POLITECNICO DI TORINO
Repository ISTITUZIONALE

ATE-SG: alternate through the epochs stochastic gradient for multi-task neural networks

*Original*
ATE-SG: alternate through the epochs stochastic gradient for multi-task neural networks / Bellavia, S., Della Santa, F., Papini, A.. - In: OPTIMIZATION METHODS & SOFTWARE. - ISSN 1055-6788. - (2026), pp. 1-33. [10.1080/10556788.2026.2659033]

*Availability:*
This version is available at: 11583/3010559 since: 2026-05-05T12:58:14Z

*Publisher:*
Taylor & Francis

*Published*
DOI:10.1080/10556788.2026.2659033

*Terms of use:*

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

*Publisher copyright*

(Article begins on next page)

29 November 2026

Optimization Methods and Software

ISSN: 1055-6788 (Print) 1029-4937 (Online) Journal homepage: www.tandfonline.com/journals/goms20

# ATE-SG: alternate through the epochs stochastic gradient for multi-task neural networks

Stefania Bellavia, Francesco Della Santa & Alessandra Papini

**To cite this article:** Stefania Bellavia, Francesco Della Santa & Alessandra Papini (28 Apr 2026): ATE-SG: alternate through the epochs stochastic gradient for multi-task neural networks, Optimization Methods and Software, DOI: 10.1080/10556788.2026.2659033

**To link to this article:** https://doi.org/10.1080/10556788.2026.2659033

© 2026 The Author(s). Published by Informa UK Limited, trading as Taylor & Francis Group.

Published online: 28 Apr 2026.

Submit your article to this journal

Article views: 31

View related articles

View Crossmark data

Full Terms & Conditions of access and use can be found at
https://www.tandfonline.com/action/journalInformation?journalCode=goms20

OPTIMIZATION METHODS & SOFTWARE
https://doi.org/10.1080/10556788.2026.2659033

Taylor & Francis
by informa

OPEN ACCESS Check for updates

# ATE-SG: alternate through the epochs stochastic gradient for multi-task neural networks

Stefania Bellavia[a,b], Francesco Della Santa[c,b] and Alessandra Papini[a,b]

[a]Dipartimento di Ingegneria Industriale, Università degli Studi di Firenze, Firenze, Italy; [b]Gruppo Nazionale per il Calcolo Scientifico, Istituto Nazionale di Alta Matematica, Rome, Italy; [c]Dipartimento di Scienze Matematiche, Politecnico di Torino, Turin, Italy

**ABSTRACT**

This paper introduces novel alternate training procedures for hard-parameter sharing Multi-Task Neural Networks (MTNNs). Traditional MTNN training faces challenges in managing conflicting loss gradients, often yielding sub-optimal performance. The proposed alternate training method updates shared and task-specific weights alternately through the epochs, exploiting the multi-head architecture of the model. This approach reduces computational costs per epoch and memory requirements. Convergence properties similar to those of the classical stochastic gradient method are established. Empirical experiments demonstrate enhanced training regularization and reduced computational demands. In summary, our alternate training procedures offer a promising advancement for the training of hard-parameter sharing MTNNs.

**ARTICLE HISTORY**
Received 26 March 2025
Accepted 7 April 2026

**KEYWORDS**
Alternate stochastic gradient; multi-task learning; neural networks; deep learning

**2020 MATHEMATICS SUBJECT CLASSIFICATIONS**
49M37; 65K05; 68T05; 68W40; 90C15

## 1. Introduction

Multi-Task Learning (MTL) consists of jointly learning multiple tasks rather than individually, in such a way that the knowledge obtained by learning a task can be exploited for learning other tasks, hopefully improving the generalization performance of all the tasks at hand [31]. For the case of Neural Networks (NNs), MTL is approached by building NN architectures characterized by multiple output layers, one for each task, connected to (at least) one shared input layer; then, Multi-Task NNs (MTNNs) are characterized by an inherent layer sharing property and, historically, can be divided into *hard-parameter sharing* MTNNs and *soft-parameter sharing* MTNNs [27]. In this work, we focus on the hard-parameter sharing case, i.e. on MTNNs characterized by a so-called multi-head design architecture, where a first block of shared layers connects the inputs to multiple task-specific blocks of layers (see Figure 1). Summarizing, the idea behind these MTNNs is to build a shared encoder that branches out into multiple task-specific decoders [27] (e.g. see [4,11,13,26]).

The NN model is generally trained to simultaneously make predictions for all tasks, where the loss is a weighted sum of all the task-specific loss functions (*aggregate loss*

**CONTACT** Stefania Bellavia stefania.bellavia@unifi.it

© 2026 The Author(s). Published by Informa UK Limited, trading as Taylor & Francis Group.
This is an Open Access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited. The terms on which this article has been published allow the posting of the Accepted Manuscript in a repository by the author(s) or with their consent.

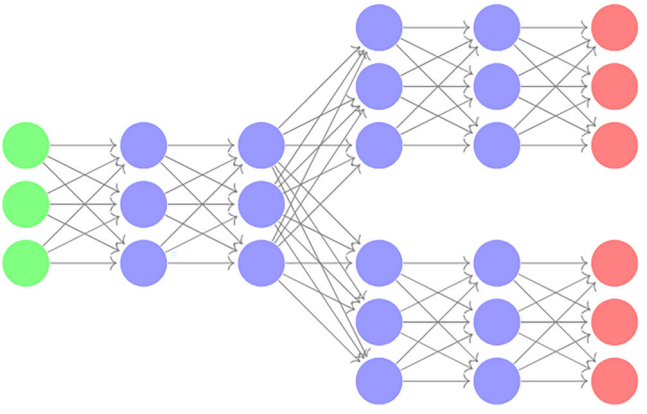

**Figure 1.** Example of MTNN with 2 tasks. On the left half of the figure, there is $N_0$ (with input layer in green); on the right half of the figure, there are $N_1$ and $N_2$ (with output layers in red).

*function*) [31]. This approach presents several difficulties. Descent directions of different loss functions at the current iterate may conflict and the direction used to update the NN parameters may produce an increase of a single loss despite the aggregate loss function decreases. Then, this approach often yields lower performance than its corresponding single-task counterparts [22]. Different approaches have been proposed by researchers to overcome these difficulties and obtain more robust procedures. These approaches can be divided into two main types: *i*) modify the training procedure by considering also the gradients of the task-specific losses, and/or by training all or a subset of the weights with respect to single tasks, e.g. see [15,19,21,22,25]; *ii*) adaptively choose a good setting of the weights in the aggregate loss function for a good balance of the magnitudes of the task-specific losses, e.g. see [5,7,8,14,16,20]. Among these, papers [16,20] deal with a general multiobjective problem, of which the MTNN training is a special case, and aim at approximating the entire Pareto front. In order to adaptively compute the aggregate loss weights these approaches require the solution of a minimization subproblem at each iteration.

Further, we also recall the Block Coordinate Descent (BCD) method, where the parameters of the MTL model are partitioned and updated with respect to the corresponding subproblems (see [31] and the references there-in); especially in non-Deep Learning MTL problems, such a kind of approach is useful to reduce the complexity of the optimization learning problem (e.g. see [17]).

### 1.1. Contribution

In this paper we propose a novel approach for training a generic hard-parameter sharing MTNN. Though inspired both by the approaches based on task-specific gradients and by the BCD method, it is distinguished by the following characteristics.

- We always aim at reducing the aggregate loss function, but rather than alternating among stochastic gradient steps for a single task as in [15,21,22] we alternate stochastic estimators of the gradient with respect to the shared NN parameters and stochastic estimators of the gradient with respect to the task-specific NN parameters. The shared

and task-specific parameters are then updated alternately; in this way, when the task-specific weights are updated, all specific-task losses are reduced simultaneously. This important property depends on the multi-head architecture that characterizes the type of MTNN we consider in this work. We stress that we do not need to solve optimization subproblems as in the multi-gradient method in [16,20].

- Our approach is theoretically well-founded. We consider the case of nonconvex, differentiable functions and analyse both the case where shared and task-specific parameters are alternately updated at each iteration, and the case where we keep updating the shared (task-specific) parameters of the NN for one or more epochs, and then alternate, i.e. we alternate through the epochs. We show that the convergence properties of the classical stochastic gradient method are maintained. We carried out the convergence analysis assuming Lipschitz continuity of the partial gradients with respect to both shared and task-specific parameters. This yields to the use of potentially larger sequences of learning rates.
- The alternate training we present is a new stochastic gradient training procedure for hard-parameter sharing MTNNs, that compared with the classical stochastic gradient approach both reduces memory requirements and computational costs, and allows to use different sequences of learning rates in the shared and task-specific updating phases. Further, numerical experiments show that our approach regularizes the training phase.
- One of our objectives was to devise easy-to-apply procedures and to provide a ready-to-use version of our proposed training routine for MTNNs (see Appendix), implementable within the most used Deep Learning frameworks in literature (e.g. see [2,6]). We also stress that our alternate framework can be combined with any other optimizer. As an example, in Section 4 we provide results obtained using Adam in place of the Stochastic Gradient.

The content of this work is organized as follows. We start by introducing the MTNNs and analysing the properties of gradients computed with respect to shared or task-specific weights (Section 2). Then, we describe the new alternate training method, and discuss its convergence properties (Section 3). After that, a section of numerical experiments (Section 4) illustrates a comparison between MTNNs trained classically and trained using the proposed alternate training. Finally, conclusions about advantages and properties of the proposed method are summarized (Section 5).

## 2. Multiple-task neural networks

A hard-parameter sharing MTNN for a MTL problem made of $K \in \mathbb{N}$ tasks is an NN model with an architecture characterized by: one main block of layers, called *trunk*, connected to the input layer(s); $K$ independent blocks of layers, called *branches*, connected to the last layer of the trunk. The last layer of the $k$th branch is the output layer of the MTNN for the $k$th task, for each $k = 1, \ldots, K$.

The main idea behind this type of architecture (see Figure 1) is that the trunk encodes the inputs, learning the new representation characterized by features important for all the $K$ tasks. Then, each branch reads this representation (i.e. the output of the last trunk's layer) and decodes it independently from other branches, learning its task. In other words, we

can interpret the output of a hard-parameter sharing MTNN as a concatenation of $K$ independent decoding operations applied to an encoding operation applied to the same input signals.

In the following, we formalize the definition of hard-parameter sharing MTNN and the observation about the outputs of an MTNN. From now on, for simplicity, we take for granted that when we talk about MTNNs we are considering a hard-parameter sharing MTNN as in the next definition.

**Definition 2.1 (Hard Parameter Sharing Multi-Task Neural Network):** Let N be an NN with characterizing function $\widehat{\boldsymbol{F}} : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^{m_1} \times \cdots \times \mathbb{R}^{m_K}$, where the domain $\mathbb{R}^n \times \mathbb{R}^p$ represents the Cartesian product between the space of the NN inputs ( $\mathbb{R}^n$) and the space of the NN trainable parameters ( $\mathbb{R}^p$). Then, N is a *hard parameter sharing multi-task NN* (MTNN) with respect to $K$ tasks in $\mathbb{R}^{m_1}, \ldots, \mathbb{R}^{m_K}$, respectively, if N's architecture is characterized by $K+1$ smaller NNs, $\mathrm{N}_0, \ldots, \mathrm{N}_K$, such that:

(1) the characterizing function of $\mathrm{N}_0$ is a function $\widehat{\boldsymbol{F}}_0 : \mathbb{R}^n \times \mathbb{R}^{p_0} \to \mathbb{R}^{m_0}$;
(2) the characterizing function of $\mathrm{N}_k$ is a function $\widehat{\boldsymbol{F}}_k : \mathbb{R}^{m_0} \times \mathbb{R}^{p_k} \to \mathbb{R}^{m_k}$, for each $k = 1, \ldots, K$;
(3) N is obtained by connecting the output layer of $\mathrm{N}_0$ to the first layers of $\mathrm{N}_1, \ldots, \mathrm{N}_K$.

In particular, we define $\mathrm{N}_0$ as the architecture's block shared by the $K$ tasks, while $\mathrm{N}_k$ is defined as the architecture's block specific of task $k$, for each $k = 1, \ldots, K$.

Let $\boldsymbol{w}_k \in \mathbb{R}^{p_k}$ be the vector of trainable parameters (i.e. weights and biases) of $\mathrm{N}_k$, for each $k = 0, \ldots, K$: the parameters in $\boldsymbol{w}_0$ are defined as *shared parameters* of N, while the parameters in $\boldsymbol{w}_k$ of $\mathrm{N}_k$ are defined as *task-specific parameters* of N with respect to task $k$, for each $k = 1, \ldots, K$. Then, $\sum_{k=0}^{K} p_k = p$ and $\widehat{\boldsymbol{F}}$ is such that

$$\widehat{\boldsymbol{F}}(\boldsymbol{x}; \boldsymbol{w}) = \begin{bmatrix} \widehat{\boldsymbol{F}}_1(\widehat{\boldsymbol{F}}_0(\boldsymbol{x}; \boldsymbol{w}_0); \boldsymbol{w}_1) \\ \vdots \\ \widehat{\boldsymbol{F}}_K(\widehat{\boldsymbol{F}}_0(\boldsymbol{x}; \boldsymbol{w}_0); \boldsymbol{w}_K) \end{bmatrix}$$

for each $\boldsymbol{x} \in \mathbb{R}^n$, where $\boldsymbol{w} = (\boldsymbol{w}_0^T, \ldots, \boldsymbol{w}_K^T)^T \in \mathbb{R}^p$.

### 2.1. Loss differentiation and multiple tasks

In MTL, the loss function of a model typically is a weighted sum of different losses, evaluated with respect to each task. In Notation 2.1 below, we introduce the symbols and the formalization we use to describe the aggregated loss and the task-specific losses of an MTNN, as well as the batches of corresponding input-output pairs. In this work, we always assume that the task-specific loss functions of the MTNN, and hence the aggregated loss also, are differentiable functions with respect to all the NN parameters.

**Notation 2.1 (Aggregated and task-specific batches and losses):** Let N be an MTNN as in Definition 2.1 and let $\mathcal{B} \subset \mathbb{R}^n \times \mathbb{R}^m$ be a batch of input-output pairs for N, where $m = \sum_{k=1}^{K} m_k$; a batch is always assumed finite and non-empty. Then, we introduce the following notations:

(1) we denote by $\mathcal{B}^k \subset \mathbb{R}^n \times \mathbb{R}^{m_k}$ the batch of input-output pairs related to the $k^{\text{th}}$ task of N obtained from the batch $\mathcal{B}$; i.e.:

$$\mathcal{B}^k := \left\{(\boldsymbol{x}, \boldsymbol{y}_k) \in \mathbb{R}^n \times \mathbb{R}^{m_k} \mid (\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{B}\right\}, \quad k = 1, \dots, K,$$

where $\boldsymbol{y}_k = (y_1^{(k)}, \dots, y_{m_k}^{(k)})^T$ and $\boldsymbol{y} = (\boldsymbol{y}_1^T, \dots, \boldsymbol{y}_K^T)^T \in \mathbb{R}^m$;

(2) we denote by $\ell_k : \mathrm{P}^*(\mathbb{R}^n \times \mathbb{R}^{m_k}) \times \mathbb{R}^{p_0+p_k} \to \mathbb{R}$ a loss function defined for the task $k$ of N, for each $k = 1, \dots, K$, where $\mathrm{P}^*(A)$ denotes the set of finite and non-empty subsets of $A$, for each set $A$. For example, assuming $\ell_k$ as the *Mean Square Error* (MSE), we have that

$$\ell_k(\mathcal{B}^k; \boldsymbol{w}_0, \boldsymbol{w}_k) = \frac{1}{|\mathcal{B}^k|} \sum_{(\boldsymbol{x}, \boldsymbol{y}_k) \in \mathcal{B}^k} \left(\widehat{\boldsymbol{F}}_k(\widehat{\boldsymbol{F}}_0(\boldsymbol{x}; \boldsymbol{w}_0); \boldsymbol{w}_k) - \boldsymbol{y}_k\right)^2,$$

for each batch $\mathcal{B}^k \subset \mathbb{R}^n \times \mathbb{R}^{m_k}$;

(3) we denote by $\ell : \mathrm{P}^*(\mathbb{R}^n \times \mathbb{R}^m) \times \mathbb{R}^p \to \mathbb{R}$ the aggregated loss function of N, such that $\ell$ is a linear combination with positive coefficients of the task-specific losses $\ell_1, \dots, \ell_K$:

$$\ell(\mathcal{B}; \boldsymbol{w}) := \sum_{k=1}^{K} \lambda_k \ell_k(\mathcal{B}^k; \boldsymbol{w}_0, \boldsymbol{w}_k), \quad \text{with } \lambda_1, \dots, \lambda_k \in \mathbb{R}^+. \tag{1}$$

Our alternate training method takes inspiration both from the BCD method and from those methods based on exploiting the task-specific gradients (see Section 1). Indeed, it is easily seen that the gradient of $\ell$ with respect to the task-specific parameters of task $k$ is equal to the gradient of $\lambda_k \ell_k$, i.e.:

$$\nabla_{\boldsymbol{w}_k} \ell(\mathcal{B}; \boldsymbol{w}) = \lambda_k \nabla_{\boldsymbol{w}_k} \ell_k(\mathcal{B}^k; \boldsymbol{w}_0, \boldsymbol{w}_k),$$

for each $k = 1, \dots, K$, and for any batch $\mathcal{B} \in \mathrm{P}^*(\mathbb{R}^n \times \mathbb{R}^m)$. Then, once observed this characteristic, it is almost immediate to notice also that the gradient of $\ell(\mathcal{B}; \boldsymbol{w})$ with respect to all the task-specific parameters is just a concatenation of the $K$ gradients of the task-specific losses with respect to their own task-specific parameters (and multiplied by the coefficients), namely:

$$\nabla_{\boldsymbol{w}_{\text{ts}}} \ell(\mathcal{B}; \boldsymbol{w}) = \begin{bmatrix} \lambda_1 \nabla_{\boldsymbol{w}_1} \ell_1(\mathcal{B}^1; \boldsymbol{w}_0, \boldsymbol{w}_1) \\ \vdots \\ \lambda_K \nabla_{\boldsymbol{w}_K} \ell_K(\mathcal{B}^K; \boldsymbol{w}_0, \boldsymbol{w}_K) \end{bmatrix},$$

for each $\mathcal{B} \in \mathrm{P}^*(\mathbb{R}^n \times \mathbb{R}^m)$, where $\boldsymbol{w}_{\text{ts}} \in \mathbb{R}^{p_{\text{ts}}}$ is the vector denoting the concatenation of all the task-specific parameters; i.e.:

$$\boldsymbol{w}_{\text{ts}} := (\boldsymbol{w}_1^T, \dots, \boldsymbol{w}_K^T)^T \in \mathbb{R}^{p_{\text{ts}}}, \quad \text{with } p_{\text{ts}} = \sum_{k=1}^{K} p_k.$$

As a consequence of these results, we state in Proposition 2.1 the descent properties of the two anti-gradients of $\ell$ with respect to the shared parameters and the task-specific

parameters, respectively, for any batch $\mathcal{B}$. The proof of the proposition is omitted, since it is trivial.

**Proposition 2.1 (Gradients and Descent Directions):** *Let* N *be an MTNN as in Definition 2.1 and let* $\ell, \ell_1, \ldots, \ell_K$ *be the losses in* (1). *Let* $\boldsymbol{w} \in \mathbb{R}^p$ *be the vector of trainable parameters of* N; *specifically,* $\boldsymbol{w}$ *is the concatenation of the shared parameters* $\boldsymbol{w}_0 \in \mathbb{R}^{p_0}$ *and the task-specific parameters* $\boldsymbol{w}_{\rm ts} \in \mathbb{R}^{p_{\rm ts}}$*:*

$$\boldsymbol{w} = \begin{bmatrix} \boldsymbol{w}_0 \\ \boldsymbol{w}_{\rm ts} \end{bmatrix}.$$

*Then, for each fixed batch* $\mathcal{B}$ *the vectors*

$$\begin{bmatrix} -\nabla_{\boldsymbol{w}_0} \ell(\mathcal{B}; \boldsymbol{w}) \\ \boldsymbol{0} \end{bmatrix}, \quad \begin{bmatrix} \boldsymbol{0} \\ -\nabla_{\boldsymbol{w}_{\rm ts}} \ell(\mathcal{B}; \boldsymbol{w}) \end{bmatrix} \in \mathbb{R}^p \tag{2}$$

*are descent directions for the loss* $\ell(\mathcal{B}; \boldsymbol{w})$ *at* $\boldsymbol{w}$, *if* $\nabla_{\boldsymbol{w}_0} \ell(\mathcal{B}; \boldsymbol{w}) \neq \boldsymbol{0} \in \mathbb{R}^{p_0}$ *and* $\nabla_{\boldsymbol{w}_{\rm ts}} \ell(\mathcal{B}; \boldsymbol{w}) \neq \boldsymbol{0} \in \mathbb{R}^{p_{\rm ts}}$, *respectively. Moreover,* $(\boldsymbol{0}^T, -\nabla^T_{\boldsymbol{w}_k} \ell_k(\mathcal{B}^k; \boldsymbol{w}_0, \boldsymbol{w}_k))^T$ *(subvector of the second vector in* (2)*) is also a descent direction for* $\ell_k(\mathcal{B}^k; \boldsymbol{w}_0, \boldsymbol{w}_k)$, *for each* $k = 1, \ldots, K$.

We conclude this section remarking the different implications of the two descent directions (2):

- $\boldsymbol{w}_0$-based direction: it is a descent direction for $\ell(\mathcal{B}; \boldsymbol{w})$ that updates the shared parameters only. It allows reducing the loss with respect to the weights that affect all the tasks, then there are no guarantees of reducing all the task-specific losses too;
- $\boldsymbol{w}_{\rm ts}$-based direction: is a descent direction for $\ell(\mathcal{B}; \boldsymbol{w})$ but also for all the task-specific losses $\ell_1, \ldots, \ell_K$, and it updates the task-specific parameters only. It allows to reduce both the main loss and the task-specific losses with respect to the weights $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_K$ that affect only the losses $\ell_1, \ldots, \ell_K$, respectively (i.e. only the corresponding tasks).

Starting from these properties, in the next section we formulate new alternate training procedures and prove their convergence properties assuming Lipschitz continuity of the gradients. Specifically, there exist shared Lipschitz constants $L_0$, $L_0^{\rm ts}$, and task-specific Lipschitz constants $L_{\rm ts}$, $L_{\rm ts}^0$, such that for any $\boldsymbol{w} = (\boldsymbol{w}_0^T, \boldsymbol{w}_{\rm ts}^T)^T$, $\boldsymbol{d} = (\boldsymbol{d}_0^T, \boldsymbol{d}_{\rm ts}^T)^T \in \mathbb{R}^p$:

$$\|\nabla_{\boldsymbol{w}_0} \ell(\cdot\,; \boldsymbol{w}) - \nabla_{\boldsymbol{w}_0} \ell(\cdot\,; \boldsymbol{w} + \boldsymbol{d})\| \leq L_0 \|\boldsymbol{d_0}\|, \quad \text{when } \boldsymbol{d}_{\rm ts} = \boldsymbol{0}, \tag{3a}$$

$$\|\nabla_{\boldsymbol{w}_0} \ell(\cdot\,; \boldsymbol{w}) - \nabla_{\boldsymbol{w}_0} \ell(\cdot\,; \boldsymbol{w} + \boldsymbol{d})\| \leq L_0^{\rm ts} \|\boldsymbol{d}_{\rm ts}\|, \quad \text{when } \boldsymbol{d}_0 = \boldsymbol{0}, \tag{3b}$$

$$\|\nabla_{\boldsymbol{w}_{\rm ts}} \ell(\cdot\,; \boldsymbol{w}) - \nabla \boldsymbol{w}_{\rm ts} \ell(\cdot\,; \boldsymbol{w} + \boldsymbol{d})\| \leq L_{\rm ts} \|\boldsymbol{d_{\rm ts}}\|, \quad \text{when } \boldsymbol{d}_0 = \boldsymbol{0}, \tag{3c}$$

$$\|\nabla_{\boldsymbol{w}_{\rm ts}} \ell(\cdot\,; \boldsymbol{w}) - \nabla \boldsymbol{w}_{\rm ts} \ell(\cdot\,; \boldsymbol{w} + \boldsymbol{d})\| \leq L_{\rm ts}^0 \|\boldsymbol{d_0}\|, \quad \text{when } \boldsymbol{d}_{\rm ts} = \boldsymbol{0}. \tag{3d}$$

It can be easily seen that $\max\{L_0, L_0^{\rm ts}, L_{\rm ts}, L_{\rm ts}^0\} \leq L$ (see e.g. [29])), where $L$ denotes the standard Lipschitz constant of the gradient:

$$\|\nabla \ell(\cdot\,; \boldsymbol{w}) - \nabla \ell(\cdot\,; \boldsymbol{w} + \boldsymbol{d})\| \leq L \|\boldsymbol{d}\|.$$

Where not explicitly specified, as above, gradients are taken with respect to all the parameters $\boldsymbol{w}$.

## 3. Alternate training

The idea of an alternate training can be realized in many different ways. In this work, we define alternate training methods which are modifications of a classical Stochastic Gradient (SG) procedure.

Proposition 2.1 defines two alternative descent directions for the loss function. Devising a training procedure based on the alternate usage of these directions or their stochastic estimators may yield the following practical advantages.

(1) *Alternate training for reduced memory usage.* The advantage of the alternate training concerning memory is almost evident in situations where $p_0 \not\approx p$ and $p_{\text{ts}} \not\approx 0$ (or vice-versa). Indeed, at each step, an alternate procedure requires the storage of a gradient $\nabla_{\boldsymbol{w}_0}\ell$ with dimension $p_0$ or a gradient $\nabla_{\boldsymbol{w}_{\text{ts}}}\ell$ with dimension $p_{\text{ts}}$; in both cases, the gradient has dimension smaller than the 'global' gradient computed with respect to all the NN's weights (i.e. with dimension $p = p_0 + p_{\text{ts}}$).
(2) *Alternate training for reduced computational cost.* We observe that the computation of $\nabla_{\boldsymbol{w}_{\text{ts}}}\ell$ is cheaper than the computation of both $\nabla_{\boldsymbol{w}_0}\ell$ and $\nabla_{\boldsymbol{w}}\ell$, since only the task-specific layers of the NN are involved during the back-propagation (see e.g. [24] and Section 4); then, at equal number of epochs, training the model with a procedure that alternates between $\nabla_{\boldsymbol{w}_0}\ell$ and $\nabla_{\boldsymbol{w}_{\text{ts}}}\ell$ to update the current iterate is less costly than the classical stochastic gradient which always uses $\nabla_{\boldsymbol{w}}\ell$.
(3) *Alternate training for regularization.* Relying on the $\boldsymbol{w}_{\text{ts}}$-based direction partially reduces the typical difficulty of MTL models about selecting a direction that is not a descent direction for all the tasks, and therefore reduces the possibility of having an increase of some losses despite the overall objective function decreases (see Section 1). This interesting regularization property of the training phase is illustrated in the numerical experiments of Section 4.

In the next subsections, we describe a couple of alternate training strategies. To analyse their convergence properties, we will make use of the following theorem.

**Theorem 3.1 ([23]):** *Let $U_i, \beta_i, \xi_i, \rho_i$ be nonnegative $\mathcal{A}_i$-measurable random variables such that*

$$\mathbb{E}[U_{i+1}|\mathcal{A}_i] \leq (1+\beta_i)U_i + \xi_i - \rho_i \quad i = 0, 1, 2 \ldots.$$

*Then, on the set $\{\sum_i \beta_i < \infty, \sum_i \xi_i < \infty\}$, $U_i$ converges almost surely to a random variable $U$ and $\sum_i \rho_i < \infty$ almost surely.*

### 3.1. Simple alternate training

The Simple Alternate Training (SAT) method is a two steps iterative process that alternately updates $\boldsymbol{w}_0$ and $\boldsymbol{w}_{\text{ts}}$ in an MTNN. The variable $\boldsymbol{w}_0$ is updated at each iteration using a stochastic estimator of $\nabla_{\boldsymbol{w}_0}\ell$, while $\boldsymbol{w}_{\text{ts}}$ is updated using a stochastic estimator of $\nabla_{\boldsymbol{w}_{\text{ts}}}\ell$. In what follows we denote the training set as $\mathcal{T}$. We name SAT-SG this procedure in order to emphasize the relationship with SG, and we describe one iteration in Algorithm 3.1.

Convergence properties of SAT-SG are proven in Theorem 3.3; both cases of constant and diminishing learning rates are considered.

**Algorithm 3.1** SAT-SG - Simple Alternate Training SG for MTNNs

**Data:** $(\boldsymbol{w}_0, \boldsymbol{w}_{\rm ts}) = \boldsymbol{w}^{(i)}$ (current iterate for the trainable parameters), $\mathcal{T}$ (training set), $B$ (mini-batch size), $\eta_i^0$ and $\eta_i^{\rm ts}$ (learning rates), $\ell = \ell(\mathcal{B}; \boldsymbol{w})$ (loss function).
**Iteration** $i$**:**

1: Sample randomly a batch $\mathcal{B}_1$ from $\mathcal{T}$ s.t. $|\mathcal{B}_1| = B$
2: $\boldsymbol{w}_0 \leftarrow \boldsymbol{w}_0 - \eta_i^0 \nabla_{\boldsymbol{w}_0} \ell(\mathcal{B}_1; \boldsymbol{w}^{(i)})$
3: $\boldsymbol{z}^{(i)} \leftarrow (\boldsymbol{w}_0, \boldsymbol{w}_{\rm ts})$
4: Sample randomly a batch $\mathcal{B}_2$ from $\mathcal{T}$ s.t. $|\mathcal{B}_2| = B$
5: $\boldsymbol{w}_{\rm ts} \leftarrow \boldsymbol{w}_{\rm ts} - \eta_i^{\rm ts} \nabla_{\boldsymbol{w}_{\rm ts}} \ell(\mathcal{B}_2; \boldsymbol{z}^{(i)})$
6: $\boldsymbol{w}^{(i+1)} \leftarrow (\boldsymbol{w}_0, \boldsymbol{w}_{\rm ts})$
7: **return** $\boldsymbol{w}^{(i+1)}$ (updated iterate for MTNN's weights)

**Notation 3.1:** From now on, to shorten the notation, we will omit to explicitly indicate the dependence of the loss function from the batch of data when this coincides with the whole training set $\mathcal{T}$, namely we will write $\ell(\boldsymbol{w})$ for $\ell(\mathcal{T}; \boldsymbol{w})$, and $\nabla \ell(\boldsymbol{w})$ for $\nabla \ell(\mathcal{T}; \boldsymbol{w})$.

We first provide in Lemma 3.2 an upper bound for the expected conditioned value of the loss function $\ell(\boldsymbol{w})$ for sufficiently small learning rates $\eta_i^0$ and $\eta_i^{\rm ts}$.

**Lemma 3.2 (SAT-SG):** *Let $\{\boldsymbol{w}^{(i)}\}_{i\geq 0}, \{\boldsymbol{z}^{(i)}\}_{i\geq 0} \subset \mathbb{R}^p$ be two sequences generated by the SAT-SG method and $\{\eta_i^0\}_{i\geq 0}$ $\{\eta_i^{\rm ts}\}_{i\geq 0}$, be the sequences of the learning rates used. Let $\mathcal{A}_i$ denote the $\sigma$-algebra induced by $\boldsymbol{w}^{(0)}, \boldsymbol{z}^{(0)}, \boldsymbol{w}^{(1)}, \boldsymbol{z}^{(1)}, \ldots, \boldsymbol{w}^{(i)}$ and $\mathcal{A}_{i+\frac{1}{2}}$ the $\sigma$-algebra induced by $\boldsymbol{w}^{(0)}, \boldsymbol{z}^{(0)}, \boldsymbol{w}^{(1)}, \boldsymbol{z}^{(1)}, \ldots, \boldsymbol{w}^{(i)}, \boldsymbol{z}^{(i)}$.*

*Assume that the batches $\mathcal{B}_1$ and $\mathcal{B}_2$ are sampled randomly and uniformly, that there exist two positive constants $M_1$ and $M_2$ such that*

$$\mathbb{E}[\|\nabla_{\boldsymbol{w}_0} \ell(\mathcal{B}; \boldsymbol{w}^{(i)})\|^2 | \mathcal{A}_i] \leq M_2 \|\nabla_{\boldsymbol{w}_0} \ell(\boldsymbol{w}^{(i)})\|^2 + M_1 \tag{4}$$

$$\mathbb{E}[\|\nabla_{\boldsymbol{w}_{\rm ts}} \ell(\mathcal{B}; \boldsymbol{z}^{(i)})\|^2 | \mathcal{A}_{i+\frac{1}{2}}] \leq M_2 \|\nabla_{\boldsymbol{w}_{\rm ts}} \ell(\boldsymbol{z}^{(i)})\|^2 + M_1 \tag{5}$$

*for any batch of data $\mathcal{B}$, and that $\nabla_{\boldsymbol{w}_0} \ell(\cdot\,; \boldsymbol{w})$ and $\nabla_{\boldsymbol{w}_{\rm ts}} \ell(\cdot\,; \boldsymbol{w})$ satisfy the Lipschitz continuity conditions (3a).*

*Then the following properties hold:*

$$\mathbb{E}[\ell(\boldsymbol{z}^{(i)}) | \mathcal{A}_i] \leq \ell(\boldsymbol{w}^{(i)}) - \eta_i^0 \, G_i^0 \, \|\nabla_{\boldsymbol{w}_0} \ell(\boldsymbol{w}^{(i)})\|^2 + \frac{L_0}{2} (\eta_i^0)^2 M_1, \tag{6}$$

*with $G_i^0 = 1 - \frac{L_0}{2} \eta_i^0 M_2 > 0$ for any $i \geq 0$ and $0 < \eta_i^0 < \frac{1}{L_0} \frac{2}{M_2}$,*

$$\mathbb{E}[\ell(\boldsymbol{w}^{(i+1)}) | \mathcal{A}_{i+\frac{1}{2}}] \leq \ell(\boldsymbol{z}^{(i)}) - \eta_i^{\rm ts} \, G_i^{\rm ts} \, \|\nabla_{\boldsymbol{w}_{\rm ts}} \ell(\boldsymbol{z}^{(i)})\|^2 + \frac{L_{\rm ts}}{2} (\eta_i^{\rm ts})^2 M_1, \tag{7}$$

*with $G_i^{\rm ts} = 1 - \frac{L_{\rm ts}}{2} \eta_i^{\rm ts} M_2 > 0$ for any $i \geq 0$ and $0 < \eta_i^{\rm ts} < \frac{1}{L_{\rm ts}} \frac{2}{M_2}$.*

*Moreover, for sufficiently small values of the learning rates, e.g.*

$$0 < \eta_i^0 < \min\left\{\frac{1-C}{L_0}\frac{2}{M_2}, \frac{C}{L_{\text{ts}}^0}\right\} \quad \text{and} \quad 0 < \eta_i^{\text{ts}} < \min\left\{\frac{1-C}{L_{\text{ts}}}\frac{2}{M_2}, \frac{C}{L_{\text{ts}}^0}\right\} \tag{8}$$

*for any* $C \in (0,1)$ *and* $i \geq 0$*, the following properties hold:*

$$G_i^0 > C, \quad G_i^{\text{ts}} > C, \tag{9}$$

$$\mathbb{E}[\ell(\boldsymbol{w}^{(i+1)})|\mathcal{A}_i] \leq \ell(\boldsymbol{w}^{(i)}) - \eta_i^{\min} G_i \|\nabla\ell(\boldsymbol{w}^{(i)})\|^2 + M_1 \frac{L_0 (\eta_i^0)^2 + L_{\text{ts}} (\eta_i^{\text{ts}})^2}{2} \tag{10}$$

*with* $G_i = C - L_{\text{ts}}^0 \eta_i^{\max} > 0$, $\eta_i^{\min} = \min\{\eta_i^0, \eta_i^{\text{ts}}\}$ *and* $\eta_i^{\max} = \max\{\eta_i^0, \eta_i^{\text{ts}}\}$.

***Proof:*** First, we consider the updating of the shared parameters $\boldsymbol{w}_0$ (see steps 2 and 3 of Algorithm 3.1), and use Taylor's theorem and inequality (3a.a) to obtain

$$\ell(\boldsymbol{z}^{(i)}) \leq \ell(\boldsymbol{w}^{(i)}) + \eta_i^0 \nabla\ell(\boldsymbol{w}^{(i)})^T \begin{bmatrix} -\nabla_{\boldsymbol{w}_0}\ell(\mathcal{B}_1; \boldsymbol{w}^{(i)}) \\ \boldsymbol{0} \end{bmatrix} + \frac{L_0}{2} (\eta_i^0)^2 \|\nabla_{\boldsymbol{w}_0}\ell(\mathcal{B}_1; \boldsymbol{w}^{(i)})\|^2.$$

Then, taking the conditioned expected value on both sides, exploiting assumption (4) and the fact that the subsampled gradient $\nabla_{\boldsymbol{w}_0}\ell(\mathcal{B}_1; \boldsymbol{w}^{(i)})$ is an unbiased estimator, we have:

$$\begin{aligned}
\mathbb{E}[\ell(\boldsymbol{z}^{(i)})|\mathcal{A}_i] &\leq \ell(\boldsymbol{w}^{(i)}) - \eta_i^0 \nabla_{\boldsymbol{w}_0}\ell(\boldsymbol{w}^{(i)})^T \mathbb{E}[\nabla_{\boldsymbol{w}_0}\ell(\mathcal{B}_1; \boldsymbol{w}^{(i)})|\mathcal{A}_i] \\
&\quad + \frac{L_0}{2} (\eta_i^0)^2 \mathbb{E}[\|\nabla_{\boldsymbol{w}_0}\ell(\mathcal{B}_1; \boldsymbol{w}^{(i)})\|^2 |\mathcal{A}_i] \\
&\leq \ell(\boldsymbol{w}^{(i)}) - \eta_i^0 \|\nabla_{\boldsymbol{w}_0}\ell(\boldsymbol{w}^{(i)})\|^2 + \frac{L_0}{2} (\eta_i^0)^2 (M_2 \|\nabla_{\boldsymbol{w}_0}\ell(\boldsymbol{w}^{(i)})\|^2 + M_1),
\end{aligned}$$

which, by setting $G_i^0 = 1 - \frac{L_0}{2} \eta_i^0 M_2$, reduces to (6), with $G_i^0 > 0$ for sufficiently small values of $\eta_i^0$, namely $\eta_i^0 < \frac{1}{L_0}\frac{2}{M_2}$.

Similarly, after updating the task-specific parameters $\boldsymbol{w}_{\text{ts}}$ we have

$$\ell(\boldsymbol{w}^{(i+1)}) \leq \ell(\boldsymbol{z}^{(i)}) + \eta_i^{\text{ts}} \nabla\ell(\boldsymbol{z}^{(i)})^T \begin{bmatrix} \boldsymbol{0} \\ -\nabla_{\boldsymbol{w}_{\text{ts}}}\ell(\mathcal{B}_2; \boldsymbol{z}^{(i)}) \end{bmatrix} + \frac{L_{\text{ts}}}{2} (\eta_i^{\text{ts}})^2 \|\nabla_{\boldsymbol{w}_{\text{ts}}}\ell(\mathcal{B}_2; \boldsymbol{z}^{(i)})\|^2,$$

so that, proceeding as before in conditioned expected value and using assumption (5), we get (7) if $\eta_i^{\text{ts}} < \frac{1}{L_{\text{ts}}}\frac{2}{M_2}$, so that $G_i^{\text{ts}} = 1 - \frac{L_{\text{ts}}}{2} \eta_i^{\text{ts}} M_2 > 0$.

Further, (9) trivially follows from (8), since $G_i^0$ and $G_i^{\text{ts}}$ are bounded below by a positive constant $C \in (0,1)$ for sufficiently small values of $\eta_i^0$ and $\eta_i^{\text{ts}}$, respectively $\eta_i^0 \leq \frac{1-C}{L_0}\frac{2}{M_2}$ and $\eta_i^{\text{ts}} \leq \frac{1-C}{L_{\text{ts}}}\frac{2}{M_2}$.

Now, using inequality (3a.d) and the definition of $\boldsymbol{z}^{(i)}$, we observe that

$$\begin{aligned}
\|\nabla_{\boldsymbol{w}_{\text{ts}}}\ell(\boldsymbol{z}^{(i)})\|^2 &= \|\nabla_{\boldsymbol{w}_{\text{ts}}}\ell(\boldsymbol{w}^{(i)}) + \nabla_{\boldsymbol{w}_{\text{ts}}}\ell(\boldsymbol{z}^{(i)}) - \nabla_{\boldsymbol{w}_{\text{ts}}}\ell(\boldsymbol{w}^{(i)})\|^2 \\
&= \|\nabla_{\boldsymbol{w}_{\text{ts}}}\ell(\boldsymbol{w}^{(i)})\|^2 + \|\nabla_{\boldsymbol{w}_{\text{ts}}}\ell(\boldsymbol{z}^{(i)}) - \nabla_{\boldsymbol{w}_{\text{ts}}}\ell(\boldsymbol{w}^{(i)})\|^2 \\
&\quad + 2\nabla_{\boldsymbol{w}_{\text{ts}}}\ell(\boldsymbol{w}^{(i)})^T \left(\nabla_{\boldsymbol{w}_{\text{ts}}}\ell(\boldsymbol{z}^{(i)}) - \nabla_{\boldsymbol{w}_{\text{ts}}}\ell(\boldsymbol{w}^{(i)})\right) \\
&\geq \|\nabla_{\boldsymbol{w}_{\text{ts}}}\ell(\boldsymbol{w}^{(i)})\|^2 - 2\|\nabla_{\boldsymbol{w}_{\text{ts}}}\ell(\boldsymbol{w}^{(i)})\| \, \|\nabla_{\boldsymbol{w}_{\text{ts}}}\ell(\boldsymbol{w}^{(i)}) - \nabla_{\boldsymbol{w}_{\text{ts}}}\ell(\boldsymbol{z}^{(i)})\|
\end{aligned}$$

$$\geq \|\nabla_{\boldsymbol{w}_{\rm ts}}\ell(\boldsymbol{w}^{(i)})\|^2 - 2L_{\rm ts}^0\,\eta_i^0\,\|\nabla_{\boldsymbol{w}_{\rm ts}}\ell(\boldsymbol{w}^{(i)})\|\,\|\nabla_{\boldsymbol{w}_0}\ell(\mathcal{B}_1;\boldsymbol{w}^{(i)})\|;$$

then using this last inequality in (7) we have

$$\begin{aligned}\mathbb{E}[\ell(\boldsymbol{w}^{(i+1)})|\mathcal{A}_{i+\frac{1}{2}}] \leq\ & \ell(\boldsymbol{z}^{(i)}) - \eta_i^{\rm ts}G_i^{\rm ts}\|\nabla_{\boldsymbol{w}_{\rm ts}}\ell(\boldsymbol{w}^{(\boldsymbol{i})})\|^2 \\ & + 2L_{\rm ts}^0\eta_i^0\eta_i^{\rm ts}G_i^{\rm ts}\|\nabla_{\boldsymbol{w}_{\rm ts}}\ell(\boldsymbol{w}^{(i)})\|\|\nabla_{\boldsymbol{w}_0}\ell(\mathcal{B}_1;\boldsymbol{w}^{(i)})\| \\ & + \frac{L_{\rm ts}}{2}(\eta_i^{\rm ts})^2M_1. \end{aligned} \tag{11}$$

Finally, recalling that

$$\mathbb{E}[\ell(\boldsymbol{w}^{(i+1)})|\mathcal{A}_i] = \mathbb{E}[\mathbb{E}[\ell(\boldsymbol{w}^{(i+1)})|\mathcal{A}_{i+\frac{1}{2}}]\,|\,\mathcal{A}_i],$$

combining (6) and (11), using (9) and the upper bound $G_i^{\rm ts} < 1$, we have

$$\begin{aligned}&\mathbb{E}[\ell(\boldsymbol{w}^{(i+1)})|\mathcal{A}_i] \\ &\leq \mathbb{E}[\ell(\boldsymbol{z}^{(i)})|\mathcal{A}_i] - \eta_i^{\rm ts}G_i^{\rm ts}\|\nabla_{\boldsymbol{w}_{\rm ts}}\ell(\boldsymbol{w}^{(\boldsymbol{i})})\|^2 \\ &\quad + 2L_{\rm ts}^0\,\eta_i^0\eta_i^{\rm ts}G_i^{\rm ts}\,\|\nabla_{\boldsymbol{w}_{\rm ts}}\ell(\boldsymbol{w}^{(i)})\|\,\|\nabla_{\boldsymbol{w}_0}\ell(\boldsymbol{w}^{(i)})\| + \frac{L_{\rm ts}}{2}\,(\eta_i^{\rm ts})^2\,M_1 \\ &\leq \ell(\boldsymbol{w}^{(i)}) - \eta_i^0\,G_i^0\,\|\nabla_{\boldsymbol{w}_0}\ell(\boldsymbol{w}^{(\boldsymbol{i})})\|^2 + \frac{L_0}{2}\,(\eta_i^0)^2\,M_1 - \eta_i^{\rm ts}\,G_i^{\rm ts}\,\|\nabla_{\boldsymbol{w}_{\rm ts}}\ell(\boldsymbol{w}^{(\boldsymbol{i})})\|^2 \\ &\quad + 2L_{\rm ts}^0\,\eta_i^0\eta_i^{\rm ts}G_i^{\rm ts}\,\|\nabla_{\boldsymbol{w}_{\rm ts}}\ell(\boldsymbol{w}^{(i)})\|\,\|\nabla_{\boldsymbol{w}_0}\ell(\boldsymbol{w}^{(i)})\| + \frac{L_{\rm ts}}{2}\,(\eta_i^{\rm ts})^2\,M_1 \\ &\leq \ell(\boldsymbol{w}^{(i)}) - C\,(\eta_i^0\,\|\nabla_{\boldsymbol{w}_0}\ell(\boldsymbol{w}^{(\boldsymbol{i})})\|^2 + \eta_i^{\rm ts}\,\|\nabla_{\boldsymbol{w}_{\rm ts}}\ell(\boldsymbol{w}^{(\boldsymbol{i})})\|^2) \\ &\quad + 2L_{\rm ts}^0\,\eta_i^0\eta_i^{\rm ts}\,\|\nabla_{\boldsymbol{w}_{\rm ts}}\ell(\boldsymbol{w}^{(i)})\|\,\|\nabla_{\boldsymbol{w}_0}\ell(\boldsymbol{w}^{(i)})\| + M_1\frac{L_0\,(\eta_i^0)^2\,+L_{\rm ts}\,(\eta_i^{\rm ts})^2}{2}. \end{aligned}$$

From the last inequality, using the relations $\eta_i^{\min} \leq \eta_i^0, \eta_i^{\rm ts} \leq \eta_i^{\max}$ and $2ab \leq a^2 + b^2$, we obtain

$$\begin{aligned}\mathbb{E}[\ell(\boldsymbol{w}^{(i+1)})|\mathcal{A}_i] \leq\ & \ell(\boldsymbol{w}^{(i)}) - C\,\eta_i^{\min}\,(\,\|\nabla_{\boldsymbol{w}_0}\ell(\boldsymbol{w}^{(\boldsymbol{i})})\|^2 + \|\nabla_{\boldsymbol{w}_{\rm ts}}\ell(\boldsymbol{w}^{(\boldsymbol{i})})\|^2) \\ & + L_{\rm ts}^0\,\eta_i^{\max}\eta_i^{\min}\,(\|\nabla_{\boldsymbol{w}_0}\ell(\boldsymbol{w}^{(\boldsymbol{i})})\|^2 + \|\nabla_{\boldsymbol{w}_{\rm ts}}\ell(\boldsymbol{w}^{(\boldsymbol{i})})\|^2) \\ & + M_1\frac{L_0\,(\eta_i^0)^2\,+L_{\rm ts}\,(\eta_i^{\rm ts})^2}{2}, \end{aligned}$$

which reduces to (10) by setting $G_i = C - L_{\rm ts}^0\,\eta_i^{\max}$, with $G_i > 0$ for $\eta_i^{\max} < \frac{C}{L_{\rm ts}^0}$. ■

Inequality (10) in Lemma 3.2 paths the way for the convergence Theorem 3.3 below, in the spirit of classical convergence results on stochastic gradient methods (see Theorems 4.8 and 4.10 in [3]).

**Theorem 3.3 (SAT-SG convergence):** *Assume that the hypotheses in Lemma 3.2 hold and that the loss function $\ell$ is bounded below by $\ell_{low}$. Let $\{\boldsymbol{w}^{(i)}\}_{i\geq 0}$ be a sequence of iterates generated by Algorithm 3.1 with learning rates $\{\eta_i^0\}_{i\geq 0}$ and $\{\eta_i^{\rm ts}\}_{i\geq 0}$ satisfying (8). Then,*

(i) *for sequences of learning rates such that*

$$\sum_{i=0}^{\infty} \eta_i^{\min} = \infty, \tag{12a}$$

$$\sum_{i=0}^{\infty} (\eta_i^{\max})^2 < \infty, \tag{12b}$$

*the following holds:*

$$\liminf_{i\to\infty} ||\nabla \ell(\boldsymbol{w}^{(i)})||^2 = 0 \quad \text{a.s.;} \tag{13}$$

(ii) *for fixed learning rates, $\eta_i^0 = \eta_0$ and $\eta_i^{\text{ts}} = \eta_{\text{ts}}$ for all $i \geq 0$, the average-squared gradients of $\ell$ satisfy the following inequality at any iteration $J \in \mathbb{N}$:*

$$\mathbb{E}[\frac{1}{J+1}\sum_{i=0}^{J} \|\nabla \ell(\boldsymbol{w}^{(i)})\|^2] \leq M_1 \frac{L_0\, \eta_0^2 + L_{\text{ts}}\, \eta_{\text{ts}}^2}{2\, G\, \min\{\eta_0, \eta_{\text{ts}}\}} + \frac{\ell(\boldsymbol{w}^{(0)}) - \ell_{low}}{(J+1)\, G\, \min\{\eta_0, \eta_{\text{ts}}\}} \tag{14}$$

$$\xrightarrow{J\to\infty} M_1 \frac{L_0\, \eta_0^2 + L_{\text{ts}}\, \eta_{\text{ts}}^2}{2\, G\, \min\{\eta_0, \eta_{\text{ts}}\}}, \tag{15}$$

*with $G = C - L_{\text{ts}}^0 \max\{\eta_0, \eta_{\text{ts}}\} \geq C/2 > 0$, for $C \in (0,1)$ and $\max\{\eta_0, \eta_{\text{ts}}\} \leq \frac{1}{2}\frac{C}{L_{\text{ts}}^0}$.*

***Proof:*** In case (i) we are going to show that

$$\sum_{i=0}^{\infty} \eta_i^{\min} \|\nabla \ell(\boldsymbol{w}^{(i)})\|^2 < \infty \quad \text{a.s.,} \tag{16}$$

from which (13) follows using assumption (12a). To this aim, we first rewrite (10) as follows:

$$\begin{aligned}
&\mathbb{E}[\ell(\boldsymbol{w}^{(i+1)})|\mathcal{A}_i] - \ell_{low} \\
&\quad \leq \ell(\boldsymbol{w}^{(i)}) - \ell_{low} - G_i \eta_i^{\min} \|\nabla \ell(\boldsymbol{w}^{(i)})\|^2 + M_1 \frac{L_0\, (\eta_i^0)^2 + L_{\text{ts}}\, (\eta_i^{\text{ts}})^2}{2}.
\end{aligned}$$

Then, using Theorem 3.1 with $U_i = \ell(\boldsymbol{w}^{(i)}) - \ell_{low} > 0$, $\xi_i = M_1 \frac{L_0\, (\eta_i^0)^2 + L_{\text{ts}}\, (\eta_i^{\text{ts}})^2}{2}$, $\beta_i = 0$, and $\rho_i = \eta_i^{\min} G_i \|\nabla \ell(\boldsymbol{w}^{(i)})\|^2$, we have that the sequence $\{\ell(\boldsymbol{w}^{(i)})\}$ is convergent almost surely and

$$\sum_{i=0}^{\infty} \eta_i^{\min} G_i \|\nabla \ell(\boldsymbol{w}^{(i)})\|^2 < \infty \quad \text{a.s.}$$

Since by assumption (12b) $\eta_i^{\max} \to 0$, $G_i = C - L_{\text{ts}}^0\, \eta_i^{\max}$ is positive and bounded away from zero for all $i$ sufficiently large, so that also (16) holds.

In case (ii), since $\eta_i^0 = \eta_0$, $\eta_i^{\rm ts} = \eta_{\rm ts}$ for all $i$, and $G_i = G = C - L_{\rm ts}^0 \max\{\eta_0, \eta_{\rm ts}\} > 0$ by (8), taking the total expectation of (10) we get

$$\mathbb{E}[\ell(\boldsymbol{w}^{(i+1)})] - \mathbb{E}[\ell(\boldsymbol{w}^{(i)})] \leq -\min\{\eta_0, \eta_{\rm ts}\} G \mathbb{E}[\|\nabla \ell(\boldsymbol{w}^{(i)})\|^2] + M_1 \frac{L_0\, \eta_0^2 + L_{\rm ts}\, \eta_{\rm ts}^2}{2}.$$

Then summing up for $i = 0$ to $i = J$ and recalling that $\ell$ is bounded from below by $\ell_{low}$ we obtain

$$\begin{aligned} \ell_{low} - \ell(\boldsymbol{w}^{(0)}) &\leq \mathbb{E}[\ell(\boldsymbol{w}^{(J+1)})] - \ell(\boldsymbol{w}^{(0)}) \\ &\leq -\min\{\eta_0, \eta_{\rm ts}\}\, G \sum_{i=0}^{J} E[\|\nabla \ell(\boldsymbol{w}^{(i)})\|^2] + M_1 \sum_{i=0}^{J} \frac{L_0\, \eta_0^2 + L_{\rm ts}\, \eta_{\rm ts}^2}{2}. \end{aligned}$$

By rearranging the previous inequality we finally have

$$\mathbb{E}[\sum_{i=0}^{J} \|\nabla \ell(\boldsymbol{w}^{(i)})\|^2] \leq (J+1) M_1 \frac{L_0\, \eta_0^2 + L_{\rm ts}\, \eta_{\rm ts}^2}{2\, G\, \min\{\eta_0, \eta_{\rm ts}\}} + \frac{\ell(\boldsymbol{w}^{(0)}) - \ell_{low}}{G\, \min\{\eta_0, \eta_{\rm ts}\}},$$

from which (14) follows by dividing for $J+1$. ■

Let us assume to set $C = \frac{1}{2}$ in Theorem 3.3 and compare (14) with the corresponding result for the SG method [3, Th. 4.8]. In (14) we have on the right-hand side the additional factor $1/G < 4$ rather than a factor 2 as in the corresponding result for the SG method. On the other hand, we expect to be able to take larger learning rates as the Lipschitz constants involved are smaller than the Lipschitz constant of the full gradient. Regarding the optimality gap, in addition to potentially smaller Lipschitz constants, also the constant $M_1$ in (4)–(5) may be smaller than the corresponding constant used to bound the expected value of $\|\nabla \ell(\boldsymbol{w})\|^2$ with SG.

We finally observe that SAT-SG can also be seen as a block stochastic gradient method and has some similarities with the approach proposed in [30]. We stress that in [30] the same batch $\mathcal{B} = \mathcal{B}_1 = \mathcal{B}_2$ is used for each block and, using our notation, the convergence analysis is carried out under the stronger assumption

$$\|\mathbb{E}[\nabla_{\boldsymbol{w}_{\rm ts}} \ell(\mathcal{B}; \boldsymbol{z}^{(\boldsymbol{i})}) - \nabla_{\boldsymbol{w}_{\rm ts}} \ell(\boldsymbol{z}^{(\boldsymbol{i})}) | \mathcal{A}_i]\| \leq A\, \eta_i^{\max},$$

where $A$ is a positive constant, at any iteration $i$. Further, in [30] it is proved that such an assumption is verified in case the batch $\mathcal{B}$ is a singleton and the following Lipschitz conditions hold:

$$\|\nabla_{\boldsymbol{w}_0} \ell(\cdot\,;\, \boldsymbol{w}) - \nabla_{\boldsymbol{w}_0} \ell(\cdot\,;\, \boldsymbol{w} + \boldsymbol{d})\| \leq L \|\boldsymbol{d}\| \tag{17}$$

$$\|\nabla_{\boldsymbol{w}_{\rm ts}} \ell(\cdot\,;\, \boldsymbol{w}) - \nabla_{\boldsymbol{w}_{\rm ts}} \ell(\cdot\,;\, \boldsymbol{w} + \boldsymbol{d})\| \leq L \|\boldsymbol{d}\|. \tag{18}$$

### 3.2. Alternate training through the epochs

From the practical point of view, it can be more effective to alternate the training procedure through the epochs, since within each epoch it is assumed that the model sees all the

available training samples; moreover, concerning compatibility with Deep Learning frameworks, it is easier to develop a training procedure that alternatively switches the trainable weights ($\boldsymbol{w}_0$ and $\boldsymbol{w}_{\text{ts}}$) at given epochs instead at each mini-batch.[1]

With an alternate training through the epochs, the weights of the NN are alternatively updated for some epochs with respect to $\boldsymbol{w}_0$, and for some other epochs with respect to $\boldsymbol{w}_{\text{ts}}$. We can outline the procedure as follows:

- train the MTNN with SG for $E_0 \in \mathbb{N}$ epochs, with respect to the shared parameters $\boldsymbol{w}_0$;
- train the MTNN with SG for $E_{\text{ts}} \in \mathbb{N}$ epochs, with respect to the task-specific parameters $\boldsymbol{w}_{\text{ts}}$;
- repeat until convergence (or a stopping criterion is satisfied).

So a complete cycle of alternate training (see Algorithm 3.2) consists of $E_0 + E_{\text{ts}}$ epochs and $t \cdot (E_0 + E_{\text{ts}})$ updating iterations, where $t \in \mathbb{N}$ is the number of used batches per epoch, tipically the integer part of $|\mathcal{T}|/B$ for a given batch size $B$. We name Alternate Training through the Epochs SG (ATE-SG) this procedure. In the rest of this section, we denote by $e$ the cycle counter and by $i$ the step (or iteration) counter, namely each cycle $e$ starts at the iterate $\boldsymbol{w}^{(i_{\text{ts}})}$ with $i_{\text{ts}} = (E_0 + E_{\text{ts}})\, e\, t$.

To analyse the convergence properties of ATE-SG it is convenient to split the set of iteration indexes into two subsets, $I_0$ and $I_{\text{ts}}$, corresponding to the shared phase and to the task-specific phase, respectively. In other words, gradients with respect to the shared (task-specific) parameters are evaluated at iterate $\boldsymbol{w}^{(i)}$ when $i \in I_0$ ($I_{\text{ts}}$).

We now prove the almost sure convergence of ATE-SG assuming to use diminishing sequences of learning rates $\{\eta_i\}_{i\in I_0}$ and $\{\eta_i\}_{i\in I_{\text{ts}}}$. We first provide, in the next Lemma, a '*lim-inf* ' result for $||\nabla_{\boldsymbol{w}_0}\ell(\boldsymbol{w}^{(i)})||$ along the shared iterations and for $||\nabla_{\boldsymbol{w}_{ts}}\ell(\boldsymbol{w}^{(i)})||$ along the task-specific iterations.

**Lemma 3.4 (ATE-SG):** *Given $\boldsymbol{w}^{(0)} \in \mathbb{R}^p$, let $\{\boldsymbol{w}^{(i)}\}_{i\geq 0} \subset \mathbb{R}^p$ be the sequence generated by iterating ATE-SG Algorithm 3.2, with diminishing sequences of learning rates $\{\eta_i\}_{i\in I_0}$ and $\{\eta_i\}_{i\in I_{\text{ts}}}$ such that*

$$\sum_{i\in I_0} \eta_i = \infty, \quad \sum_{i\in I_{\text{ts}}} \eta_i = \infty, \tag{19a}$$

$$\sum_{i=0}^{\infty} \eta_i^2 < \infty, \tag{19b}$$

*where $I_0$ and $I_{\text{ts}}$ are the sets of iteration indexes corresponding to the shared phase and to the task-specific phase, respectively. Assume that the loss function $\ell$ is bounded below by $\ell_{low}$, that $\nabla_{\boldsymbol{w}_0}\ell(\cdot\,;\boldsymbol{w})$ and $\nabla_{\boldsymbol{w}_{\text{ts}}}\ell(\cdot\,;\boldsymbol{w})$ satisfy the Lipschitz continuity conditions (3a), and there exist two positive constants $M_1$ and $M_2$ such that for any batch $\mathcal{B}$*

$$\mathbb{E}[\|\nabla_{\boldsymbol{w}_0}\ell(\mathcal{B};\boldsymbol{w}^{(i)})\|^2|\mathcal{A}_i] \leq M_2\,\|\nabla_{\boldsymbol{w}_0}\ell(\boldsymbol{w}^{(i)})\|^2 + M_1, \quad \text{for } i \in I_0,$$

$$\mathbb{E}[\|\nabla_{\boldsymbol{w}_{\text{ts}}}\ell(\mathcal{B};\boldsymbol{w}^{(i)})\|^2|\mathcal{A}_i] \leq M_2\,\|\nabla_{\boldsymbol{w}_{\text{ts}}}\ell(\boldsymbol{w}^{(i)})\|^2 + M_1, \quad \text{for } i \in I_{\text{ts}},$$

**Algorithm 3.2** ATE-SG - Alternate Training through the Epochs SG for MTNNs

**Data:** $(\boldsymbol{w}_0^{(i_{\rm ts})}, \boldsymbol{w}_{\rm ts}^{(i_{\rm ts})}) = \boldsymbol{w}^{(i_{\rm ts})}$ (current iterate for the trainable parameters), $\mathcal{T}$ (training set), $B$ (mini-batch size), $t$ (number of mini-batches per epoch), $E_0, E_{\rm ts}$ (number of epochs for alternate training), $\{\eta_i, i = i_{\rm ts}, \ldots, i_{\rm ts} + (E_0 + E_{\rm ts})t - 1\}$ (learning rates), $\ell = \ell(\mathcal{B}; \boldsymbol{w})$ (loss function).

**Cycle** $e$**:**

$i \leftarrow i_{\rm ts}$ (iteration counter, here $i = (E_0 + E_{\rm ts})\, e\, t$)
**for** $e_0 = 1, 2, \ldots, E_0$ (epochs counter, shared phase) **do**
  **for** $\tau = 0, 1, \ldots, t-1$ **do**
    Sample randomly and uniformly a batch $\mathcal{B}_\tau$ from $\mathcal{T}$ s.t. $|\mathcal{B}_\tau| = B$
    $\boldsymbol{w}_0^{(i+1)} \leftarrow \boldsymbol{w}_0^{(i)} - \eta_i \nabla_{\boldsymbol{w}_0} \ell(\mathcal{B}_\tau; \boldsymbol{w}^{(i)})$
    $i \leftarrow i + 1$
    $\boldsymbol{w}^{(i)} \leftarrow (\boldsymbol{w}_0^{(i)}, \boldsymbol{w}_{\rm ts}^{(i_{\rm ts})})$
  **end for**
**end for**
$i_0 \leftarrow i$ (here $i = i_{\rm ts} + tE_0$)
**for** $e_{\rm ts} = 1, 2, \ldots, E_{\rm ts}$ (epochs counter, task-specific phase) **do**
  **for** $\tau = 0, 1, \ldots, t-1$ **do**
    Sample randomly and uniformly a batch $\mathcal{B}_\tau$ from $\mathcal{T}$ s.t. $|\mathcal{B}_\tau| = B$
    $\boldsymbol{w}_{\rm ts}^{(i+1)} \leftarrow \boldsymbol{w}_{\rm ts}^{(i)} - \eta_i \nabla_{\boldsymbol{w}_{\rm ts}} \ell(\mathcal{B}_\tau; \boldsymbol{w}^{(i)})$
    $i \leftarrow i + 1$
    $\boldsymbol{w}^{(i)} \leftarrow (\boldsymbol{w}_0^{(i_0)}, \boldsymbol{w}_{\rm ts}^{(i)})$
  **end for**
**end for**
$i_{\rm ts} \leftarrow i$ (here $i = i_0 + tE_{\rm ts}$)
**return** $\boldsymbol{w}^{(i_{\rm ts})}$ (updated iterate after $E_0 + E_{\rm ts}$ epochs)

*where $\mathcal{A}_i$ denotes the $\sigma$-algebra induced by $\boldsymbol{w}^{(0)}, \ldots, \boldsymbol{w}^{(i)}$. Then*

$$\liminf_{i \in I_0} ||\nabla_{\boldsymbol{w}_0} \ell(\boldsymbol{w}^{(i)})||^2 = \liminf_{i \in I_{\rm ts}} ||\nabla_{\boldsymbol{w}_{ts}} \ell(\boldsymbol{w}^{(i)})||^2 = 0 \quad \text{a.s.}$$

***Proof:*** As in the proof of Lemma 3.2, we start by considering the updating of the shared parameters $\boldsymbol{w}_0$ and the inequality:

$$\ell(\boldsymbol{w}^{(i+1)}) \leq \ell(\boldsymbol{w}^{(i)}) + \eta_i \nabla \ell(\boldsymbol{w}^{(i)})^T \begin{bmatrix} -\nabla_{\boldsymbol{w}_0} \ell(\mathcal{B}_\tau; \boldsymbol{w}^{(i)}) \\ \mathbf{0} \end{bmatrix} + \frac{L_0}{2} \eta_i^2 \, \|\nabla_{\boldsymbol{w}_0} \ell(\mathcal{B}_\tau; \boldsymbol{w}^{(i)})\|^2,$$

which holds for $i \in I_0$, and in conditioned expected value becomes

$$\begin{aligned} \mathbb{E}[\ell(\boldsymbol{w}^{(i+1)})|\mathcal{A}_i] &\leq \ell(\boldsymbol{w}^{(i)}) - \eta_i \, \|\nabla_{\boldsymbol{w}_0} \ell(\boldsymbol{w}^{(i)})\|^2 + \frac{L_0}{2} \eta_i^2 \, (M_2 \, \|\nabla_{\boldsymbol{w}_0} \ell(\boldsymbol{w}^{(i)})\|^2 + M_1) \\ &= \ell(\boldsymbol{w}^{(i)}) - \eta_i \, G_i \, \|\nabla_{\boldsymbol{w}_0} \ell(\boldsymbol{w}^{(i)})\|^2 + \frac{L_0}{2} \eta_i^2 \, M_1, \end{aligned}$$

with $G_i = 1 - \frac{L_0}{2} \eta_i M_2 \geq G_0 = 1 - \frac{L_0}{2} \eta_0 M_2 > 0$ for sufficiently small values of $\eta_0$.

Similarly, in the task-specific case we have

$$\mathbb{E}[\ell(\boldsymbol{w}^{(i+1)})|\mathcal{A}_i] \leq \ell(\boldsymbol{w}^{(i)}) - \eta_i\, G_i\, \|\nabla_{\boldsymbol{w}_{\text{ts}}}\ell(\boldsymbol{w}^{(i)})\|^2 + \frac{L_{\text{ts}}}{2}\,\eta_i^2\, M_1,$$

with $G_i = 1 - \frac{L_{\text{ts}}}{2}\eta_i M_2 \geq G_{tE_0} = 1 - \frac{L_{\text{ts}}}{2}\eta_{tE_0} M_2 > 0$ for any $i \in I_{\text{ts}}$ and sufficiently small values of the initial learning rate, $\eta_{tE_0}$, of the task-specific updating phase.

Then, setting $U_i = \ell(\boldsymbol{w}^{(i)}) - \ell_{low} > 0$ and $\beta_i = 0$ for any $i \geq 0$, $\xi_i = \frac{L_{\text{ts}}}{2}\,\eta_i^2\, M_1$ and $\rho_i = \eta_i G_i \|\nabla_{\boldsymbol{w}_{\text{ts}}}\ell(\boldsymbol{w}^{(i)})\|^2$ for $i \in I_{\text{ts}}, \xi_i = \frac{L_0}{2}\,\eta_i^2\, M_1$ and $\rho_i = \eta_i G_i \|\nabla_{\boldsymbol{w}_0}\ell(\boldsymbol{w}^{(i)})\|^2$ for $i \in I_0$, since assumption (19b) ensures $\sum_{i=0}^{\infty}\xi_i < \infty$, by Theorem 3.1 the sequence $\{\ell(\boldsymbol{w}^{(i)})\}$ is convergent almost surely, and

$$\sum_{i=0}^{\infty}\rho_i = \sum_{i\in I_{\text{ts}}}\eta_i G_i \|\nabla_{\boldsymbol{w}_{\text{ts}}}\ell(\boldsymbol{w}^{(i)})\|^2 + \sum_{i\in I_0}\eta_i G_i \|\nabla_{\boldsymbol{w}_0}\ell(\boldsymbol{w}^{(i)})\|^2 < \infty \quad \text{a.s.} \tag{20}$$

Recall now that for $\eta_0$ and $\eta_{tE_0}$ sufficiently small, $0 < \min\{G_0, G_{tE_0}\} \leq G_i < 1$ for all $i \geq 0$. Hence from (20) it also holds

$$\sum_{i\in I_{\text{ts}}}\eta_i \|\nabla_{\boldsymbol{w}_{\text{ts}}}\ell(\boldsymbol{w}^{(i)})\|^2 + \sum_{i\in I_0}\eta_i \|\nabla_{\boldsymbol{w}_0}\ell(\boldsymbol{w}^{(i)})\|^2 < \infty \quad \text{a.s.} \tag{21}$$

Finally, the thesis follows from assumption (19a). ■

We are now ready to provide our main result, where we prove a '*lim-inf* ' result for the norm of the gradient of $\ell$.

**Theorem 3.5 (ATE-SG convergence):** *Under the assumptions of Lemma 3.4, and using the same notations, let us further assume that there exists $M > 0$ such that*

$$\|\nabla\ell(\mathcal{B}; \boldsymbol{w}^{(i)})\|^2 \leq M \quad \textit{for any } \mathcal{B} \subset \mathcal{T}, \tag{22}$$

*and that*

$$\sum_{i=0}^{\infty}\eta_i^{\min} = \infty, \tag{23}$$

*where $\eta_i^{\min}$ denotes the minimum learning rate within one cycle of Algorithm 3.2:*

$$\eta_i^{\min} = \min_{j\in I_e}\{\eta_j\} \quad \text{for } i \in I_e, \tag{24}$$

*with $I_e = \{i \in \mathbb{N} : (E_0 + E_{\text{ts}})et \leq i \leq (E_0 + E_{\text{ts}})\,(e+1)t - 1\}$. Then*

$$\liminf_{i\to\infty} ||\nabla\ell(\boldsymbol{w}^{(i)})||^2 = 0 \quad \textit{a.s.} \tag{25}$$

***Proof:*** Similarly to Lemma 3.4, we are going to show that

$$\sum_{i=0}^{\infty}\eta_i^{\min}\|\nabla\ell(\boldsymbol{w}^{(i)})\|^2 < \infty \quad \text{a.s.},$$

then the thesis follows by assumption (23). To this aim, we first rewrite the above series as

$$\sum_{i=0}^{\infty}\eta_i^{\min}\|\nabla\ell(\boldsymbol{w}^{(i)})\|^2 = \sum_{i\in I_{\text{ts}}}\eta_i^{\min}\|\nabla_{\boldsymbol{w}_{\text{ts}}}\ell(\boldsymbol{w}^{(i)})\|^2 + \sum_{i\in I_0}\eta_i^{\min}\|\nabla_{\boldsymbol{w}_0}\ell(\boldsymbol{w}^{(i)})\|^2$$

$$+ \sum_{i \in I_{\rm ts}} \eta_i^{\min} \|\nabla_{\boldsymbol{w}_0} \ell(\boldsymbol{w}^{(i)})\|^2 + \sum_{i \in I_0} \eta_i^{\min} \|\nabla_{\boldsymbol{w}_{\rm ts}} \ell(\boldsymbol{w}^{(i)})\|^2. \tag{26}$$

Since $\eta_i^{\min} \leq \eta_i$, recalling (21) we only need to show that the last two terms in (26) are bounded. Both terms can be treated in a very similar way, hence we will prove the result in detail only for the last one. Moreover, for simplicity's sake, we consider the case $E_0 = E_{\rm ts} = 1$, where one cycle of ATE-SG consists of $t$ SG steps taken first with respect to the shared parameters $\boldsymbol{w}_0$, and then with respect to the task-specific parameters $\boldsymbol{w}_{\rm ts}$. This allows us to write down and exploit the following representations of the sets $I_0$ and $I_{\rm ts}$:

$$I_0 = [0 : t-1] \cup [2\,t : 3\,t-1] \cup \ldots = \bigcup_{e=0}^{\infty} [2\,e\,t : (2\,e+1)\,t-1] \tag{27}$$

$$I_{\rm ts} = [t : 2\,t-1] \cup [3\,t : 4\,t-1] \cup \ldots = \bigcup_{e=0}^{\infty} [(2\,e+1)\,t : 2(e+1)t-1]. \tag{28}$$

Nonetheless, we remark that with a bit more technicalities the proof can be extended to the general case $E_0 \geq 1$ and $E_{\rm ts} \geq 1$.

So, let us consider the last term in (26), which by using (27) can be rewritten as follows:

$$\begin{aligned}\sum_{i \in I_0} \eta_i^{\min} \|\nabla_{\boldsymbol{w}_{\rm ts}} \ell(\boldsymbol{w}^{(i)})\|^2 &= \sum_{e=0}^{\infty} \sum_{\tau=0}^{t-1} \eta_{2et+\tau}^{\min} \|\nabla_{\boldsymbol{w}_{\rm ts}} \ell(\boldsymbol{w}^{(2et+\tau)})\|^2 \\ &= \sum_{\tau=0}^{t-1} \eta_{\tau}^{\min} \|\nabla_{\boldsymbol{w}_{\rm ts}} \ell(\boldsymbol{w}^{(\tau)})\|^2 + \sum_{e=1}^{\infty} \sum_{\tau=0}^{t-1} \eta_{2et+\tau}^{\min} \|\nabla_{\boldsymbol{w}_{\rm ts}} \ell(\boldsymbol{w}^{(2et+\tau)})\|^2,\end{aligned} \tag{29}$$

where, for any $e \geq 1$ and $\tau = 0, 1, \ldots, t-1$,

$$\begin{aligned}\|\nabla_{\boldsymbol{w}_{\rm ts}} \ell(\boldsymbol{w}^{(2et+\tau)})\| &\leq \|\nabla_{\boldsymbol{w}_{\rm ts}} \ell(\boldsymbol{w}^{(2et+\tau)}) - \nabla_{\boldsymbol{w}_{\rm ts}} \ell(\boldsymbol{w}^{(2et-1)})\| + \|\nabla_{\boldsymbol{w}_{\rm ts}} \ell(\boldsymbol{w}^{(2et-1)})\| \\ &\leq L_{\rm ts}^0 \|\boldsymbol{w}^{(2et+\tau)} - \boldsymbol{w}^{(2et-1)}\| + \|\nabla_{\boldsymbol{w}_{\rm ts}} \ell(\boldsymbol{w}^{(2et-1)})\| \\ &= L_{\rm ts}^0 \|\eta_{2et-1} \nabla_{\boldsymbol{w}_{\rm ts}} \ell(\mathcal{B}_{t-1}; \boldsymbol{w}^{(2et-1)}) + \sum_{k=0}^{\tau-1} \eta_{2et+k} \nabla_{\boldsymbol{w}_0} \ell(\mathcal{B}_k; \boldsymbol{w}^{(2et+k)}) \| \\ &\quad + \|\nabla_{\boldsymbol{w}_{\rm ts}} \ell(\boldsymbol{w}^{(2et-1)})\|.\end{aligned}$$

The idea of the proof is that, when damped by the learning rates, gradient norms can be bounded by assumption (22). Instead, to tackle the norm of $\nabla_{\boldsymbol{w}_{\rm ts}} \ell(\boldsymbol{w}^{(2et-1)})$ we can resort to Lemma 3.4, because by (28) it is easily seen that $2et - 1 \in I_{\rm ts}$ for any $e \geq 1$. Then

$$\|\nabla_{\boldsymbol{w}_{\rm ts}} \ell(\boldsymbol{w}^{(2et+\tau)})\| \leq L_{\rm ts}^0 M \sum_{k=-1}^{\tau-1} \eta_{2et+k} + \|\nabla_{\boldsymbol{w}_{\rm ts}} \ell(\boldsymbol{w}^{(2et-1)})\|,$$

and

$$\|\nabla_{\boldsymbol{w}_{\rm ts}} \ell(\boldsymbol{w}^{(2et+\tau)})\|^2 \leq (L_{\rm ts}^0)^2 M^2 \left( \sum_{k=-1}^{\tau-1} \eta_{2et+k} \right)^2$$

$$+ 2L_{\mathrm{ts}}^0 M^2 \sum_{k=-1}^{\tau-1} \eta_{2et+k} + \|\nabla_{\boldsymbol{w}_{\mathrm{ts}}} \ell(\boldsymbol{w}^{(2et-1)})\|^2. \tag{30}$$

Now using (22) and (30) in (29) we have

$$\begin{aligned}\sum_{i\in I_0} \eta_i^{\min} \|\nabla_{\boldsymbol{w}_{\mathrm{ts}}} \ell(\boldsymbol{w}^{(i)})\|^2 \le M^2 \sum_{\tau=0}^{t-1} \eta_\tau + (L_{\mathrm{ts}}^0)^2 M^2 \sum_{e=1}^{\infty} \sum_{\tau=0}^{t-1} \eta_{2et+\tau}^{\min} \left( \sum_{k=-1}^{\tau-1} \eta_{2et+k} \right)^2 \\ + 2L_{\mathrm{ts}}^0 M^2 \sum_{e=1}^{\infty} \sum_{\tau=0}^{t-1} \eta_{2et+\tau}^{\min} \left( \sum_{k=-1}^{\tau-1} \eta_{2et+k} \right) \\ + \sum_{e=1}^{\infty} \sum_{\tau=0}^{t-1} \eta_{2et+\tau}^{\min} \|\nabla_{\boldsymbol{w}_{\mathrm{ts}}} \ell(\boldsymbol{w}^{(2et-1)})\|^2. \end{aligned} \tag{31}$$

It is worth to point out here a technical fact that we exploit to conclude the proof: by (27) and (28) it is easily seen that $\{2et, 2et+1, \ldots, 2et+t-1\} \subset I_0$ for all $e \ge 0$, and $\{2et-1, 2et+t, 2et+t+1, \ldots, 2et+2t-1\} \subset I_{\mathrm{ts}}$ for all $e \ge 1$. The corresponding sequences of learning rates are diminishing, so that recalling definition (24) we have

$$\eta_{2et+\tau}^{\min} = \min\{\eta_{2et+t-1}, \eta_{2et+2t-1}\} < \eta_{2et-1}, \quad \forall\, e \ge 1 \quad \text{and} \quad 0 \le \tau \le t-1. \tag{32}$$

Hence the last term in (31) can be bounded almost surely by exploiting (21) as follows:

$$\begin{aligned}\sum_{e=1}^{\infty} \sum_{\tau=0}^{t-1} \eta_{2et+\tau}^{\min} \|\nabla_{\boldsymbol{w}_{\mathrm{ts}}} \ell(\boldsymbol{w}^{(2et-1)})\|^2 &< t \sum_{e=1}^{\infty} \eta_{2et-1} \|\nabla_{\boldsymbol{w}_{\mathrm{ts}}} \ell(\boldsymbol{w}^{(2et-1)})\|^2 \\ &< t \sum_{i\in I_{\mathrm{ts}}} \eta_i \|\nabla_{\boldsymbol{w}_{\mathrm{ts}}} \ell(\boldsymbol{w}^{(i)})\|^2 < \infty. \end{aligned}$$

The other terms can be bounded using the assumption $\sum_{i=0}^{\infty} \eta_i^2 < \infty$. Indeed, besides (32), since $\{2et, 2et+1, \ldots, 2et+t-1\} \subset I_0$ and the corresponding sequence of learning rate is diminishing, it holds also that

$$\eta_{2et+\tau}^{\min} \le \eta_{2et+\tau} < \eta_{2et+k} \le \eta_{2et}, \quad \text{for all } e \ge 1, 1 \le \tau \le t-1 \quad \text{and} \quad 0 \le k \le \tau - 1;$$

then we have

$$\begin{aligned}\sum_{e=1}^{\infty} \sum_{\tau=0}^{t-1} \eta_{2et+\tau}^{\min} \left( \sum_{k=-1}^{\tau-1} \eta_{2et+k} \right) &= \sum_{e=1}^{\infty} \left( \sum_{\tau=0}^{t-1} \eta_{2et+\tau}^{\min}\, \eta_{2et-1} + \sum_{\tau=1}^{t-1} \sum_{k=0}^{\tau-1} \eta_{2et+\tau}^{\min} \eta_{2et+k} \right) \\ &< \sum_{e=1}^{\infty} \left( \sum_{\tau=0}^{t-1} \eta_{2et-1}^2 + \sum_{\tau=1}^{t-1} \sum_{k=0}^{\tau-1} \eta_{2et}^2 \right) \\ &< t \sum_{e=1}^{\infty} \eta_{2et-1}^2 + (t-1)^2 \sum_{e=1}^{\infty} \eta_{2et}^2 < \infty, \end{aligned}$$

and

$$\sum_{e=1}^{\infty}\sum_{\tau=0}^{t-1}\eta_{2et+\tau}^{\min}\left(\sum_{k=-1}^{\tau-1}\eta_{2et+k}\right)^2 < \left(\sum_{k=-1}^{\tau-1}\eta_{2t+k}\right)\sum_{e=1}^{\infty}\sum_{\tau=0}^{t-1}\eta_{2et+\tau}^{\min}\left(\sum_{k=-1}^{\tau-1}\eta_{2et+k}\right) < \infty.$$

■

**Remark 3.1:** It is worth to observe that assumption (22) in Theorem 3.5 can be relaxed. Indeed, with a slight modification of the theorem's proof, we can prove that the last term in (26) is bounded assuming

$$\sum_{e=1}^{\infty}\eta_{2et-1}^2 M_{2et-1}^2 < \infty \quad \text{and} \quad \sum_{e=1}^{\infty}\eta_{2et}^2 M_{2et-1}^2 < \infty,$$

where

$$M_{2et-1} = \max\{\|\nabla_{\boldsymbol{w}_{\text{ts}}}\ell(\mathcal{B}_{t-1};\boldsymbol{w}^{(2et-1)})\|, \max_{k=0,1,\ldots,t-1}\|\nabla_{\boldsymbol{w}_0}\ell(\mathcal{B}_k;\boldsymbol{w}^{(2et+k)})\|\}.$$

These conditions are clearly weaker than (22), and can be satisfied for example whenever $\{\eta_i\} = \{\frac{1}{i}\}$ and $M_{2et-1} \le (2et-1)^p$ with $p \in (0, 1/2)$.

## 4. Numerical experiments

In this section we report the results of our numerical experimentation, where we compare the performance of an MTNN trained with a standard stochastic gradient procedure and by using alternate training procedures. We consider three different problems. The first one takes into account a synthetic dataset representing a 4-classes classification task and a binary classification task for a set of points in $\mathbb{R}^2$; the second experiment is based on a real-world dataset, where signals must be classified with respect to two different multiclass classification tasks; the third one is a three-task classification problem, based on a real-world dataset of mammography images. In all cases, the used loss function is a weighted sum of the task-specific losses where the weights are fixed and equal to one (i.e. $\lambda_k = 1$ for each $k = 1, \ldots, K$, see (1)). Indeed, we want to analyse the effects of our method without any balancing of the task-specific losses or emphasizing the action of a task with respect to the other.

The alternate procedure used for the experiments of this section is illustrated in Algorithm A.1 of Appendix: we call it *implemented* ATE-SG method. The main difference between this implemented version and ATE-SG (see Algorithm 3.2) lies in the way mini-batches are generated. Indeed, for theoretical purposes, in ATE-SG the mini-batches are randomly sampled from the training set $\mathcal{T}$, instead of being obtained through a random split of $\mathcal{T}$ as in Algorithm A.1. This modification is necessary for optimal compatibility with the Deep Learning frameworks *Keras* [6] and *TensorFlow* [2] used in the experiments. Starting from a given initial value, the learning rates are adaptively reduced as specified in the training options below.

To compare the performance of the implemented ATE-SG procedures with the standard SG procedure, we investigated their dependence both on the initial learning rate and on the number of alternate update epochs $E_0 = E_{\text{ts}}$. In addition, each test case was repeated

several times using different random seeds. Specifically, we trained the MTNN for each combination of the following hyperparameters:

- 11 distinct random seeds;
- Starting learning rates: $10^{-2}$, $10^{-3}$;
- Alternate training sub-epochs: $E_0 = E_{\text{ts}} = 1, 10, 100$.

Training options common to both experiments are the following:

- Preprocessing: standard scaler for inputs (based on training data only);
- Optimization: maximum number of epochs 50,00; early stopping (patience 350, restore best weights); reduce learning rate on plateaus (patience 50, factor 0.75, min-delta 0.0001).

Additionally, for the third multi-task problem (the one based on mammography images) we combined our ATE strategy also with Adam [12] and tested the new ATE-Adam alternate training strategy. The motivation of this test is to show that the behaviour observed for the ATE-SG method is confirmed also when another reference optimization method is used, though the theoretical analysis was carried out assuming to use SG.

For classification tasks, typical performance measures of the methods are the weighted average precision, the weighted average recall, and the weighted average $F_1$-score [1].

For the reader's convenience, we report a brief description of the quantities: precision, recall, and $F_1$-score, for any class $C$ of a general classification problem (see [18]). The precision is the percentage of correct predictions among all the elements predicted as $C$; the recall is the percentage of elements predicted as $C$ among all the $C$ elements in the set; the $F_1$-score is defined as the weighted harmonic mean of precision and recall. Then, the weighted average precision/recall/$F_1$-score is the weighted average of these quantities with respect to the cardinalities of each class in the set. For this reason, we have that the weighted average recall is equivalent to the accuracy.

Performance measures reported in Tables 2–3, 5–6, and 8 are averaged with respect to the 11 models trained with different random seeds. We also analyse the behaviour of the loss functions (on both the training and the validation set) over the epochs and over the computational cost. Plots are reported in Figures 3–4, 6–7, and 8–9 where coloured areas represent the values between the minimum and the maximum loss obtained after 11 realizations; coloured lines are the median values. The computational cost is estimated as a function of the number of full-gradient evaluations: namely, one unit corresponds to the cost of one full-gradient epoch. These plots allow to show the saving in terms of computational cost of ATE-SG due to the fact that SG computes and stores the full gradient $\nabla_{\boldsymbol{w}}\ell$, while ATE-SG computes and stores either $\nabla_{\boldsymbol{w}_0}\ell$ or $\nabla_{\boldsymbol{w}_{\text{ts}}}\ell$ and one task-specific epoch costs $p_{\text{ts}}/p$ units, because the back-propagation [24] stops before propagating through the trunk $\widehat{\boldsymbol{F}}_0$ of the MTNN (see Definition 2.1).

### 4.1. Experiments on synthetic data

We consider a dataset $\mathcal{D}$ made of $N = 10{,}000$ points uniformly sampled in the square $D = [-2, 2]^2 \subset \mathbb{R}^2$ and labelled with respect to two different criteria: (1) to belong to one of the

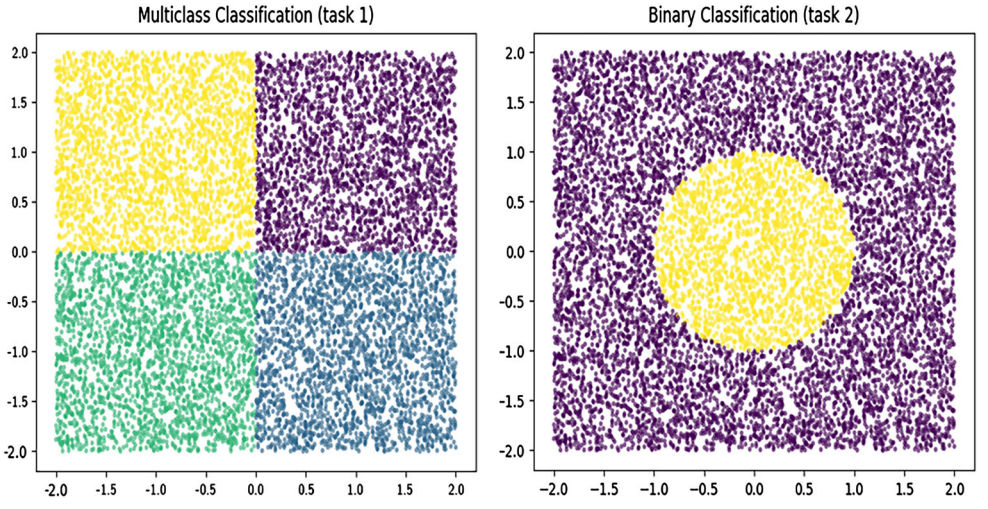


**Figure 2.** The synthetic dataset $\mathcal{D}$ with respect to its two tasks. Task 1 on the left, task 2 on the right. Different colours denote different labels for the points.

**Table 1.** Synthetic dataset cases. Keras [2,6] architecture of the MTNN.

| Layer name | Layer type (Keras) | Output shape | Param. # | Param. # (Grouped) | Connected to |
|---|---|---|---|---|---|
| input | InputLayer | (None, 2) | 0 | – | – |
| trunk_01 | Dense (relu) | (None, 512) | 1 536 | ($\boldsymbol{w}_0$-param.s) | input |
| trunk_02 | Dense (relu) | (None, 512) | 262 656 | 526 848 | trunk_01 |
| trunk_03 | Dense (relu) | (None, 512) | 262 656 | | trunk_02 |
| quad_01 | Dense (relu) | (None, 512) | 262 656 | ($\boldsymbol{w}_{\text{ts}}$-param.s, task 1) | trunk_03 |
| quad_02 | Dense (relu) | (None, 512) | 262 656 | 527 364 | quad_01 |
| quad_out | Dense (softmax) | (None, 4) | 2 052 | | quad_02 |
| circ_01 | Dense (relu) | (None, 512) | 262 656 | ($\boldsymbol{w}_{\text{ts}}$-param.s, task 2) | trunk_03 |
| circ_02 | Dense (relu) | (None, 512) | 262 656 | 525 825 | circ_01 |
| circ_out | Dense (linear) | (None, 1) | 513 | | circ_02 |

four quadrants of $\mathbb{R}^2$ (4-classes classification task); (2) to be inside or outside the unitary circle centred in the origin (binary classification task). For simplicity, we denote these tasks by *task 1* and *task 2*, respectively.

Then, the dataset $\mathcal{D}$ is made up of samples $(\boldsymbol{x}_i, (q_i, c_i))$ such that $\boldsymbol{x}_i \in D = [-2, 2]^2$, $q_i \in \{0, \ldots, 3\}$, and $c_i \in \{0, 1\}$, for each $i = 1, \ldots, N$ (see Figure 2), where $q_i$ and $c_i$ denote the quadrant label and the circle label of $\boldsymbol{x}_i$, respectively.

We randomly split $\mathcal{D}$, so that the training set $\mathcal{T}$, the validation set $\mathcal{V}$, and the test set $\mathcal{P}$ are made up of $T = 5\,600$ samples, $V = 1\,400$ samples, and $P = 3\,000$ samples, respectively. Then, we build an MTNN with architecture as described in Table 1 and, both for the classic SG and the implemented ATE-SG procedures, we train it by setting the mini-batch size to 256 and minimizing the sum of the following task-specific loss functions:

- task 1: categorical cross-entropy;
- task 2: binary cross-entropy (from logits).

**Table 2.** Synthetic dataset. Starting learning rate $10^{-2}$. Performance measures of the MTNNs on the test set $\mathcal{P}$.

| | Recall (accuracy) | | Precision | | $F_1$-score | |
|---|---|---|---|---|---|---|
| Training Type | task 1 | task 2 | task 1 | task 2 | task 1 | task 2 |
| ATE-SG ($E_0 = E_{ts} = 1$) | 0.997909 | 0.996667 | 0.997914 | 0.996677 | 0.997909 | 0.996665 |
| ATE-SG ($E_0 = E_{ts} = 10$) | 0.997727 | 0.996758 | 0.997730 | 0.996766 | 0.997727 | 0.996758 |
| ATE-SG ($E_0 = E_{ts} = 100$) | 0.997848 | 0.996576 | 0.997851 | 0.996587 | 0.997848 | 0.996575 |
| Classic SG | 0.997788 | 0.996697 | 0.997791 | 0.996703 | 0.997788 | 0.996694 |

**Table 3.** Synthetic dataset. Starting learning rate $10^{-3}$. Performance measures of the MTNNs on the test set $\mathcal{P}$.

| | Recall (accuracy) | | Precision | | $F_1$-score | |
|---|---|---|---|---|---|---|
| Training type | task1 | task 2 | task1 | task 2 | task1 | task 2 |
| ATE-SG ($E_0 = E_{ts} = 1$) | 0.998061 | 0.996606 | 0.998063 | 0.996614 | 0.998060 | 0.996605 |
| ATE-SG ($E_0 = E_{ts} = 10$) | 0.998030 | 0.996606 | 0.998033 | 0.996616 | 0.998030 | 0.996606 |
| ATE-SG ($E_0 = E_{ts} = 100$) | 0.998333 | 0.996818 | 0.998335 | 0.996826 | 0.998333 | 0.996817 |
| Classic SG | 0.998242 | 0.996758 | 0.998244 | 0.996767 | 0.998242 | 0.996757 |

After training the model with respect to all the configurations, the first thing we observe is that the performance measured in terms of recall, precision, and $F_1$-score is almost identical for all the training methods and all the starting learning rate values. See Tables 2–3 for the average performance measures of the MTNN on the test set.

We also analyse the behaviour of the loss functions in Figures 3 and 4 for $\eta = 10^{-2}$ and $\eta = 10^{-3}$ respectively.

As already noticed, at an equal number of epochs, ATE-SG is less expensive and requires less memory than SG. In our test, the dimension of $\nabla_{\boldsymbol{w}_0}\ell$ is approximately one-third of $\nabla_{\boldsymbol{w}}\ell$'s dimension (see Table 1). Then, when computing $\nabla_{\boldsymbol{w}_{ts}}\ell$ in the task-specific phase, ATE-SG saves one-third of memory allocation and one-third of back-propagation operations. The computational savings are shown in Figures 3(b,d) and 4(b,d).

We further observe what follows.

- For the case $E_0 = E_{ts} = 1$, ATE-SG appears to have regularization effects on the training, leading to a marked reduction in its oscillations with respect to SG as well as to larger values of $E_0$ and $E_{ts}$, both in the training set and in the validation set.
- For the case $E_0 = E_{ts} = 10$, we observe a particular phenomenon for ATE-SG: the training yields oscillations for the loss function which show spikes approximately every 10 epochs and are characterized by a range of values larger than the typical fluctuations of the loss of a classic MTNN training. Nonetheless, the overall trend of the loss on the validation set is still decreasing with the epochs (see Figures 3(c) and 4(c)). Analogously to the other cases, these oscillations tend to reduce when decreasing the learning rate (e.g. compare Figures 3(c) and 4(c)).
- The larger the value chosen for $E_0 = E_{ts}$, the more similar the loss behaviour of ATE-SG is to the loss of SG. In particular, with $E_0 = E_{ts} = 100$ the loss behaviours versus the epochs (on both training and validation set) are very similar, both in the range of values and in the stochastic oscillations (see Figures 3(a,c) and 4(a,c)). In general, these oscillations tend to reduce with smaller learning rates due to the shorter steps in the domain.

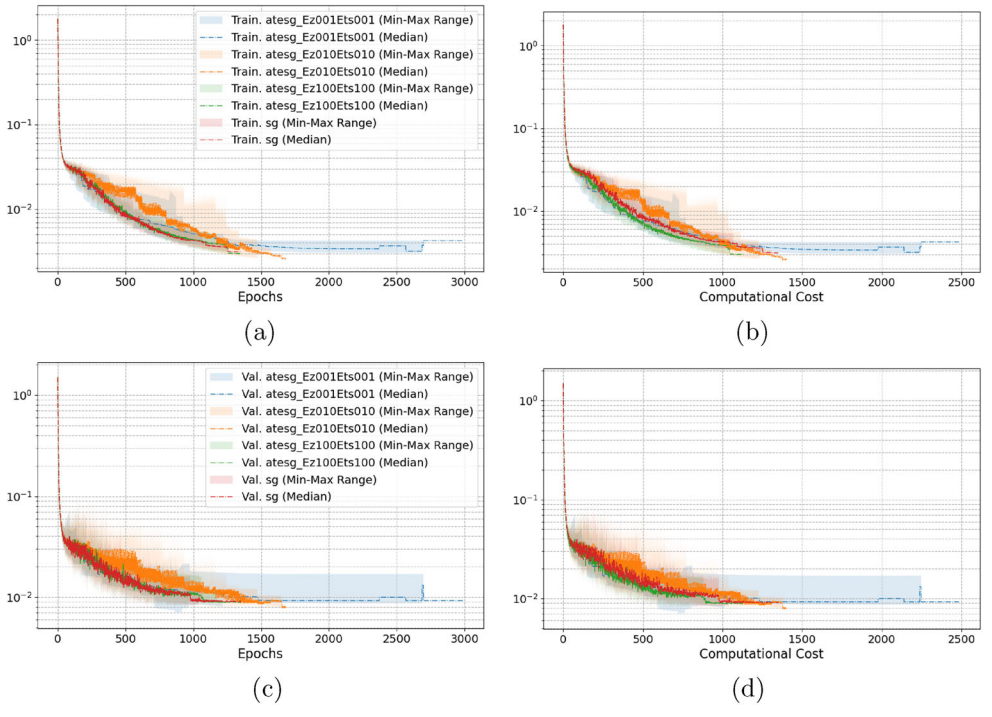


**Figure 3.** Synthetic dataset case. Starting learning rate $10^{-2}$. (a) Training loss through the epochs. (b) Training loss with respect to the computational cost. (c) Validation loss through the epochs and (d) Validation loss with respect to the computational cost.

- In Figure 5 we plot the used learning rate against the epochs for each seed and for each method, for the starting learning rate $10^{-3}$. We can observe that SG and ATE-SG with $E_0 = E_{\text{ts}} = 100$ shows a similar behaviour, while with $E_0 = E_{\text{ts}} = 1, 10$ the initial learning rate is used for a larger number of epochs in ATE-SG, resulting in overall larger sequences of learning rates. We observed a similar behaviour also with the starting learning rate equal to $10^{-2}$.

## 4.2. Experiments on real-world data

For the experiments on real-world data, we consider two multi-task learning problems based on different datasets: in Subsection 4.2.1 we report the results obtained on a dataset used for wireless signal recognition tasks [10], while Subsection 4.2.2 is devoted to the experiments we conducted on a dataset for mammography calcification recognition tasks, available from the multi-domain, multi-task MedIMeta dataset (see *mammo_calc* sub-dataset at [28]).

### 4.2.1. Wireless signal dataset

Typically, signal recognition is segmented into sub-tasks like the modulation recognition or the wireless technology (i.e. signal type); nonetheless, in [11] the authors suggest an

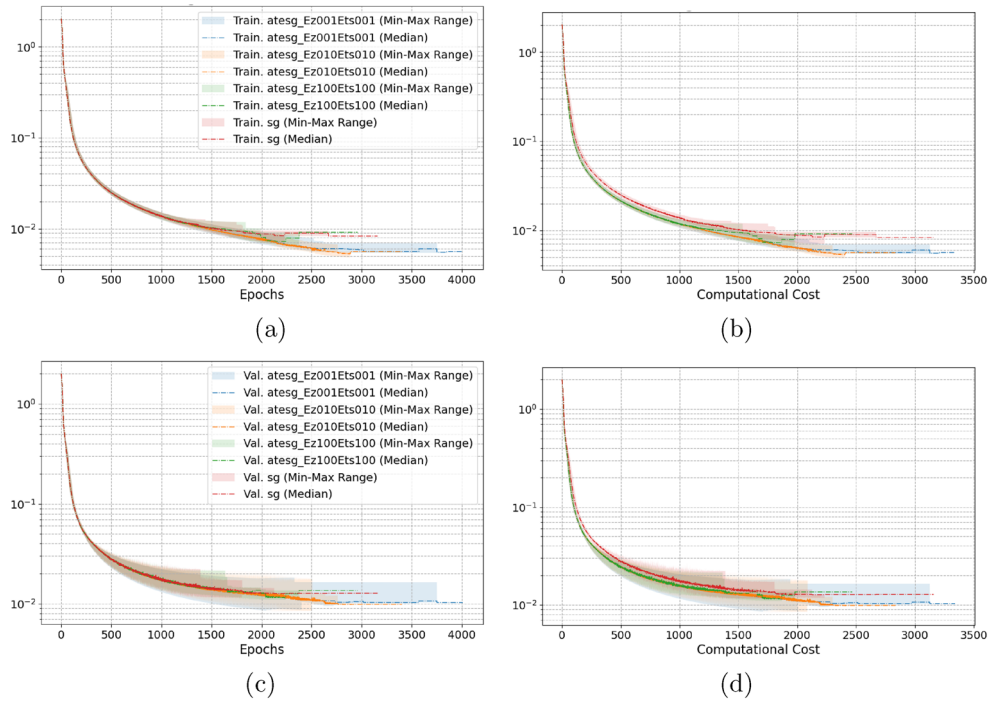


**Figure 4.** Synthetic dataset case. Starting learning rate $10^{-3}$. (a) Training loss through the epochs. (b) Training loss with respect to the computational cost. (c) Validation loss through the epochs and (d) Validation loss with respect to the computational cost.

approach to the problem that exploits a multi-task setting for classifying at the same time both the modulation and the signal type of a wireless signal.

Here we focus on signals characterized by 0dB Signal-to-Noise Ratio (SNR). The dataset $\mathcal{D}$ is made of $N = 63\,000$ signals, each one represented as a vector $\boldsymbol{s}_i \in \mathbb{R}^{256}$ obtained from 128 complex samples of the original signal. In the dataset, associated to each signal $\boldsymbol{s}_i$, we have a label $\mu_i \in \{0, \dots, 5\}$ for the corresponding modulation (*task 1*) and a label $\sigma_i \in \{0, \dots, 7\}$ for the corresponding signal type (*task 2*). For more details about the data see [10,11].

We split $\mathcal{D}$ randomly, so that the training set $\mathcal{T}$, the validation set $\mathcal{V}$, and the test set $\mathcal{P}$ are made of $T = 35\,280$ samples, $V = 8\,820$ samples, and $P = 18\,900$ samples, respectively (i.e. same ratios used for the synthetic data in Section 4.1). Then, we build an MTNN with architecture as described in Table 4 and, both for the classic SG and the implemented ATE-SG procedures, we train it by setting the mini-batch size to 512 and minimizing the sum of the task-specific loss functions (categorical cross-entropy for both tasks). In particular, given this architecture (see Table 4 and Remark 4.1 below) when computing $\nabla_{\boldsymbol{w}_{\text{ts}}} \ell$ in the task-specific phase, ATE-SG saves one-half of memory allocation and one-half of back-propagation operations; on the other hand, in the shared phase, one-half of memory allocation is saved.

**Remark 4.1 (MTNN architecture and hyper-parameters):** Since the aim of the experiment is to study how the training performance of an MTNN change when using an

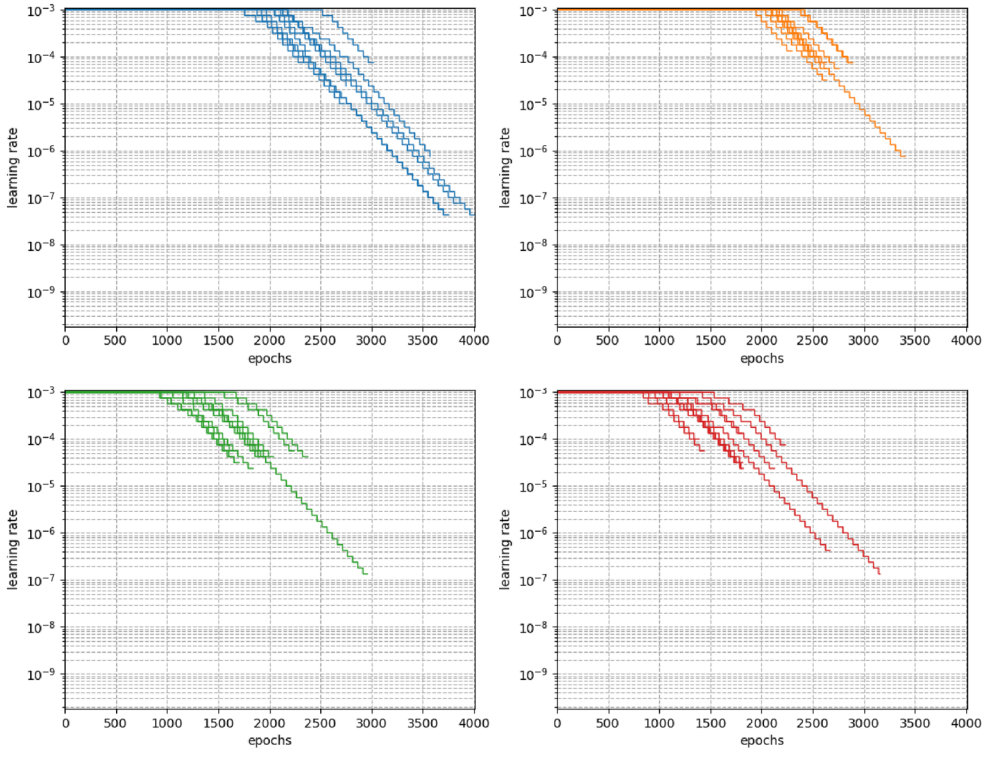


**Figure 5.** Synthetic dataset case. Learning rate sequence, starting value: $10^{-3}$. From left to right, from top to bottom: ATE-SG $E_0 = E_{\text{ts}} = 1$, ATE-SG $E_0 = E_{\text{ts}} = 10$, ATE-SG $E_0 = E_{\text{ts}} = 100$, and SG.

**Table 4.** Wireless signals dataset. Keras [2,6] architecture of the MTNN.

| Layer name | Layer type (Keras) | Output shape | Param. # | Param. # (Grouped) | Connected to |
|---|---|---|---|---|---|
| input | InputLayer | (None, 256) | 0 | – | – |
| trunk_00 | Reshape | (None, 128, 2) | 0 | | input |
| trunk_01 | Conv1D (relu) | (None, 128, 64) | 576 | ($\boldsymbol{w}_0$-param.s) | trunk_00 |
| trunk_02 | Conv1D (relu) | (None, 128, 32) | 8 224 | 12 928 | trunk_01 |
| trunk_03 | Conv1D (relu) | (None, 128, 32) | 4 128 | | trunk_02 |
| trunk_end | GlobalMaxPooling1D | (None, 32) | 0 | | trunk_03 |
| mod_01 | Dense (relu) | (None, 64) | 2 112 | ($\boldsymbol{w}_{\text{ts}}$-param.s, task 1) | trunk_end |
| mod_02 | Dense (relu) | (None, 64) | 4 160 | 6 662 | mod_01 |
| mod_out | Dense (softmax) | (None, 6) | 390 | | mod_02 |
| sig_01 | Dense (relu) | (None, 64) | 2 112 | ($\boldsymbol{w}_{\text{ts}}$-param.s, task 2) | trunk_end |
| sig_02 | Dense (relu) | (None, 64) | 4 160 | 6 792 | sig_01 |
| sig_out | Dense (softmax) | (None, 8) | 520 | | sig_02 |

alternate training procedure, we do not focus on hyper-parameter and/or architecture tuning for obtaining the best predictions. Then, for simplicity, in this experiment we choose a simple but efficient architecture (see Table 4) based on 1-dimensional convolutional layers, after a brief, manual hyper-parameter tuning. The 1-dimensional convolutional layers are useful to exploit the signals reshaped as 128 complex signals (see layer *trunk_00* in Table 4), keeping the NN relatively small.

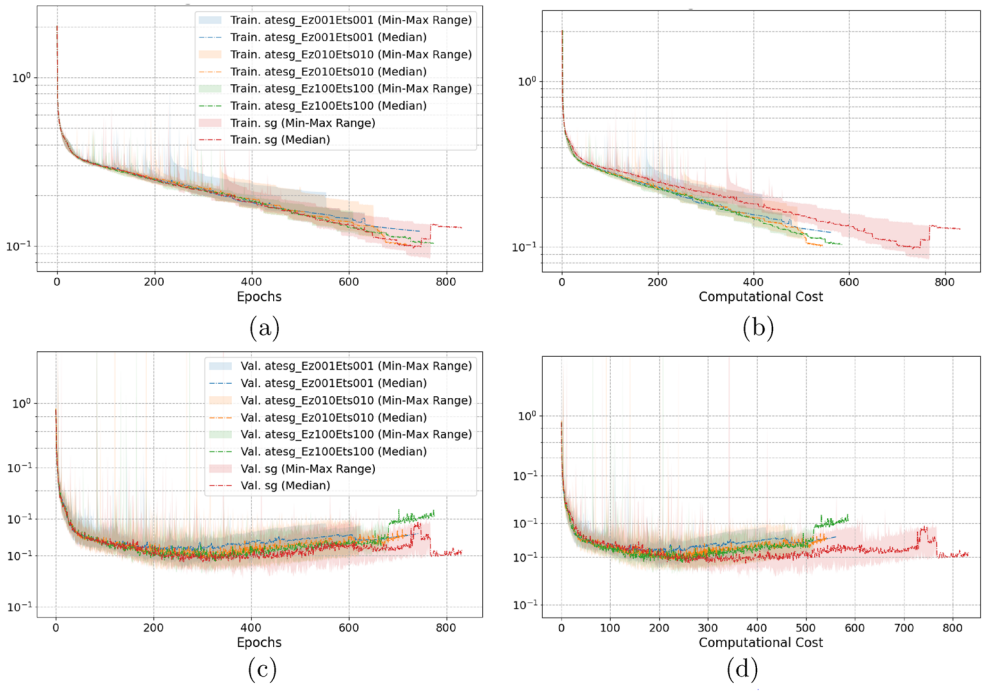


**Figure 6.** Wireless signals dataset. Starting learning rate $10^{-2}$. (a) Training loss through the epochs. (b) Training loss with respect to the computational cost. (c) Validation loss through the epochs and (d) Validation loss with respect to the computational cost.

**Table 5.** Wireless signals dataset. Starting learning rate $10^{-2}$. Performance measures of the MTNNs on the test set $\mathcal{P}$.

| | Recall (accuracy) | | Precision | | $F_1$-score | |
|---|---|---|---|---|---|---|
| Training Type | task 1 | task 2 | task 1 | task 2 | task 1 | task 2 |
| ATE-SG ($E_0 = E_{ts} = 1$) | 0.916840 | 0.961592 | 0.921696 | 0.962326 | 0.917194 | 0.960846 |
| ATE-SG ($E_0 = E_{ts} = 10$) | 0.922131 | 0.964031 | 0.923422 | 0.964155 | 0.922331 | 0.963479 |
| ATE-SG ($E_0 = E_{ts} = 100$) | 0.923906 | 0.965613 | 0.925943 | 0.965629 | 0.924248 | 0.965063 |
| Classic SG | 0.926763 | 0.967181 | 0.927541 | 0.967079 | 0.926777 | 0.966837 |

Looking at Tables 5–6, all the trained models show approximately the same performance, but from Figures 6–7 we can see that experiments on this test case are more challenging than the previous ones. Although achieving good accuracy, with the initial learning rate $10^{-2}$ overfitting is reached; starting with learning rate $10^{-3}$ overfitting is no longer a problem, rather a plateau appears to be reached, and the training procedures have difficulties in further decreasing the validation loss.

Further, some general trends are confirmed:

- the choice $E_0 = E_{ts} = 1$ yields a regularization effect on the training;
- oscillations tend to reduce with smaller learning rates;
- for larger values of $E_0$ and $E_{ts}$ the loss behaviour of ATE-SG is closer to that of SG.
- Figures 6(b) and 7(b) clearly show that the ATE-SG procedures are generally less expensive than SG.

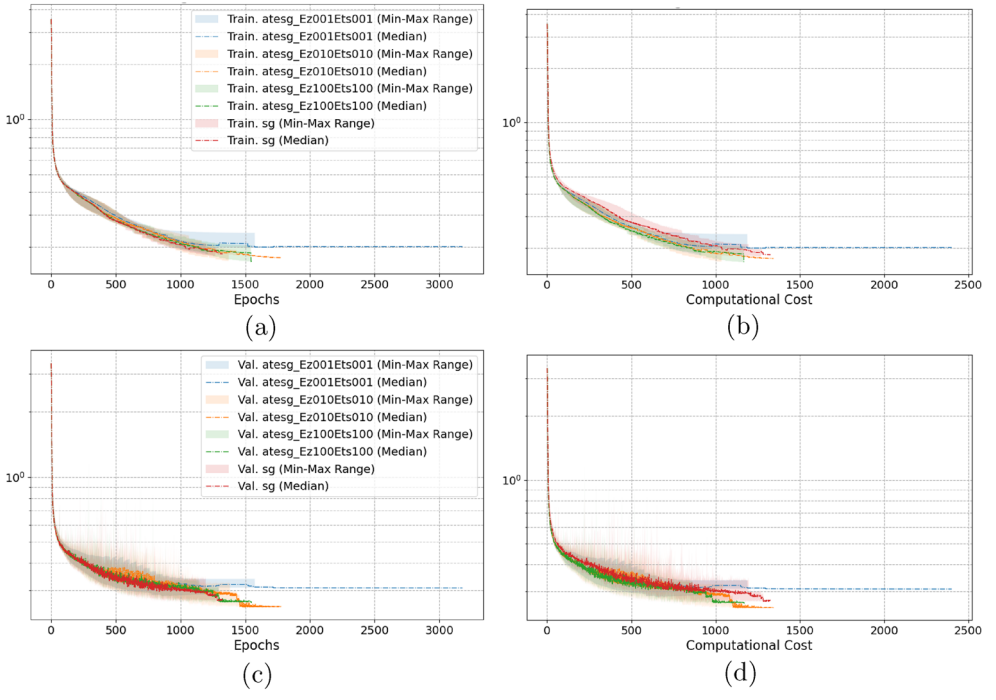


**Figure 7.** Wireless signals dataset. Starting learning rate $10^{-3}$. (a) Training loss through the epochs. (b) Training loss with respect to the computational cost. (c) Validation loss through the epochs and (d) Validation loss with respect to the computational cost.

**Table 6.** Wireless signals dataset. Starting learning rate $10^{-3}$. Performance measures of the MTNNs on the test set $\mathcal{P}$.

| | Recall (accuracy) | | Precision | | $F_1$-score | |
|---|---|---|---|---|---|---|
| Training Type | task 1 | task 2 | task 1 | task 2 | task 1 | task 2 |
| ATE-SG ($E_0 = E_{ts} = 1$) | 0.921241 | 0.964786 | 0.921954 | 0.964736 | 0.921131 | 0.964292 |
| ATE-SG ($E_0 = E_{ts} = 10$) | 0.924411 | 0.965830 | 0.924818 | 0.965756 | 0.924168 | 0.965472 |
| ATE-SG ($E_0 = E_{ts} = 100$) | 0.923430 | 0.965166 | 0.924457 | 0.965090 | 0.923388 | 0.964671 |
| Classic SG | 0.923516 | 0.965527 | 0.923931 | 0.965450 | 0.923406 | 0.965076 |

### *4.2.2. Mammography calcification dataset*

In the mammography calcification dataset taken from [28], we have $N = 1872$ black and white, $224 \times 224$ images split into training set (1332 images), validation set (214 images), and test set (326 images). Three different labels, $\boldsymbol{\gamma}'$, $\boldsymbol{\gamma}''$, $\boldsymbol{\gamma}'''$, are associated to these images, each one corresponding to a different task; specifically, we have two multi-binary classification tasks of 5 classes and 14 classes, respectively, and a binary classification task, that is $\boldsymbol{\gamma}' \in \{0,1\}^5$, $\boldsymbol{\gamma}'' \in \{0,1\}^{14}$, $\gamma''' \in \{0,1\}$.

For this three-task problem, we build an MTNN with architecture as described in Table 7 and we train it by setting the mini-batch size to 32 and minimizing the sum of the task-specific loss functions. For simplicity, in this experiment we choose for the trunk of the MTNN (i.e. the encoding phase of the MTNN) one of the typical NN architectures used for image recognition, the ResNet50 [9]. In particular, the architecture used is based on a

**Table 7.** Mammography calcification dataset. Keras [2,6] architecture of the MTNN.

| Layer Name | Layer Type (Keras) | Output Shape | Param. # | Param. # (Grouped) | Connected to |
|---|---|---|---|---|---|
| input_00 | InputLayer | (None, 224, 224) | 0 | – | – |
| input_01 | Rescale | (None, 224, 224) | 0 | – | input_00 |
| input_end | Concatenate | (None, 3, 224, 224) | 0 | – | input_01 |
| trunk_00 | | | | | |
| ⋮ | ResNet50 | (None, 2048) | – | 26 945 012 | input_end |
| trunk_end | | | | | |
| task1_01 | Dense (leaky relu) | (None, 512) | 1 049 088 | ($\boldsymbol{w}_{\text{ts}}$-param.s, task 1) | trunk_end |
| task1_02 | Dense (leaky relu) | (None, 128) | 65 664 | 1 119 045 | task1_01 |
| task1_03 | Dense (leaky relu) | (None, 32) | 4 128 | | task1_02 |
| task1_out | Dense (sigmoid) | (None, 5) | 165 | | task1_03 |
| task2_01 | Dense (leaky relu) | (None, 512) | 1 049 088 | ($\boldsymbol{w}_{\text{ts}}$-param.s, task 2) | trunk_end |
| task2_02 | Dense (leaky relu) | (None, 128) | 65 664 | 1 119 342 | task1_01 |
| task2_03 | Dense (leaky relu) | (None, 32) | 4 128 | | task2_02 |
| task2_out | Dense (sigmoid) | (None, 14) | 462 | | task2_03 |
| task3_01 | Dense (leaky relu) | (None, 512) | 1 049 088 | ($\boldsymbol{w}_{\text{ts}}$-param.s, task 3) | trunk_end |
| task3_02 | Dense (leaky relu) | (None, 128) | 65 664 | 1 118 913 | task1_01 |
| task3_03 | Dense (leaky relu) | (None, 32) | 4 128 | | task3_02 |
| task3_out | Dense (sigmoid) | (None, 1) | 1 | | task3_03 |

trunk that is an untrained, newly initialized ResNet50, followed by three branches made of fully connected layers. Such an architecture makes extremely cheap the task-specific phase of the alternate trainings. In fact, when computing $\nabla_{\boldsymbol{w}_{\text{ts}}}\ell$ in the task-specific phase, ATE procedures save approximately 87.5% of memory allocation and back-propagation operations; on the other hand, in the shared phase, only 12.5% of memory allocation is saved.

Concerning the task-specific losses, we use the binary cross-entropy loss for the third task and a multi-binary cross-entropy loss for the first two tasks, i.e.: for $k = 1, 2$

$$\ell_k(\mathcal{B}^k; \boldsymbol{w}_0, \boldsymbol{w}_k) = \frac{1}{|\mathcal{B}^k|} \sum_{(\text{Img},\boldsymbol{\gamma})\in\mathcal{B}^k} \sum_{i=1}^{m} -\mu_1 \gamma_i \log(\widehat{\gamma}_i) - \mu_0(1-\gamma_i)\log(1-\widehat{\gamma}_i),$$

for any batch $\mathcal{B}^k$ of input-output data $(\text{Img}, \boldsymbol{\gamma}) \in [0, 255]^{224\times 224} \times \{0, 1\}^m$, where $\widehat{\boldsymbol{\gamma}} \in [0, 1]^m$ is the predicted vector for the task, and $m = 5, 14$ for $k = 1, 2$, respectively. Namely, $\ell_k$ is the average sum of the component-wise application of the binary cross-entropy loss. The hyperparameters $\mu_1$ and $\mu_0$ are used to suitably weight the classification of positive and negative classes, respectively; in particular, we used $\mu_0 = 0.5$ and $\mu_1 = 1.5$.

For this problem we illustrate the results obtained with starting learning rate equal to $10^{-3}$, $E_0 = E_{\text{ts}} = 1, 10$, and using SG and Adam as optimizers. Looking at Table 8, we can observe that all the trained models show approximately the same performance, with a particular difficulty in learning the third task, due to overfitting (probably caused by the unbalanced, small training set). The overfitting phenomenon can be clearly seen in Figures 8–9, where the validation losses start to increase while the training losses are still decreasing. In addition we observe what follows:

- Adam-based training is characterized by higher oscillations in the losses, both in classic training and ATE procedures; nonetheless, for the ATE procedures, we observe a smoother shape of the min-max range, describing a weak regularization effect on the training;

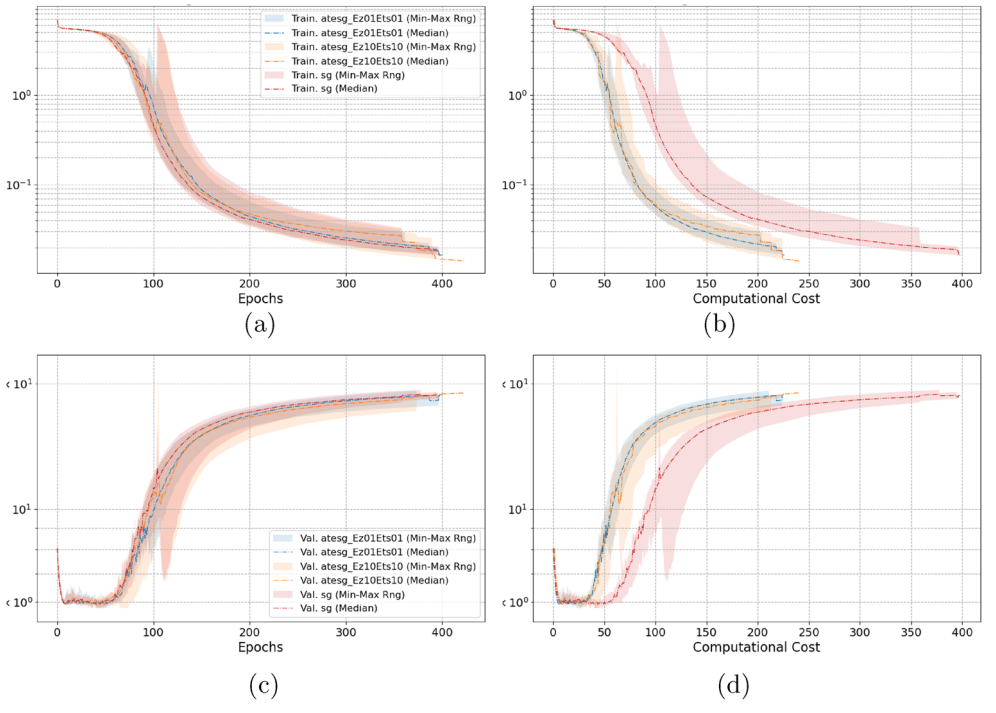


**Figure 8.** Mammography calcification dataset. Starting learning rate $10^{-3}$, SG as reference optimization method. (a) Training loss through the epochs. (b) Training loss with respect to the computational cost. (c) Validation loss through the epochs and (d) Validation loss with respect to the computational cost.

**Table 8.** Mammography calcification dataset. Starting learning rate $10^{-3}$. Performance measures of the MTNNs on the test set $\mathcal{P}$.

| | Recall (Accuracy) | | | Precision | | | $F_1$-score | | |
|---|---|---|---|---|---|---|---|---|---|
| Training Type | task 1 | task 2 | task 3 | task 1 | task 2 | task 3 | task 1 | task 2 | task 3 |
| ATE-SG ($E_0 = E_{ts} = 1$) | 0.8679 | 0.9145 | 0.6430 | 0.8797 | 0.9189 | 0.6324 | 0.8725 | 0.9165 | 0.6315 |
| ATE-SG ($E_0 = E_{ts} = 10$) | 0.8698 | 0.9110 | 0.6534 | 0.8821 | 0.9178 | 0.6418 | 0.8746 | 0.9141 | 0.6364 |
| Classic SG | 0.8721 | 0.9116 | 0.6485 | 0.8838 | 0.9169 | 0.6383 | 0.8767 | 0.9141 | 0.6323 |
| ATE-Adam ($E_0 = E_{ts} = 1$) | 0.8741 | 0.9180 | 0.6368 | 0.8848 | 0.9197 | 0.6296 | 0.8783 | 0.9186 | 0.5853 |
| ATE-Adam ($E_0 = E_{ts} = 10$) | 0.8783 | 0.9176 | 0.6301 | 0.8849 | 0.9189 | 0.6202 | 0.8809 | 0.9181 | 0.5619 |
| Classic Adam | 0.8726 | 0.9186 | 0.6436 | 0.8822 | 0.9202 | 0.6376 | 0.8765 | 0.9194 | 0.5846 |

- Adam-based trainings reach lower values of the training loss than the SG-based procedures. Indeed, the training loss at the end of the training is of the order of $10^{-5}$ for ATE-Adam and Adam, while ATE-SG and SG do not manage to push the loss below $10^{-2}$ (see Figures 8(a) and 9(a)). However, the accuracy reached on the test set is almost the same.
- All the alternated procedures are less expensive than the corresponding standard ones (see Figures 8(b) and 9(b)).

With this latest experiment, we confirm the general behaviour of ATE-SG in training MTNNs, as compared to standard SG. More importantly, we show that the ATE procedure

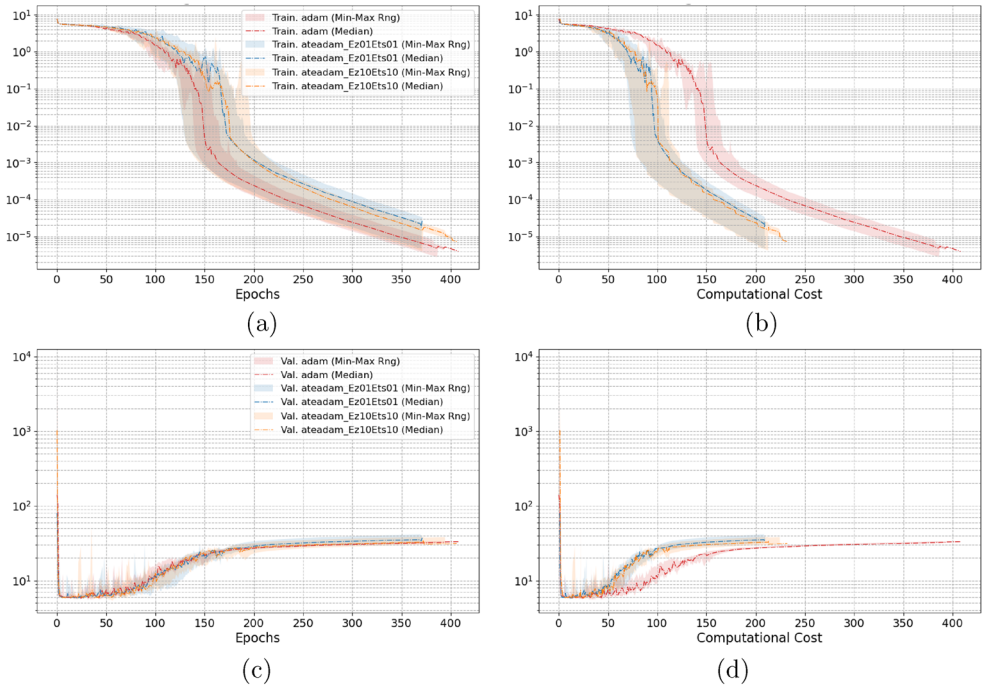


**Figure 9.** Mammography calcification dataset. Starting learning rate $10^{-3}$, Adam as reference optimization method. (a) Training loss through the epochs. (b) Training loss with respect to the computational cost. (c) Validation loss through the epochs and (d) Validation loss with respect to the computational cost.

can also be successfully combined with other optimization methods, such as Adam, with a behaviour similar to that of ATE-SG against to SG.

## 5. Conclusion

In this work we presented new alternate training procedures for hard-parameter sharing MTNNs. We started illustrating the properties of the task-specific gradients of the loss function of an MTNN, and explaining the motivations behind the proposed alternate method. Then, in Section 3, we introduced a first formulation of alternate training called Simple Alternate Training (SAT) and a second one called Alternate Training through the Epochs (ATE); both formulations are based on the Stochastic Gradient (SG). For these methods we provided a stochastic convergence analysis.

We concluded the work illustrating the results of three numerical experiments, where the prediction abilities of an MTNN trained using the implemented ATE-SG algorithm are compared with the prediction abilities of the same MTNN trained using the SG. The experiments show very interesting properties of the implemented ATE-SG; in particular, in the training phase, we observe a reduction of the computational costs and regularization effects on the training (i.e. marked reduction in loss value oscillations), when high-frequency alternation between the shared parameters and the task-specific parameters is adopted. We also provided numerical results using the new ATE-Adam method obtained combining the

ATE strategy with the Adam method in place of SG; obtained results are very encouraging and motivate a future work to analyse how the convergence properties of the ATE strategy may change by replacing the stochastic gradient with other optimization methods.

Additionally, we will focus on studying the behaviour of ATE-SG varying adaptively the number of epochs in the shared and task-specific phases along the iterations, taking also into account the learning rate schedule. For a ready-to-use version of the implemented ATE-SG algorithm (see Algorithm A.1, Appendix A), visit https://github.com/Fra0013To/ATEforMTNN.

## Note

1. For example, in our TensorFlow implementation, we just label as '*untrainable*' the weights $\boldsymbol{w}_0$ in the task-specific training and viceversa; this approach is particularly efficient, and does not involve any repetition of the typical overheads of initial steps of a training phase (e.g. JIT compilation, etc.).

## Acknowledgements

The authors acknowledge support from INdAM-GNCS Group.

## Disclosure statement

No potential conflict of interest was reported by the author(s).

## Funding

S.B. acknowledges the support from the Italian PRIN project "Numerical Optimization with Adaptive accuracy and applications to machine learning", CUP E53D23007690006, Progetti di Ricerca di Interesse nazionale 2022 and of the PRIN-PNRR project "Advanced optimization METhods for automated central veIn Sign detection in multiple sclerosis from magneTic resonAnce imag-ing (AMETISTA)', (CUP E53D23017980001). F.D. acknowledges that this study was carried out within the FAIR-Future Artificial Intelligence Research and received funding from the European Union Next-GenerationEU (PIANO NAZIONALE DI RIPRESA E RESILIENZA (PNRR) - MISSIONE 4 COMPONENTE 2, INVESTIMENTO 1.3 - D.D. 1555 11/10/2022, PE00000013). This manuscript reflects only the authors' views and opinions; neither the European Union nor the European Commission can be considered responsible for them. F.D. acknowledges support from Italian MUR PRIN project 20227K44ME, Full and Reduced order modeling of coupled systems: focus on non-matching methods and automatic learning (FaReX).

## Notes on contributors

***Stefania Bellavia*** received the degree in mathematics from the University of Florence, Italy, in 1993, and the Ph.D. degree in computational mathematics and informatics from the University of Padua, in 1997. Since 1999, she has been with the University of Florence. She is currently a Full Professor in numerical analysis. She is an expert in numerical methods for optimization problems. Her recent research interests include stochastic methods for optimization problems in in machine learning.

***Francesco Della Santa*** received the master's degree in mathematics from the University of Florence and the joint Ph.D. degree in pure and applied mathematics from the University of Turin and Politecnico di Torino. He is currently a Research Associate with Politecnico di Torino. His main scientific research interests include deep learning, surrogate models, uncertainty quantification, and numerical optimization.

***Alessandra Papini*** received the Laurea in Mathematics at the University of Florence, Italy, on July 1986. She was first assistant professor (1992-2001) and then associate professor (2001-to present) in Numerical Analysis at the University of Florence. Her research interests include numerical methods for nonlinear optimization and multi-objective problems, with special focus on derivative-free methods of direct search type, and for matrix computation.

## References


[1] *Scikit-learn*. Available at https://scikit-learn.org. Accessed on: 2023-09-09.

[2] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G.S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, *TensorFlow: Large-scale machine learning on heterogeneous systems* (2015). Available at https://www.tensorflow.org/, Software available from tensorflow.org.

[3] L. Bottou, F.E. Curtis, and J. Nocedal, *Optimization methods for large-scale machine learning*, SIAM Rev. 60 (2018), pp. 223–311https://doi.org/10.1137/16M1080173

[4] F. Bragman, R. Tanno, S. Ourselin, D. Alexander, and J. Cardoso, *Stochastic filter groups for multi-task CNNs: Learning specialist and generalist convolution kernels*, in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 1385–1394.

[5] Z. Chen, V. Badrinarayanan, C.Y. Lee, and A. Rabinovich, *GradNorm: Gradient normalization for adaptive loss balancing in deep multitask networks*, in *Proceedings of the 35th International Conference on Machine Learning*, J. Dy and A. Krause, eds., Proceedings of Machine Learning Research Vol. 80, 10–15 Jul. PMLR, 2018, pp. 794–803. Available at https://proceedings.mlr.press/v80/chen18a.html.

[6] F. Chollet, *et al.*, *Keras* (2015). Available at https://keras.io.

[7] R. Cipolla, Y. Gal, and A. Kendall, *Multi-task learning using uncertainty to weigh losses for scene geometry and semantics*, in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7482–7491.

[8] M. Guo, A. Haque, D.A. Huang, S. Yeung, and L. Fei-Fei, *Dynamic task prioritization for multitask learning*, in *Computer Vision – ECCV 2018*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, eds., Springer International Publishing, Cham, 2018, pp. 282–299.

[9] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition 2016-Decem*, 2016, pp. 770–778.

[10] A. Jagannath and J. Jagannath, *Dataset for modulation classification and signal type classification for multi-task and single task learning*, Comput. Netw. 199 (2021), pp.108441https://doi.org/10.1016/j.comnet.2021.108441

[11] A. Jagannath and J. Jagannath, *Multi-task learning approach for automatic modulation and wireless signal classification*, in *Proceedings of the of IEEE International Conference on Communications (ICC), June, Montreal, Canada*, 2021.

[12] D.P. Kingma and J.L. Ba, *Adam: A method for stochastic optimization*, in *3rd International Conference on Learning Representations, ICLR 2015 – Conference Track Proceedings*, 2015, pp. 1–15.

[13] I. Kokkinos, *Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory*, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[14] S. Liu and L. Vicente, *Accuracy and fairness trade-offs in machine learning: A stochastic multi-objective approach*, Comput. Manage. Sci.19 (2022), pp. 513–537https://doi.org/10.1007/s10287-022-00425-z

[15] S. Liu and L. Vicente, *Convergence rates of the stochastic alternating algorithm for bi-objective optimization*, J. Optim. Theory Appl. 198 (2023), pp. 165–186https://doi.org/10.1007/s10957-023-02253-w

[16] S. Liu and L. Vicente, *The stochastic multi-gradient algorithm for multi-objective optimization and its application to supervised machine learning*, Ann. Oper. Res. 339 (2024), pp. 1119–1148.
[17] H. Liu, M. Palatucci, and J. Zhang, *Blockwise coordinate descent procedures for the multi-task lasso, with applications to neural semantic basis discovery*, in *Proceedings of the 26th Annual International Conference on Machine Learning*, Association for Computing Machinery, ICML '09, New York, NY, USA, 2009, pp. 649–656. https://doi.org/10.1145/1553374.1553458
[18] J. Makhoul, F. Kubala, R. Schwartz, and R. Weischedel, *Performance measures for information extraction*, in *Proceedings of DARPA Broadcast News Workshop*, 1999.
[19] K.K. Maninis, I. Radosavovic, and I. Kokkinos, *Attentive single-tasking of multiple tasks*, in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 1851–1860.
[20] Q. Mercier, F. Poirion, and J. Désidéri, *A stochastic multiple gradient descent algorithm*, Eur. J. Oper. Res. 271 (2018), pp. 808–817.
[21] A. Navon, A. Shamsian, I. Achituve, H. Maron, K. Kawaguchi, G. Chechik, and E. Fetaya, *Multi-task learning as a bargaining game*, in *Proceedings of the 39th International Conference on Machine Learning*, Vol. 162, 2022.
[22] L. Pascal, P. Michiardi, X. Bost, B. Huet, and M.A. Zuluaga, *Improved optimization strategies for deep multi-task Networks*, arXiv preprint (2021).
[23] H. Robbins and D. Siegmund, *A convergence theorem for non negative almost supermartingales and some applications*, in *Optimizing Methods in Statistics*, J. S. Rustagi, ed., Academic Press, 1971, pp. 233–257.
[24] D.E. Rumelhart, G.E. Hinton, and R.J. Williams, *Learning representations by back-propagating errors*, Nature 323 (1986), pp. 533–536.
[25] G. Strezoski, N.v. Noord, and M. Worring, *Many task learning with task routing*, in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.
[26] M. Teichmann, M. Weber, M. Zollner, R. Cipolla, and R. Urtasun, *Multinet: Real-time joint semantic reasoning for autonomous driving* (2018). Available at https://www.repository.cam.ac.uk/handle/1810/279403.
[27] S. Vandenhende, S. Georgoulis, W. Van Gansbeke, M. Proesmans, D. Dai, and L. Van Gool, *Multi-task learning for dense prediction tasks: A survey*, IEEE Trans. Pattern Anal. Mach. Intell. 44 (2022), pp. 3614–3633.
[28] S. Woerner, A. Jaques, and C. Baumgartner, *Medimeta: A comprehensive and easy-to-use multi-domain multi-task medical imaging meta-dataset* (2024). https://doi.org/10.5281/zenodo.7884735
[29] S.J. Wright, *Coordinate descent algorithms*, Math. Program. Ser. B 151 (2015), pp. 3–34.
[30] Y. Xu and W. Yin, *Block stochastic gradient iteration for convex and nonconvex optimization*, SIAM J. Optim. 25 (2015), pp. 1686–1716.
[31] Y. Zhang and Q. Yang, *A survey on multi-task learning*, IEEE Trans. Knowl. Data Eng. 34 (2022), pp. 5586–5609.

## Appendix. Implemented ATE-SG

The pseudo-code of the implemented ATE-SG method is given in Algorithm A.1. For simplicity, we consider the total number of epochs as the only stopping criterion. For a ready-to-use version of this algorithm, see https://github.com/Fra0013To/ATEforMTNN.

**Algorithm A.1** Implemented ATE-SG

**Data:** $(\boldsymbol{w}_0, \boldsymbol{w}_{\text{ts}}) = \boldsymbol{w}$ (initial guesses for the trainable parameters), $\mathcal{T}$ (training set), $B$ (mini-batch size), $\{\eta_i, i \geq 0\}$ (learning rates), $\ell$ (loss function), $E_0, E_{\text{ts}}$ (number of epochs for alternate training), $E$ (total number of epochs).

**Procedure:**

$\boldsymbol{w}^{(0)} \leftarrow (\boldsymbol{w}_0, \boldsymbol{w}_{\text{ts}})$
$e \leftarrow 0$ (epochs counter, total)
$i \leftarrow 0$ (iteration counter)
**while** $e \leq E$ **do**
  $e_0 \leftarrow 0$ (epochs counter, shared phase)
  **while** $e_0 \leq E_0$ and $e \leq E$ (alternate training, shared phase) **do**
    $\{\mathcal{B}_1, \ldots, \mathcal{B}_t\} \leftarrow$ random split of $\mathcal{T}$ into mini-batches w.r.t. $B$
    **for** $\tau = 0, 1, \ldots, t-1$ **do**
      $\boldsymbol{w}_0 \leftarrow \boldsymbol{w}_0 - \eta_i \nabla_{\boldsymbol{w}_0} \ell(\mathcal{B}_\tau; \boldsymbol{w}^{(i)})$
      $i \leftarrow i + 1$
      $\boldsymbol{w}^{(i)} \leftarrow (\boldsymbol{w}_0, \boldsymbol{w}_{\text{ts}})$
    **end for**
    $e_0 \leftarrow e_0 + 1$ and $e \leftarrow e + 1$
  **end while**
  $e_{\text{ts}} \leftarrow 0$ (epochs counter, task-specific phase)
  **while** $e_{\text{ts}} \leq E_{\text{ts}}$ and $e \leq E$ (alternate training, task-specific phase) **do**
    $\{\mathcal{B}_1, \ldots, \mathcal{B}_t\} \leftarrow$ random split of $\mathcal{T}$ into mini-batches w.r.t. $B$
    **for** $\tau = 0, 1, \ldots, t-1$ **do**
      $\boldsymbol{w}_{\text{ts}} \leftarrow \boldsymbol{w}_{\text{ts}} - \eta_i \nabla_{\boldsymbol{w}_{\text{ts}}} \ell(\mathcal{B}_\tau; \boldsymbol{w}^{(i)})$
      $i \leftarrow i + 1$
      $\boldsymbol{w}^{(i)} \leftarrow (\boldsymbol{w}_0, \boldsymbol{w}_{\text{ts}})$
    **end for**
    $e_{\text{ts}} \leftarrow e_{\text{ts}} + 1$ and $e \leftarrow e + 1$
  **end while**
**end while**
**return** $\boldsymbol{w}^{(i)}$ (final MTNN's weights)