results for the multilevel hierarchy using the resolution of layers for MLSGD without gradient correction with different step size regimen.

	MLSGD	**SGD**	**MLSGD**	**SGD**
start step size	0.05	0.1	0.1	
step size regimen	–		reduce by 0.7	
train acc	91.70%	92.93%	94.06%	93.98%
val acc	85.49%	82.97%	86.29%	83.99%
test acc	83.08%	82.96%	84.50%	83.04%
training time	2:52:16	2:57:10	2:51:33	2:54:54
time to 80% val acc	1:03:37	0:58:39	0:37:12	0:52:26
time to 84% val acc	1:57:58	2:37:40	1:18:17	2:21:39
time to 85% val acc	2:29:42	–	1:42:24	–

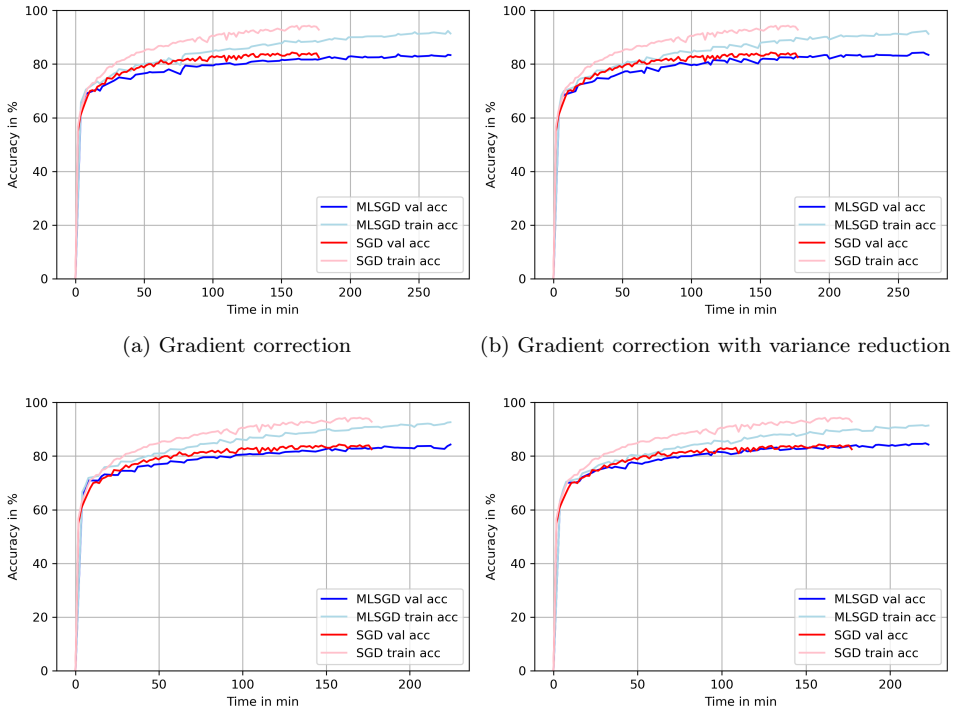
With the reducing step size regimen, MLSGD gains a significant advantage not only reaching a higher validation and test accuracy of 86.29% and 84.50% compared to SGD's 83.99% and 83.04% respectively but also reaching validation thresholds of 80% and 84% faster. Specifically, MLSGD reaches 80% validation accuracy in 37 minutes, whereas SGD takes 52 minutes. Similarly in reaching a 84% validation accuracy, MLSGD takes only 0.55 of the time SGD needs. SGD even fails to reach the 85% threshold within the recorded training time.

Moreover, we present results using gradient correction. Based on Section 3.3, we distinguish between using a deterministic gradient correction with and without variance reduction and using a stochastic gradient correction with and without variance reduction. Again, the first V-cycle does not use a gradient correction in any setting. For the stochastic gradient correction we use again a percentage of 3.125% of the samples to compute the stochastic approximation of the gradient. For all settings, the best step size tested is 0.1. Again we perform 23 V-cycles for MLSGD.

The plots of training and validation accuracy over time during training compared to the performance of SGD is shown in Figure 4.8. The results are shown in Table 4.6.

Once again, the networks using gradient correction need more time for the same number of V-cycles as without gradient correction. Similar to Section 4.2.3, using a stochastic gradient correction reduces the training time from four and a half hours to three hours and forty minutes.

However, other than in Section 4.2.3, in this hierarchy, the performance of MLSGD with gradient correction is almost comparable to that of SGD. As seen in Figure 4.8, MLSGD with a stochastic gradient correction and variance reduction can compete with SGD regarding validation accuracy. Even though MLSGD in this case does not outperform SGD, the training accuracy of SGD is much higher indicating earlier overfitting of SGD.



(a) Gradient correction

(b) Gradient correction with variance reduction

(c) Stochastic gradient correction

(d) Stochastic gradient correction with variance reduction

Figure 4.8 Plot of the results for the multilevel hierarchy using the resolution of the images for different variants of MLSGD with a fixed stepsize. Training and validation accuracy are plotted against the time.

This could indicate that MLSGD is more robust, particularly when using a stochastic gradient correction with variance reduction.

4.4 Discussion

We find that the gradient correction has not performed as effectively in practice as the theory would suggest. While it is crucial in theory for ensuring consistency between the levels, in our numerical results it either added no benefit or slowed down the training process. One key issue is the computational cost of calculating a full gradient for the gradient correction is expensive. Even with a stochastic gradient correction, MLSGD loses the time advantage it otherwise has over SGD. This indicates that the computational costs of the gradient correction outweighs the theoretical benefit it adds to the convergence.

Table 4.6 Results for the multilevel hierarchy using the resolution of the images for different variants of MLSGD with a fixed stepsize.

	MLSGD			
gradient correction	deterministic		stochastic	
variance reduction	yes	no	yes	no
Figure 4.8	a)	b)	c)	d)
train acc	91.37%	91.43%	91.31%	92.82%
val acc	83.64%	83.23%	83.82%	84.41%
test acc	83.09%	82.83%	83.29%	84.20%
training time	4:31:57	4:33:04	3:42:44	3:44:19
time to 80% val acc	1:29:55	1:52:20	1:23:37	1:36:12
time to 84% val acc	4:19:58	–	2:59:45	3:44:19
time to 85% val acc	–	–	–	–

Another possible explanation is that first order coherence may not be the best criterion for ensuring the helpfulness of the lower level directions, especially given the highly nonlinear loss landscape of deep neural networks. The gradient at the higher level might change too fast to provide helpful directions over several steps for the lower level. A different approach might be to start without gradient correction and integrate it later on in the training process when the method is close to a minimum. This way the gradient correction would not slow down the beginning of the training process but yield good approximations when only small improvements need to be made.

MLSGD also presents other challenges. Aside from the increased number of hyperparameters that have to be tuned, such as the constant in the angle condition or the step size regularization, it is only applicable to problems where a sufficient hierarchy can be defined. This is a non trivial task in most applications. Both of our hierarchical approaches also face limitations when it comes to scaling to more levels. In Section 4.2 introducing more layers increases the depth of the network. However, there is a practical limit to network depth when training becomes too expensive and adding more layers no longer improves accuracy but rather increases the risk of overfitting. On the other side, no further lower levels can be introduced once the minimum amount of one layer in a residual block is reached. Similarly, for the approach in Section 4.3, the structure of the convolutional layer bounds the possible number of layers by the size of the convolution kernel. By that we mean, that to introduce another level in a similar way, 1×1 or 4×4 convolutions are needed. Both are less common in architectures than 3×3 convolutions.

Despite these limitations, MLSGD without gradient correction consistenly demonstrates faster training and outperforms SGD in reaching target accuracy thresholds. Furthermore, the final test accuracy of MLSGD is better than that of SGD. A possible

explanation might be that the minima found by MLSGD are more stable leading to better generalization.

In conclusion, our experiments show the usability and efficiency multilevel methods can provide in a machine learning setup. However, further research is necessary to address the inefficiency of the gradient correction. Nonetheless, MLSGD can be an efficient alternative to SGD in structured problems, where hierarchies can be defined.

CHAPTER 5

Conclusion

This work deals with a multilevel stochastic gradient descent method. The results of this thesis can be split into two parts. The first part focuses on the theoretical convergence of the method and explores which variations can be implemented to improve said convergence. The MLSGD method is based on multigrid methods for solving PDEs. As is common in such methods, a gradient correction is used to establish first order coherence. To further guarantee the helpfulness of the lower level direction, we introduce step size regularization and termination criteria. The results from Section 3.3 provide an upper bound for the expected value of the mean of the norm of the gradients consisting of a constant residual term and a term that approaches zero as the number of cycles increases. By using a stochastic direction in the gradient correction instead of the full gradient, the computational effort to compute an update is reduced but the upper bounds become potentially worse. If a variance reduction term is used with a deterministic gradient correction, then the residual vanishes and the bound converges to zero as the iteration number increases.

In Chapter 4, we apply MLSGD to the CIFAR-10 dataset. Based on a common architecture, we explore two different approaches to establish a multilevel hierarchy. The first approach varies the number of layers to create networks of varying depth and thus complexity. The second approach coarsens the resolution of the dataset and decreases the size of the convolutional kernels in the network. In both cases, we find that when not using a gradient correction, MLSGD outperforms SGD. On the other hand, with gradient correction MLSGD is slower than SGD.

This thesis points to potential directions for further research. First of all, even though gradient correction is crucial in theory, in practice it appears to be less effective than expected often slowing down training without improving accuracy. Addressing

this inefficiency is a key direction for research as gradient correction with the correct modification or adaptive regimen could prove still valuable. Additionally as seen in Chapter 4, the choice of a hierarchy and the transfer operators for restriction and prolongation plays a critical role in the performance of MLSGD. While our approaches work well, there might be other transfer operators that could yield even better results. Currently, for each application one must find a new hierarchy and there is no universal way to choose promising transfer operators. Finally, although our theoretical analysis and practical experiments focus on multilevel SGD, most modern training algorithms use advanced variants such as Adagrad or Adam. Such adaptive methods often surpass SGD in performance, suggesting that multilevel variants of these methods could also be interesting extensions.

List of Figures

List of Tables

Bibliography

[Abd+14] Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, Li Deng, Gerald Penn, and Dong Yu. "Convolutional Neural Networks for Speech Recognition." *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 22.10 (2014), pp. 1533–1545.

[AGH18] Peter Martey Addo, Dominique Guegan, and Bertrand Hassani. "Credit Risk Analysis Using Machine and Deep Learning Models." *Risks* 6.2 (2018), p. 38.

[Bor+18] Nicolas Borisov, Victor Tkachev, Maria Suntsova, Olga Kovalchuk, Alex Zhavoronkov, Ilya Muchnik, and Anton Buzdin. "A Method of Gene Expression Data Transfer from Cell Lines to Cancer Patients for Machine-Learning Prediction of Drug Efficiency." *Cell Cycle* 17.4 (2018), pp. 486–491.

[Bor05] Alfio Borzì. "On the Convergence of the MG/OPT Method." *PAMM* 5 (2005), pp. 735–736.

[BCN18] Léon Bottou, Frank E. Curtis, and Jorge Nocedal. "Optimization Methods for Large-Scale Machine Learning." *SIAM Rev.* 60.2 (2018), pp. 223–311.

[BKK20] Vanessa Braglia, Alena Kopanicáková, and Rolf Krause. "A Multilevel Approach to Training." *CoRR* abs/2006.15602 (2020). arXiv: 2006.15602.

[Bra93] James H. Bramble. *Multigrid methods*. Vol. 294. Pitman Research Notes in Mathematics Series. Harlow: Longman Scientific & Technical, 1993.

[Bra73] Achi Brandt. "Multi-Level Adaptive Technique (MLAT) For Fast Numerical Solution to Boundary Value Problems." *Proceedings of the Third International Conference on Numerical Methods in Fluid Mechanics*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1973, pp. 82–89.

[Bra77] Achi Brandt. "Multi-Level Adaptive Solutions to Boundary Value Problems." *Mathematics of Computation* 31 (1977), pp. 333–390.

[BMR85] Achi Brandt, Steve McCormick, and John Ruge. "Algebraic Multigrid (AMG) for Sparse Matrix Equations." *Sparsity and its Applications*. Cambridge: Cambridge University Press, 1985, pp. 257–284.

[Bre01] Leo Breiman. "Random Forests." *Machine Learning* 45 (2001), pp. 5–32.

[Can+15] Diego Cantor-Rivera, Ali R. Khan, Maged Goubran, Seyed M. Mirsattari, and Terry M. Peters. "Detection of Temporal Lobe Epilepsy Using Support Vector Machines in Multi-Parametric Quantitative MR Imaging." *Computerized Medical Imaging and Graphics* 41 (2015), pp. 14–28.

[Cha+17] Bo Chang, Lili Meng, Eldad Haber, Frederick Tung, and David Begert. "Multi-level Residual Networks from Dynamical Systems View." *CoRR* abs/1710.10348 (2017). arXiv: `1710.10348`.

[Cho+14] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation." *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, 2014, pp. 1724–1734.

[CKP03] James Clark, Irena Koprinska, and Josiah Poon. "A Neural Network Based Approach to Automated E-Mail Classification." *Proceedings IEEE/WIC International Conference on Web Intelligence (WI 2003)*. 2003, pp. 702–705.

[Cla90] Frank H. Clarke. *Optimization and Nonsmooth Analysis*. Society for Industrial and Applied Mathematics, 1990.

[CV95] Corinna Cortes and Vladimir Naumovich Vapnik. "Support-Vector Networks." *Machine Learning* 20 (1995), pp. 273–297.

[Dah+12] George E. Dahl, Dong Yu, Li Deng, and Alex Acero. "Context-Dependent Pre-Trained Deep Neural Networks for Large-Vocabulary Speech Recognition." *IEEE Transactions on Audio, Speech, and Language Processing* 20.1 (2012), pp. 30–42.

[Déf+22] Alexandre Défossez, Leon Bottou, Francis Bach, and Nicolas Usunier. "A Simple Convergence Proof of Adam and Adagrad." *Transactions on Machine Learning Research* (2022).

[DHS11] John Duchi, Elad Hazan, and Yoram Singer. "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization." *Journal of Machine Learning Research* 12 (2011), pp. 2121–2159.

[Fed64] Radi Petrowitsch Fedorenko. "The Speed of Convergence of One Iterative Process." *USSR Computational Mathematics and Mathematical Physics* 4.3 (1964), pp. 227–235.

[Fed62] Radii P. Fedorenko. "A Relaxation Method for Solving Elliptic Difference Equations." *USSR Computational Mathematics and Mathematical Physics* 1.4 (1962), pp. 1092–1096.

[FH89] Evelyn Fix and J. L. Hodges. "Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties." *International Statistical Review / Revue Internationale de Statistique* 57.3 (1989), pp. 238–247.

[GKK20] Lisa Gaedke-Merzhäuser, Alena Kopanicáková, and Rolf Krause. "Multilevel Minimization for Deep Residual Networks." *CoRR* abs/2004.06196 (2020). arXiv: 2004.06196.

[GB10] Xavier Glorot and Yoshua Bengio. "Understanding the Difficulty of Training Deep Feedforward Neural Networks." *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Yee Whye Teh and Mike Titterington. Vol. 9. Proceedings of Machine Learning Research. Chia Laguna Resort, Sardinia, Italy: PMLR, 2010, pp. 249–256.

[GBB11] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. "Deep Sparse Rectifier Neural Networks." *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Geoffrey Gordon, David Dunson, and Miroslav Dudík. Vol. 15. Fort Lauderdale, FL, USA: PMLR, 2011, pp. 315–323.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[GKT23] Serge Gratton, Alena Kopanicáková, and Philippe L. Toint. "Multilevel Objective-Function-Free Optimization with an Application to Neural Networks Training." *SIAM Journal on Optimization* 33.4 (2023), pp. 2772–2800.

[Gra+08] Serge Gratton, Mélodie Mouffe, Philippe L. Toint, and Melissa Weber-Mendonça. "A recursive ℓ_∞-trust-region method for bound-constrained nonlinear optimization." *IMA Journal of Numerical Analysis* 28.4 (2008), pp. 827–861.

[GST08] Serge Gratton, Annick Sartenaer, and Philippe L. Toint. "Recursive Trust-Region Methods for Multiscale Nonlinear Optimization." *SIAM Journal on Optimization* 19.1 (2008), pp. 414–444.

[GMH13] Alex Graves, Abdel-rahman Mohamed, and Geoffrey E. Hinton. "Speech Recognition with Deep Recurrent Neural Networks." *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. Vancouver, BC, Canada: IEEE, 2013, pp. 6645–6649.

[GK09] Christian Gross and Rolf Krause. "On the Convergence of Recursive Trust-Region Methods for Multiscale Nonlinear Optimization and Applications to Nonlinear Mechanics." *SIAM Journal on Numerical Analysis* 47 (2009), pp. 3044–3069.

[GK18] Pooja Gupta and Avleen Kaur Malhi. "Using Deep Learning to Enhance Head and Neck Cancer Diagnosis and Classification." *2018 IEEE International Conference on System, Computation, Automation and Networking*. 2018, pp. 1–6.

[HR17a] Eldad Haber and Lars Ruthotto. "Stable Architectures for Deep Neural Networks." *Inverse Problems* 34.1 (2017), p. 014004.

[Hab+18] Eldad Haber, Lars Ruthotto, Elliot Holtham, and Seong-Hwan Jun. "Learning across Scales — Multiscale Methods for Convolution Neural Networks." *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*. New Orleans, Louisiana, USA: AAAI Press, 2018.

[Hac76] Wolfgang Hackbusch. *Ein iteratives Verfahren zur schnellen Auflösung elliptischer Randwertprobleme*. Tech. rep. Mathematisches Institut der Universität Köln, 1976.

[Hac78] Wolfgang Hackbusch. "A Fast Iterative Method for Solving Poisson's Equation in a General Region." *Numerical Treatment of Differential Equations*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1978, pp. 51–62.

[Hac85] Wolfgang Hackbusch. *Multi-Grid Methods and Applications*. Vol. 4. Springer Series in Computational Mathematics. Springer, Cham, 1985.

[He+16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition." *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778.

[Ho95] Tin Kam Ho. "Random Decision Forests." *Proceedings of 3rd International Conference on Document Analysis and Recognition*. Vol. 1. 1995, pp. 278–282.

[Hor91] Kurt Hornik. "Approximation Capabilities of Multilayer Feedforward Networks." *Neural Networks* 4.2 (1991), pp. 251–257.

[HR17b] Peiyun Hu and Deva Ramanan. "Finding Tiny Faces." *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 1522–1530.

[IS15] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift." *Proceedings of the 32nd International Conference on International Conference on Machine Learning*. Vol. 37. Lille, France: JMLR.org, 2015, pp. 448–456.

[Jar+09] Kevin Jarrett, Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann LeCun. "What is the Best Multi-Stage Architecture for Object Recognition?" *2009 IEEE 12th International Conference on Computer Vision*. 2009, pp. 2146–2153.

[JZ13] Rie Johnson and Tong Zhang. "Accelerating Stochastic Gradient Descent using Predictive Variance Reduction." *Advances in Neural Information Processing Systems*. Ed. by C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger. Vol. 26. Curran Associates, Inc., 2013.

[Kap+14] Olga Kapralova, John Alex, Eugene Weinstein, Pedro Moreno, and Olivier Siohan. "A Big Data Approach to Acoustic Model Training Corpus Selection." *Conference of the International Speech Communication Association (Interspeech)*. 2014.

[KKL10] Amir E. Khandani, Adlar J. Kim, and Andrew W. Lo. "Consumer Credit-Risk Models via Machine-Learning Algorithms." *Journal of Banking & Finance* 34.11 (2010), pp. 2767–2787.

[KB14] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization." *CoRR* abs/1412.6980 (2014). arXiv: `1412.6980`.

[Kop20] Alena Kopanicáková. "Multilevel minimization in trust-region framework: algorithmic and software developments." PhD thesis. Università della Svizzera italiana, 2020.

[KK23] Alena Kopanicáková and Rolf Krause. "Globally Convergent Multilevel Training of Deep Residual Networks." *SIAM Journal on Scientific Computing* 45.3 (2023), pp. 254–280.

[Kri09] Alex Krizhevsky. *Learning Multiple Layers of Features from Tiny Images*. Tech. rep. 0. Toronto, Ontario: University of Toronto, 2009.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks." *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*. Lake Tahoe, Nevada: Curran Associates Inc., 2012, pp. 1097–1105.

[LB98] Yann LeCun and Yoshua Bengio. "Convolutional Networks for Images, Speech, and Time Series." *The Handbook of Brain Theory and Neural Networks*. Cambridge, MA, USA: MIT Press, 1998, pp. 255–258.

[Lec+98] Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. "Gradient-Based Learning Applied to Document Recognition." *Proceedings of the IEEE*. Vol. 86. 11. 1998, pp. 2278–2324.

[Les+93] Moshe Leshno, Vladimir Ya. Lin, Allan Pinkus, and Shimon Schocken. "Multilayer Feedforward Networks with a Nonpolynomial Activation Function Can Approximate any Function." *Neural Networks* 6.6 (1993), pp. 861–867.

[Li+18] Hao Li, Zheng Xu, Gavin Taylor, and Tom Goldstein. "Visualizing the Loss Landscape of Neural Nets." *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. Montréal, Canada: Curran Associates Inc., 2018, pp. 6391–6401.

[MHN13] Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. "Rectifier nonlinearities improve neural network acoustic models." *Proceedings of the 30th International Conference on Machine Learning*. Vol. 28. Atlanta, Georgia, USA, 2013.

[MNT19] Matthieu Martin, Fabio Nobile, and Panagiotis Tsilifis. "A Multilevel Stochastic Gradient method for PDE-constrained Optimal Control Problems with uncertain parameters" (2019). arXiv: 1912.11900.

[MP43] Warren S. Mcculloch and Walter Pitts. "A Logical Calculus of the Ideas Immanent in Nervous Activity." *Journal of Symbolic Logic* 9.2 (1943), pp. 49–50.

[Men+13] Michael P. Menden, Francesco Iorio, Mathew Garnett, Ultan McDermott, Cyril H. Benes, Pedro J. Ballester, and Julio Saez-Rodriguez. "Machine Learning Prediction of Cancer Cell Sensitivity to Drugs Based on Genomic and Chemical Properties." *PLoS ONE* 8.4 (2013), e61318.

[NH10] Vinod Nair and Geoffrey Hinton. "Rectified Linear Units Improve Restricted Boltzmann Machines Vinod Nair." *Proceedings of ICML*. Vol. 27. 2010, pp. 807–814.

[Nas00] Stephen G. Nash. "A Multigrid Approach to Discretized Optimization Problems." *Optimization Methods and Software* 14.1-2 (2000), pp. 99–116.

[Par17] Panos Parpas. "A Multilevel Proximal Gradient Algorithm for a Class of Composite Optimization Problems." *SIAM Journal on Scientific Computing* 39.5 (2017), pp. 681–701.

[Pol64] Boris Polyak. "Some methods of speeding up the convergence of iteration methods." *USSR Computational Mathematics and Mathematical Physics* 4.5 (1964), pp. 1–17.

[Qia99] Ning Qian. "On the Momentum Term in Gradient Descent Learning Algorithms." *Neural Networks* 12.1 (1999), pp. 145–151.

[RKK18] Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. "On the Convergence of Adam and Beyond." *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. 2018.

[Red+16] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. "You Only Look Once: Unified, Real-Time Object Detection." *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 779–788.

[RM51] Herbert Robbins and Sutton Monro. "A Stochastic Approximation Method." *The Annals of Mathematical Statistics* 22.3 (1951), pp. 400–407.

[Rud16] Sebastian Ruder. "An Overview of Gradient Descent Optimization Algorithms." *CoRR* (2016).

[RH20] Lars Ruthotto and Eldad Haber. "Deep Neural Networks Motivated by Partial Differential Equations." *Journal of Mathematical Imaging and Vision* 62.3 (2020), pp. 352–364.

[SA19] Tawseef Ayoub Shaikh and Rashid Ali. "Applying Machine Learning Algorithms for Early Diagnosis and Prediction of Breast Cancer Risk." *Proceedings of 2nd International Conference on Communication, Computing and Networking*. Ed. by C. Rama Krishna, Maitreyee Dutta, and Rakesh Kumar. Singapore: Springer Singapore, 2019, pp. 589–598.

[SB14] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

[Sha08] Yair Shapira. *Matrix-Based Multigrid. Theory and Applications*. 2nd ed. Vol. 2. Numerical Methods and Algorithms. New York, NY: Springer, 2008.

[She+22] Mohammad Shehab, Laith Abualigah, Qusai Shambour, Muhannad A. Abu-Hashem, Mohd Khaled Yousef Shambour, Ahmed Izzat Alsalibi, and Amir H. Gandomi. "Machine Learning in Medical Applications: A Review of State-of-the-Art Methods." *Computers in Biology and Medicine* 145 (2022), p. 105458.

[Sho+23] Muhammad Shoaib, Babar Shah, Shaker EI-Sappagh, Akhtar Ali, Asad Ullah, Fayadh Alenezi, Tsanko Gechev, Tariq Hussain, and Farman Ali. "An Advanced Deep Learning Models-Based Plant Disease Detection: A Review of Recent Research." *Frontiers in Plant Science* 14 (2023).

[SZT19] Vishwanath A. Sindagi, Yin Zhou, and Oncel Tuzel. "MVX-Net: Multimodal VoxelNet for 3D Object Detection." *2019 International Conference on Robotics and Automation (ICRA)*. Montreal, QC, Canada: IEEE Press, 2019, pp. 7276–7282.

[Sin+12] Gajendra Singh, Ravindra Gupta, Ashish Rastogi, Mahiraj Chandel, and Riyaz Ahemad. "A Machine Learning Approach for Detection of Fraud based on SVM." *International Journal of Scientific Engineering and Technology* 1 (2012), p. 194.

[Sun+14] Yi Sun, Yuheng Chen, Xiaogang Wang, and Xiaoou Tang. "Deep Learning Face Representation by Joint Identification-Verification." *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*. Montreal, Canada: MIT Press, 2014, pp. 1988–1996.

[Sut86] Richard S. Sutton. "Two Problems with Backpropagation and Other Steepest-Descent Learning Procedures for Networks." *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*. Hillsdale, NJ: Erlbaum, 1986, pp. 823–831.

[Tai+14] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. "DeepFace: Closing the Gap to Human-Level Performance in Face Verification." *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 1701–1708.

[Tea17] Computer Vision Machine Learning Team. *An On-device Deep Neural Network for Face Detection*. 2017. URL: https://machinelearning.apple.com/research/face-detection (visited on 01/16/2024).

[Tea21] Computer Vision Machine Learning Team. *Recognizing People in Photos Through Private On-Device Machine Learning*. 2021. URL: https://machinelearning.apple.com/research/recognizing-people-photos (visited on 01/16/2024).

[TOS01] Ulrich Trottenberg, Cornelis W. Oosterlee, and Anton Schüller. *Multigrid. With guest contributions by A. Brandt, P. Oswald, K. Stüben*. Orlando, FL: Academic Press, 2001.

[UU12] Michael Ulbrich and Stefan Ulbrich. *Nichtlineare Optimierung*. Mathematik Kompakt. Basel: Birkhäuser, 2012.

[Wan22] Chenguang Wang. "Efficient Customer Segmentation in Digital Marketing Using Deep Learning with Swarm Intelligence Approach." *Information Processing & Management* 59.6 (2022), p. 103085.

[WG10] Zaiwen Wen and Donald Goldfarb. "A Line Search Multigrid Method for Large-Scale Nonlinear Optimization." *SIAM Journal on Optimization* 20.3 (2010), pp. 1478–1503.

[Wu+19] Chao-Yuan Wu, Ross B. Girshick, Kaiming He, Christoph Feichtenhofer, and Philipp Krähenbühl. "A Multigrid Method for Efficiently Training Video Models." *CoRR* abs/1912.00998 (2019). arXiv: 1912.00998.

[WRS10] Jionglin Wu, Jason Roy, and Walter F. Stewart. "Prediction Modeling Using EHR Data: Challenges, Strategies, and a Comparison of Machine Learning Approaches." *Medical Care* 48.6 (2010), pp. 106–113.

[Zha+22] Yu Zhang, James Qin, Daniel S. Park, Wei Han, Chung-Cheng Chiu, Ruoming Pang, Quoc V. Le, and Yonghui Wu. "Pushing the Limits of Semi-Supervised Learning for Automatic Speech Recognition" (2022). arXiv: 2010.10504.

Wissenschaftlicher Werdegang

12/2019 – 03/2025	**Wissenschaftliche Mitarbeiterin** am Fachbereich Mathematik der Technischen Universität Darmstadt, Arbeitsgruppe Nichtlineare Optimierung
10/2017 – 10/2019	**Master of Science** in Mathematik an der Technischen Universität Darmstadt
10/2014 – 10/2017	**Bachelor of Science** in Mathematik (bilingual) an der Technischen Universität Darmstadt