Jacobian Descent for Multi-Objective Optimization

Pierre Quinton *
LTHI, EPFL
pierre.quinton@epfl.ch

Valérian Rey *
SimplexLab
valerian.rey@gmail.com

Abstract

Many optimization problems require balancing multiple conflicting objectives. To generalize gradient descent, which is limited to single-objective optimization, we introduce Jacobian descent (JD) and its stochastic variants.[1] This algorithm iteratively updates parameters using the Jacobian matrix of a vector-valued objective function, in which each row is the gradient of an individual objective. While several methods to combine gradients already exist in the literature, they generally struggle when the objectives conflict. In contrast, we propose projecting gradients to fully resolve conflict. We prove convergence to the Pareto front in the smooth convex case with this approach, a stronger guarantee than the usual convergence to a weakly Pareto stationary point. Our method also enables instance-wise risk minimization (IWRM), a novel learning paradigm in which the loss of each training example is considered a separate objective. Early experimentation on image classification shows promising results for IWRM. Lastly, we provide an efficient implementation of JD using the Gramian of the Jacobian matrix to drastically reduce memory requirements.

Algorithm 1: Gradient descent

Input: $\boldsymbol{x} \in \mathbb{R}^n$, $0 < \eta$, $T \in \mathbb{N}$
for $t \leftarrow 1$ **to** T **do**
  $\boldsymbol{x} \leftarrow \boldsymbol{x} - \eta \nabla f(\boldsymbol{x})$
Output: $\boldsymbol{x}$

Algorithm 2: Jacobian descent with aggregator $\mathcal{A}$

Input: $\boldsymbol{x} \in \mathbb{R}^n$, $0 < \eta$, $T \in \mathbb{N}$, $\mathcal{A} : \mathbb{R}^{m \times n} \to \mathbb{R}^n$
for $t \leftarrow 1$ **to** T **do**
  $\boldsymbol{x} \leftarrow \boldsymbol{x} - \eta \mathcal{A}\big(\mathcal{J}\boldsymbol{f}(\boldsymbol{x})\big)$
Output: $\boldsymbol{x}$

1 Introduction

The field of multi-objective optimization studies minimization of vector-valued objective functions (Sawaragi et al., 1985; Ehrgott, 2005; Branke, 2008; Deb et al., 2016). In deep learning, a widespread approach to train a model with multiple objectives is to combine those into a scalar loss function minimized by stochastic gradient descent. While this method is simple, it comes at the expense of potentially degrading some individual objectives. Without prior knowledge of their relative importance, this is undesirable. In contrast, multi-objective optimization methods typically attempt to optimize all objectives simultaneously, without making arbitrary compromises: the goal is to find points for which no improvement can be made on some objectives without degrading others. Such points are called *Pareto optimal*, and their image by the objective function is called the *Pareto front*.

Early works have attempted to extend gradient descent (GD) to consider several objectives simultaneously, and thus several gradients (Fliege & Svaiter, 2000; Désidéri, 2012). Essentially, they propose a heuristic to prevent the degradation of any individual objective. Several other works have built upon

*Equal contribution
[1] Our library enabling JD with `PyTorch` is available at https://github.com/SimplexLab/TorchJD

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

this method, analyzing its convergence properties or extending it to a stochastic setting (Poirion et al., 2017; Mercier et al., 2018; Fliege et al., 2019). Later, this has been applied to multi-task learning to tackle conflict between tasks, illustrated by contradicting gradient directions (Sener & Koltun, 2018). Many studies have followed, proposing various other algorithms for the training of multi-task models (Yu et al., 2020; Chen et al., 2020; Liu et al., 2021a,b; Lin et al., 2021; Navon et al., 2022; Senushkin et al., 2023). They commonly rely on an *aggregator*: a function mapping the Jacobian matrix, whose rows are the gradients of the individual objectives, to a single parameter update.

We propose to unify all such methods under the *Jacobian descent* (JD) algorithm, specified by an aggregator. This algorithm aims to minimize a differentiable vector-valued function $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^m$ iteratively without relying on a scalarization of the objective. Intuitively, at each iteration, JD computes the gradient of each objective, combines these gradients into a single update direction using the aggregator, and takes a small step in that direction, just like GD does with a single gradient. The role of the aggregator is thus to find a direction that is good for all objectives at once. Under this formulation, the existing methods are simply distinguished by their aggregator. Consequently, studying its properties is essential for understanding the behavior and convergence of JD. Under significant conflict, existing aggregators often fail to provide strong convergence guarantees. To address this, we propose $\mathcal{A}_{\text{UPGrad}}$, specifically designed to resolve conflicts while naturally preserving the relative influence of individual gradients.

Furthermore, we introduce a novel stochastic variant of JD that enables the training of neural networks with a large number of objectives. This unlocks a particularly interesting perspective: considering the minimization of instance-wise loss vectors rather than the usual minimization of the average training loss. As this paradigm is a direct generalization of the well-known empirical risk minimization (ERM) (Vapnik, 1995), we name it *instance-wise risk minimization* (IWRM).

Contributions. (1) We formalize the JD algorithm and its stochastic variants. (2) We define a new aggregator $\mathcal{A}_{\text{UPGrad}}$, based on gradient projections, and satisfying important properties. (3) We show convergence of JD with $\mathcal{A}_{\text{UPGrad}}$ to the Pareto front in the smooth convex case, a stronger guarantee than the one usually provided for existing methods. (4) We define formally the IWRM learning paradigm, backed by some positive preliminary experiments. (5) We provide a novel formulation of Jacobian descent, relying on the Gramian of the Jacobian, decreasing memory usage from $\mathcal{O}(m \times n)$ to $\mathcal{O}(n + m^2)$. (6) We provide an open source `PyTorch`-based library to do Jacobian descent with any aggregator, with either the naive or the Gramian-based implementation.

2 Conceptual and theoretical foundations

To optimize a vector-valued objective, a partial order between vectors must be considered. Throughout this paper, $\preceq$ denotes the relation defined for any pair of vectors $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^m$ as $\boldsymbol{u} \preceq \boldsymbol{v}$ whenever $u_i \leq v_i$ for all coordinates i. Similarly, $\prec$ is the relation defined by $\boldsymbol{u} \prec \boldsymbol{v}$ whenever $u_i < v_i$ for all coordinates i. Furthermore, $\boldsymbol{u} \precneqq \boldsymbol{v}$ indicates that both $\boldsymbol{u} \preceq \boldsymbol{v}$ and $\boldsymbol{u} \neq \boldsymbol{v}$ hold. The Euclidean vector norm and the Frobenius matrix norm are denoted by $\|\cdot\|$ and $\|\cdot\|_{\text{F}}$, respectively. Finally, for any $m \in \mathbb{N}$, the symbol $[m]$ represents the range $\{i \in \mathbb{N} : 1 \leq i \leq m\}$.

2.1 Jacobian descent

In the following, we introduce Jacobian descent, a natural extension of gradient descent supporting the optimization of vector-valued functions.

Suppose that $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^m$ is continuously differentiable. Let $\mathcal{J}\boldsymbol{f}(\boldsymbol{x}) \in \mathbb{R}^{m \times n}$ be the Jacobian matrix of $\boldsymbol{f}$ at $\boldsymbol{x}$, i.e.

$$\mathcal{J}\boldsymbol{f}(\boldsymbol{x}) = \begin{bmatrix} \nabla f_1(\boldsymbol{x})^\top \\ \nabla f_2(\boldsymbol{x})^\top \\ \vdots \\ \nabla f_m(\boldsymbol{x})^\top \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial x_1} f_1(\boldsymbol{x}) & \frac{\partial}{\partial x_2} f_1(\boldsymbol{x}) & \cdots & \frac{\partial}{\partial x_n} f_1(\boldsymbol{x}) \\ \frac{\partial}{\partial x_1} f_2(\boldsymbol{x}) & \frac{\partial}{\partial x_2} f_2(\boldsymbol{x}) & \cdots & \frac{\partial}{\partial x_n} f_2(\boldsymbol{x}) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial}{\partial x_1} f_m(\boldsymbol{x}) & \frac{\partial}{\partial x_2} f_m(\boldsymbol{x}) & \cdots & \frac{\partial}{\partial x_n} f_m(\boldsymbol{x}) \end{bmatrix} \tag{1}$$

Given $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$, Taylor's theorem yields

$$\boldsymbol{f}(\boldsymbol{x} + \boldsymbol{y}) = \boldsymbol{f}(\boldsymbol{x}) + \mathcal{J}\boldsymbol{f}(\boldsymbol{x}) \cdot \boldsymbol{y} + o(\|\boldsymbol{y}\|), \tag{2}$$

The term $\boldsymbol{f}(\boldsymbol{x})+\mathcal{J}\boldsymbol{f}(\boldsymbol{x})\cdot\boldsymbol{y}$ is the first-order Taylor approximation of $\boldsymbol{f}(\boldsymbol{x}+\boldsymbol{y})$. Via this approximation, we aim to select a small update $\boldsymbol{y}$ that reduces $\boldsymbol{f}(\boldsymbol{x}+\boldsymbol{y})$, ideally achieving $\boldsymbol{f}(\boldsymbol{x}+\boldsymbol{y}) \preceq \boldsymbol{f}(\boldsymbol{x})$. As the approximation depends on $\boldsymbol{y}$ only through $\mathcal{J}\boldsymbol{f}(\boldsymbol{x})\cdot\boldsymbol{y}$, selecting the update based on the Jacobian is natural. A mapping $\mathcal{A}:\mathbb{R}^{m\times n}\to\mathbb{R}^n$ reducing such a matrix into a vector is called an *aggregator*. For any $J\in\mathbb{R}^{m\times n}$, $\mathcal{A}(J)$ is called the *aggregation* of J by $\mathcal{A}$.

To minimize $\boldsymbol{f}$, consider the update $\boldsymbol{y}=-\eta\mathcal{A}\big(\mathcal{J}\boldsymbol{f}(\boldsymbol{x})\big)$, where η is an appropriate step size, and $\mathcal{A}$ is an appropriate aggregator. Jacobian descent simply consists in applying this update iteratively, as shown in Algorithm 2. To put it into perspective, we also provide a minimal version of GD in Algorithm 1. Remarkably, when $m=1$, the Jacobian has a single row, so GD is a special case of JD where the aggregator is the identity.

Note that other gradient-based optimization algorithms, e.g. Adam (Kingma & Ba, 2014), can similarly be extended to the multi-objective case.

Stochastic variants. In some settings, the exact computation of the update can be prohibitively slow or even intractable. When dealing with a single objective, stochastic gradient descent (SGD) replaces the gradient $\nabla f(\boldsymbol{x})$ with some estimation. More generally, *stochastic Jacobian descent* (SJD) relies on estimates of the aggregation of the Jacobian. One approach, that we call *stochastically estimated Jacobian descent* (SEJD), is to compute and aggregate an estimation of the Jacobian. Alternatively, when the number of objectives is very large, we propose to aggregate a matrix whose rows are a random subset of the rows of the true Jacobian. We call this approach *stochastic sub-Jacobian descent* (SSJD).

2.2 Desirable properties for aggregators

An inherent challenge of multi-objective optimization is to manage conflicting objectives (Sener & Koltun, 2018; Yu et al., 2020; Liu et al., 2021a). Substituting the update $\boldsymbol{y}=-\eta\mathcal{A}\big(\mathcal{J}\boldsymbol{f}(\boldsymbol{x})\big)$ into the first-order Taylor approximation $\boldsymbol{f}(\boldsymbol{x})+\mathcal{J}\boldsymbol{f}(\boldsymbol{x})\cdot\boldsymbol{y}$ yields $\boldsymbol{f}(\boldsymbol{x})-\eta\mathcal{J}\boldsymbol{f}(\boldsymbol{x})\cdot\mathcal{A}\big(\mathcal{J}\boldsymbol{f}(\boldsymbol{x})\big)$. In particular, if $\mathbf{0}\preceq\mathcal{J}\boldsymbol{f}(\boldsymbol{x})\cdot\mathcal{A}\big(\mathcal{J}\boldsymbol{f}(\boldsymbol{x})\big)$, then no coordinate of the approximation of $\boldsymbol{f}$ will increase. A pair of vectors $\boldsymbol{x},\boldsymbol{y}\in\mathbb{R}^n$ is said to *conflict* if $\boldsymbol{x}^\top\boldsymbol{y}<0$. Hence, for a sufficiently small η, if any row of $\mathcal{J}\boldsymbol{f}(\boldsymbol{x})$ conflicts with $\mathcal{A}\big(\mathcal{J}\boldsymbol{f}(\boldsymbol{x})\big)$, the corresponding coordinate of $\boldsymbol{f}$ will increase. When minimizing $\boldsymbol{f}$, avoiding conflict between the aggregation and any gradient is thus desirable, motivating the first property.

Definition 1 (Non-conflicting)**.** Let $\mathcal{A}:\mathbb{R}^{m\times n}\to\mathbb{R}^n$ be an aggregator. If for all $J\in\mathbb{R}^{m\times n}$, $\mathbf{0}\preceq J\cdot\mathcal{A}(J)$, then $\mathcal{A}$ is said to be *non-conflicting*.

With a non-conflicting aggregator and a sufficiently small step size, no objective increases at any iteration of JD, even when the objectives strongly conflict. Without this property, some objectives can be improved at the expense of others, which is precisely what multi-objective optimization aims to avoid.

For any collection of vectors $C\subseteq\mathbb{R}^n$, the *dual cone* of C is $\{\boldsymbol{x}\in\mathbb{R}^n : \forall\boldsymbol{y}\in C, 0\leq\boldsymbol{x}^\top\boldsymbol{y}\}$ (Boyd & Vandenberghe, 2004). Notice that an aggregator $\mathcal{A}$ is non-conflicting if and only if for any J, $\mathcal{A}(J)$ is in the dual cone of the rows of J, as illustrated in Figure 1.

In a step of GD, the update scales proportionally to the gradient norm. Small gradients thus lead to small updates, and conversely, large gradients lead to large updates. To maintain coherence with GD, it would be natural that the rows of the Jacobian also contribute to the aggregation proportionally to their norm. Scaling each row of $\mathcal{J}\boldsymbol{f}(\boldsymbol{x})$ by the corresponding element of some vector $\boldsymbol{c}\in\mathbb{R}^m$ yields $\operatorname{diag}(\boldsymbol{c})\cdot\mathcal{J}\boldsymbol{f}(\boldsymbol{x})$, where $\operatorname{diag}(\boldsymbol{c})\in\mathbb{R}^{m\times m}$ is the diagonal matrix whose diagonal is $\boldsymbol{c}$. This insight can then be formalized as the following property.

Definition 2 (Linear under scaling)**.** Let $\mathcal{A}:\mathbb{R}^{m\times n}\to\mathbb{R}^n$ be an aggregator. If for all $J\in\mathbb{R}^{m\times n}$, the mapping from any $\mathbf{0}\prec\boldsymbol{c}\in\mathbb{R}^m$ to $\mathcal{A}\big(\operatorname{diag}(\boldsymbol{c})\cdot J\big)$ is linear in $\boldsymbol{c}$, then $\mathcal{A}$ is said to be *linear under scaling*.

Each gradient then contributes to the aggregation proportionally to its norm, so small gradients can neither dominate the update nor make it vanish. For instance, $\mathcal{A}_{\text{MGDA}}$, presented in Section 3, lacks this property: if any row of the Jacobian approaches zero, so does its aggregation.

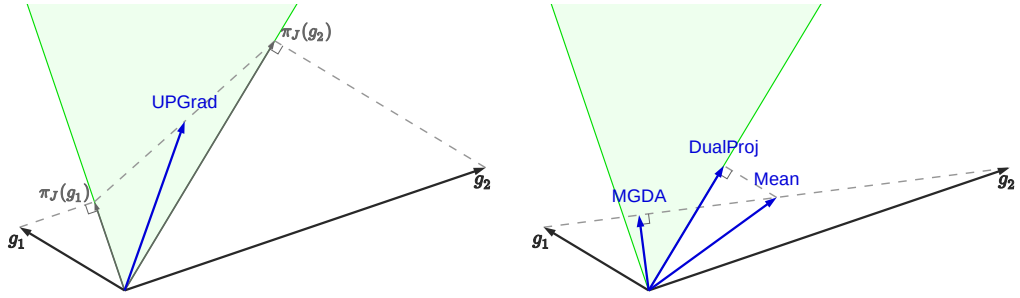
Finally, as $\|\boldsymbol{y}\|$ decreases asymptotically to 0, the precision of the first-order Taylor approximation $\boldsymbol{f}(\boldsymbol{x}) + \mathcal{J}\boldsymbol{f}(\boldsymbol{x}) \cdot \boldsymbol{y}$ improves, as highlighted in (2). The projection $\boldsymbol{y}'$ of any candidate update $\boldsymbol{y}$ onto the span of the rows of $\mathcal{J}\boldsymbol{f}(\boldsymbol{x})$ satisfies $\mathcal{J}\boldsymbol{f}(\boldsymbol{x}) \cdot \boldsymbol{y}' = \mathcal{J}\boldsymbol{f}(\boldsymbol{x}) \cdot \boldsymbol{y}$ and $\|\boldsymbol{y}'\| \leq \|\boldsymbol{y}\|$ (see Lemma 1 in Appendix C.1), so this projection decreases the norm of the update while preserving the value of the approximation. Without additional information about $\boldsymbol{f}$, it is thus reasonable to select $\boldsymbol{y}$ directly in the row span of $\mathcal{J}\boldsymbol{f}(\boldsymbol{x})$, i.e. to have a vector of weights $\boldsymbol{w} \in \mathbb{R}^m$ satisfying $\boldsymbol{y} = \mathcal{J}\boldsymbol{f}(\boldsymbol{x})^\top \cdot \boldsymbol{w}$. This yields the last desirable property.

Definition 3 (Weighted)**.** Let $\mathcal{A} : \mathbb{R}^{m \times n} \to \mathbb{R}^n$ be an aggregator. If for all $J \in \mathbb{R}^{m \times n}$, there exists $\boldsymbol{w} \in \mathbb{R}^m$ satisfying $\mathcal{A}(J) = J^\top \cdot \boldsymbol{w}$, then $\mathcal{A}$ is said to be *weighted*.

A weighted aggregator thus never wastes part of the update on directions along which, to first order, all objectives are constant. Moreover, its aggregation is entirely described by the weights $\boldsymbol{w}$, which give the contribution of each gradient to the update.

2.3 Unconflicting projection of gradients

We now define the *unconflicting projection of gradients* aggregator $\mathcal{A}_{\text{UPGrad}}$, specifically designed to be non-conflicting, linear under scaling, and weighted. In essence, it projects each gradient onto the dual cone of the rows of the Jacobian and averages the results, as illustrated in Figure 1a.



(a) $\mathcal{A}_{\text{UPGrad}}(J)$ (ours)

(b) $\mathcal{A}_{\text{Mean}}(J)$, $\mathcal{A}_{\text{MGDA}}(J)$ and $\mathcal{A}_{\text{DualProj}}(J)$

Figure 1: Aggregation of $J = [\boldsymbol{g}_1 \ \boldsymbol{g}_2]^\top \in \mathbb{R}^{2 \times 2}$ by four different aggregators. The dual cone of $\{\boldsymbol{g}_1, \boldsymbol{g}_2\}$ is represented in green.
(a) $\mathcal{A}_{\text{UPGrad}}$ projects $\boldsymbol{g}_1$ and $\boldsymbol{g}_2$ onto the dual cone and averages the results.
(b) The mean $\mathcal{A}_{\text{Mean}}(J) = \frac{1}{2}(\boldsymbol{g}_1 + \boldsymbol{g}_2)$ conflicts with $\boldsymbol{g}_1$. $\mathcal{A}_{\text{DualProj}}$ projects this mean onto the dual cone, so it lies on its boundary. $\mathcal{A}_{\text{MGDA}}(J)$ is almost orthogonal to $\boldsymbol{g}_2$ because of its larger norm.

For any $J \in \mathbb{R}^{m \times n}$ and $\boldsymbol{x} \in \mathbb{R}^n$, the *projection* of $\boldsymbol{x}$ onto the dual cone of the rows of J is

$$\pi_J(\boldsymbol{x}) = \underset{\boldsymbol{y} \in \mathbb{R}^n \colon \boldsymbol{0} \preceq J\boldsymbol{y}}{\arg\min} \ \|\boldsymbol{y} - \boldsymbol{x}\|^2. \tag{3}$$

Denoting by $\boldsymbol{e}_i \in \mathbb{R}^m$ the ith standard basis vector, $J^\top \boldsymbol{e}_i$ is the ith row of J. $\mathcal{A}_{\text{UPGrad}}$ is defined as

$$\mathcal{A}_{\text{UPGrad}}(J) = \frac{1}{m} \sum_{i \in [m]} \pi_J(J^\top \boldsymbol{e}_i). \tag{4}$$

Since the dual cone is convex, it is closed under positive combinations of its elements. For any J, $\mathcal{A}_{\text{UPGrad}}(J)$ is thus always in the dual cone of the rows of J, so $\mathcal{A}_{\text{UPGrad}}$ is non-conflicting. Note that if no pair of gradients conflicts, $\mathcal{A}_{\text{UPGrad}}$ simply averages the rows of the Jacobian.

Since π_J is a projection onto a closed convex cone, if $\boldsymbol{x} \in \mathbb{R}^n$ and $0 < a \in \mathbb{R}$, then $\pi_J(a \cdot \boldsymbol{x}) = a \cdot \pi_J(\boldsymbol{x})$ (see Lemma 5). Moreover, for any $\boldsymbol{0} \prec \boldsymbol{c} \in \mathbb{R}^m$, the dual cone of the rows of $\operatorname{diag}(\boldsymbol{c}) \cdot J$ is the same as that of the rows of J. By (4), $\mathcal{A}_{\text{UPGrad}}$ is thus linear under scaling.

When n is large, the projection in (3) is prohibitively expensive to compute. An alternative but equivalent approach is to use its dual formulation, which is independent of n.

Proposition 1. Let $J \in \mathbb{R}^{m \times n}$. For any $\boldsymbol{u} \in \mathbb{R}^m$, $\pi_J(J^\top \boldsymbol{u}) = J^\top \boldsymbol{w}$ with

$$\boldsymbol{w} \in \underset{\boldsymbol{v} \in \mathbb{R}^m \colon \boldsymbol{u} \preceq \boldsymbol{v}}{\arg\min} \ \boldsymbol{v}^\top J J^\top \boldsymbol{v}. \tag{5}$$

Proof. See Appendix C.2.

The problem defined in (5) can be solved efficiently using a quadratic programming solver, such as those bundled in `qpsolvers` (Caron et al., 2026). For any $i \in [m]$, let $\boldsymbol{w}_i$ be given by (5) when substituting $\boldsymbol{u}$ with $\boldsymbol{e}_i$. Then, by Proposition 1,

$$\mathcal{A}_{\text{UPGrad}}(J) = J^\top \left(\frac{1}{m} \sum_{i \in [m]} \boldsymbol{w}_i \right). \tag{6}$$

This provides an efficient implementation of $\mathcal{A}_{\text{UPGrad}}$ and proves that it is weighted. $\mathcal{A}_{\text{UPGrad}}$ can also be easily extended to incorporate a vector of preferences by replacing the average in (4) and (6) by a weighted sum with positive weights. This extension remains non-conflicting, linear under scaling, and weighted.

2.4 Convergence to the Pareto front

We now provide theoretical convergence guarantees of JD with $\mathcal{A}_{\text{UPGrad}}$ when minimizing some $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^m$ satisfying standard assumptions. If for a given $\boldsymbol{x} \in \mathbb{R}^n$, there exists no $\boldsymbol{y} \in \mathbb{R}^n$ satisfying $\boldsymbol{f}(\boldsymbol{y}) \precnsim \boldsymbol{f}(\boldsymbol{x})$, then $\boldsymbol{x}$ is said to be *Pareto optimal*. The set $X^* \subseteq \mathbb{R}^n$ of Pareto optimal points is called the *Pareto set*, and its image $\boldsymbol{f}(X^*)$ is called the *Pareto front*.

Whenever $\boldsymbol{f}\big(\lambda \boldsymbol{x} + (1-\lambda)\boldsymbol{y}\big) \preceq \lambda \boldsymbol{f}(\boldsymbol{x}) + (1-\lambda)\boldsymbol{f}(\boldsymbol{y})$ holds for any pair of vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ and any $\lambda \in [0, 1]$, $\boldsymbol{f}$ is said to be $\preceq$*-convex*. Moreover, $\boldsymbol{f}$ is said to be β*-smooth* whenever $\big\|\mathcal{J}\boldsymbol{f}(\boldsymbol{x}) - \mathcal{J}\boldsymbol{f}(\boldsymbol{y})\big\|_{\text{F}} \leq \beta \|\boldsymbol{x} - \boldsymbol{y}\|$ holds for any pair of vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$.

Theorem 1. Let $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^m$ be a β-smooth and $\preceq$-convex function. Suppose that the Pareto front $\boldsymbol{f}(X^*)$ is compact and that for any $\boldsymbol{x} \in \mathbb{R}^n$, there is $\boldsymbol{x}^* \in X^*$ satisfying $\boldsymbol{f}(\boldsymbol{x}^*) \preceq \boldsymbol{f}(\boldsymbol{x})$.[2] Let $\boldsymbol{x}_1 \in \mathbb{R}^n$, and for all $t \geq 1$, $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \eta \mathcal{A}_{\text{UPGrad}}\big(\mathcal{J}\boldsymbol{f}(\boldsymbol{x}_t)\big)$, with $\eta = \frac{1}{\beta\sqrt{m}}$. Let $\boldsymbol{w}_t$ be the weights defining $\mathcal{A}_{\text{UPGrad}}\big(\mathcal{J}\boldsymbol{f}(\boldsymbol{x}_t)\big)$ as per (6), i.e. $\mathcal{A}_{\text{UPGrad}}\big(\mathcal{J}\boldsymbol{f}(\boldsymbol{x}_t)\big) = \mathcal{J}\boldsymbol{f}(\boldsymbol{x}_t)^\top \cdot \boldsymbol{w}_t$. If $\boldsymbol{w}_t$ is bounded, then $\boldsymbol{f}(\boldsymbol{x}_t)$ converges to $\boldsymbol{f}(\boldsymbol{x}^*)$ for some $\boldsymbol{x}^* \in X^*$. In other words, $\boldsymbol{f}(\boldsymbol{x}_t)$ converges to the Pareto front.

Proof sketch. The proof follows the standard template used in smooth convex optimization. The first step is to show that $\boldsymbol{f}(\boldsymbol{x}_t)$ converges to some limit $\boldsymbol{f}^* \in \mathbb{R}^m$. For each t, there exists $\boldsymbol{x}_t^* \in X^*$ such that $\boldsymbol{f}(\boldsymbol{x}_t^*) \preceq \boldsymbol{f}(\boldsymbol{x}_t)$. By compactness of the Pareto front, a subsequence of $\boldsymbol{f}(\boldsymbol{x}_t^*)$ converges to some $\boldsymbol{f}(\boldsymbol{x}^*)$ with $\boldsymbol{x}^* \in X^*$, satisfying $\boldsymbol{f}(\boldsymbol{x}^*) \preceq \boldsymbol{f}^*$. The remainder of the proof shows that this inequality cannot be strict, as this would contradict the convergence of $\boldsymbol{f}(\boldsymbol{x}_t)$ to $\boldsymbol{f}^*$. Hence $\boldsymbol{f}(\boldsymbol{x}^*) = \boldsymbol{f}^*$. The full proof is given in Appendix C.3.

Another common theoretical result for multi-objective optimization is convergence to a weakly stationary point. If for a given $\boldsymbol{x} \in \mathbb{R}^n$, there exists $\boldsymbol{0} \precnsim \boldsymbol{w}$ satisfying $\mathcal{J}\boldsymbol{f}(\boldsymbol{x})^\top \boldsymbol{w} = 0$ then $\boldsymbol{x}$ is said to be *weakly Pareto stationary*. Even though every Pareto optimal point is weakly Pareto stationary, the converse does not hold, even in the convex case. Despite being necessary, convergence to a weakly Pareto stationary point is thus not a sufficient condition for optimality and, hence, constitutes a rather weak guarantee. To the best of our knowledge, $\mathcal{A}_{\text{UPGrad}}$ is the first non-conflicting aggregator that provably converges to the Pareto front in the smooth convex case.

Appendix C.4 provides additional results and discussions on the convergence rate and about guarantees in the non-convex setting.

2.5 Instance-wise risk minimization

In machine learning, we generally have access to a training set consisting of m examples. The goal of empirical risk minimization (ERM) (Vapnik, 1995) is simply to minimize the average loss over the whole training set. More generally, we introduce instance-wise risk minimization (IWRM), which considers the loss associated with each training example as a distinct objective. Formally, if $\boldsymbol{\theta} \in \mathbb{R}^n$

[2] This condition is a generalization to the case $m \geq 1$ of the existence of a minimizer $\boldsymbol{x}^* \in \mathbb{R}^n$ when $m = 1$.

are the parameters of the model and $f_i(\boldsymbol{\theta})$ is the loss associated to the ith example, the respective objective functions of ERM and IWRM are:

$$\text{(Empirical risk)} \qquad \bar{f}(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i \in [m]} f_i(\boldsymbol{\theta}) \tag{7}$$

$$\text{(Instance-wise risk)} \qquad \boldsymbol{f}(\boldsymbol{\theta}) = \begin{bmatrix} f_1(\boldsymbol{\theta}) & f_2(\boldsymbol{\theta}) & \cdots & f_m(\boldsymbol{\theta}) \end{bmatrix}^\top \tag{8}$$

Naively using GD for ERM is inefficient in most practical cases, so a prevalent alternative is to use SGD or one of its variants. Similarly, using JD for IWRM is typically intractable. Indeed, it would require computing a Jacobian matrix with one row per training example at each iteration. In contrast, we can use the Jacobian of a random batch of training example losses. Since it consists of a subset of the rows of the full Jacobian, this approach is a form of stochastic sub-Jacobian descent, as introduced in Section 2.1. IWRM can also be extended to cases where each f_i is a vector-valued function. The objective would then be the concatenation of the losses of all examples.

We discuss about more standard applications, such as multitask learning, adversarial training and distributed optimization, in Appendix B.

3 Existing aggregators

In the context of multi-task learning, several works have proposed iterative optimization algorithms based on the combination of task-specific gradients (Sener & Koltun, 2018; Yu et al., 2020; Liu et al., 2021b,a; Lin et al., 2021; Navon et al., 2022; Senushkin et al., 2023). These methods can be formulated as variants of JD parameterized by different aggregators. More specifically, since the gradients are stochastically estimated from batches of data, these are cases of what we call SEJD. In the following, we briefly present the most prominent aggregators and summarize their properties in Table 1. As a baseline, we also consider $\mathcal{A}_{\text{Mean}}$, which simply averages the rows of the Jacobian. Their formal definitions are provided in Appendix D. Some of them are also illustrated in Figure 1b.

$\mathcal{A}_{\text{RGW}}$ aggregates the matrix using a random vector of weights (Lin et al., 2021). $\mathcal{A}_{\text{MGDA}}$ gives the aggregation that maximizes the smallest improvement (Fliege & Svaiter, 2000; Désidéri, 2012; Sener & Koltun, 2018), which is a non-conflicting update. $\mathcal{A}_{\text{CAGrad}}$ maximizes the smallest improvement in a ball around the average gradient whose radius is parameterized by $c \in [0, 1[$ (Liu et al., 2021a). $\mathcal{A}_{\text{PCGrad}}$ projects each gradient onto the orthogonal hyperplane of other gradients in case of conflict, iteratively and in a random order (Yu et al., 2020), thereby removing some conflict. It is, however, only non-conflicting when $m \leq 2$, in which case $\mathcal{A}_{\text{PCGrad}} = m \cdot \mathcal{A}_{\text{UPGrad}}$. IMTL-G is a method to balance some gradients with impartiality (Liu et al., 2021b): it seeks a combination of the gradients having the same projection onto each of their directions. It is only defined for linearly independent gradients, but we generalize it as a formal aggregator, denoted $\mathcal{A}_{\text{IMTL-G}}$, in Appendix D.6. Aligned-MTL orthonormalizes the Jacobian, so that no gradient dominates the others, and weights its rows according to some preferences (Senushkin et al., 2023). We denote by $\mathcal{A}_{\text{Aligned-MTL}}$ this method with uniform preferences. $\mathcal{A}_{\text{Nash-MTL}}$ aggregates Jacobians by finding the Nash equilibrium between task-specific gradients (Navon et al., 2022), seen as players bargaining over the update direction. Lastly, the GradDrop layer (Chen et al., 2020) defines a custom backward pass that combines gradients with respect to some internal activation. The corresponding aggregator, denoted $\mathcal{A}_{\text{GradDrop}}$, randomly drops out some gradient coordinates based on their sign and sums the remaining ones.

In the context of continual learning, to limit forgetting, an idea is to project the gradient onto the dual cone of gradients computed with past examples (Lopez-Paz & Ranzato, 2017). This idea can be translated into an aggregator that projects the mean gradient onto the dual cone of the rows of the Jacobian, i.e. that finds the non-conflicting vector closest to the mean gradient. We name this $\mathcal{A}_{\text{DualProj}}$.

Several other works consider the gradients to be noisy when making their theoretical analysis (Zhou et al., 2022; Fernando et al., 2023; Chen et al., 2023; Xiao et al., 2023; Liu & Vicente, 2024). Their solutions for combining gradients are typically stateful. Although this could enhance practical convergence rates, we have restricted our focus to the analysis of stateless aggregators. Exploring and analyzing a generalized Jacobian descent algorithm, that would preserve some state over the iterations, is a promising future direction.

In the federated learning setting, several aggregators have been proposed to combine the model updates while being robust to adversaries (Blanchard et al., 2017; Chen et al., 2017; El Mhamdi et al., 2018; Yin et al., 2018). We do not study them here as they mainly focus on security aspects.

Table 1: Properties satisfied for any number of objectives. Proofs are provided in Appendix D.

Ref.	Aggregator	Non-conflicting	Linear under scaling	Weighted
—	$\mathcal{A}_{\text{Mean}}$	✗	✓	✓
Désidéri (2012)	$\mathcal{A}_{\text{MGDA}}$	✓	✗	✓
Lopez-Paz & Ranzato (2017)	$\mathcal{A}_{\text{DualProj}}$	✓	✗	✓
Yu et al. (2020)	$\mathcal{A}_{\text{PCGrad}}$	✗	✓	✓
Chen et al. (2020)	$\mathcal{A}_{\text{GradDrop}}$	✗	✗	✗
Liu et al. (2021b)	$\mathcal{A}_{\text{IMTL-G}}$	✗	✗	✓
Liu et al. (2021a)	$\mathcal{A}_{\text{CAGrad}}$	✗	✗	✓
Lin et al. (2021)	$\mathcal{A}_{\text{RGW}}$	✗	✓	✓
Navon et al. (2022)	$\mathcal{A}_{\text{Nash-MTL}}$	✓	✗	✓
Senushkin et al. (2023)	$\mathcal{A}_{\text{Aligned-MTL}}$	✗	✗	✓
(ours)	$\mathcal{A}_{\text{UPGrad}}$	✓	✓	✓

4 Experiments

Appendix A.1 shows the optimization trajectories of the methods from Table 1 when optimizing three simple convex quadratic functions. In particular, it confirms the following points: (1) convergence to a weakly Pareto stationary point is insufficient, (2) non-conflicting aggregators do not increase the value of any objective over the course of the optimization, (3) aiming for a particular point of the Pareto front can take much longer than aiming for the Pareto front in general.

In the following, we present empirical results for instance-wise risk minimization on some simple vision datasets. IWRM is performed by stochastic sub-Jacobian descent, as described in Section 2.5, with $f_i(\boldsymbol{\theta}) = \mathcal{L}_{\text{CE}}\big(\boldsymbol{g}(\boldsymbol{x}_i; \boldsymbol{\theta}), \boldsymbol{y}_i\big)$, where $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ is the i'th input/target pair, $\boldsymbol{g}(\boldsymbol{x}_i; \boldsymbol{\theta})$ is the model output, and $\mathcal{L}_{\text{CE}}$ is the cross-entropy loss. A key consideration is that when the aggregator is $\mathcal{A}_{\text{Mean}}$, this approach becomes equivalent to empirical risk minimization with SGD (indeed, the average gradient is always equal to the gradient of the average objective). It is thus used as a baseline for comparison.

We train convolutional neural networks for image classification on subsets of SVHN (Netzer et al., 2011), CIFAR-10 (Krizhevsky & Hinton, 2009), EuroSAT (Helber et al., 2019), MNIST (LeCun et al., 1998), Fashion-MNIST (Xiao et al., 2017) and Kuzushiji-MNIST (Clanuwat et al., 2018). To make the comparisons as fair as possible, we have tuned the learning rate very precisely for each aggregator, as explained in detail in Appendix E.1. We have also run the same experiments several times independently to gain confidence in our results. Since this leads to a total of 43776 training runs across all of our experiments, we have limited the size of each training dataset to 1024 images, greatly reducing computational costs. Other experimental settings, such as the network architectures and the total computational budget used to run our experiments, are given in Appendix E. Figure 2a reports the evolution of the training loss on CIFAR-10, one of the datasets with the largest performance gap. Figure 2b shows the similarity between the update of each aggregator and the update given by $\mathcal{A}_{\text{Mean}}$ (i.e. the update of SGD).

Here, we compare the aggregators in terms of their average loss over the training set: the goal of ERM. For this reason, it is rather surprising that $\mathcal{A}_{\text{Mean}}$, which directly optimizes this objective, shows a slower convergence rate than $\mathcal{A}_{\text{UPGrad}}$ and, to a lesser extent, $\mathcal{A}_{\text{DualProj}}$. This performance gap can be observed on all datasets, as reported in Appendix A.2.

For $\mathcal{A}_{\text{UPGrad}}$, a low similarity with the SGD update indicates that there are some conflicting gradients with imbalanced norms (i.e. the setting illustrated in Figure 1). Our interpretation is thus that $\mathcal{A}_{\text{UPGrad}}$ prevents gradients of hard examples from being dominated by those of easier examples early into the training. Since fitting those is more complex and time-consuming, it could be beneficial to consider

(a) CIFAR-10: training loss

(b) CIFAR-10: update similarity to the SGD update

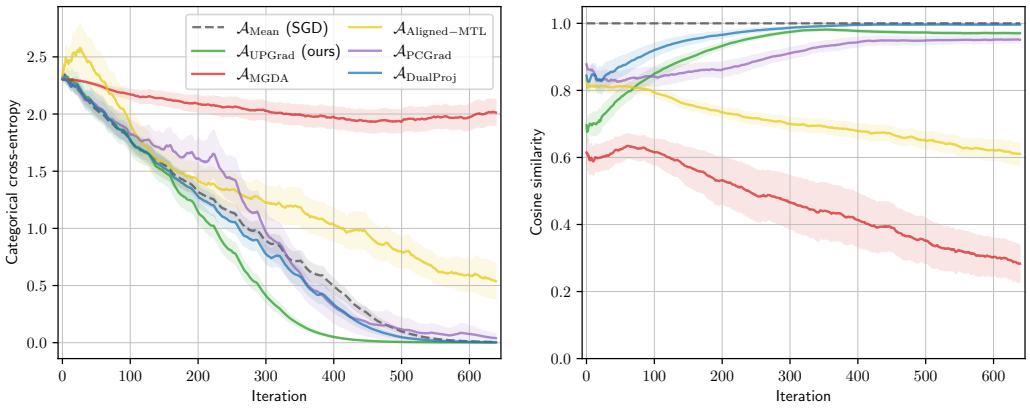


Figure 2: Optimization metrics obtained with IWRM with 1024 training examples and a batch size of 32, averaged over 8 independent runs. The shaded area around each curve shows the estimated standard error of the mean over the 8 runs. Curves are smoothed for readability. Best viewed in color.

Table 2: Test set accuracies (%) obtained by IWRM with different aggregators across datasets. The corresponding box plots are shown in Figure 12.

Aggregator	SVHN	CIFAR-10	EuroSAT	MNIST	F-MNIST	K-MNIST
$\mathcal{A}_{\text{Mean}}$	69.6	47.7	68.5	**96.3**	80.7	76.9
$\mathcal{A}_{\text{UPGrad}}$	**69.7**	**48.1**	**73.0**	96.0	80.8	**77.1**
$\mathcal{A}_{\text{MGDA}}$	41.7	32.9	62.5	85.2	73.4	53.7
$\mathcal{A}_{\text{Aligned-MTL}}$	61.3	47.3	65.3	88.3	78.5	67.9
$\mathcal{A}_{\text{PCGrad}}$	49.1	**48.1**	59.3	85.9	69.3	54.7
$\mathcal{A}_{\text{DualProj}}$	69.3	47.2	72.7	96.1	80.8	76.4
$\mathcal{A}_{\text{RGW}}$	66.2	39.5	67.9	94.0	77.1	73.6
$\mathcal{A}_{\text{GradDrop}}$	68.9	46.8	71.9	**96.3**	**80.9**	76.9
$\mathcal{A}_{\text{CAGrad}}$	60.7	42.4	68.1	90.3	78.1	65.3
$\mathcal{A}_{\text{IMTL-G}}$	58.7	44.8	65.2	87.2	75.0	65.9
$\mathcal{A}_{\text{Nash-MTL}}$	39.7	31.3	60.8	87.6	72.0	52.7

them earlier. Another possible interpretation is that $\mathcal{A}_{\text{UPGrad}}$'s updates could be safer than those of $\mathcal{A}_{\text{Mean}}$, allowing it to operate at a higher learning rate.

The sub-optimal performance of $\mathcal{A}_{\text{MGDA}}$ in this setting can be attributed to its sensitivity to small gradients: if any row of the Jacobian approaches zero, the aggregation by $\mathcal{A}_{\text{MGDA}}$ will also approach zero. This observation illustrates the discrepancy between aiming for weak stationarity and aiming for optimality, as discussed in Section 2.4. A notable advantage of linearity under scaling is to explicitly prevent this from happening. Although $\mathcal{A}_{\text{PCGrad}}$ is also based on gradient projections, its stochastic nature and lack of the non-conflicting property appear to result in instability and significantly worse performance compared to $\mathcal{A}_{\text{UPGrad}}$.

Table 2 reports the test set accuracies, showing that IWRM with $\mathcal{A}_{\text{UPGrad}}$ can also lead to a better generalization. While increasing the batch size in SGD reduces variance, the effect of doing so in SSJD combined with $\mathcal{A}_{\text{UPGrad}}$ is non-trivial, as it also tightens the dual cone. Additional results obtained when varying the batch size or updating the parameters with the `Adam` optimizer are available in Appendices A.4 and A.5, respectively. Appendix F provides memory usage and computation time considerations for our methods.

This experiment provides an example of a setting, that is traditionally considered as single-objective, benefitting from a multi-objective formulation. Even though more work is required to reduce IWRM's overhead and to validate it on more realistic learning tasks, we believe that it deserves further attention.

5 Gramian-based Jacobian descent

When the number of objectives is dominated by the number of parameters of the model, the main overhead of JD comes from storing the Jacobian matrix, m times larger than a single gradient, rather than the gradient itself. In the following, we introduce an alternative implementation of JD that only requires the $m(m+1)/2$ inner products between each pair of gradients.

For any $J \in \mathbb{R}^{m \times n}$, the matrix $G = JJ^\top$ is called the *Gramian* of J and is positive semi-definite. Let $\mathcal{M}_m \subseteq \mathbb{R}^{m \times m}$ be the set of positive semi-definite matrices. The Gramian of the Jacobian, denoted $\mathcal{G}\boldsymbol{f}(\boldsymbol{x}) = \mathcal{J}\boldsymbol{f}(\boldsymbol{x}) \cdot \mathcal{J}\boldsymbol{f}(\boldsymbol{x})^\top \in \mathcal{M}_m$, contains all the inner products between pairs of gradients. It thus includes the gradient norms, and captures the conflict between them. Whenever $\mathcal{A}$ is a weighted aggregator, the update of JD is $\boldsymbol{y} = -\eta \mathcal{J}\boldsymbol{f}(\boldsymbol{x})^\top \boldsymbol{w}$ for some vector of weights $\boldsymbol{w} \in \mathbb{R}^m$. Substituting this into the Taylor approximation of (2) gives (see Appendix C.5)

$$\boldsymbol{f}(\boldsymbol{x} + \boldsymbol{y}) = \boldsymbol{f}(\boldsymbol{x}) - \eta \mathcal{G}\boldsymbol{f}(\boldsymbol{x}) \cdot \boldsymbol{w} + o\left(\eta \sqrt{\boldsymbol{w}^\top \cdot \mathcal{G}\boldsymbol{f}(\boldsymbol{x}) \cdot \boldsymbol{w}}\right). \tag{9}$$

This expression only depends on the Jacobian through its Gramian. It is thus sensible to focus on aggregators whose weights are only a function of the Gramian. Denoting this function as $\mathcal{W} : \mathcal{M}_m \to \mathbb{R}^m$, those aggregators satisfy $\mathcal{A}(J) = J^\top \cdot \mathcal{W}(G)$. Remarkably, all weighted aggregators of Table 1 can be expressed in this form. In the case of $\mathcal{A}_{\text{UPGrad}}$, this is clearly demonstrated in Proposition 1, which shows that the weights depend on G. For such aggregators, substitution and linearity of differentiation then yield (see Appendix C.5), with $\mathcal{W}\big(\mathcal{G}\boldsymbol{f}(\boldsymbol{x})\big)$ treated as a constant during differentiation,

$$\mathcal{A}\big(\mathcal{J}\boldsymbol{f}(\boldsymbol{x})\big) = \nabla\Big(\mathcal{W}\big(\mathcal{G}\boldsymbol{f}(\boldsymbol{x})\big)^\top \cdot \boldsymbol{f}\Big)(\boldsymbol{x}). \tag{10}$$

After computing $\mathcal{W}\big(\mathcal{G}\boldsymbol{f}(\boldsymbol{x})\big)$, a step of JD would thus only require the backpropagation of a scalar function. The computational cost of applying $\mathcal{W}$ depends on the aggregator and is often dominated by the cost of computing the Gramian.

We now outline a method to compute the Gramian of the Jacobian without ever having to store the full Jacobian in memory. Similarly to the backpropagation algorithm, we can use the chain rule. Let $\boldsymbol{g} : \mathbb{R}^n \to \mathbb{R}^k$ and $\boldsymbol{f} : \mathbb{R}^k \to \mathbb{R}^m$, then for any $\boldsymbol{x} \in \mathbb{R}^n$, the chain rule for Gramians is

$$\mathcal{G}(\boldsymbol{f} \circ \boldsymbol{g})(\boldsymbol{x}) = \mathcal{J}\boldsymbol{f}\big(\boldsymbol{g}(\boldsymbol{x})\big) \cdot \mathcal{G}\boldsymbol{g}(\boldsymbol{x}) \cdot \mathcal{J}\boldsymbol{f}\big(\boldsymbol{g}(\boldsymbol{x})\big)^\top. \tag{11}$$

Moreover, when the function has multiple inputs, the Gramian can be computed as a sum of individual Gramians. Let $\boldsymbol{f} : \mathbb{R}^{n_1 + \cdots + n_k} \to \mathbb{R}^m$ and $\boldsymbol{x} = \begin{bmatrix}\boldsymbol{x}_1^\top & \cdots & \boldsymbol{x}_k^\top\end{bmatrix}^\top$. We can write $\mathcal{J}\boldsymbol{f}(\boldsymbol{x})$ as the concatenation of Jacobians $\begin{bmatrix}\mathcal{J}_{\boldsymbol{x}_1}\boldsymbol{f}(\boldsymbol{x}) & \cdots & \mathcal{J}_{\boldsymbol{x}_k}\boldsymbol{f}(\boldsymbol{x})\end{bmatrix}$, where $\mathcal{J}_{\boldsymbol{x}_i}\boldsymbol{f}(\boldsymbol{x})$ is the Jacobian of $\boldsymbol{f}$ with respect to $\boldsymbol{x}_i$ evaluated at $\boldsymbol{x}$. For any $i \in [k]$, let $\mathcal{G}_{\boldsymbol{x}_i}\boldsymbol{f}(\boldsymbol{x}) = \mathcal{J}_{\boldsymbol{x}_i}\boldsymbol{f}(\boldsymbol{x}) \cdot \mathcal{J}_{\boldsymbol{x}_i}\boldsymbol{f}(\boldsymbol{x})^\top$. Then

$$\mathcal{G}\boldsymbol{f}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k) = \sum_{i \in [k]} \mathcal{G}_{\boldsymbol{x}_i}\boldsymbol{f}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k). \tag{12}$$

When a function is made of compositions and concatenations of elementary functions, the Gramian of the Jacobian can thus be expressed with sums and products of partial Jacobians.

We now provide an example algorithm to compute the Gramian of a sequence of layers. For $0 \leq i < k$, let $\boldsymbol{f}_i : \mathbb{R}^{n_i} \times \mathbb{R}^{\ell_i} \to \mathbb{R}^{n_{i+1}}$ be a layer parameterized by $\boldsymbol{p}_i \in \mathbb{R}^{\ell_i}$. Given $\boldsymbol{x}_0 \in \mathbb{R}^{n_0}$, for $0 \leq i < k$, the activations are recursively defined as $\boldsymbol{x}_{i+1} = \boldsymbol{f}_i(\boldsymbol{x}_i, \boldsymbol{p}_i)$. Algorithm 3 illustrates how (11) and (12) can be combined to compute the Gramian of the network with respect to its parameters.

Algorithm 3: *Gramian reverse accumulation* for a sequence of layers

$J_x \leftarrow I$ `# Identity matrix of size` $n_k \times n_k$
$G \leftarrow 0$ `# Zero matrix of size` $n_k \times n_k$
for $i \leftarrow k-1$ **to** 0 **do**
  $J_p \leftarrow J_x \cdot \mathcal{J}_{\boldsymbol{p}_i}\boldsymbol{f}_i(\boldsymbol{x}_i, \boldsymbol{p}_i)$ `# Jacobian of` $\boldsymbol{x}_k$ `w.r.t.` $\boldsymbol{p}_i$
  $G \leftarrow G + J_p J_p^\top$
  $J_x \leftarrow J_x \cdot \mathcal{J}_{\boldsymbol{x}_i}\boldsymbol{f}_i(\boldsymbol{x}_i, \boldsymbol{p}_i)$ `# Jacobian of` $\boldsymbol{x}_k$ `w.r.t.` $\boldsymbol{x}_i$
Output: G

Algorithm 3 is analogous to the backpropagation of a gradient, except that it backpropagates the Jacobian J_x of $\boldsymbol{x}_k$ with respect to the current activation, so vector-Jacobian products become Jacobian-Jacobian products. Whenever the Jacobian J_p with respect to the parameters of a layer is obtained, its Gramian is accumulated into G, as justified by (12), and J_p can then be freed, so the full Jacobian is never stored.

Our library provides a working implementation of Algorithm 3 supporting any computational graph (rather than just sequential models). Note that it also provides a Gramian-based weighting implementation for most weighted aggregators, to make them compatible with this algorithm.

The practical memory usage and computation time obtained with this algorithm on the IWRM experiment described in Section 4 are provided in Appendix F. We can see that SSJD with $\mathcal{A}_{\text{UPGrad}}$ becomes between 42% and 89% faster with the Gramian-based implementation compared to the naive implementation, despite being mathematically equivalent. This implementation also leads to a reduction of the memory usage between 87% and 93% depending on the architecture. It remains an open challenge to make this method even more efficient, and in particular to make it handle the block-diagonal sparsity of J_x when using it for IWRM.

6 Conclusion

We introduced Jacobian descent and its stochastic variants, together with a novel non-conflicting aggregator that provides stronger convergence guarantees than prior approaches. Through a Gramian-based implementation, we significantly reduced the memory and time overhead of existing methods. Combined with our library, these contributions should provide a stronger conceptual foundation for future works and largely improve the practical usability of such methods.

Limitations and future directions. The empirical evaluation of Jacobian descent is limited to synthetic functions and to IWRM, a setting with moderately conflicting objectives. An essential future direction is thus to develop proper benchmarks to compare aggregators on a wider variety of real learning tasks. Also, the efficiency of SSJD on the IWRM objective, and of $\mathcal{A}_{\text{UPGrad}}$ in general, can still be largely improved through the development of efficient block-diagonal sparse tensors and of a GPU-based quadratic program solver, respectively.

Acknowledgments

We would like to express our sincere thanks to Scott Pesme, Emre Telatar, Matthieu Buot de l'Épine, Adrien Vandenbroucque, Ye Zhu, Alix Jeannerot, and Damian Dudzicz for their careful and thorough review. The many insightful discussions that we shared with them were essential to this project. We are also grateful to Khush Patel, Praneth Paruchuri, and Rasa Khosrowshahi, our most dedicated open-source contributors to TorchJD.

This work received no external funding. The authors declare no competing interests.

References

Tameem Adel, Isabel Valera, Zoubin Ghahramani, and Adrian Weller. One-network adversarial fairness. In *AAAI Conference on Artificial Intelligence*, volume 33, pp. 2412–2420, 2019.

Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Stephen P Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

Jürgen Branke. *Multiobjective Optimization: Interactive and Evolutionary Approaches*. Springer Science & Business Media, 2008.

Stéphane Caron, Daniel Arnström, Suraj Bonagiri, Antoine Dechaume, Nikolai Flowers, Adam Heins, Takuma Ishikawa, Dustin Kenefake, Giacomo Mazzamuto, Donato Meoli, Brendan O'Donoghue, Adam A. Oppenheimer, Pavel Otta, Abhishek Pandala, Juan José Quiroz Omaña, Nikitas Rontsis,

Paarth Shah, Samuel St-Jean, Nicola Vitucci, Soeren Wolfers, Fengyu Yang, Brian Delhaisse, MeindertHH, rimaddo, urob, shaoanlu, Uriel Sandoval, Ahmed Khalil, Lev Kozlov, Antoine Groudiev, João Sousa Pinto, Roland Schwan, Rohan Budhiraja, Johannes Keustermans, Yubin Wu, and Hongpei Li. qpsolvers: Quadratic Programming Solvers in Python, 2026. URL https://github.com/qpsolvers/qpsolvers.

Lisha Chen, Heshan Fernando, Yiming Ying, and Tianyi Chen. Three-way trade-off in multi-objective learning: Optimization, generalization and conflict-avoidance. In *Advances in Neural Information Processing Systems*, volume 36, pp. 70045–70093, 2023.

Yudong Chen, Lili Su, and Jiaming Xu. Distributed statistical machine learning in adversarial settings: Byzantine gradient descent. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 1(2):1–25, 2017.

Zhao Chen, Jiquan Ngiam, Yanping Huang, Thang Luong, Henrik Kretzschmar, Yuning Chai, and Dragomir Anguelov. Just pick a sign: Optimizing deep multitask models with gradient sign dropout. In *Advances in Neural Information Processing Systems*, volume 33, pp. 2039–2050, 2020.

Tarin Clanuwat, Mikel Bober-Irizar, Asanobu Kitamoto, Alex Lamb, Kazuaki Yamamoto, and David Ha. Deep learning for classical Japanese literature. In *NeurIPS Workshop on Machine Learning for Creativity and Design*, 2018.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (ELUs). *arXiv preprint arXiv:1511.07289*, 2015.

Kalyanmoy Deb, Karthik Sindhya, and Jussi Hakanen. Multi-objective optimization. In *Decision Sciences*, pp. 161–200. CRC Press, 2016.

Timothy Dozat. Incorporating Nesterov momentum into Adam. In *International Conference on Learning Representations Workshop*, 2016.

Jean-Antoine Désidéri. Multiple-gradient descent algorithm (MGDA) for multiobjective optimization. *Comptes Rendus Mathematique*, 350(5-6):313–318, 2012.

Matthias Ehrgott. *Multicriteria Optimization*. Springer Science & Business Media, 2005.

El Mahdi El Mhamdi, Rachid Guerraoui, and Sébastien Rouault. The hidden vulnerability of distributed learning in Byzantium. In *International Conference on Machine Learning*, pp. 3521–3530, 2018.

Heshan Devaka Fernando, Han Shen, Miao Liu, Subhajit Chaudhury, Keerthiram Murugesan, and Tianyi Chen. Mitigating gradient bias in multi-objective learning: A provably convergent approach. In *International Conference on Learning Representations*, 2023.

Jörg Fliege, A Ismael F Vaz, and Luís Nunes Vicente. Complexity of gradient descent for multiobjective optimization. *Optimization Methods and Software*, 34(5):949–959, 2019.

Jörg Fliege and Benar Fux Svaiter. Steepest descent methods for multicriteria optimization. *Mathematical Methods of Operations Research*, 51(3):479–494, 2000.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario March, and Victor Lempitsky. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(59):1–35, 2016.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, volume 27, 2014.

Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. EuroSAT: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12(7):2217–2226, 2019.

Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaid Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konečný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Hang Qi, Daniel Ramage, Ramesh Raskar, Mariana Raykova, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2): 1–210, 2021.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Vitaly Kurin, Alessandro De Palma, Ilya Kostrikov, Shimon Whiteson, and Pawan K Mudigonda. In defense of the unitary scalarization for deep multi-task learning. In *Advances in Neural Information Processing Systems*, volume 35, pp. 12169–12183, 2022.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Baijiong Lin, Feiyang Ye, Yu Zhang, and Ivor W Tsang. Reasonable effectiveness of random weighting: A litmus test for multi-task learning. *arXiv preprint arXiv:2111.10603*, 2021.

Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. Conflict-averse gradient descent for multi-task learning. In *Advances in Neural Information Processing Systems*, volume 34, pp. 18878–18890, 2021a.

Liyang Liu, Yi Li, Zhanghui Kuang, Jing-Hao Xue, Yimin Chen, Wenming Yang, Qingmin Liao, and Wayne Zhang. Towards impartial multi-task learning. In *International Conference on Learning Representations*, 2021b.

Suyun Liu and Luis Nunes Vicente. The stochastic multi-gradient algorithm for multi-objective optimization and its application to supervised machine learning. *Annals of Operations Research*, 339(3):1119–1148, 2024.

David Lopez-Paz and Marc' Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Quentin Mercier, Fabrice Poirion, and Jean-Antoine Désidéri. A stochastic multiple gradient descent algorithm. *European Journal of Operational Research*, 271(3):808–817, 2018.

Aviv Navon, Aviv Shamsian, Idan Achituve, Haggai Maron, Kenji Kawaguchi, Gal Chechik, and Ethan Fetaya. Multi-task learning as a bargaining game. In *International Conference on Machine Learning*, pp. 16428–16446, 2022.

Yurii Nesterov. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. *Proceedings of the USSR Academy of Sciences*, 269(3):543–547, 1983.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Fabrice Poirion, Quentin Mercier, and Jean-Antoine Désidéri. Descent algorithm for nonsmooth stochastic multiobjective optimization. *Computational Optimization and Applications*, 68(2): 317–331, 2017.

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

Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

Yoshikazu Sawaragi, Hirotaka Nakayama, and Tetsuzo Tanino. *Theory of Multiobjective Optimization*. Elsevier, 1985.

Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Dmitry Senushkin, Nikolay Patakin, Arseny Kuznetsov, and Anton Konushin. Independent component alignment for multi-task learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20083–20093, 2023.

Vladimir Naumovich Vapnik. *The Nature of Statistical learning theory*. Springer-Verlag, 1995.

Weiyun Wang, Zhe Chen, Wenhai Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Jinguo Zhu, Xizhou Zhu, Lewei Lu, Yu Qiao, and Jifeng Dai. Enhancing the reasoning ability of multimodal large language models via mixed preference optimization. *arXiv preprint arXiv:2411.10442*, 2024.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Peiyao Xiao, Hao Ban, and Kaiyi Ji. Direction-oriented multi-objective learning: Simple and provable stochastic algorithms. In *Advances in Neural Information Processing Systems*, volume 36, pp. 4509–4533, 2023.

Derrick Xin, Behrooz Ghorbani, Justin Gilmer, Ankush Garg, and Orhan Firat. Do current multi-task optimization methods in deep learning even help? In *Advances in Neural Information Processing Systems*, volume 35, pp. 13597–13609, 2022.

Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. In *International Conference on Machine Learning*, pp. 5650–5659, 2018.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 5824–5836, 2020.

Shiji Zhou, Wenpeng Zhang, Jiyan Jiang, Wenliang Zhong, Jinjie Gu, and Wenwu Zhu. On the convergence of stochastic multi-objective gradient manipulation and beyond. In *Advances in Neural Information Processing Systems*, volume 35, pp. 38103–38115, 2022.

A Additional experimental results

In this appendix, we provide additional experimental results about Jacobian descent and IWRM.

A.1 Optimization trajectories

Figures 3, 4 and 5 show the optimization trajectories of the aggregators of Table 1 for three different convex quadratic functions, with several initializations.[3]

The first function is the element-wise quadratic function $\boldsymbol{f}_{\text{EWQ}} : \mathbb{R}^2 \to \mathbb{R}^2$, defined as

$$\boldsymbol{f}(\boldsymbol{x}) = \begin{bmatrix} x_1^2 & x_2^2 \end{bmatrix}^\top. \tag{13}$$

Notably, its Pareto set only contains the origin, while its set of weakly Pareto stationary points is the union of the two axes $x_1 = 0$ and $x_2 = 0$. The Jacobian of $\boldsymbol{f}$ at $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ is $\begin{bmatrix} 2x_1 & 0 \\ 0 & 2x_2 \end{bmatrix}$, indicating that the gradients never conflict, which makes this function relatively simple to optimize. Nevertheless, Figure 3 shows that $\mathcal{A}_{\text{MGDA}}$, $\mathcal{A}_{\text{IMTL-G}}$ and $\mathcal{A}_{\text{CAGrad}}$ fail to converge to the Pareto set for some initializations. For instance, $\mathcal{A}_{\text{MGDA}}$ converges to weakly Pareto stationary points, which can be arbitrarily far from the Pareto set. This trivial example shows the importance of proving convergence to the Pareto front (as we do for $\mathcal{A}_{\text{UPGrad}}$ in Section 2.4) rather than to a weakly Pareto stationary point.

The second function is a convex quadratic function $\boldsymbol{g} : \mathbb{R}^2 \to \mathbb{R}^2$, constructed to sometimes reproduce the setting of Figure 1 with conflicting and imbalanced gradients:

$$\boldsymbol{g}(\boldsymbol{x}) = \begin{bmatrix} (\boldsymbol{x} - \boldsymbol{v}_1)^\top \cdot A_1 \cdot (\boldsymbol{x} - \boldsymbol{v}_1) \\ (\boldsymbol{x} - \boldsymbol{v}_2)^\top \cdot A_2 \cdot (\boldsymbol{x} - \boldsymbol{v}_2) \end{bmatrix} \tag{14}$$

$$\text{with } A_1 = \begin{bmatrix} 1 & 0.2 \\ 0.2 & 0.05 \end{bmatrix}, A_2 = \begin{bmatrix} 3 & -0.6 \\ -0.6 & 0.2 \end{bmatrix} \tag{15}$$

$$\boldsymbol{v}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \text{and } \boldsymbol{v}_2 = \begin{bmatrix} -1 \\ 0 \end{bmatrix} \tag{16}$$

Figure 4 shows that most existing aggregators have some unusual behavior in this setting. In particular, on Figure 4b, we can see that with $\mathcal{A}_{\text{Mean}}$, $\mathcal{A}_{\text{GradDrop}}$, $\mathcal{A}_{\text{IMTL-G}}$, $\mathcal{A}_{\text{CAGrad}}$, $\mathcal{A}_{\text{RGW}}$ and $\mathcal{A}_{\text{Aligned-MTL}}$, the updates can make one objective's value increase. In contrast, with the non-conflicting aggregators $\mathcal{A}_{\text{MGDA}}$, $\mathcal{A}_{\text{DualProj}}$, $\mathcal{A}_{\text{Nash-MTL}}$ and $\mathcal{A}_{\text{UPGrad}}$, no objective value increases during the optimization. Due to its linearity under scaling, $\mathcal{A}_{\text{UPGrad}}$ also shows improved smoothness compared to the other non-conflicting aggregators.

The third function, $\boldsymbol{h} : \mathbb{R}^2 \to \mathbb{R}^2$, is an even simpler convex quadratic function with conflicting gradients, but with even greater gradient norm imbalance.

$$\boldsymbol{h}(\boldsymbol{x}) = \begin{bmatrix} (\boldsymbol{x} - \boldsymbol{u}_1)^\top \cdot A \cdot (\boldsymbol{x} - \boldsymbol{u}_1) \\ (\boldsymbol{x} - \boldsymbol{u}_2)^\top \cdot 10A \cdot (\boldsymbol{x} - \boldsymbol{u}_2) \end{bmatrix} \tag{17}$$

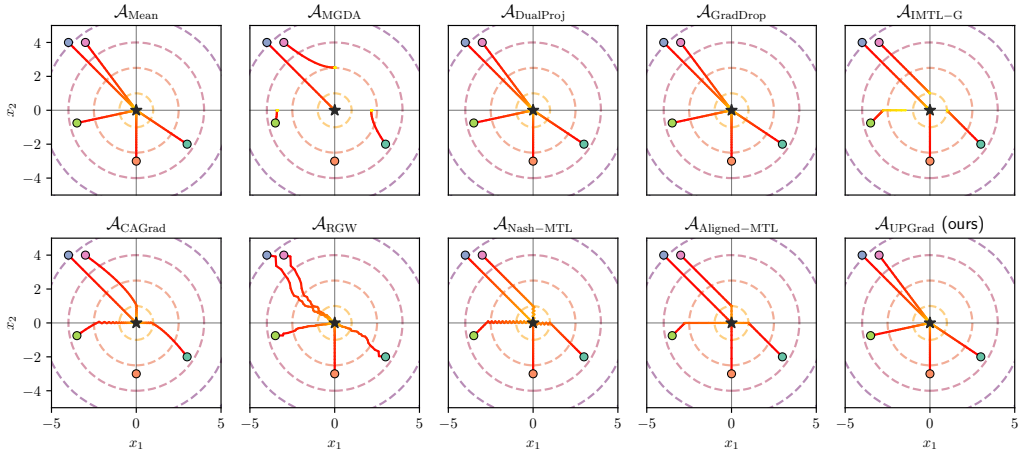
$$\text{with } A = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix} \tag{18}$$

$$\boldsymbol{u}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \text{and } \boldsymbol{u}_2 = \begin{bmatrix} -10 \\ 0 \end{bmatrix} \tag{19}$$

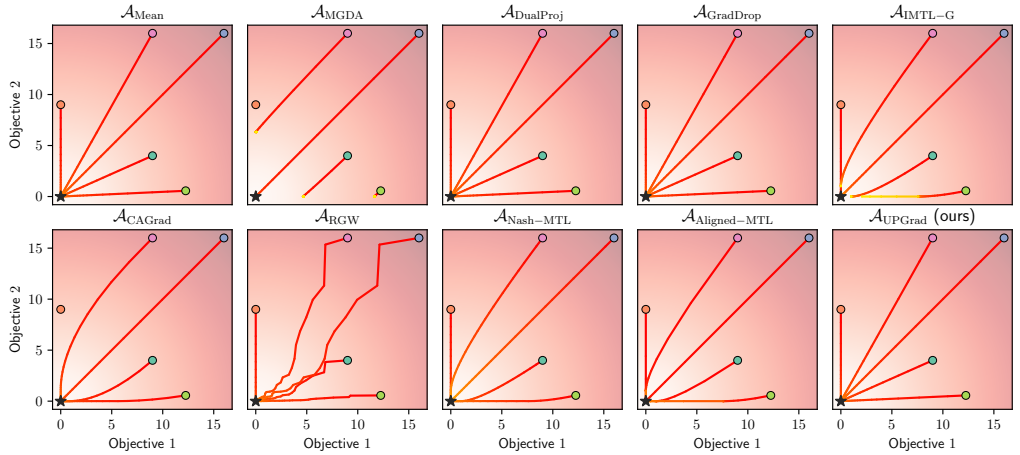
Figure 5 shows that in such a setting, $\mathcal{A}_{\text{Mean}}$ can take a much longer trajectory to reach the Pareto front compared to $\mathcal{A}_{\text{UPGrad}}$. This shows that the advantage of aiming for the Pareto front in general rather than one of its points, can sometimes be very large. Note that the gap between $\mathcal{A}_{\text{Mean}}$ and $\mathcal{A}_{\text{UPGrad}}$ can be arbitrarily large if the gradient imbalance increases.

[3] We do not show the trajectories of $\mathcal{A}_{\text{PCGrad}}$ because it is equivalent to $2 \cdot \mathcal{A}_{\text{UPGrad}}$ when there are only two objectives.

(a) Parameter space. Black star: Pareto set. Dashed lines: contour lines of the mean objective.



(b) Objective space. Black star: Pareto front. Background color: distance to the Pareto front.



(c) Distance to Pareto front over the iterations for each initial point. Each line has the color of the corresponding initial point in the trajectory plots.

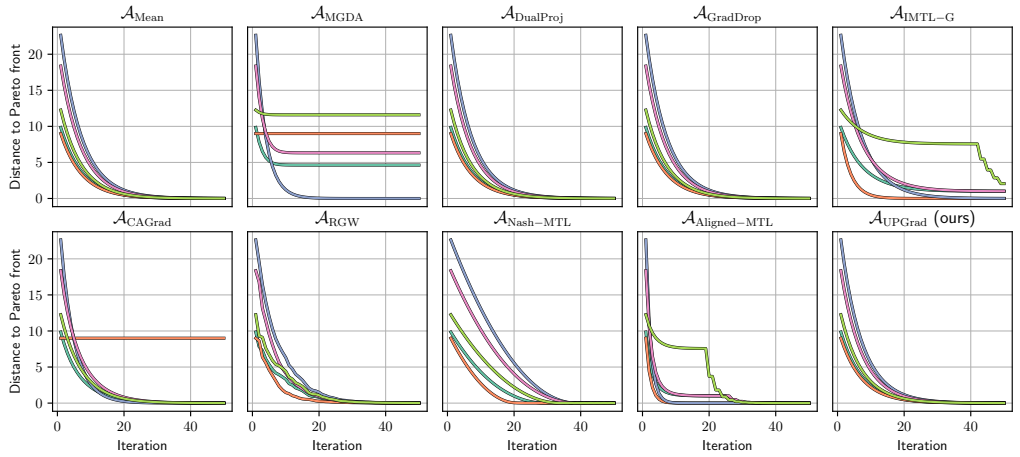
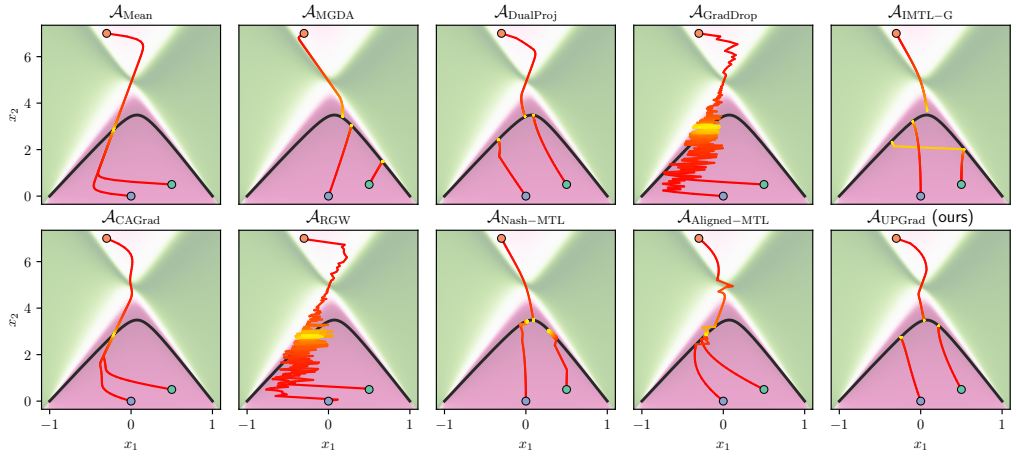
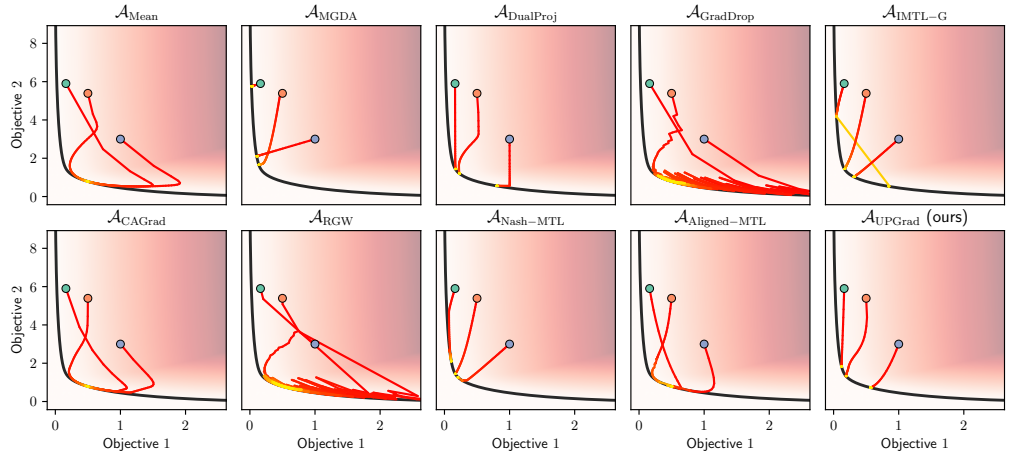


Figure 3: Optimization trajectories of various aggregators when optimizing $\boldsymbol{f} : \begin{bmatrix} x_1 & x_2 \end{bmatrix}^\top \mapsto \begin{bmatrix} x_1^2 & x_2^2 \end{bmatrix}^\top$ with JD. Colored dots represent the initial points. The trajectories start in red and evolve towards yellow.

(a) Parameter space. Black line: Pareto set. Background color: cosine similarity between the two gradients, between -1 (fully conflicting) in pink and 1 (fully aligned) in green. For instance, the similarity is -0.925 at the initial point $\begin{bmatrix}0 & 0\end{bmatrix}^\top$ and is -1 on (and only on) the Pareto set.



(b) Objective space. Black line: Pareto front. Background color: distance to the Pareto front.



(c) Distance to Pareto front over the iterations for each initial point.

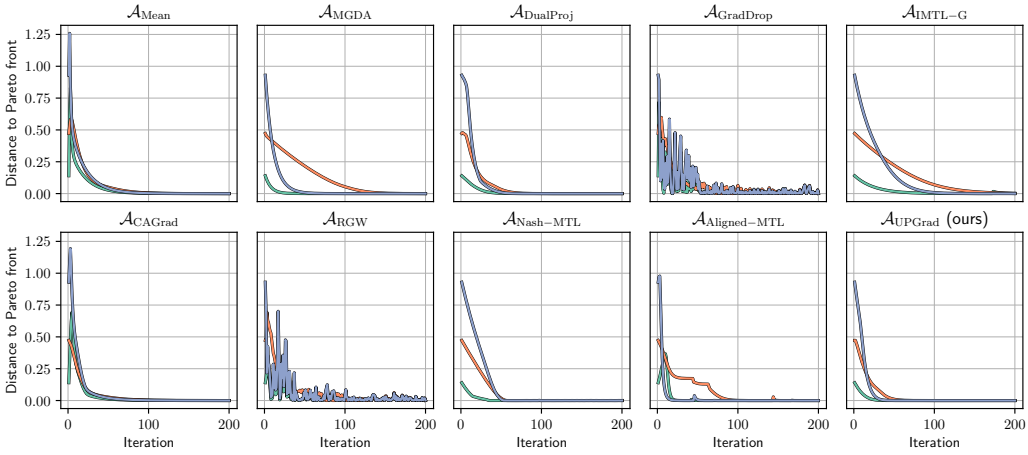
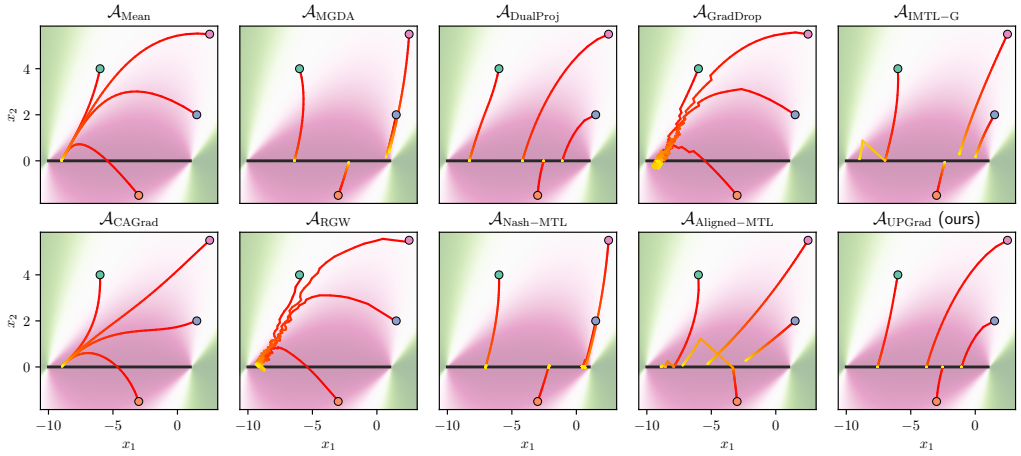
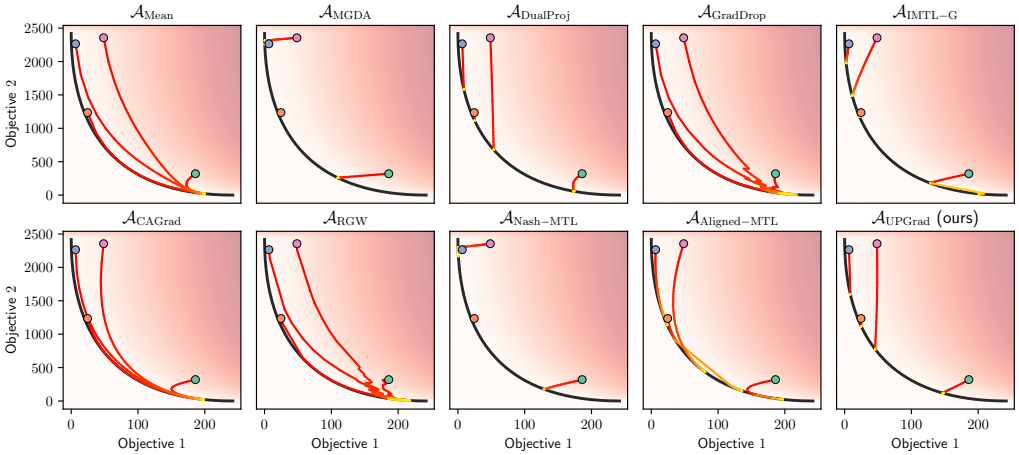


Figure 4: Optimization trajectories of various aggregators when optimizing the convex quadratic function $\boldsymbol{g}$ of (14) with JD. Colored dots represent initial parameter values.

(a) Parameter space. Black line: Pareto set. Background color: cosine similarity between the two gradients, between -1 (fully conflicting) in pink and 1 (fully aligned) in green.



(b) Objective space. Black line: Pareto front. Background color: distance to the Pareto front.



(c) Distance to Pareto front over the iterations for each initial point. Each line has the color of the corresponding initial point in the trajectory plots.

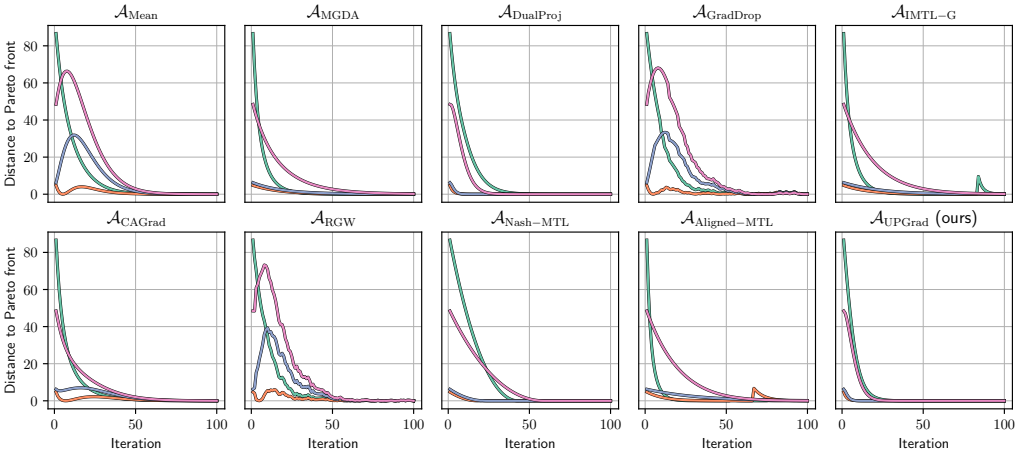


Figure 5: Optimization trajectories of various aggregators when optimizing the convex quadratic function $\boldsymbol{h}$ of (17) with JD. Colored dots represent initial parameter values. The trajectories start in red and evolve towards yellow.

A.2 All datasets and all aggregators

Figures 6, 7, 8, 9, 10 and 11 show the full results of the experiments described in Section 4 on SVHN, CIFAR-10, EuroSAT, MNIST, Fashion-MNIST and Kuzushiji-MNIST, respectively. For readability, the results are displayed on three different plots for each dataset. We always show $\mathcal{A}_{\text{UPGrad}}$ and $\mathcal{A}_{\text{Mean}}$ for reference. The exact experimental settings are described in Appendix E.

It should be noted that some of these aggregators were not developed as general-purpose aggregators, but mainly for the use case of multi-task learning, with one gradient per task. Our experiments present a more challenging setting than multi-task learning optimization because conflict between rows of the Jacobian is typically higher.

We can see that $\mathcal{A}_{\text{UPGrad}}$ provides a significant improvement over $\mathcal{A}_{\text{Mean}}$ on all datasets. Moreover, the performance gaps seem to be linked to the difficulty of the dataset, which suggests that experimenting with harder tasks is a promising future direction.

A.3 Test set accuracies

Figure 12 shows the distribution, over the 8 independent runs described in Appendix E.2, of the test set accuracies reported in Table 2, with one subfigure per dataset. This makes it easier to compare aggregators and to assess the significance of the differences between them.

A.4 Varying the batch size

Figure 13 shows the results on CIFAR-10 with $\mathcal{A}_{\text{UPGrad}}$ when varying the batch size from 4 to 64. Concretely, because we are using SSJD, this makes the number of rows of the sub-Jacobian aggregated at each step vary from 4 to 64. Recall that IWRM with SSJD and $\mathcal{A}_{\text{Mean}}$ is equivalent to ERM with SGD. We observe that with a small batch size, $\mathcal{A}_{\text{UPGrad}}$ becomes very similar to $\mathcal{A}_{\text{Mean}}$. This is not surprising since both would be equivalent with a batch size of 1. Conversely, a larger batch size increases the gap between $\mathcal{A}_{\text{UPGrad}}$ and $\mathcal{A}_{\text{Mean}}$. Since the projections of $\mathcal{A}_{\text{UPGrad}}$ are onto the dual cone of more rows, each step becomes non-conflicting with respect to more of the original 1024 objectives, pushing even further the benefits of the non-conflicting property. In other words, increasing the batch size refines the dual cone, thereby improving the quality of the projections.

A.5 Compatibility with Adam

Figure 14 gives the results on CIFAR-10 and SVHN when using `Adam` rather than the `SGD` optimizer. Concretely, this corresponds to the Adam algorithm in which the gradient is replaced by the aggregation of the Jacobian. The learning rate is still tuned as described in Appendix E.1, but the other hyperparameters of Adam are fixed to the default values of `PyTorch`, i.e. $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. Because optimization with Adam is faster, the number of epochs for SVHN and CIFAR-10 is reduced to 20 and 15, respectively. While the performance gap is smaller with this optimizer, it is still significant and suggests that our methods are beneficial with other optimizers than the simple `SGD`. The thorough investigation of the interplay between aggregators and momentum-based optimizers is a compelling future research direction.

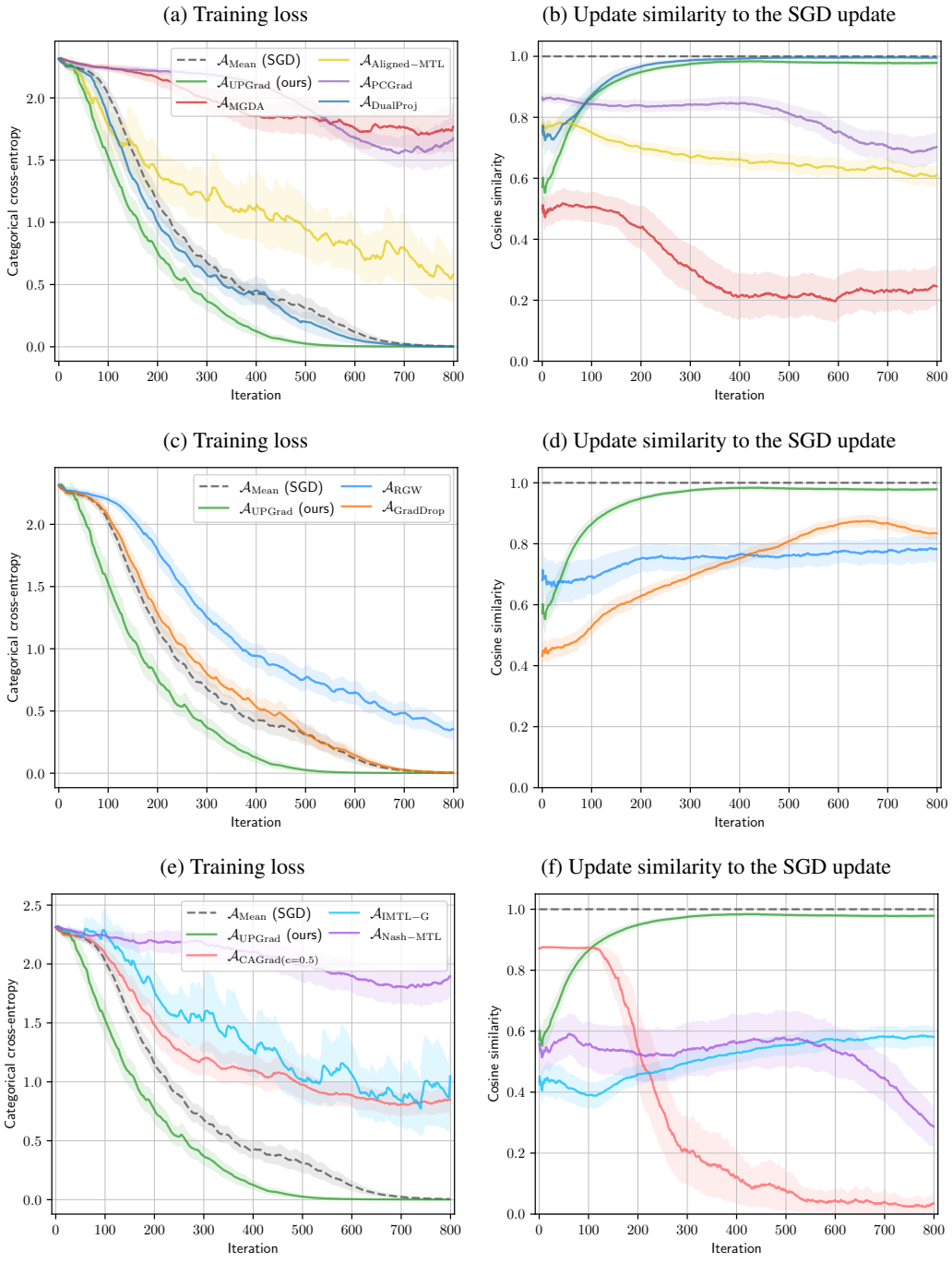


Figure 6: SVHN results.

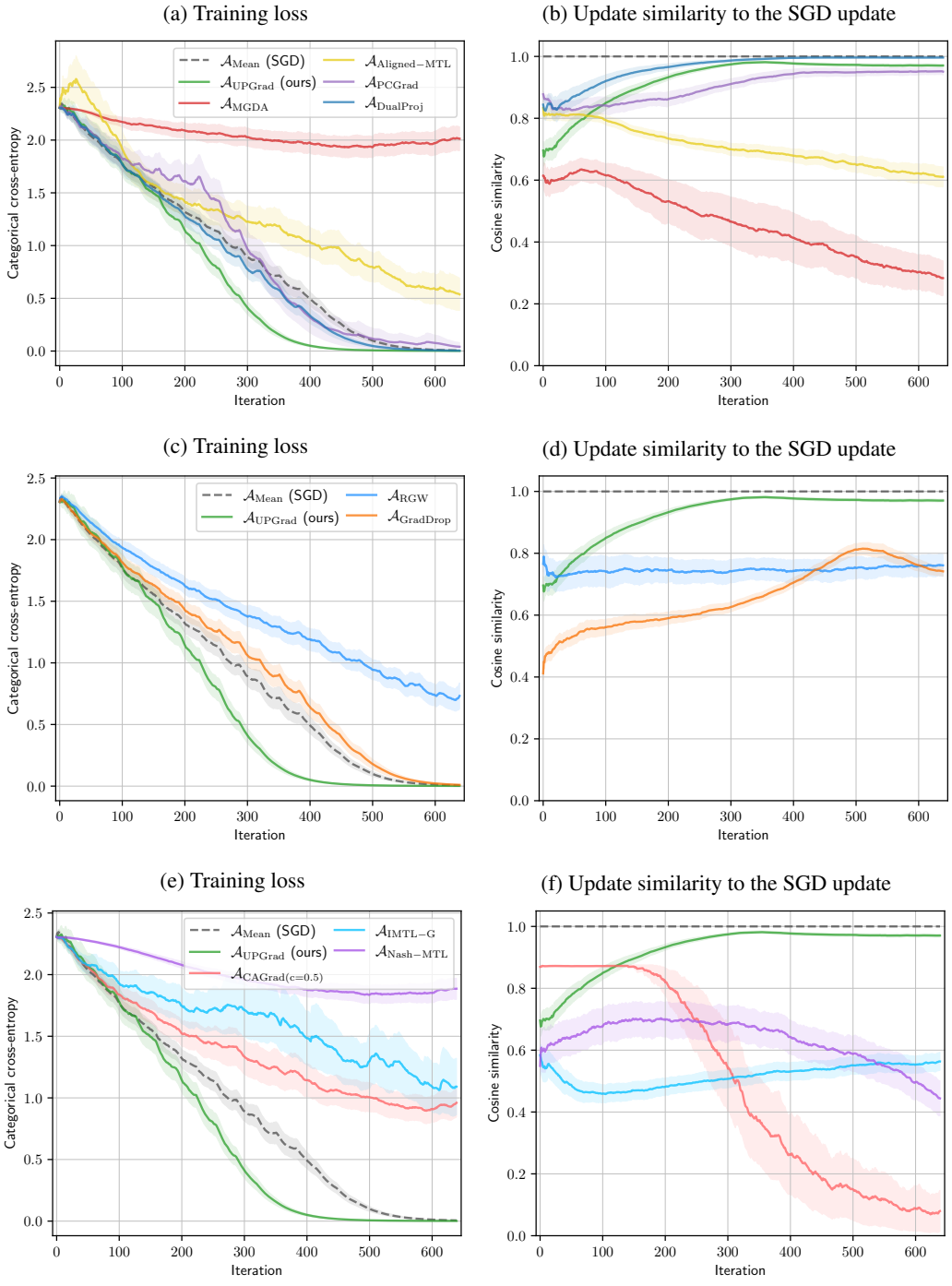


Figure 7: CIFAR-10 results.

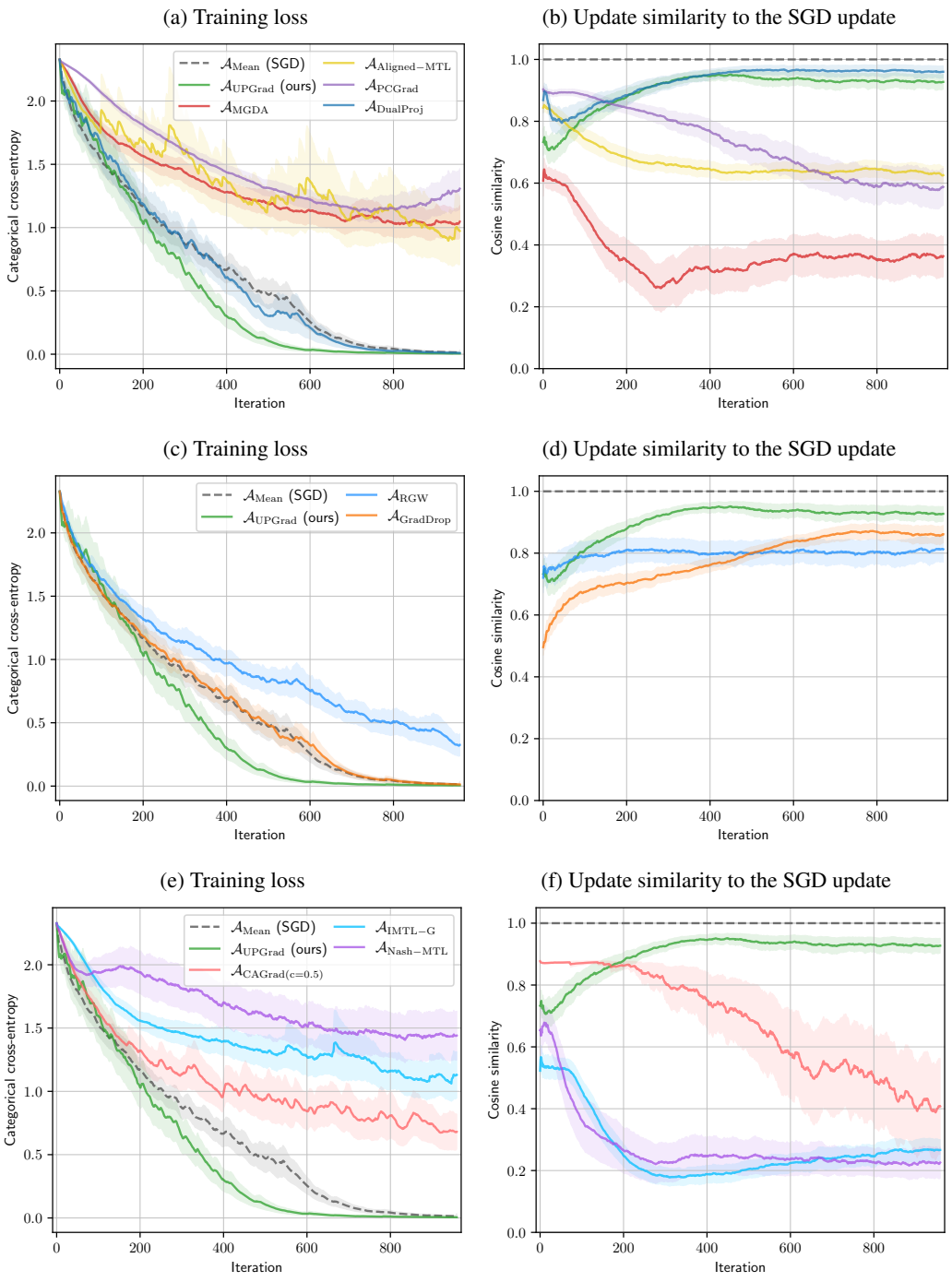


Figure 8: EuroSAT results.

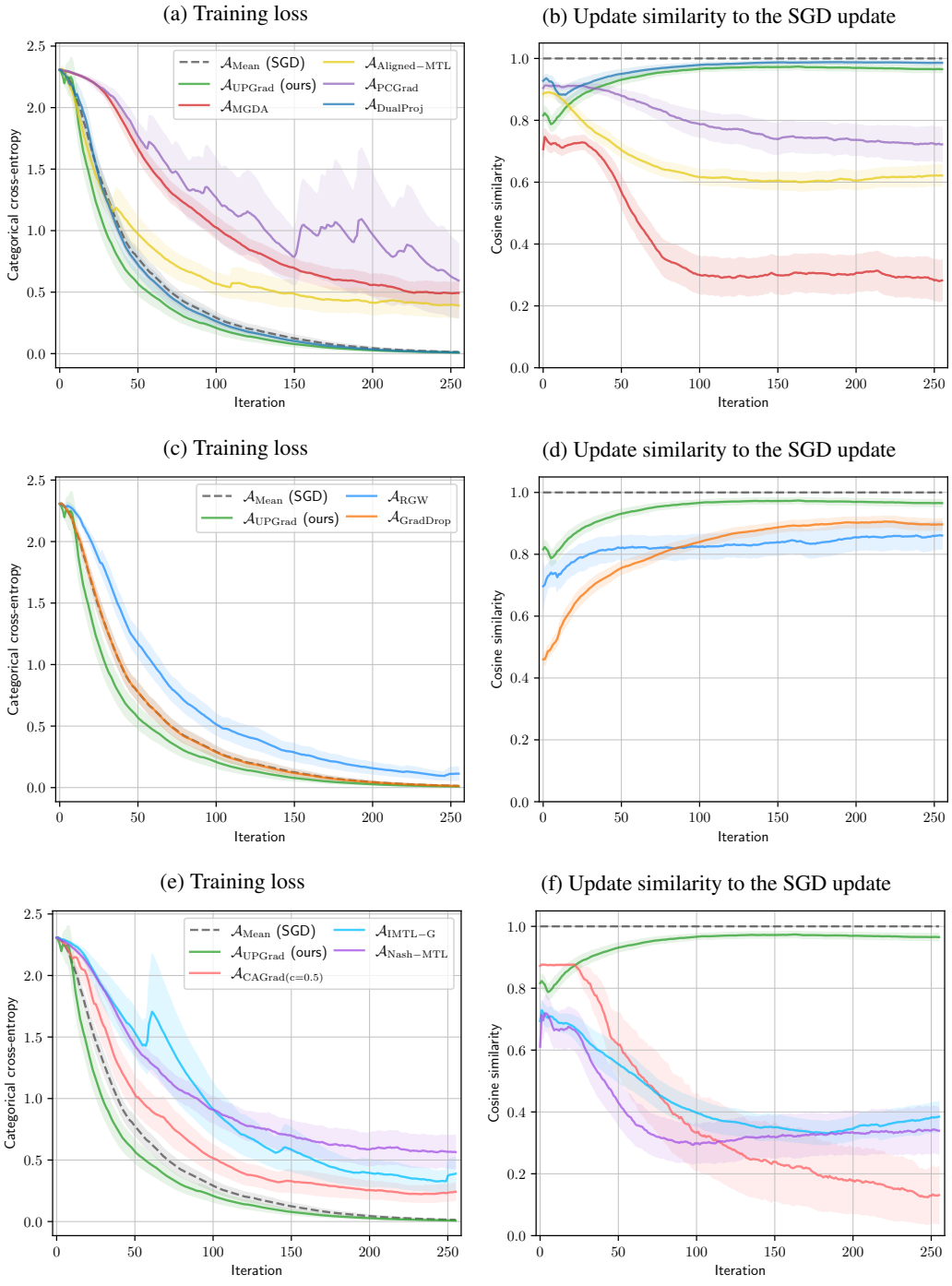


Figure 9: MNIST results.

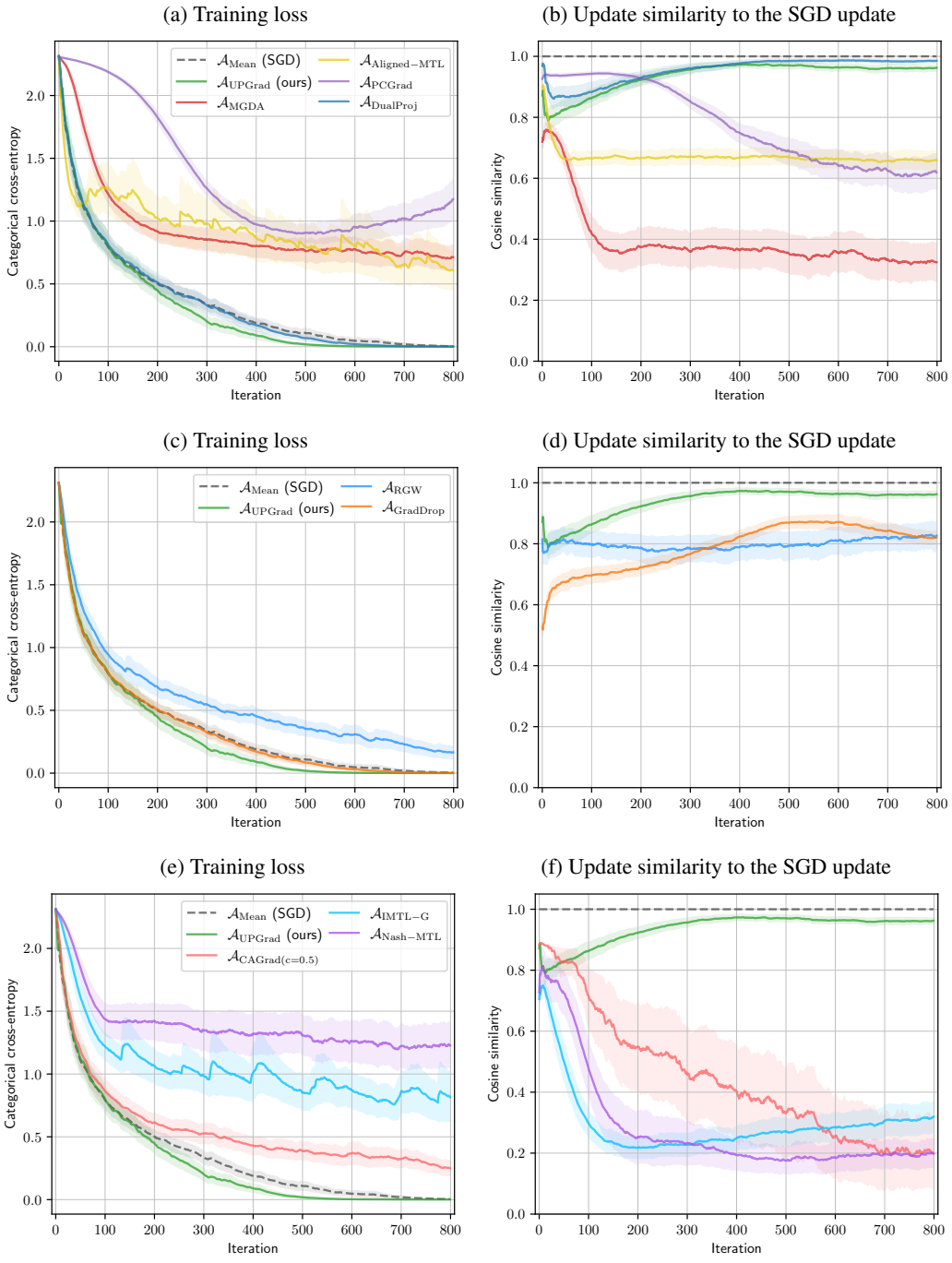


Figure 10: Fashion-MNIST results.

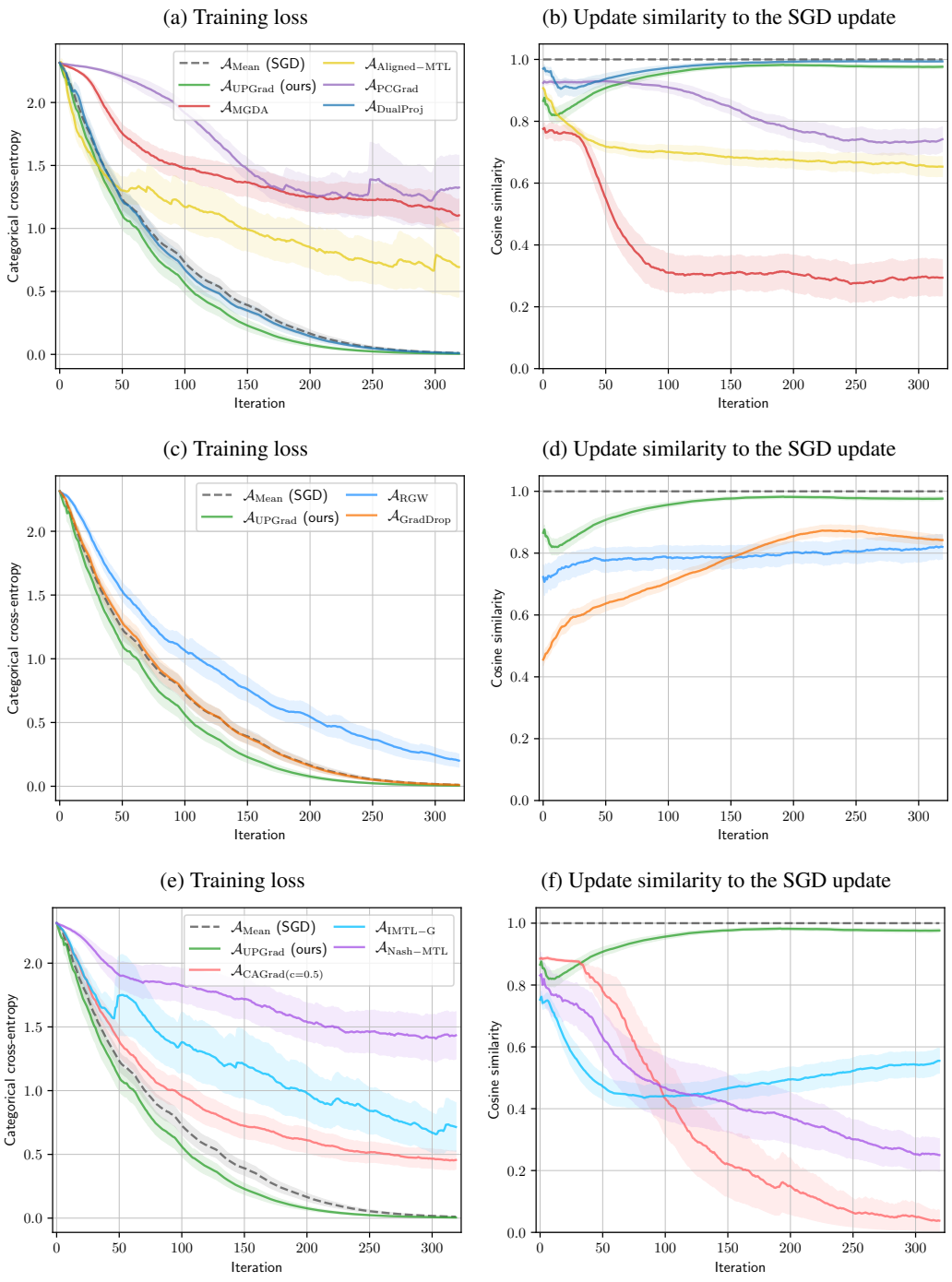


Figure 11: Kuzushiji-MNIST results.

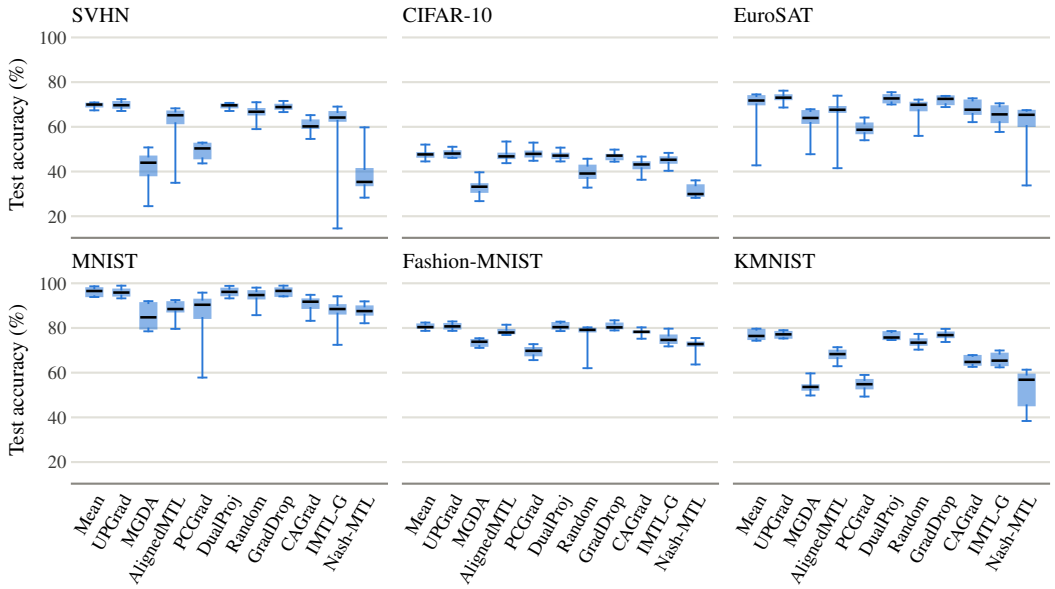


Figure 12: Box plots of the test set accuracies (%) obtained by IWRM with different aggregators across datasets, over 8 independent runs.

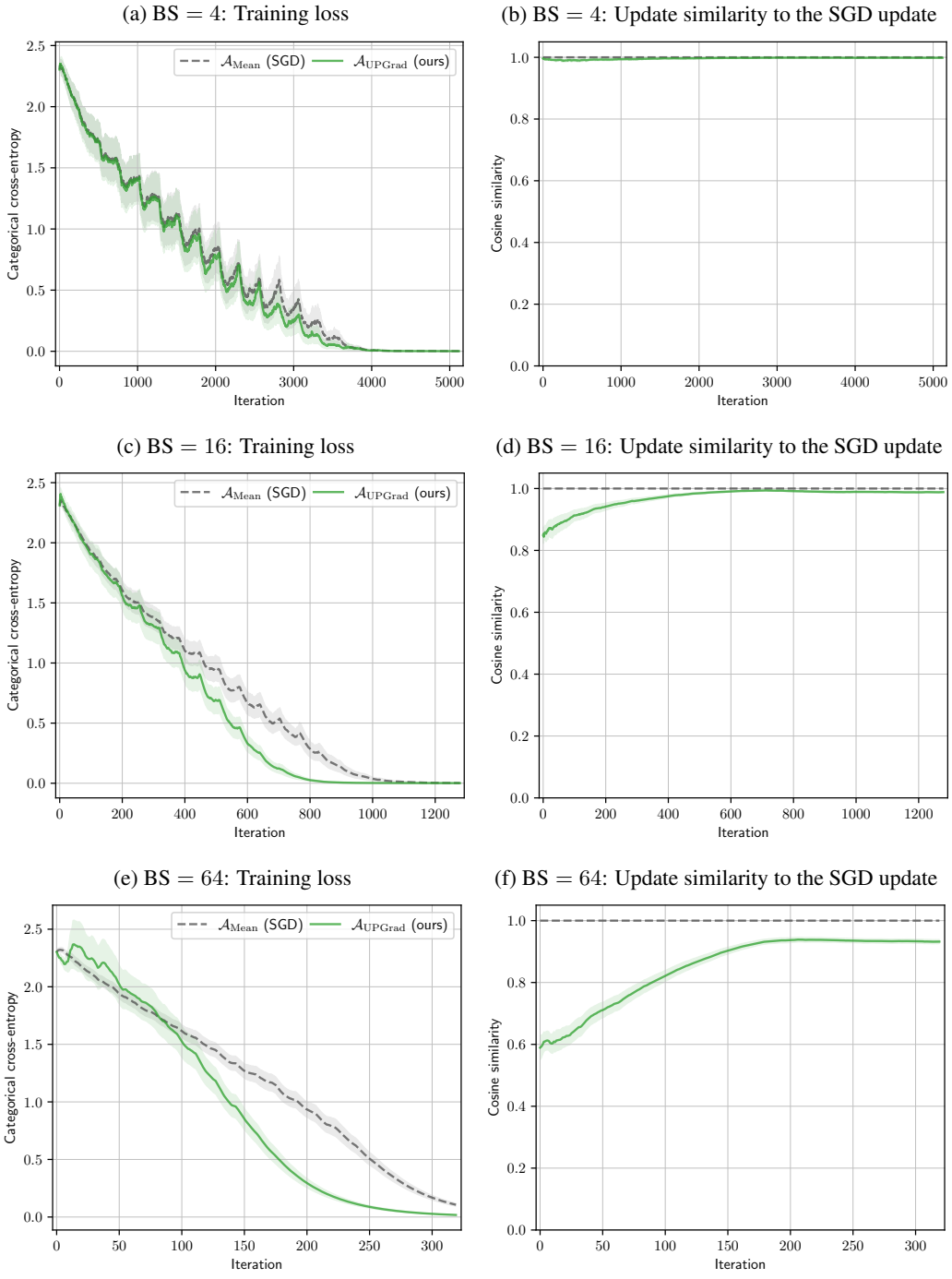
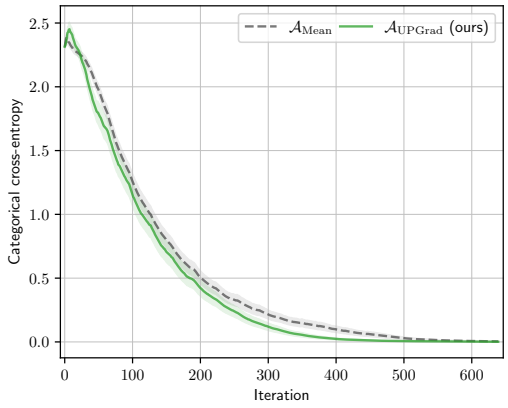
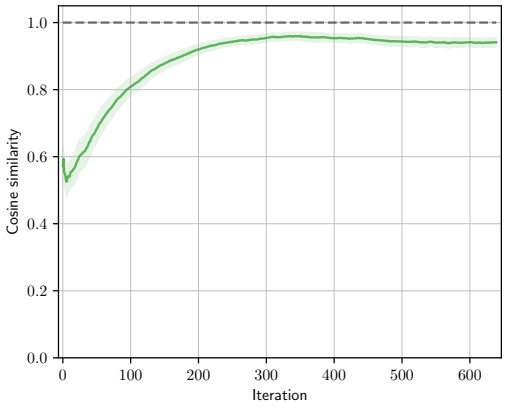


Figure 13: CIFAR-10 results with different batch sizes (BS). The number of epochs is always 20, so the number of iterations varies.

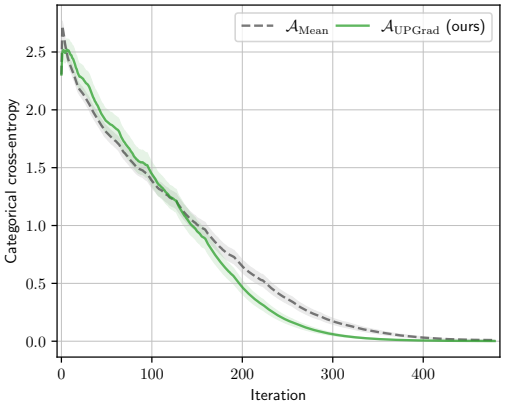
(a) SVHN: Training loss



(b) SVHN: Update similarity to the SGD update



(c) CIFAR-10: Training loss



(d) CIFAR-10: Update similarity to the SGD update

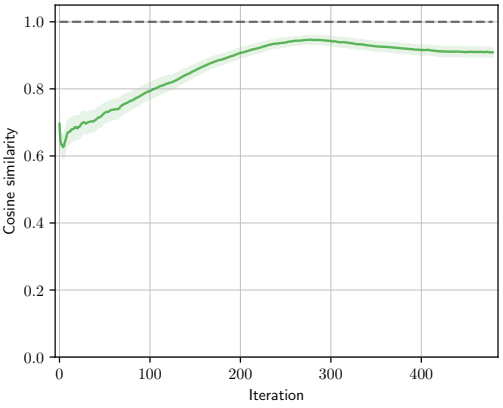


Figure 14: Results with the `Adam` optimizer.

B Applications

In this appendix, we discuss briefly about some potential applications for Jacobian descent.

Multi-task learning. In multi-task learning, a single model is trained to perform several related tasks simultaneously, leveraging shared representations to improve overall performance (Ruder, 2017). At its core, multi-task learning is a multi-objective optimization problem (Sener & Koltun, 2018), making it a straightforward application for Jacobian descent. Yet, the conflict between tasks is often too limited to justify the overhead of computing all task-specific gradients, i.e. the whole Jacobian (Kurin et al., 2022; Xin et al., 2022). In such cases, a practical approach is to minimize some linear scalarization of the objectives using an SGD-based method. Nevertheless, we believe that a setting with inherent conflict between tasks naturally prescribes Jacobian descent with a non-conflicting aggregator. We analyze several related works applied to multi-task learning in Section 3.

Constrained optimization. Constrained optimization problems can be approximated by turning each constraint into an additional objective. In such a situation, Jacobian descent could be used to respect these extra objectives without having to find weights for them.

Regularization as an additional loss. A common technique of regularization is to add an extra term to the loss function. This term could be considered as a distinct loss, and optimized jointly with the main objective.

Adversarial training. In adversarial domain adaptation, the feature extractor of a model is trained with two conflicting objectives: The features should be helpful for the main task and should be unable to discriminate the domain of the input (Ganin et al., 2016). Likewise, in adversarial fairness, the feature extractor is trained to both minimize the predictability of sensitive attributes, such as race or gender, and maximize the performance on the main task (Adel et al., 2019). Combining the corresponding gradients with a non-conflicting aggregator could improve the optimization efficiency of such methods. We believe that the training of generative adversarial networks (Goodfellow et al., 2014) could be similarly formulated as a multi-objective optimization problem. The generator and discriminator could then be jointly optimized with JD.

Momentum-based optimization. In gradient-based single-objective optimization, several methods use some form of gradient momentum to improve their convergence speed (Polyak, 1964). Essentially, their updates consider an exponential moving average of past gradients rather than just the last one. An appealing idea is to modify those algorithms to make them combine the gradient and the momentum with some aggregator, such as $\mathcal{A}_{\text{UPGrad}}$, instead of summing them. This would apply to many popular optimizers, like SGD with Nesterov momentum (Nesterov, 1983), Adam (Kingma & Ba, 2014), AdamW (Loshchilov & Hutter, 2019) and NAdam (Dozat, 2016).

Distributed optimization. In a distributed data-parallel setting with multiple machines or multiple GPUs, model updates are computed in parallel. This can be viewed as multi-objective optimization with one objective per data share. Rather than the typical averaging, a specialized aggregator, such as $\mathcal{A}_{\text{UPGrad}}$, could thus combine the model updates. This consideration can even be extended to federated learning, in which multiple entities participate in the training of a common model from their own private data by sharing model updates (Kairouz et al., 2021). In this setting, as security is one of the main challenges, the non-conflicting property of the aggregator could be interesting.

Mixed preference optimization. When optimizing large language models for human preferences, we may use several loss functions, such as the preference loss, quality loss and generation loss (Wang et al., 2024). These losses could be optimized simultaneously using Jacobian descent, reducing the sensitivity to the weights we give them.

C Proofs

C.1 Preliminary theoretical results

Lemma 1. Let $J \in \mathbb{R}^{m\times n}$, let $\boldsymbol{y} \in \mathbb{R}^n$ and let $\boldsymbol{y}'$ be the orthogonal projection of $\boldsymbol{y}$ onto the span of the rows of J, then $J\boldsymbol{y}' = J\boldsymbol{y}$ and $\|\boldsymbol{y}'\| \leq \|\boldsymbol{y}\|$.

Proof. Let $P \in \mathbb{R}^{n\times n}$ be the orthogonal projection matrix onto the span of the rows of J, so that $\boldsymbol{y}' = P\boldsymbol{y}$. Since every row of J belongs to this span, $J = JP$, and therefore $J\boldsymbol{y}' = JP\boldsymbol{y} = J\boldsymbol{y}$. Moreover, $I - P$ is the projection matrix onto the orthogonal complement of this span, so by the Pythagorean theorem,

$$\|\boldsymbol{y}\|^2 = \|P\boldsymbol{y}\|^2 + \big\|(I-P)\boldsymbol{y}\big\|^2 \geq \|P\boldsymbol{y}\|^2 = \|\boldsymbol{y}'\|^2,$$

which concludes the proof. □

Recall that a function $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^m$ is $\preceq$-convex if for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ and any $\lambda \in [0,1]$,

$$\boldsymbol{f}\big(\lambda\boldsymbol{x} + (1-\lambda)\boldsymbol{y}\big) \preceq \lambda\boldsymbol{f}(\boldsymbol{x}) + (1-\lambda)\boldsymbol{f}(\boldsymbol{y}).$$

Lemma 2. If $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^m$ is a continuously differentiable $\preceq$-convex function, then for any pair of vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$, $\mathcal{J}\boldsymbol{f}(\boldsymbol{x})(\boldsymbol{y}-\boldsymbol{x}) \preceq \boldsymbol{f}(\boldsymbol{y}) - \boldsymbol{f}(\boldsymbol{x})$.

Proof.

$$\begin{aligned}
\mathcal{J}\boldsymbol{f}(\boldsymbol{x})(\boldsymbol{y}-\boldsymbol{x}) &= \lim_{\lambda\to 0^+} \frac{\boldsymbol{f}\big(\boldsymbol{x} + \lambda(\boldsymbol{y}-\boldsymbol{x})\big) - \boldsymbol{f}(\boldsymbol{x})}{\lambda} && \big(\text{differentiation}\big)\\
&\preceq \lim_{\lambda\to 0^+} \frac{\boldsymbol{f}(\boldsymbol{x}) + \lambda\big(\boldsymbol{f}(\boldsymbol{y}) - \boldsymbol{f}(\boldsymbol{x})\big) - \boldsymbol{f}(\boldsymbol{x})}{\lambda} && \big(\preceq\text{-convexity}\big)\\
&= \boldsymbol{f}(\boldsymbol{y}) - \boldsymbol{f}(\boldsymbol{x}),
\end{aligned}$$

which concludes the proof. □

Lemma 3. Let $J \in \mathbb{R}^{m\times n}$, let $\boldsymbol{u} \in \mathbb{R}^m$ and let $\boldsymbol{x} \in \mathbb{R}^n$, then

$$\boldsymbol{u}^\top J\boldsymbol{x} \leq \|\boldsymbol{u}\| \cdot \|J\|_{\mathrm{F}} \cdot \|\boldsymbol{x}\|$$

Proof. Let J_i be the ith row of J, then

$$\begin{aligned}
\big(\boldsymbol{u}^\top J\boldsymbol{x}\big)^2 &\leq \|\boldsymbol{u}\|^2 \cdot \|J\boldsymbol{x}\|^2 && \left(\begin{smallmatrix}\text{Cauchy-Schwarz}\\ \text{inequality}\end{smallmatrix}\right)\\
&= \|\boldsymbol{u}\|^2 \cdot \sum_{i\in[m]} \big(J_i^\top \boldsymbol{x}\big)^2\\
&\leq \|\boldsymbol{u}\|^2 \cdot \sum_{i\in[m]} \|J_i\|^2 \cdot \|\boldsymbol{x}\|^2 && \left(\begin{smallmatrix}\text{Cauchy-Schwarz}\\ \text{inequality}\end{smallmatrix}\right)\\
&= \|\boldsymbol{u}\|^2 \cdot \|J\|_{\mathrm{F}}^2 \cdot \|\boldsymbol{x}\|^2,
\end{aligned}$$

which concludes the proof. □

Recall that a function $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^m$ is β-smooth if for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$,

$$\big\|\mathcal{J}\boldsymbol{f}(\boldsymbol{x}) - \mathcal{J}\boldsymbol{f}(\boldsymbol{y})\big\|_{\mathrm{F}} \leq \beta\|\boldsymbol{x}-\boldsymbol{y}\| \tag{20}$$

Lemma 4. Let $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^m$ be β-smooth, then for any $\boldsymbol{w} \in \mathbb{R}^m$ and any $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$,

$$\boldsymbol{w}^\top\big(\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{f}(\boldsymbol{y}) - \mathcal{J}\boldsymbol{f}(\boldsymbol{y})(\boldsymbol{x}-\boldsymbol{y})\big) \leq \frac{\beta}{2}\|\boldsymbol{w}\| \cdot \|\boldsymbol{x}-\boldsymbol{y}\|^2 \tag{21}$$

Proof.

$$
\begin{aligned}
&\boldsymbol{w}^\top \big(\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{f}(\boldsymbol{y}) - \mathcal{J}\boldsymbol{f}(\boldsymbol{y})(\boldsymbol{x}-\boldsymbol{y})\big) \\
&= \boldsymbol{w}^\top \left(\int_0^1 \mathcal{J}\boldsymbol{f}\big(\boldsymbol{y} + t(\boldsymbol{x}-\boldsymbol{y})\big)(\boldsymbol{x}-\boldsymbol{y})\, dt - \mathcal{J}\boldsymbol{f}(\boldsymbol{y})(\boldsymbol{x}-\boldsymbol{y}) \right) && \left(\begin{smallmatrix}\text{fundamental}\\ \text{theorem}\\ \text{of calculus}\end{smallmatrix}\right) \\
&= \int_0^1 \boldsymbol{w}^\top \Big(\mathcal{J}\boldsymbol{f}\big(\boldsymbol{y} + t(\boldsymbol{x}-\boldsymbol{y})\big) - \mathcal{J}\boldsymbol{f}(\boldsymbol{y}) \Big)(\boldsymbol{x}-\boldsymbol{y})\, dt \\
&\leq \int_0^1 \|\boldsymbol{w}\| \cdot \Big\| \mathcal{J}\boldsymbol{f}\big(\boldsymbol{y} + t(\boldsymbol{x}-\boldsymbol{y})\big) - \mathcal{J}\boldsymbol{f}(\boldsymbol{y}) \Big\|_{\mathrm{F}} \cdot \|\boldsymbol{x}-\boldsymbol{y}\|\, dt && \left(\text{Lemma 3}\right) \\
&\leq \int_0^1 \|\boldsymbol{w}\| \cdot \beta t \cdot \|\boldsymbol{x}-\boldsymbol{y}\|^2\, dt && \left(\beta\text{-smoothness 20}\right) \\
&= \frac{\beta}{2} \|\boldsymbol{w}\| \cdot \|\boldsymbol{x}-\boldsymbol{y}\|^2,
\end{aligned}
$$

which concludes the proof. □

Lemma 5. Let $J \in \mathbb{R}^{m\times n}$, let $\boldsymbol{x} \in \mathbb{R}^n$ and let $0 < a \in \mathbb{R}$, then $\pi_J(a \cdot \boldsymbol{x}) = a \cdot \pi_J(\boldsymbol{x})$.

Proof. Let $C = \{\boldsymbol{y} \in \mathbb{R}^n : \boldsymbol{0} \leq J\boldsymbol{y}\}$ be the dual cone of the rows of J. Since C is a closed convex set, the projection onto C is unique, and $\pi_J(\boldsymbol{x})$ is the point of C closest to $\boldsymbol{x}$. Since $\pi_J(\boldsymbol{x}) \in C$ and $0 < a$, we have $\boldsymbol{0} \leq a \cdot J\pi_J(\boldsymbol{x}) = J\big(a \cdot \pi_J(\boldsymbol{x})\big)$, so $a \cdot \pi_J(\boldsymbol{x}) \in C$. Any $\boldsymbol{y} \in C$ satisfies $\boldsymbol{0} \leq \frac{1}{a} J\boldsymbol{y} = J\left(\frac{1}{a}\boldsymbol{y}\right)$, so $\frac{1}{a}\boldsymbol{y} \in C$. Therefore

$$
\begin{aligned}
\big\|a \cdot \pi_J(\boldsymbol{x}) - a \cdot \boldsymbol{x}\big\| &= a\big\|\pi_J(\boldsymbol{x}) - \boldsymbol{x}\big\| \\
&\leq a \left\| \frac{1}{a}\boldsymbol{y} - \boldsymbol{x} \right\| && \left(\begin{smallmatrix}\text{definition of } \pi_J(\boldsymbol{x})\\ \text{and } \frac{1}{a}\boldsymbol{y} \in C\end{smallmatrix}\right) \\
&= \|\boldsymbol{y} - a \cdot \boldsymbol{x}\|.
\end{aligned}
$$

Hence $a \cdot \pi_J(\boldsymbol{x})$ is a point of C at least as close to $a \cdot \boldsymbol{x}$ as any point of C. By uniqueness of the projection onto C, $\pi_J(a \cdot \boldsymbol{x}) = a \cdot \pi_J(\boldsymbol{x})$. □

C.2 Proposition 1

Proposition 1. Let $J \in \mathbb{R}^{m\times n}$. For any $\boldsymbol{u} \in \mathbb{R}^m$, $\pi_J(J^\top \boldsymbol{u}) = J^\top \boldsymbol{w}$ with

$$
\boldsymbol{w} \in \operatorname*{arg\,min}_{\boldsymbol{v}\in\mathbb{R}^m:\ \boldsymbol{u}\preceq\boldsymbol{v}} \boldsymbol{v}^\top J J^\top \boldsymbol{v}. \tag{5}
$$

Proof. This is a direct consequence of Lemma 6. □

Lemma 6. Let $J \in \mathbb{R}^{m\times n}$, $G = JJ^\top$, $\boldsymbol{u} \in \mathbb{R}^m$. For any $\boldsymbol{w} \in \mathbb{R}^m$ satisfying

$$
\begin{cases}
\boldsymbol{u} \preceq \boldsymbol{w} & \text{(22a)} \\
\boldsymbol{0} \preceq G\boldsymbol{w} & \text{(22b)} \\
\boldsymbol{u}^\top G\boldsymbol{w} = \boldsymbol{w}^\top G\boldsymbol{w} & \text{(22c)}
\end{cases}
$$

we have $\pi_J(J^\top \boldsymbol{u}) = J^\top \boldsymbol{w}$. Moreover, $\boldsymbol{w}$ satisfies these conditions if and only if

$$
\boldsymbol{w} \in \operatorname*{arg\,min}_{\boldsymbol{u}\preceq\boldsymbol{v}} \boldsymbol{v}^\top G\boldsymbol{v}.
$$

Proof. Let $\boldsymbol{w} \in \mathbb{R}^m$ satisfy (22a), (22b) and (22c), and let $\boldsymbol{v} = \boldsymbol{w} - \boldsymbol{u}$ and $\boldsymbol{x} = J^\top \boldsymbol{w}$. The projection

$$
\pi_J(J^\top \boldsymbol{u}) = \operatorname*{arg\,min}_{\substack{\boldsymbol{x}\in\mathbb{R}^n:\\ \boldsymbol{0}\preceq J\boldsymbol{x}}} \frac{1}{2}\|\boldsymbol{x} - J^\top \boldsymbol{u}\|^2
$$

is a convex program with linear constraints. Consequently, the KKT conditions are both necessary and sufficient. The Lagrangian is given by $\mathcal{L}(\boldsymbol{x}, \boldsymbol{v}) = \frac{1}{2}\|\boldsymbol{x} - J^\top \boldsymbol{u}\|^2 - \boldsymbol{v}^\top J\boldsymbol{x}$. The KKT conditions are then given by

$$\begin{cases} \nabla_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{v}) = \mathbf{0} \\ \mathbf{0} \preceq \boldsymbol{v} \\ \mathbf{0} \preceq J\boldsymbol{x} \\ 0 = \boldsymbol{v}^\top J\boldsymbol{x} \end{cases}$$

We now verify that $\boldsymbol{x}$ and $\boldsymbol{v}$ satisfy each of these conditions:

$$\begin{aligned} \nabla_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{v}) = \boldsymbol{x} - J^\top \boldsymbol{u} - J^\top \boldsymbol{v} = J^\top \boldsymbol{w} - J^\top \boldsymbol{u} - J^\top(\boldsymbol{w} - \boldsymbol{u}) = \mathbf{0} & \\ \mathbf{0} \preceq \boldsymbol{w} - \boldsymbol{u} = \boldsymbol{v} & \quad \text{(by (22a))} \\ \mathbf{0} \preceq G\boldsymbol{w} = J\boldsymbol{x} & \quad \text{(by (22b))} \\ \boldsymbol{v}^\top J\boldsymbol{x} = (\boldsymbol{w} - \boldsymbol{u})^\top G\boldsymbol{w} = \boldsymbol{w}^\top G\boldsymbol{w} - \boldsymbol{u}^\top G\boldsymbol{w} = 0 & \quad \text{(by (22c))} \end{aligned}$$

Since the KKT conditions are sufficient, $\boldsymbol{x} = J^\top \boldsymbol{w}$ is the solution of the projection problem, i.e. $\pi_J(J^\top \boldsymbol{u}) = J^\top \boldsymbol{w}$. This proves the first part.

The program $\min_{\boldsymbol{u} \preceq \boldsymbol{v}} \frac{1}{2}\boldsymbol{v}^\top G\boldsymbol{v}$ is also convex with linear constraints, so its KKT conditions are also both necessary and sufficient. Its Lagrangian is given by $\mathcal{L}'(\boldsymbol{v}, \boldsymbol{\mu}) = \frac{1}{2}\boldsymbol{v}^\top G\boldsymbol{v} - \boldsymbol{\mu}^\top(\boldsymbol{v} - \boldsymbol{u})$, and its KKT conditions are given by

$$\begin{cases} \nabla_{\boldsymbol{v}} \mathcal{L}'(\boldsymbol{v}, \boldsymbol{\mu}) = \mathbf{0} \\ \mathbf{0} \preceq \boldsymbol{\mu} \\ \boldsymbol{u} \preceq \boldsymbol{v} \\ 0 = \boldsymbol{\mu}^\top(\boldsymbol{v} - \boldsymbol{u}) \end{cases}$$

$$\Leftrightarrow \begin{cases} \boldsymbol{\mu} = G\boldsymbol{v} \\ \mathbf{0} \preceq G\boldsymbol{v} \\ \boldsymbol{u} \preceq \boldsymbol{v} \\ \boldsymbol{u}^\top G\boldsymbol{v} = \boldsymbol{v}^\top G\boldsymbol{v} \end{cases}$$

where the equivalence uses the symmetry of G. These are exactly (22a), (22b) and (22c), so $\boldsymbol{w}$ satisfies them if and only if it is a minimizer, which concludes the proof. □

C.3 Theorem 1

Theorem 1. Let $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^m$ be a β-smooth and $\preceq$-convex function. Suppose that the Pareto front $\boldsymbol{f}(X^*)$ is compact and that for any $\boldsymbol{x} \in \mathbb{R}^n$, there is $\boldsymbol{x}^* \in X^*$ satisfying $\boldsymbol{f}(\boldsymbol{x}^*) \preceq \boldsymbol{f}(\boldsymbol{x})$. Let $\boldsymbol{x}_1 \in \mathbb{R}^n$, and for all $t \in \mathbb{N}$, $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \eta \mathcal{A}_{\text{UPGrad}}\big(\mathcal{J}\boldsymbol{f}(\boldsymbol{x}_t)\big)$, with $\eta = \frac{1}{\beta\sqrt{m}}$. Let $\boldsymbol{w}_t$ be the weights defining $\mathcal{A}_{\text{UPGrad}}\big(\mathcal{J}\boldsymbol{f}(\boldsymbol{x}_t)\big)$ as per (6), i.e. $\mathcal{A}_{\text{UPGrad}}\big(\mathcal{J}\boldsymbol{f}(\boldsymbol{x}_t)\big) = \mathcal{J}\boldsymbol{f}(\boldsymbol{x}_t)^\top \cdot \boldsymbol{w}_t$. If $\boldsymbol{w}_t$ is bounded, then $\boldsymbol{f}(\boldsymbol{x}_t)$ converges to $\boldsymbol{f}(\boldsymbol{x}^*)$ for some $\boldsymbol{x}^* \in X^*$. In other words, $\boldsymbol{f}(\boldsymbol{x}_t)$ converges to the Pareto front.

To prove the theorem we will need Lemmas 7, 8 and 9 below.

Lemma 7. Let $J \in \mathbb{R}^{m \times n}$ and $\boldsymbol{w} = \frac{1}{m}\sum_{i=1}^m \boldsymbol{w}_i$ be the weights defining $\mathcal{A}_{\text{UPGrad}}(J)$ as per (6). Let, as usual, $G = JJ^\top$, then,

$$\boldsymbol{w}^\top G\boldsymbol{w} \leq \mathbf{1}^\top G\boldsymbol{w}.$$

Proof. Since $\boldsymbol{v}^\top G \boldsymbol{v} = \|J^\top \boldsymbol{v}\|^2$ for any $\boldsymbol{v} \in \mathbb{R}^m$, the mapping $\boldsymbol{v} \mapsto \boldsymbol{v}^\top G \boldsymbol{v}$ is convex. Therefore

$$\begin{aligned}
&\boldsymbol{w}^\top G \boldsymbol{w} \\
&\leq \frac{1}{m} \sum_i \boldsymbol{w}_i^\top G \boldsymbol{w}_i && \text{(Jensen's inequality)} \\
&= \frac{1}{m} \sum_i \boldsymbol{e}_i{}^\top G \boldsymbol{w}_i && \text{(Lemma 6, (22c))} \\
&\leq \frac{1}{m} \sum_i \mathbf{1}^\top G \boldsymbol{w}_i && \left(\substack{\text{Lemma 6, (22b)} \\ \boldsymbol{e}_i \leq \mathbf{1}}\right) \\
&= \mathbf{1}^\top G \boldsymbol{w},
\end{aligned}$$

which concludes the proof. □

Lemma 8. Under the assumptions of Theorem 1, for any $\boldsymbol{w} \in \mathbb{R}^m$ and any $t \in \mathbb{N}$,

$$\boldsymbol{w}^\top \big(\boldsymbol{f}(\boldsymbol{x}_{t+1}) - \boldsymbol{f}(\boldsymbol{x}_t)\big) \leq \frac{\|\boldsymbol{w}\|}{\beta\sqrt{m}} \left(\frac{\mathbf{1}}{2\sqrt{m}} - \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}\right)^\top G_t \boldsymbol{w}_t.$$

Proof. For all $t \in \mathbb{N}$, let $J_t = \mathcal{J}\boldsymbol{f}(\boldsymbol{x}_t)$, $G_t = J_t J_t^\top$. Then $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \eta \mathcal{A}_{\text{UPGrad}}(J_t) = \boldsymbol{x}_t - \eta J_t^\top \boldsymbol{w}_t$. Therefore

$$\begin{aligned}
&\boldsymbol{w}^\top \big(\boldsymbol{f}(\boldsymbol{x}_{t+1}) - \boldsymbol{f}(\boldsymbol{x}_t)\big) \\
&\leq -\eta \boldsymbol{w}^\top J_t J_t^\top \boldsymbol{w}_t + \frac{\beta\eta^2}{2} \|\boldsymbol{w}\| \cdot \|J_t^\top \boldsymbol{w}_t\|^2 && \text{(Lemma 4)} \\
&= -\frac{1}{\beta\sqrt{m}} \boldsymbol{w}^\top G_t \boldsymbol{w}_t + \frac{1}{2\beta m} \|\boldsymbol{w}\| \cdot \boldsymbol{w}_t^\top G_t \boldsymbol{w}_t && \left(\eta = \tfrac{1}{\beta\sqrt{m}}\right) \\
&\leq -\frac{1}{\beta\sqrt{m}} \boldsymbol{w}^\top G_t \boldsymbol{w}_t + \frac{1}{2\beta m} \|\boldsymbol{w}\| \cdot \mathbf{1}^\top G_t \boldsymbol{w}_t && \text{(Lemma 7)} \\
&= \frac{\|\boldsymbol{w}\|}{\beta\sqrt{m}} \left(\frac{\mathbf{1}}{2\sqrt{m}} - \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}\right)^\top G_t \boldsymbol{w}_t,
\end{aligned}$$

which concludes the proof. □

Lemma 9. Under the assumptions of Theorem 1, if $\boldsymbol{x}^* \in X^*$ satisfies $\mathbf{1}^\top \boldsymbol{f}(\boldsymbol{x}^*) \leq \mathbf{1}^\top \boldsymbol{f}(\boldsymbol{x}_t)$ for all $t \in \mathbb{N}$, then

$$\frac{1}{T} \sum_{t \in [T]} \boldsymbol{w}_t^\top \big(\boldsymbol{f}(\boldsymbol{x}_t) - \boldsymbol{f}(\boldsymbol{x}^*)\big) \leq \frac{1}{T} \left(\mathbf{1}^\top \big(\boldsymbol{f}(\boldsymbol{x}_1) - \boldsymbol{f}(\boldsymbol{x}^*)\big) + \frac{\beta\sqrt{m}}{2} \|\boldsymbol{x}_1 - \boldsymbol{x}^*\|^2\right). \tag{23}$$

Proof. We first bound, for any $t \in \mathbb{N}$, $\mathbf{1}^\top \big(\boldsymbol{f}(\boldsymbol{x}_{t+1}) - \boldsymbol{f}(\boldsymbol{x}_t)\big)$ as follows

$$\begin{aligned}
&\mathbf{1}^\top \big(\boldsymbol{f}(\boldsymbol{x}_{t+1}) - \boldsymbol{f}(\boldsymbol{x}_t)\big) \\
&\leq -\frac{1}{2\beta\sqrt{m}} \cdot \mathbf{1}^\top G_t \boldsymbol{w}_t && \left(\substack{\text{Lemma 8} \\ \text{with } \boldsymbol{w} = \mathbf{1}}\right) \\
&\leq -\frac{1}{2\beta\sqrt{m}} \cdot \boldsymbol{w}_t^\top G_t \boldsymbol{w}_t. && \text{(Lemma 7)}
\end{aligned}$$

Summing this over $t \in [T]$ yields

$$\begin{aligned}
&\frac{1}{2\beta\sqrt{m}} \sum_{t \in [T]} \boldsymbol{w}_t^\top G_t \boldsymbol{w}_t \\
&\leq \sum_{t \in [T]} \mathbf{1}^\top \big(\boldsymbol{f}(\boldsymbol{x}_t) - \boldsymbol{f}(\boldsymbol{x}_{t+1})\big) \\
&= \mathbf{1}^\top \big(\boldsymbol{f}(\boldsymbol{x}_1) - \boldsymbol{f}(\boldsymbol{x}_{T+1})\big) && \text{(Telescoping sum)} \\
&\leq \mathbf{1}^\top \big(\boldsymbol{f}(\boldsymbol{x}_1) - \boldsymbol{f}(\boldsymbol{x}^*)\big). && \left(\substack{\text{Assumption} \\ \mathbf{1}^\top \boldsymbol{f}(\boldsymbol{x}^*) \leq \mathbf{1}^\top \boldsymbol{f}(\boldsymbol{x}_{T+1})}\right)
\end{aligned} \tag{24}$$

Since $\mathbf{0} \preceq \boldsymbol{w}_t$,

$$
\begin{aligned}
&\boldsymbol{w}_t^\top \big(\boldsymbol{f}(\boldsymbol{x}_t) - \boldsymbol{f}(\boldsymbol{x}^*)\big) \\
&\leq \boldsymbol{w}_t^\top J_t(\boldsymbol{x}_t - \boldsymbol{x}^*) && \big(\text{Lemma 2}\big) \\
&= \frac{1}{\eta}(\boldsymbol{x}_t - \boldsymbol{x}_{t+1})^\top(\boldsymbol{x}_t - \boldsymbol{x}^*) && \big(\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \eta J_t^\top \boldsymbol{w}_t\big) \\
&= \frac{1}{2\eta}\Big(\|\boldsymbol{x}_t - \boldsymbol{x}_{t+1}\|^2 + \|\boldsymbol{x}_t - \boldsymbol{x}^*\|^2 - \|\boldsymbol{x}_{t+1} - \boldsymbol{x}^*\|^2\Big) && \big(\substack{\text{Parallelogram}\\\text{law}}\big) \\
&= \frac{1}{2\beta\sqrt{m}}\boldsymbol{w}_t^\top G_t \boldsymbol{w}_t + \frac{\beta\sqrt{m}}{2}\Big(\|\boldsymbol{x}_t - \boldsymbol{x}^*\|^2 - \|\boldsymbol{x}_{t+1} - \boldsymbol{x}^*\|^2\Big). && \big(\eta = \tfrac{1}{\beta\sqrt{m}}\big)
\end{aligned}
$$

Summing this over $t \in [T]$ yields

$$
\begin{aligned}
&\sum_{t\in[T]} \boldsymbol{w}_t^\top \big(\boldsymbol{f}(\boldsymbol{x}_t) - \boldsymbol{f}(\boldsymbol{x}^*)\big) \\
&\leq \frac{1}{2\beta\sqrt{m}} \sum_{t\in[T]} \boldsymbol{w}_t^\top G_t \boldsymbol{w}_t + \frac{\beta\sqrt{m}}{2}\Big(\|\boldsymbol{x}_1 - \boldsymbol{x}^*\|^2 - \|\boldsymbol{x}_{T+1} - \boldsymbol{x}^*\|^2\Big) && \big(\text{Telescoping sum}\big) \\
&\leq \frac{1}{2\beta\sqrt{m}} \sum_{t\in[T]} \boldsymbol{w}_t^\top G_t \boldsymbol{w}_t + \frac{\beta\sqrt{m}}{2}\|\boldsymbol{x}_1 - \boldsymbol{x}^*\|^2 \\
&\leq \mathbf{1}^\top \big(\boldsymbol{f}(\boldsymbol{x}_1) - \boldsymbol{f}(\boldsymbol{x}^*)\big) + \frac{\beta\sqrt{m}}{2}\|\boldsymbol{x}_1 - \boldsymbol{x}^*\|^2. && \big(\text{By (24)}\big)
\end{aligned}
$$

Scaling down this inequality by T yields

$$
\frac{1}{T}\sum_{t\in[T]} \boldsymbol{w}_t^\top \big(\boldsymbol{f}(\boldsymbol{x}_t) - \boldsymbol{f}(\boldsymbol{x}^*)\big) \leq \frac{1}{T}\left(\mathbf{1}^\top(\boldsymbol{f}(\boldsymbol{x}_1) - \boldsymbol{f}(\boldsymbol{x}^*)) + \frac{\beta\sqrt{m}}{2}\|\boldsymbol{x}_1 - \boldsymbol{x}^*\|^2\right),
$$

which concludes the proof. □

We are now ready to prove Theorem 1.

Proof. For all $t \in \mathbb{N}$, let $J_t = \mathcal{J}\boldsymbol{f}(\boldsymbol{x}_t)$, $G_t = J_t J_t^\top$. Then

$$
\begin{aligned}
\boldsymbol{x}_{t+1} &= \boldsymbol{x}_t - \eta \mathcal{A}_{\text{UPGrad}}(J_t) \\
&= \boldsymbol{x}_t - \eta J_t^\top \boldsymbol{w}_t.
\end{aligned}
$$

Substituting $\boldsymbol{w} = \mathbf{1}$ in the term $\frac{\mathbf{1}}{2\sqrt{m}} - \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}$ of Lemma 8 yields

$$
\begin{aligned}
\frac{\mathbf{1}}{2\sqrt{m}} - \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|} &= -\frac{\mathbf{1}}{2\sqrt{m}} \\
&\prec \mathbf{0}.
\end{aligned}
$$

Therefore there exists some $\varepsilon > 0$ such that any $\boldsymbol{w} \in \mathbb{R}^m$ with $\|\mathbf{1} - \boldsymbol{w}\| < \varepsilon$ satisfies $\frac{\mathbf{1}}{2\sqrt{m}} \prec \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}$. Denote by $B_\varepsilon(\mathbf{1}) = \{\boldsymbol{w} \in \mathbb{R}^m : \|\mathbf{1} - \boldsymbol{w}\| < \varepsilon\}$, i.e. for all $\boldsymbol{w} \in B_\varepsilon(\mathbf{1})$, $\frac{\mathbf{1}}{2\sqrt{m}} \prec \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}$. By the non-conflicting property of $\mathcal{A}_{\text{UPGrad}}$, $\mathbf{0} \preceq G_t \boldsymbol{w}_t$ and therefore for all $\boldsymbol{w} \in B_\varepsilon(\mathbf{1})$,

$$
\begin{aligned}
&\boldsymbol{w}^\top \big(\boldsymbol{f}(\boldsymbol{x}_{t+1}) - \boldsymbol{f}(\boldsymbol{x}_t)\big) \\
&\leq \frac{\|\boldsymbol{w}\|}{\beta\sqrt{m}}\left(\frac{\mathbf{1}}{2\sqrt{m}} - \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}\right) G_t \boldsymbol{w}_t && \big(\text{Lemma 8}\big) \\
&\leq 0.
\end{aligned}
$$

Since $\boldsymbol{w}^\top \boldsymbol{f}(\boldsymbol{x}_t)$ is bounded and non-increasing, it converges. Since $B_\varepsilon(\mathbf{1})$ contains a basis of $\mathbb{R}^m$, $\boldsymbol{f}(\boldsymbol{x}_t)$ converges to some $\boldsymbol{f}^* \in \mathbb{R}^m$. By the assumption on $\boldsymbol{f}$, for each $t \in \mathbb{N}$, there exists $\boldsymbol{x}_t^*$ in the Pareto set such that $\boldsymbol{f}(\boldsymbol{x}_t^*) \preceq \boldsymbol{f}(\boldsymbol{x}_t)$. Given that the Pareto front is compact, the sequence $\{\boldsymbol{f}(\boldsymbol{x}_t^*)\}_{t\in\mathbb{N}}$

admits a subsequence converging to some $\boldsymbol{f}(\boldsymbol{x}^*)$ also in the Pareto front. Since $\boldsymbol{f}(\boldsymbol{x}_t) \to \boldsymbol{f}^*$, the closedness of the partial order $\preceq$ implies that $\boldsymbol{f}(\boldsymbol{x}^*) \preceq \boldsymbol{f}^*$.

We now prove that $\boldsymbol{f}(\boldsymbol{x}^*) = \boldsymbol{f}^*$. Since $\boldsymbol{f}(\boldsymbol{x}^*) \preceq \boldsymbol{f}^*$, it is sufficient to show that $\mathbf{1}^\top \big(\boldsymbol{f}^* - \boldsymbol{f}(\boldsymbol{x}^*)\big) \leq 0$.

First, the additional assumption of Lemma 9 applies since $\mathbf{1}^\top \boldsymbol{f}(\boldsymbol{x}_t)$ decreases to $\mathbf{1}^\top \boldsymbol{f}^*$ which is larger than or equal to $\mathbf{1}^\top \boldsymbol{f}(\boldsymbol{x}^*)$. Therefore

$$
\begin{aligned}
&\mathbf{1}^\top \big(\boldsymbol{f}^* - \boldsymbol{f}(\boldsymbol{x}^*)\big) \\
&\leq \left(\frac{m}{T} \sum_{t\in[T]} \boldsymbol{w}_t\right)^\top \big(\boldsymbol{f}^* - \boldsymbol{f}(\boldsymbol{x}^*)\big) && \left(\substack{\boldsymbol{f}(\boldsymbol{x}^*)\preceq \boldsymbol{f}^* \\ \mathbf{1}\leq m\boldsymbol{w}_t \\ \text{by (22a)}}\right) \\
&= \frac{m}{T} \sum_{t\in[T]} \boldsymbol{w}_t^\top \Big(\boldsymbol{f}^* - \boldsymbol{f}(\boldsymbol{x}_t) + \boldsymbol{f}(\boldsymbol{x}_t) - \boldsymbol{f}(\boldsymbol{x}^*)\Big) \\
&= \frac{m}{T} \left(\sum_{t\in[T]} \boldsymbol{w}_t^\top \big(\boldsymbol{f}^* - \boldsymbol{f}(\boldsymbol{x}_t)\big) + \sum_{t\in[T]} \boldsymbol{w}_t^\top \big(\boldsymbol{f}(\boldsymbol{x}_t) - \boldsymbol{f}(\boldsymbol{x}^*)\big)\right) \\
&\leq \frac{m}{T} \left(\sum_{t\in[T]} \boldsymbol{w}_t^\top \big(\boldsymbol{f}^* - \boldsymbol{f}(\boldsymbol{x}_t)\big) + \mathbf{1}^\top \big(\boldsymbol{f}(\boldsymbol{x}_1) - \boldsymbol{f}(\boldsymbol{x}^*)\big) + \frac{\beta\sqrt{m}}{2}\|\boldsymbol{x}_1 - \boldsymbol{x}^*\|^2\right) && (\text{Lemma 9})
\end{aligned}
\tag{25}
$$

Taking the limit as $T \to \infty$, we get

$$
\begin{aligned}
&\mathbf{1}^\top \big(\boldsymbol{f}^* - \boldsymbol{f}(\boldsymbol{x}^*)\big) \\
&\leq \lim_{T\to\infty} \frac{m}{T} \sum_{t\in[T]} \boldsymbol{w}_t^\top \big(\boldsymbol{f}^* - \boldsymbol{f}(\boldsymbol{x}_t)\big) \\
&\leq \lim_{T\to\infty} \frac{m}{T} \sum_{t\in[T]} \|\boldsymbol{w}_t\| \cdot \big\|\boldsymbol{f}^* - \boldsymbol{f}(\boldsymbol{x}_t)\big\| && \left(\substack{\text{Cauchy-Schwarz} \\ \text{inequality}}\right) \\
&= 0, && \left(\substack{\boldsymbol{w}_t \text{ bounded} \\ \boldsymbol{f}(\boldsymbol{x}_t)\to\boldsymbol{f}^*}\right)
\end{aligned}
$$

which concludes the proof. □

C.4 Supplementary theoretical results

The proof of Theorem 1 provides some additional insights about the convergence rate as well as some notion of convergence in the non-convex case.

Convergence rate. Combining the equality $\boldsymbol{f}^* = \boldsymbol{f}(\boldsymbol{x}^*)$ and (25), we have

$$
\frac{1}{T} \sum_{t\in[T]} \boldsymbol{w}_t^\top \big(\boldsymbol{f}(\boldsymbol{x}_t) - \boldsymbol{f}(\boldsymbol{x}^*)\big) \leq \frac{1}{T} \mathbf{1}^\top \big(\boldsymbol{f}(\boldsymbol{x}_1) - \boldsymbol{f}(\boldsymbol{x}^*)\big) + \frac{\beta\sqrt{m}}{2T}\|\boldsymbol{x}_1 - \boldsymbol{x}^*\|^2
$$

Furthermore, the Cauchy-Schwarz inequality yields $\mathbf{1}^\top \big(\boldsymbol{f}(\boldsymbol{x}_1) - \boldsymbol{f}(\boldsymbol{x}^*)\big) \leq \sqrt{m} \cdot \big\|\boldsymbol{f}(\boldsymbol{x}_1) - \boldsymbol{f}(\boldsymbol{x}^*)\big\|$, so

$$
\frac{1}{T} \sum_{t\in[T]} \boldsymbol{w}_t^\top \big(\boldsymbol{f}(\boldsymbol{x}_t) - \boldsymbol{f}(\boldsymbol{x}^*)\big) \leq \frac{\sqrt{m}}{T} \left(\big\|\boldsymbol{f}(\boldsymbol{x}_1) - \boldsymbol{f}(\boldsymbol{x}^*)\big\| + \frac{\beta}{2}\|\boldsymbol{x}_1 - \boldsymbol{x}^*\|^2\right) \tag{26}
$$

This hints a convergence rate of order $\mathcal{O}\left(\frac{1}{T}\right)$, whose constant depends on the initial point $\boldsymbol{x}_1$.

Non-convex setting. The proof of (24) does not require the convexity of the objective function. This bound can be equivalently formulated as

$$
\frac{1}{T} \sum_{t=1}^{T} \big\|J_t^\top w_t\big\|^2 \leq \frac{2\beta\sqrt{m}}{T} \mathbf{1}^\top \big(f(x_1) - f(x^*)\big) \tag{27}
$$

This shows the convergence of the updates in the non-convex setting, under the smoothness condition of Theorem 1.

C.5 Derivations of Section 5

In this section, we detail the derivations of (9) and (10). Let $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^m$ be continuously differentiable, let $\boldsymbol{x} \in \mathbb{R}^n$, and let $\mathcal{A}$ be a weighted aggregator.

Taylor approximation in terms of the Gramian. For a fixed $\boldsymbol{x} \in \mathbb{R}^n$, let $\boldsymbol{w} \in \mathbb{R}^m$ be such that $\mathcal{A}\big(\mathcal{J}\boldsymbol{f}(\boldsymbol{x})\big) = \mathcal{J}\boldsymbol{f}(\boldsymbol{x})^\top \boldsymbol{w}$, and let $\boldsymbol{y} = -\eta \mathcal{J}\boldsymbol{f}(\boldsymbol{x})^\top \boldsymbol{w}$ be the corresponding update of JD. Observe that

$$\|\boldsymbol{y}\|^2 = \eta^2 \boldsymbol{w}^\top \mathcal{J}\boldsymbol{f}(\boldsymbol{x}) \mathcal{J}\boldsymbol{f}(\boldsymbol{x})^\top \boldsymbol{w} = \eta^2 \boldsymbol{w}^\top \mathcal{G}\boldsymbol{f}(\boldsymbol{x}) \boldsymbol{w}.$$

Substituting $\boldsymbol{y}$ into the Taylor approximation (2) thus yields

$$\begin{aligned}
\boldsymbol{f}(\boldsymbol{x} + \boldsymbol{y}) &= \boldsymbol{f}(\boldsymbol{x}) + \mathcal{J}\boldsymbol{f}(\boldsymbol{x}) \cdot \boldsymbol{y} + o(\|\boldsymbol{y}\|) \\
&= \boldsymbol{f}(\boldsymbol{x}) - \eta \mathcal{J}\boldsymbol{f}(\boldsymbol{x}) \mathcal{J}\boldsymbol{f}(\boldsymbol{x})^\top \boldsymbol{w} + o\left(\eta \left\| \mathcal{J}\boldsymbol{f}(\boldsymbol{x})^\top \boldsymbol{w} \right\|\right) \\
&= \boldsymbol{f}(\boldsymbol{x}) - \eta \mathcal{G}\boldsymbol{f}(\boldsymbol{x}) \cdot \boldsymbol{w} + o\left(\eta \sqrt{\boldsymbol{w}^\top \cdot \mathcal{G}\boldsymbol{f}(\boldsymbol{x}) \cdot \boldsymbol{w}}\right),
\end{aligned}$$

which is (9).

Gramian-based update. Suppose now that $\mathcal{A}(J) = J^\top \cdot \mathcal{W}(G)$ for some $\mathcal{W} : \mathcal{M}_m \to \mathbb{R}^m$, where $G = JJ^\top$. For any constant vector $\boldsymbol{w} \in \mathbb{R}^m$, the function $\boldsymbol{w}^\top \boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}$ is a linear combination of the coordinates of $\boldsymbol{f}$, so by linearity of differentiation,

$$\nabla \left(\boldsymbol{w}^\top \boldsymbol{f}\right)(\boldsymbol{x}) = \sum_{i \in [m]} w_i \nabla f_i(\boldsymbol{x}) = \mathcal{J}\boldsymbol{f}(\boldsymbol{x})^\top \boldsymbol{w}.$$

Let $\boldsymbol{w} = \mathcal{W}\big(\mathcal{G}\boldsymbol{f}(\boldsymbol{x})\big)$. Note that $\boldsymbol{w}$ depends on $\boldsymbol{x}$, but it is computed before the differentiation step and then treated as a constant, i.e. we do not differentiate through $\mathcal{W}$ nor through $\mathcal{G}\boldsymbol{f}$. Therefore,

$$\begin{aligned}
\mathcal{A}\big(\mathcal{J}\boldsymbol{f}(\boldsymbol{x})\big) &= \mathcal{J}\boldsymbol{f}(\boldsymbol{x})^\top \cdot \mathcal{W}\big(\mathcal{G}\boldsymbol{f}(\boldsymbol{x})\big) \\
&= \mathcal{J}\boldsymbol{f}(\boldsymbol{x})^\top \boldsymbol{w} \\
&= \nabla \left(\boldsymbol{w}^\top \boldsymbol{f}\right)(\boldsymbol{x}) \\
&= \nabla \Big(\mathcal{W}\big(\mathcal{G}\boldsymbol{f}(\boldsymbol{x})\big)^\top \cdot \boldsymbol{f}\Big)(\boldsymbol{x}),
\end{aligned}$$

which is (10).

D Properties of existing aggregators

In the following, we prove the properties of the aggregators from Table 1. Some aggregators, e.g. $\mathcal{A}_{\text{RGW}}$, $\mathcal{A}_{\text{GradDrop}}$ and $\mathcal{A}_{\text{PCGrad}}$, are non-deterministic and are thus not technically functions but rather random variables whose distribution depends on the matrix $J \in \mathbb{R}^{m \times n}$ to aggregate. Still, the properties of Section 2.2 can be easily adapted to a random setting. If $\mathcal{A}$ is a random aggregator, then for any J, $\mathcal{A}(J)$ is a random vector in $\mathbb{R}^n$. The aggregator is non-conflicting if $\mathcal{A}(J)$ is in the dual cone of the rows of J with probability 1. It is linear under scaling if for all $J \in \mathbb{R}^{m \times n}$, there is a – possibly random – matrix $\mathfrak{J} \in \mathbb{R}^{m \times n}$ such that for all $\mathbf{0} \prec \boldsymbol{c} \in \mathbb{R}^m$, $\mathcal{A}(\text{diag}(\boldsymbol{c}) \cdot J) = \mathfrak{J}^\top \cdot \boldsymbol{c}$. Finally, $\mathcal{A}$ is weighted if for any $J \in \mathbb{R}^{m \times n}$ there is a – possibly random – weighting $\boldsymbol{w} \in \mathbb{R}^m$ satisfying $\mathcal{A}(J) = J^\top \cdot \boldsymbol{w}$.

D.1 Mean

$\mathcal{A}_{\text{Mean}}$ simply averages the rows of the input matrix, i.e. for all $J \in \mathbb{R}^{m \times n}$,

$$\mathcal{A}_{\text{Mean}}(J) = \frac{1}{m} J^\top \cdot \mathbf{1} \tag{28}$$

✗ **Non-conflicting.** $\mathcal{A}_{\text{Mean}}\left(\begin{bmatrix} -2 \\ 4 \end{bmatrix}\right) = [1]$, which conflicts with $[-2]$, so $\mathcal{A}_{\text{Mean}}$ is not non-conflicting.

✓ **Linear under scaling.** For any $\boldsymbol{c} \in \mathbb{R}^m$, $\mathcal{A}_{\text{Mean}}(\text{diag}(\boldsymbol{c}) \cdot J) = \frac{1}{m} J^\top \cdot \boldsymbol{c}$, which is linear in $\boldsymbol{c}$. $\mathcal{A}_{\text{Mean}}$ is therefore linear under scaling.

✓ **Weighted.** By (28), $\mathcal{A}_{\text{Mean}}$ is weighted with constant weighting equal to $\frac{1}{m}\mathbf{1}$.

D.2 MGDA

The optimization algorithm presented in Désidéri (2012), called MGDA, is tied to a particular method for aggregating the gradients. We thus refer to this aggregator as $\mathcal{A}_{\text{MGDA}}$. The dual problem of this method was also introduced independently in Fliege & Svaiter (2000). We show the equivalence between the two solutions to make the analysis of $\mathcal{A}_{\text{MGDA}}$ easier.

For all $J \in \mathbb{R}^{m \times n}$, the aggregation described in Désidéri (2012) is defined as

$$\mathcal{A}_{\text{MGDA}}(J) = J^\top \cdot \boldsymbol{w} \tag{29}$$

$$\text{with} \quad \boldsymbol{w} \in \underset{\substack{\mathbf{0} \preceq \boldsymbol{v}: \\ \mathbf{1}^\top \boldsymbol{v} = 1}}{\arg\min} \left\| J^\top \boldsymbol{v} \right\|^2 \tag{30}$$

In Equation (3) of Fliege & Svaiter (2000), the following problem is studied:

$$\min_{\substack{\alpha \in \mathbb{R}, \boldsymbol{x} \in \mathbb{R}^n: \\ J\boldsymbol{x} \preceq \alpha \mathbf{1}}} \alpha + \frac{1}{2}\|\boldsymbol{x}\|^2 \tag{31}$$

We show that the problems in (30) and (31) are dual to each other. Furthermore, the duality gap is null since this is a convex problem. The Lagrangian of the problem in (31) is given by $\mathcal{L}(\alpha, \boldsymbol{x}, \boldsymbol{\mu}) = \alpha + \frac{1}{2}\|\boldsymbol{x}\|^2 - \boldsymbol{\mu}^\top(\alpha \mathbf{1} - J\boldsymbol{x})$. Differentiating w.r.t. α and $\boldsymbol{x}$ gives respectively $1 - \mathbf{1}^\top \boldsymbol{\mu}$ and $\boldsymbol{x} + J^\top \boldsymbol{\mu}$. The dual problem is obtained by setting those two to $\mathbf{0}$ and then maximizing the Lagrangian on $\mathbf{0} \preceq \boldsymbol{\mu}$ and α, i.e.

$$\begin{aligned}
&\arg\max_{\substack{\alpha, \mathbf{0} \preceq \boldsymbol{\mu}: \\ \mathbf{1}^\top \boldsymbol{\mu} = 1}} \alpha + \frac{1}{2}\left\|J^\top \boldsymbol{\mu}\right\|^2 - \boldsymbol{\mu}^\top\left(\alpha \mathbf{1} + JJ^\top \boldsymbol{\mu}\right) \\
= &\arg\max_{\substack{\alpha, \mathbf{0} \preceq \boldsymbol{\mu}: \\ \mathbf{1}^\top \boldsymbol{\mu} = 1}} \alpha + \frac{1}{2}\left\|J^\top \boldsymbol{\mu}\right\|^2 - \alpha \boldsymbol{\mu}^\top \mathbf{1} - \boldsymbol{\mu}^\top JJ^\top \boldsymbol{\mu} \\
= &\arg\min_{\substack{\mathbf{0} \preceq \boldsymbol{\mu}: \\ \mathbf{1}^\top \boldsymbol{\mu} = 1}} \frac{1}{2}\left\|J^\top \boldsymbol{\mu}\right\|^2
\end{aligned}$$

Therefore, (30) and (31) are equivalent, with $\boldsymbol{x} = -J^\top \boldsymbol{w}$.

✓ **Non-conflicting.** Observe that since in (31), $\alpha = 0$ and $\boldsymbol{x} = \mathbf{0}$ is feasible, the objective is non-positive and therefore $\alpha \leq 0$. Substituting $\boldsymbol{x} = -J^\top \boldsymbol{w}$ in $J \cdot \boldsymbol{x} \preceq \alpha \mathbf{1} \preceq \mathbf{0}$ yields $\mathbf{0} \preceq JJ^\top \boldsymbol{w}$, i.e. $\mathbf{0} \preceq J \cdot \mathcal{A}_{\text{MGDA}}(J)$, so $\mathcal{A}_{\text{MGDA}}$ is non-conflicting.

✗ **Linear under scaling.** With $J = \begin{bmatrix} 2 & 0 \\ 0 & 2 \\ a & a \end{bmatrix}$, if $0 \leq a \leq 1$, $\mathcal{A}_{\text{MGDA}}(J) = \begin{bmatrix} a \\ a \end{bmatrix}$. However, if $a \geq 1$, $\mathcal{A}_{\text{MGDA}}(J) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$. This is not affine in a, so $\mathcal{A}_{\text{MGDA}}$ is not linear under scaling. In particular, if any row of J is $\mathbf{0}$, $\mathcal{A}_{\text{MGDA}}(J) = \mathbf{0}$. This implies that the optimization will stop whenever one objective has converged.

✓ **Weighted.** By (29), $\mathcal{A}_{\text{MGDA}}$ is weighted.

D.3 DualProj

The projection of a gradient of interest onto a dual cone was first described in Lopez-Paz & Ranzato (2017). When this gradient is the average of the rows of the Jacobian, we call this aggregator $\mathcal{A}_{\text{DualProj}}$. Formally,

$$\mathcal{A}_{\text{DualProj}}(J) = \frac{1}{m} \cdot \pi_J \left(J^\top \cdot \mathbf{1} \right) \tag{32}$$

where π_J is the projection operator defined in (3).

✓ **Non-conflicting.** By the constraint in (3), $\mathcal{A}_{\text{DualProj}}$ is non-conflicting.

✗ **Linear under scaling.** With $J = \begin{bmatrix} 2 & 0 \\ -2a & 2a \end{bmatrix}$, if $a \geq 1$, $\mathcal{A}_{\text{DualProj}}(J) = \begin{bmatrix} 0 \\ a \end{bmatrix}$. However, if $0.5 \leq a \leq 1$, $\mathcal{A}_{\text{DualProj}}(J) = \begin{bmatrix} 1 - a \\ a \end{bmatrix}$. This is not affine in a on $[0.5, +\infty[$, so $\mathcal{A}_{\text{DualProj}}$ is not linear under scaling.

✓ **Weighted.** By Proposition 1, $\mathcal{A}_{\text{DualProj}}(J) = \frac{1}{m} J^\top \cdot \boldsymbol{w}$, with $\boldsymbol{w} \in \arg\min_{\mathbf{1} \preceq \boldsymbol{v}} \boldsymbol{v}^\top JJ^\top \boldsymbol{v}$. $\mathcal{A}_{\text{DualProj}}$ is thus weighted.

D.4 PCGrad

$\mathcal{A}_{\text{PCGrad}}$ is described in Yu et al. (2020). It projects each gradient onto the orthogonal hyperplane of other gradients in case of conflict with them, iteratively and in random order. When $m \leq 2$, $\mathcal{A}_{\text{PCGrad}}$ is deterministic and satisfies $\mathcal{A}_{\text{PCGrad}} = m \cdot \mathcal{A}_{\text{UPGrad}}$. Therefore, in this case, it satisfies all three properties. When $m > 2$, $\mathcal{A}_{\text{PCGrad}}$ is non-deterministic, so $\mathcal{A}_{\text{PCGrad}}(J)$ is a random vector.

For any index $i \in [m]$, let $\boldsymbol{g}_i = J^\top \cdot \boldsymbol{e}_i$ and let $\mathbf{p}(i)$ be a random vector distributed uniformly on the set of permutations of the elements in $[m] \setminus \{i\}$. For instance, if $m = 3$, $\mathbf{p}(2) = [1 \quad 3]^\top$ with probability 0.5 and $\mathbf{p}(2) = [3 \quad 1]^\top$ with probability 0.5. For notation convenience, whenever i is clear from context, we denote $j_k = \mathbf{p}(i)_k$. The iterative projection of $\mathcal{A}_{\text{PCGrad}}$ is then defined recursively as:

$$\boldsymbol{g}_{i,1}^{\text{PC}} = \boldsymbol{g}_i \tag{33}$$

$$\boldsymbol{g}_{i,k+1}^{\text{PC}} = \boldsymbol{g}_{i,k}^{\text{PC}} - \mathbb{1}\{\boldsymbol{g}_{i,k}^{\text{PC}} \cdot \boldsymbol{g}_{j_k} < 0\} \frac{\boldsymbol{g}_{i,k}^{\text{PC}} \cdot \boldsymbol{g}_{j_k}}{\|\boldsymbol{g}_{j_k}\|^2} \boldsymbol{g}_{j_k} \tag{34}$$

We noticed that an equivalent formulation to the conditional projection of (34) is the projection onto the dual cone of $\{\boldsymbol{g}_{j_k}\}$:

$$\boldsymbol{g}_{i,k+1}^{\text{PC}} = \pi_{\boldsymbol{g}_{j_k}^\top} (\boldsymbol{g}_{i,k}^{\text{PC}}) \tag{35}$$

Finally, the aggregation is given by

$$\mathcal{A}_{\text{PCGrad}}(J) = \sum_{i=1}^{m} \boldsymbol{g}_{i,m}^{\text{PC}}. \tag{36}$$

✗ **Non-conflicting.** If $J = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -0.5 & -1 \end{bmatrix}$, the only non-conflicting direction is $\mathbf{0}$. However, $\mathcal{A}_{\text{PCGrad}}(J)$ is uniform over the set $\left\{ \begin{bmatrix} 0.4 \\ 0.2 \end{bmatrix}, \begin{bmatrix} 0.8 \\ 0.2 \end{bmatrix}, \begin{bmatrix} 0.4 \\ -0.2 \end{bmatrix}, \begin{bmatrix} 0.8 \\ -0.2 \end{bmatrix} \right\}$, i.e. $\mathcal{A}_{\text{PCGrad}}(J)$ is in the dual cone of the rows of J with probability 0. $\mathcal{A}_{\text{PCGrad}}$ is thus not non-conflicting. Here, $\mathbb{E}[\mathcal{A}_{\text{PCGrad}}(J)] = [0.6 \quad 0]^\top$, so $\mathcal{A}_{\text{PCGrad}}$ is neither non-conflicting in expectation.

✓ **Linear under scaling.** To show that $\mathcal{A}_{\text{PCGrad}}$ is linear under scaling, let $\mathbf{0} \prec \boldsymbol{c} \in \mathbb{R}^m$, $\boldsymbol{g}_i' = c_i \boldsymbol{g}_i$, $\boldsymbol{g}_{i,1}'^{\text{PC}} = \boldsymbol{g}_i'$ and $\boldsymbol{g}_{i,k+1}'^{\text{PC}} = \pi_{\boldsymbol{g}_{j_k}'^\top}(\boldsymbol{g}_{i,k}'^{\text{PC}})$. We show by induction that $\boldsymbol{g}_{i,k}'^{\text{PC}} = c_i \boldsymbol{g}_{i,k}^{\text{PC}}$.

The base case is given by $\boldsymbol{g}_{i,1}'^{\text{PC}} = \boldsymbol{g}_i' = c_i \boldsymbol{g}_i = c_i \boldsymbol{g}_{i,1}^{\text{PC}}$.

Then, assuming the induction hypothesis $\boldsymbol{g}_{i,k}'^{\text{PC}} = c_i \boldsymbol{g}_{i,k}^{\text{PC}}$, we show $\boldsymbol{g}_{i,k+1}'^{\text{PC}} = c_i \boldsymbol{g}_{i,k+1}^{\text{PC}}$:

$$\begin{aligned}
\boldsymbol{g}_{i,k+1}'^{\text{PC}} &= \pi_{c_{j_k} \boldsymbol{g}_{j_k}^\top}\left(c_i \boldsymbol{g}_{i,k}^{\text{PC}}\right) && \left(\text{Induction hypothesis}\right) \\
\boldsymbol{g}_{i,k+1}'^{\text{PC}} &= c_i \pi_{\boldsymbol{g}_{j_k}^\top}\left(\boldsymbol{g}_{i,k}^{\text{PC}}\right) && \left(0<c_i \text{ and } 0<c_{j_k}\right) \\
\boldsymbol{g}_{i,k+1}'^{\text{PC}} &= c_i \boldsymbol{g}_{i,k+1}^{\text{PC}} && \left(\text{By (35)}\right)
\end{aligned}$$

Therefore $\mathcal{A}_{\text{PCGrad}}\big(\operatorname{diag}(\boldsymbol{c}) \cdot J\big) = \sum_{i=1}^m c_i \boldsymbol{g}_{i,m}^{\text{PC}}$, so it can be written as $\mathcal{A}_{\text{PCGrad}}\big(\operatorname{diag}(\boldsymbol{c}) \cdot J\big) = \mathfrak{J}^\top \cdot \boldsymbol{c}$ with $\mathfrak{J} = \begin{bmatrix} \boldsymbol{g}_{1,m}^{\text{PC}} & \cdots & \boldsymbol{g}_{m,m}^{\text{PC}} \end{bmatrix}^\top$. Therefore, $\mathcal{A}_{\text{PCGrad}}$ is linear under scaling.

✓ **Weighted.** For all i, $\boldsymbol{g}_{i,m}^{\text{PC}}$ is always a random linear combination of rows of J. $\mathcal{A}_{\text{PCGrad}}$ is thus weighted.

D.5 GradDrop

The aggregator used by the GradDrop layer, which we denote $\mathcal{A}_{\text{GradDrop}}$, is described in Chen et al. (2020). It is non-deterministic, so $\mathcal{A}_{\text{GradDrop}}(J)$ is a random vector. Given $J \in \mathbb{R}^{m \times n}$, let $|J| \in \mathbb{R}^{m \times n}$ be the element-wise absolute value of J. Let $P = \frac{1}{2}\left(\mathbf{1} + \frac{J^\top \cdot \mathbf{1}}{|J|^\top \cdot \mathbf{1}}\right) \in \mathbb{R}^n$, where the division is element-wise. Each coordinate $i \in [n]$ is independently assigned to the set $\mathcal{I}_+$ with probability P_i and to the set $\mathcal{I}_-$ otherwise. The aggregation at coordinate $i \in \mathcal{I}_+$ is given by the sum of all positive J_{ji}, for $j \in [m]$. The aggregation at coordinate $i \in \mathcal{I}_-$ is given by the sum of all negative J_{ji}, for $j \in [m]$. Formally,

$$\mathcal{A}_{\text{GradDrop}}(J) = \left(\sum_{i \in \mathcal{I}_+} \boldsymbol{e}_i \sum_{\substack{j \in [m]: \\ J_{ji} > 0}} J_{ji} \right) + \left(\sum_{i \in \mathcal{I}_-} \boldsymbol{e}_i \sum_{\substack{j \in [m]: \\ J_{ji} < 0}} J_{ji} \right) \tag{37}$$

✗ **Non-conflicting.** If $J = \begin{bmatrix} -2 \\ 1 \end{bmatrix}$, then $P = [1/3]$. Therefore, $\mathbb{P}\big[\mathcal{A}_{\text{GradDrop}}(J) = [-2]\big] = 2/3$ and $\mathbb{P}\big[\mathcal{A}_{\text{GradDrop}}(J) = [1]\big] = 1/3$, i.e. $\mathcal{A}_{\text{GradDrop}}(J)$ is in the dual cone of the rows of J with probability 0. Therefore, $\mathcal{A}_{\text{GradDrop}}$ is not non-conflicting. Here, $\mathbb{E}\big[\mathcal{A}_{\text{GradDrop}}(J)\big] = [-1]^\top$, so $\mathcal{A}_{\text{GradDrop}}$ is neither non-conflicting in expectation.

✗ **Linear under scaling.** If $J = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$, then $P = \frac{1}{2} \cdot \mathbf{1}$ and the aggregation is one of the four vectors $[\pm 1 \quad \pm 1]^\top$ with equal probability. Scaling the first line of J by 2 yields $J = \begin{bmatrix} 2 & -2 \\ -1 & 1 \end{bmatrix}$ and $P = [2/3 \quad 1/3]^\top$, which cannot lead to a uniform distribution over four elements. Therefore, $\mathcal{A}_{\text{GradDrop}}$ is not linear under scaling.

✗ **Weighted.** With $J = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$, the span of J does not include $[1 \quad 1]^\top$ nor $[-1 \quad -1]^\top$. Therefore, $\mathcal{A}_{\text{GradDrop}}$ is not weighted.

D.6 IMTL-G

In Liu et al. (2021b), the authors describe a method to impartially balance gradients by weighting them. Let $\boldsymbol{g}_i$ be the i'th row of J and let $\boldsymbol{u}_i = \frac{\boldsymbol{g}_i}{\|\boldsymbol{g}_i\|}$. They want to find a combination $\boldsymbol{g} = \sum_{i=1}^m \alpha_i \boldsymbol{g}_i$ such that $\boldsymbol{g}^\top \boldsymbol{u}_i$ is equal for all i. Let $U = [\boldsymbol{u}_1 - \boldsymbol{u}_2 \quad \dots \quad \boldsymbol{u}_1 - \boldsymbol{u}_m]^\top$, $D = [\boldsymbol{g}_1 - \boldsymbol{g}_2 \quad \dots \quad \boldsymbol{g}_1 - \boldsymbol{g}_m]^\top$. If $\boldsymbol{\alpha}_{2:m} = [\alpha_2 \quad \dots \quad \alpha_m]^\top$, then $\boldsymbol{\alpha}_{2:m} = \left(UD^\top\right)^{-1} U \cdot \boldsymbol{g}_1$ and $\alpha_1 = 1 - \sum_{i=2}^m \alpha_i$. Notice that this is defined only when the gradients are linearly independent. Thus, this is not strictly speaking an aggregator since it can only be computed on matrices of rank m. We thus propose a generalization defined for matrices of any rank that is equivalent when the matrix has rank m. In the original formulation, requiring $\boldsymbol{g}^\top \boldsymbol{u}_i$ to be equal to some $c \in \mathbb{R}$ for all i, is equivalent to requiring that for all i, $\boldsymbol{g}^\top \boldsymbol{g}_i$ is equal to $c \, \|\boldsymbol{g}_i\|$. Writing $\boldsymbol{g} = J^\top \boldsymbol{\alpha}$ and letting $\boldsymbol{d} \in \mathbb{R}^m$ be the vector of norms of the rows of J, the objective is thus to find $\boldsymbol{\alpha}$ satisfying $JJ^\top \boldsymbol{\alpha} \propto \boldsymbol{d}$. Besides, to match the original formulation, the elements of $\boldsymbol{\alpha}$ should sum to 1.

Letting $\left(JJ^\top\right)^\dagger$ be the Moore-Penrose pseudo inverse of $JJ^\top$, we define

$$\mathcal{A}_{\text{IMTL-G}}(J) = J^\top \cdot \boldsymbol{w} \tag{38}$$

$$\text{with} \qquad \boldsymbol{w} = \begin{cases} \frac{\boldsymbol{v}}{\mathbf{1}^\top \boldsymbol{v}}, & \text{if } \mathbf{1}^\top \boldsymbol{v} \neq \mathbf{0} \\ \mathbf{0}, & \text{otherwise} \end{cases} \tag{39}$$

$$\text{and} \qquad \boldsymbol{v} = \left(JJ^\top\right)^\dagger \cdot \boldsymbol{d}. \tag{40}$$

✗ **Non-conflicting.** If $J = [1 \quad -1 \quad -1]^\top$, then $\boldsymbol{d} = [1 \quad 1 \quad 1]^\top$, $JJ^\top = \begin{bmatrix} 1 & -1 & -1 \\ -1 & 1 & 1 \\ -1 & 1 & 1 \end{bmatrix}$, and thus $\boldsymbol{v} = \frac{1}{9}[-1 \quad 1 \quad 1]^\top$. Therefore, $\boldsymbol{w} = [-1 \quad 1 \quad 1]^\top$, $\mathcal{A}_{\text{IMTL-G}}(J) = [-3]^\top$ and $J \cdot \mathcal{A}_{\text{IMTL-G}}(J) = [-3 \quad 3 \quad 3]^\top$. $\mathcal{A}_{\text{IMTL-G}}$ is thus not non-conflicting.

It should be noted that when J has rank m, $\mathcal{A}_{\text{IMTL-G}}$ seems to be non-conflicting. Thus, it would be possible to make a different non-conflicting generalization, for instance, by deciding $\mathbf{0}$ when J is not full rank.

✗ **Linear under scaling.** With $J = \begin{bmatrix} a & 0 \\ 0 & 1 \end{bmatrix}$ and $a > 0$, we have $\boldsymbol{v} = \begin{bmatrix} 1/a^2 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} a \\ 1 \end{bmatrix} = \begin{bmatrix} 1/a \\ 1 \end{bmatrix}$ and $\mathcal{A}_{\text{IMTL-G}}(J) = \begin{bmatrix} a & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1/a \\ 1 \end{bmatrix} \cdot \frac{1}{\frac{1}{a}+1} = \frac{1}{\frac{1}{a}+1} \cdot \begin{bmatrix} 1 \\ 1 \end{bmatrix}$. This is not affine in a, so $\mathcal{A}_{\text{IMTL-G}}$ is not linear under scaling.

✓ **Weighted.** By (38), $\mathcal{A}_{\text{IMTL-G}}$ is weighted.

D.7 CAGrad

$\mathcal{A}_{\text{CAGrad}}$ is described in Liu et al. (2021a). It is parameterized by $c \in [0, 1[$. If $c = 0$, this is equivalent to $\mathcal{A}_{\text{Mean}}$. Therefore, we restrict our analysis to the case $c > 0$. For any $J \in \mathbb{R}^{m \times n}$, let $\bar{\boldsymbol{g}}$ be the

average gradient $\frac{1}{m} J^\top \cdot \mathbf{1}$, and let ${\boldsymbol{e}_i}^\top J$ denote the i'th row of J. The aggregation is then defined as

$$\mathcal{A}_{\text{CAGrad}}(J) \in \underset{\substack{\boldsymbol{d} \in \mathbb{R}^n : \\ \|\boldsymbol{d} - \overline{\boldsymbol{g}}\| \leq c\|\overline{\boldsymbol{g}}\|}}{\arg\max} \min_{i \in [m]} {\boldsymbol{e}_i}^\top J \boldsymbol{d} \tag{41}$$

✗ **Non-conflicting.** Let $J = \begin{bmatrix} 2 & 0 \\ -2a - 2 & 2 \end{bmatrix}$, with a satisfying $-a + c\sqrt{a^2+1} < 0$. We have $\overline{\boldsymbol{g}} = \begin{bmatrix} -a & 1 \end{bmatrix}^\top$ and $\|\overline{\boldsymbol{g}}\| = \sqrt{a^2+1}$. Observe that any $\boldsymbol{d} \in \mathbb{R}^n$ satisfying the constraint $\|\boldsymbol{d} - \overline{\boldsymbol{g}}\| \leq c\|\overline{\boldsymbol{g}}\|$ has first coordinate at most $-a + c\sqrt{a^2+1}$. Because $-a + c\sqrt{a^2+1} < 0$, any feasible d has a negative first coordinate, making d conflict with the first row of J. For any $c \in [0, 1[$, $-a + c\sqrt{a^2+1} < 0$ is equivalent to $\sqrt{\frac{c^2}{1-c^2}} < a$. Thus, this provides a counter-example to the non-conflicting property for any $c \in [0, 1[$, i.e. $\mathcal{A}_{\text{CAGrad}}$ is not non-conflicting.

If we generalize to the case $c \geq 1$, as suggested in the original paper, then $\boldsymbol{d} = \mathbf{0}$ becomes feasible, which yields $\min_{i \in [m]} {\boldsymbol{e}_i}^\top J \boldsymbol{d} = 0$. Therefore the optimal $\boldsymbol{d}$ satisfies $0 \leq \min_{i \in [m]} {\boldsymbol{e}_i}^\top J \boldsymbol{d}$, i.e. $\mathbf{0} \leq J\boldsymbol{d}$. With $c \geq 1$, $\mathcal{A}_{\text{CAGrad}}$ would thus be non-conflicting.

✗ **Linear under scaling (sketch of proof).** Let $J = \begin{bmatrix} 2 & 0 \\ 0 & 2a \end{bmatrix}$, then $\overline{\boldsymbol{g}} = \begin{bmatrix} 1 & a \end{bmatrix}^\top$ and $\|\overline{\boldsymbol{g}}\| = \sqrt{1+a^2}$. One can show that the constraint $\|\boldsymbol{d} - \overline{\boldsymbol{g}}\| \leq c\|\overline{\boldsymbol{g}}\|$ needs to be satisfied with equality since, otherwise, we can scale $\boldsymbol{d}$ to make the objective larger. Substituting J in $\min_{i \in [m]} {\boldsymbol{e}_i}^\top J \boldsymbol{d}$ yields $2\min(d_1, ad_2)$. For any a satisfying $c\sqrt{1+a^2} + 1 < a^2$, it can be shown that the optimal $\boldsymbol{d}$ satisfies $d_1 < ad_2$. In that case the inner minimum over i is $2d_1$ and, to satisfy $\|\boldsymbol{d} - \overline{\boldsymbol{g}}\| = c\|\overline{\boldsymbol{g}}\|$, the KKT conditions over the Lagrangian yield $\boldsymbol{d} - \overline{\boldsymbol{g}} \propto \nabla_{\boldsymbol{d}} d_1 = \begin{bmatrix} 1 & 0 \end{bmatrix}^\top$. This yields $\boldsymbol{d} = \begin{bmatrix} c \cdot \|\overline{\boldsymbol{g}}\| + 1 \\ a \end{bmatrix} = \begin{bmatrix} c\sqrt{1+a^2} + 1 \\ a \end{bmatrix}$. This is not affine in a; therefore, $\mathcal{A}_{\text{CAGrad}}$ is not linear under scaling.

✓ **Weighted.** In Liu et al. (2021a), $\mathcal{A}_{\text{CAGrad}}$ is formulated via its dual: $\mathcal{A}_{\text{CAGrad}}(J) = \frac{1}{m} J^\top \left(\mathbf{1} + \frac{c\|J^\top \mathbf{1}\|}{\|J^\top \boldsymbol{w}\|} \boldsymbol{w} \right)$, with $\boldsymbol{w} \in \arg\min_{\boldsymbol{w} \in \Delta(m)} \mathbf{1}^\top J J^\top \boldsymbol{w} + c \cdot \|J^\top \mathbf{1}\| \cdot \|J^\top \boldsymbol{w}\|$, where $\Delta(m)$ is the probability simplex of dimension m. Therefore, $\mathcal{A}_{\text{CAGrad}}$ is weighted.

D.8 RGW

$\mathcal{A}_{\text{RGW}}$ is defined in Lin et al. (2021) as the weighted sum of the rows of the input matrix, with a random weighting. The weighting is obtained by sampling m i.i.d. normally distributed random variables and applying a softmax. Formally,

$$\mathcal{A}_{\text{RGW}}(J) = J^\top \cdot \text{softmax}(\mathbf{w}) \tag{42}$$

$$\text{with} \quad \mathbf{w} \sim \mathcal{N}(\mathbf{0}, I) \tag{43}$$

✗ **Non-conflicting.** When $J = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$, the only non-conflicting solution is $\mathbf{0}$. However, $\mathbb{P}[\mathcal{A}_{\text{RGW}}(J) = \mathbf{0}] = 0$, i.e. $\mathcal{A}_{\text{RGW}}(J)$ is in the dual cone of the rows of J with probability 0. $\mathcal{A}_{\text{RGW}}$ is thus not non-conflicting. Here,

$$\begin{aligned} \mathbb{E}[\mathcal{A}_{\text{RGW}}(J)] &= \mathbb{E}\left[\frac{e^{w_1} - 2e^{w_2}}{e^{w_1} + e^{w_2}} \right] && \text{(By (42))} \\ &= -\mathbb{E}\left[\frac{e^{w_1}}{e^{w_1} + e^{w_2}} \right] && (\mathbf{w} \sim \mathcal{N}(\mathbf{0}, I)) \\ &< 0 && (0 < e^{\mathbf{w}}) \end{aligned}$$

so $\mathcal{A}_{\text{RGW}}$ is neither non-conflicting in expectation.

✓ **Linear under scaling.** $\mathcal{A}_{\text{RGW}}\big(\text{diag}(\boldsymbol{c}) \cdot J\big) = \big(\text{diag}(\boldsymbol{c}) \cdot J\big)^\top \cdot \text{softmax}(\mathbf{w}) = J^\top \cdot \text{diag}(\boldsymbol{c}) \cdot \text{softmax}(\mathbf{w}) = J^\top \cdot \text{diag}\big(\text{softmax}(\mathbf{w})\big) \cdot \boldsymbol{c}$. We thus have $\mathcal{A}_{\text{RGW}}\big(\text{diag}(\boldsymbol{c}) \cdot J\big) = \mathfrak{J}^\top \cdot \boldsymbol{c}$ with $\mathfrak{J}^\top = J^\top \cdot \text{diag}\big(\text{softmax}(\mathbf{w})\big)$. Therefore, $\mathcal{A}_{\text{RGW}}$ is linear under scaling.

✓ **Weighted.** By (42), $\mathcal{A}_{\text{RGW}}$ is weighted.

D.9 Nash-MTL

Nash-MTL is described in Navon et al. (2022). Unfortunately, we were not able to verify the proof of Claim 3.1, and we believe that the official implementation of Nash-MTL may mismatch the desired objective by which it is defined. Therefore, we only analyze the initial objective even though our experiments for this aggregator are conducted with the official implementation.

Let $J \in \mathbb{R}^{m \times n}$ and $\varepsilon > 0$. Let also $B_\varepsilon = \left\{ \boldsymbol{d} \in \mathbb{R}^n : \|\boldsymbol{d}\| \leq \varepsilon, \mathbf{0} \preceq J\boldsymbol{d} \right\}$. With ${\boldsymbol{e}_i}^\top J$ denoting the i'th row of J, $\mathcal{A}_{\text{Nash-MTL}}$ is then defined as

$$\mathcal{A}_{\text{Nash-MTL}}(J) = \arg\max_{\boldsymbol{d} \in B_\varepsilon} \sum_{i \in [m]} \log\left({\boldsymbol{e}_i}^\top J\boldsymbol{d}\right) \tag{44}$$

✓ **Non-conflicting.** By the constraint, $\mathcal{A}_{\text{Nash-MTL}}$ is non-conflicting.

✗ **Linear under scaling.** If an aggregator $\mathcal{A}$ is linear under scaling, it should be the case that $\mathcal{A}(aJ) = a\mathcal{A}(J)$ for any scalar $a > 0$ and any $J \in \mathbb{R}^{m \times n}$. However, $\log(a{\boldsymbol{e}_i}^\top J\boldsymbol{d}) = \log({\boldsymbol{e}_i}^\top J\boldsymbol{d}) + \log(a)$. This means that scaling by a scalar does not impact aggregation. Since this is not the trivial $\mathbf{0}$ aggregator, $\mathcal{A}_{\text{Nash-MTL}}$ is not linear under scaling.

✓ **Weighted.** Suppose towards contradiction that $\boldsymbol{d}$ is both optimal for (44) and not in the span of $J^\top$. Let $\boldsymbol{d}'$ be the projection of $\boldsymbol{d}$ onto the span of $J^\top$. Since $\|\boldsymbol{d}'\| < \|\boldsymbol{d}\| \leq \varepsilon$ and $J\boldsymbol{d} = J\boldsymbol{d}'$, we have $J\boldsymbol{d} \prec J\left(\frac{\|\boldsymbol{d}\|}{\|\boldsymbol{d}'\|}\boldsymbol{d}'\right)$, contradicting the optimality of $\boldsymbol{d}$. Therefore, $\mathcal{A}_{\text{Nash-MTL}}$ is weighted.

D.10 Aligned-MTL

The Aligned-MTL method for balancing the Jacobian is described in Senushkin et al. (2023). For simplicity, we fix the vector of preferences to $\frac{1}{m}\mathbf{1}$, but the proofs can be adapted for any non-trivial vector. Given $J \in \mathbb{R}^{m \times n}$, let $V\Sigma^2 V^\top$ be the eigen-decomposition of $JJ^\top$, let $\Sigma^\dagger$ be the diagonal matrix whose non-zero elements are the inverse of corresponding non-zero diagonal elements of Σ and let $\sigma_{\min} = \min_{i \in [m], \Sigma_{ii} \neq 0} \Sigma_{ii}$. The aggregation is then defined as

$$\mathcal{A}_{\text{Aligned-MTL}}(J) = \frac{1}{m} J^\top \cdot \boldsymbol{w} \tag{45}$$

$$\text{with} \qquad \boldsymbol{w} = \sigma_{\min} \cdot V\Sigma^\dagger V^\top \cdot \mathbf{1} \tag{46}$$

✗ **Non-conflicting.** If the SVD of J is $V\Sigma U^\top$, then $J^\top \boldsymbol{w} = \sigma_{\min} UPV^\top \mathbf{1}$ with $P = \Sigma^\dagger \Sigma$ a diagonal projection matrix with 1s corresponding to non zero elements of Σ and 0s everywhere else. Further, $J \cdot \mathcal{A}_{\text{Aligned-MTL}}(J) = \frac{\sigma_{\min}}{m} \cdot V\Sigma V^\top \mathbf{1}$. If $V = \frac{1}{2}\begin{bmatrix} \sqrt{3} & 1 \\ -1 & \sqrt{3} \end{bmatrix}$ and $\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$, we have $J \cdot \mathcal{A}_{\text{Aligned-MTL}}(J) = \frac{1}{2} \cdot V\Sigma V^\top \mathbf{1} = \frac{1}{8}\begin{bmatrix} 3 - \sqrt{3} & 1 - \sqrt{3} \end{bmatrix}^\top$ which is not non-negative. $\mathcal{A}_{\text{Aligned-MTL}}$ is thus not non-conflicting.

✗ **Linear under scaling.** If $J = \begin{bmatrix} 1 & 0 \\ 0 & a \end{bmatrix}$, then $U = V = I$, $\Sigma = J$. For $0 < a \leq 1$, $\sigma_{\min} = a$, therefore $\mathcal{A}_{\text{Aligned-MTL}}(J) = \frac{a}{2} \cdot \mathbf{1}$. For $1 < a$, $\sigma_{\min} = 1$ and therefore $\mathcal{A}_{\text{Aligned-MTL}}(J) = \frac{1}{2} \cdot \mathbf{1}$, which makes $\mathcal{A}_{\text{Aligned-MTL}}$ not linear under scaling.

✓ **Weighted.** By (45), $\mathcal{A}_{\text{Aligned-MTL}}$ is weighted.

E Experimental settings

For all of our experiments, we used `PyTorch` (Paszke et al., 2019). We have developed an open-source library[4] on top of it to enable Jacobian descent easily. This library is designed to be reusable for many other use cases than the experiments presented in our work. To separate them from the library, the experiments have been conducted with a different code repository[5] mainly using `PyTorch` and our library.

E.1 Learning rate selection

The learning rate has a very important impact on the speed of optimization. To make the comparisons as fair as possible, we always show the results corresponding to the best learning rate. Concretely, for each random rerun and each aggregator, we first try 22 learning rates from 10^{-5} to 10^{2}, increasing by a factor $10^{\frac{1}{3}}$ every time. The two best learning rates from this range then define a refined range of plausible good learning rates, going from the smallest of those two multiplied by $10^{-\frac{1}{3}}$ to the largest of those two multiplied by $10^{\frac{1}{3}}$. This margin makes it unlikely for the best learning rate to lie out of the refined range. After this, 50 learning rates from the refined range are tried. These learning rates are evenly spaced in the exponent domain. The best one is then selected and presented in the plots. For simplicity, we have always used a constant learning rate, i.e. no learning rate scheduler was used.

This approach has the advantage of being simple and precise, thus giving trustworthy results. However, it requires 72 trainings for each aggregator, random rerun, and dataset, i.e. a total of 43776 trainings for all of our experiments. For this reason, we have opted to work on small subsets of the original datasets.

E.2 Random reruns and standard error of the mean

To get an idea of confidence in our results, every experiment is performed 8 times on a different seed and a different subset, of size 1024, of the training dataset. The seed used for run $i \in [8]$ is always simply set to i. Because each random rerun includes the full learning rate selection method described in Appendix E.1, it is sensible to consider the 8 sets of results as i.i.d. For each point of both the loss curves and the cosine similarity curves, we thus compute the estimated standard error of the mean with the usual formula $\frac{1}{\sqrt{8}}\sqrt{\frac{\sum_{i\in[8]}(v_i-\bar{v})^2}{8-1}}$, where v_i is the value of a point of the curve for random rerun i, and $\bar{v}$ is the average value of this point over the 8 runs.

E.3 Model architectures

In all experiments, the models are simple convolutional neural networks. All convolutions always have a stride of 1×1, a kernel size of 3×3, a learnable bias, and no padding. All linear layers always have a learnable bias. The activation function is the exponential linear unit (Clevert et al., 2015). The full architectures are given in Tables 3, 4, 5 and 6. Note that these architectures have been fixed arbitrarily, i.e. they were not optimized through some hyper-parameter selection. The weights of the model have been initialized with the default initialization scheme of `PyTorch`.

E.4 Optimizer

For all experiments except those described in Appendix A.5, we always use the basic `SGD` optimizer of `PyTorch`, without any regularization or momentum. Here, `SGD` refers to the `PyTorch` optimizer that updates the parameters of the model in the opposite direction of the gradient, which, in our case, is replaced by the aggregation of the Jacobian matrix. In the rest of this paper, SGD refers to the whole stochastic gradient descent algorithm. In the experiments of Appendix A.5, we instead use `Adam` to study its interactions with JD.

E.5 Loss function

The loss function is always the usual cross-entropy, with the default parameters of `PyTorch`.

[4] Available at https://github.com/SimplexLab/TorchJD

[5] Available at https://github.com/ValerianRey/jde

Table 3: Architecture used for SVHN

`Conv2d` (3 input channels, 16 output channels, 1 group), `ELU`
`Conv2d` (16 input channels, 32 output channels, 16 groups)
`MaxPool2d` (stride of 2×2, kernel size of 2×2), `ELU`
`Conv2d` (32 input channels, 32 output channels, 32 groups)
`MaxPool2d` (stride of 3×3, kernel size of 3×3), `ELU`, `Flatten`
`Linear` (512 input features, 64 output features), `ELU`
`Linear` (64 input features, 10 outputs)

Table 4: Architecture used for CIFAR-10

`Conv2d` (3 input channels, 32 output channels, 1 group), `ELU`
`Conv2d` (32 input channels, 64 output channels, 32 groups)
`MaxPool2d` (stride of 2×2, kernel size of 2× 2), `ELU`
`Conv2d` (64 input channels, 64 output channels, 64 groups)
`MaxPool2d` (stride of 3×3, kernel size of 3×3), `ELU`, `Flatten`
`Linear` (1024 input features, 128 output features), `ELU`
`Linear` (128 input features, 10 outputs)

Table 5: Architecture used for EuroSAT

`Conv2d` (3 input channels, 32 output channels, 1 group)
`MaxPool2d` (stride of 2×2, kernel size of 2×2), `ELU`
`Conv2d` (32 input channels, 64 output channels, 32 groups)
`MaxPool2d` (stride of 2×2, kernel size of 2×2), `ELU`
`Conv2d` (64 input channels, 64 output channels, 64 groups)
`MaxPool2d` (stride of 3×3, kernel size of 3×3), `ELU`, `Flatten`
`Linear` (1024 input features, 128 output features), `ELU`
`Linear` (128 input features, 10 outputs)

Table 6: Architecture used for MNIST, Fashion-MNIST and Kuzushiji-MNIST

`Conv2d` (1 input channel, 32 output channels, 1 group), `ELU`
`Conv2d` (32 input channels, 64 output channels, 1 group)
`MaxPool2d` (stride of 2×2, kernel size of 2×2), `ELU`
`Conv2d` (64 input channels, 64 output channels, 1 group)
`MaxPool2d` (stride of 3×3, kernel size of 3×3), `ELU`, `Flatten`
`Linear` (576 input features, 128 output features), `ELU`
`Linear` (128 input features, 10 outputs)

E.6 Preprocessing

The inputs are always normalized per channel based on the mean and standard deviation computed on the entire training split of the dataset.

E.7 Iterations and computational budget

The numbers of epochs and the corresponding numbers of iterations for all datasets are provided in Table 7, along with the required number of NVIDIA L4 GPU-hours, to run all 72 learning rates for the 11 aggregators on a single seed. The total computational budget to run the main experiments on 8 seeds was thus around 760 GPU-hours. Additionally, we used a total of about 100 GPU hours for the experiments varying the batch size and using Adam, and about 200 more GPU hours were used for early investigations.

Table 7: Numbers of epochs, iterations, and GPU-hours for each dataset

Dataset	Epochs	Iterations	GPU-Hours
SVHN	25	800	17
CIFAR-10	20	640	15
EuroSAT	30	960	32
MNIST	8	256	6
Fashion-MNIST	25	800	17
Kuzushiji-MNIST	10	320	8

F Computation time and memory usage

F.1 Empirical memory usage

Table 8 compares the practical memory usage of the different methods. Since we used the same architecture for MNIST, Fashion-MNIST and Kuzushiji-MNIST, we only report the results for one of them. With the Gramian-based method proposed in Section 5, only the Gramian, which is typically small, has to be stored: the memory requirement becomes much closer to that of SGD. We believe that a better implementation of the Gramian-based approach can reduce this memory cost even further, to the level of SGD's memory usage.

Table 8: Peak GPU memory used during one epoch of training on the ERM objective with SGD, on the IWRM objective with SSJD, and on the IWRM objective with Gramian-based SSJD (as described in Section 5). The batch size is always 32.

Objective	Method	SVHN	CIFAR-10	EuroSAT	MNIST
ERM	SGD	32.0 MB	44.6 MB	72.6 MB	54.2 MB
IWRM	SSJD	315.4 MB	619.3 MB	1447.5 MB	786.0 MB
IWRM	SSJD (Gramian-based)	39.8 MB	75.3 MB	104.2 MB	65.6 MB

F.2 Empirical computational times

In Table 9, we compare the computation time of SGD with that of SSJD for all the aggregators that we experimented with. Since we used the same architecture for MNIST, Fashion-MNIST and Kuzushiji-MNIST, we only report the results for one of them. Several factors affect this computation time. First, the batch size affects the number of rows in the Jacobian to aggregate. Increasing the batch size thus requires more GPU memory and the aggregation of a taller matrix. In contrast, when using the Gramian-based implementation, the memory usage is greatly reduced, which, on a GPU, improves parallelization and thus computational time. Then, some aggregators, e.g. $\mathcal{A}_{\text{Nash-MTL}}$ and $\mathcal{A}_{\text{MGDA}}$, seem to greatly increase the run time. When the aggregation is the bottleneck, a faster implementation will be necessary to make them usable in practice. Lastly, our current implementation of Gramian-based JD and of $\mathcal{A}_{\text{UPGrad}}$ can still be greatly improved.

F.3 Time complexity of the unconflicting projection of gradients

Let $J \in \mathbb{R}^{m \times n}$ be the matrix to aggregate. Apart from computing the Gramian $JJ^\top$, applying $\mathcal{A}_{\text{UPGrad}}$ to J requires solving m instances of the quadratic program (5) of Proposition 1. Solvers for such problems of dimension m typically have a computational complexity upper bounded by $\mathcal{O}(m^4)$ or less in recent implementations (e.g. $\mathcal{O}(m^{3.67} \log m)$). This induces a $\mathcal{O}(m^5)$ time complexity for extracting the weights of $\mathcal{A}_{\text{UPGrad}}$. Note that solving these m problems in parallel would reduce this complexity to $\mathcal{O}(m^4)$.

Table 9: Time required in seconds for one epoch of training on the ERM objective with SGD, on the IWRM objective with SSJD, and on the IWRM objective with Gramian-based SSJD (as described in Section 5), for different aggregators, on an NVIDIA L4 GPU. The batch size is always 32.

Objective	Method	SVHN	CIFAR-10	EuroSAT	MNIST
ERM	SGD	0.83	0.44	0.76	0.43
IWRM	SSJD–$\mathcal{A}_{\text{UPGrad}}$	1.63	1.78	2.86	1.74
IWRM	SSJD–$\mathcal{A}_{\text{UPGrad}}$ (Gramian-based)	1.15	1.12	1.51	1.11
IWRM	SSJD–$\mathcal{A}_{\text{Mean}}$	1.36	1.67	2.71	1.62
IWRM	SSJD–$\mathcal{A}_{\text{Mean}}$ (Gramian-based)	1.01	1.00	1.32	0.97
IWRM	SSJD–$\mathcal{A}_{\text{MGDA}}$	3.92	3.95	5.36	4.05
IWRM	SSJD–$\mathcal{A}_{\text{MGDA}}$ (Gramian-based)	3.56	3.31	3.98	3.41
IWRM	SSJD–$\mathcal{A}_{\text{DualProj}}$	1.40	1.69	2.72	1.64
IWRM	SSJD–$\mathcal{A}_{\text{DualProj}}$ (Gramian-based)	1.03	1.02	1.33	1.00
IWRM	SSJD–$\mathcal{A}_{\text{PCGrad}}$	2.55	2.89	3.89	2.83
IWRM	SSJD–$\mathcal{A}_{\text{PCGrad}}$ (Gramian-based)	2.20	2.22	2.50	2.20
IWRM	SSJD–$\mathcal{A}_{\text{GradDrop}}$	1.34	1.69	2.67	1.59
IWRM	SSJD–$\mathcal{A}_{\text{IMTL-G}}$	1.43	1.69	2.73	1.65
IWRM	SSJD–$\mathcal{A}_{\text{IMTL-G}}$ (Gramian-based)	1.04	1.03	1.33	1.00
IWRM	SSJD–$\mathcal{A}_{\text{CAGrad}}$	1.74	2.02	3.08	1.98
IWRM	SSJD–$\mathcal{A}_{\text{CAGrad}}$ (Gramian-based)	1.36	1.35	1.68	1.34
IWRM	SSJD–$\mathcal{A}_{\text{RGW}}$	1.38	1.67	2.70	1.62
IWRM	SSJD–$\mathcal{A}_{\text{RGW}}$ (Gramian-based)	1.00	1.00	1.30	0.97
IWRM	SSJD–$\mathcal{A}_{\text{Nash-MTL}}$	5.72	6.20	7.06	6.13
IWRM	SSJD–$\mathcal{A}_{\text{Aligned-MTL}}$	1.49	1.73	2.75	1.68
IWRM	SSJD–$\mathcal{A}_{\text{Aligned-MTL}}$ (Gramian-based)	1.06	1.05	1.36	1.03

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We have stated our contributions as accurately as we could in the abstract and in the introduction. In particular, we have tried to clarify that Jacobian descent is more of a formalized unification of existing methods, rather than an entirely new concept.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The main limitation of Jacobian descent is the memory requirement, and to some extent, the computational cost. We discuss this issue at the end of Section 4. Section 5 provides promising future directions to address this. The limited empirical evaluation is also discussed in a dedicated paragraph at the end of the conclusion.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: All proofs are available in the appendices and were thoroughly reviewed. In Appendix C.4, we provide additional results for reference. In all of Appendix C, we give proofs of the stated results, together with Lemmas that help proving them. In Appendix D, we provide justifications for the results stated in Table 1, about properties of many existing aggregators.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We have provided a very detailed explanation of the experimental settings in Appendix E. Besides, the code necessary to run our experiments is publicly available at https://github.com/ValerianRey/jde. To make Jacobian descent even more reproducible, we have published a PyTorch library for it, available at https://github.com/SimplexLab/TorchJD.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The code used for our experiments is publicly available at https://github.com/ValerianRey/jde. It can be used to reproduce our results, by following the instructions of the README file. The data always comes from torchvision, so it will be downloaded automatically as needed.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The experimental settings are all given in Appendix E. We have tried to give as many details as possible, and to justify our choices.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: In our experiments, we show the standard error of the mean to get some idea of confidence in our results. We explicitly state how it is computed in Appendix E.2. We also provide box plots of the test set accuracies in Appendix A.3.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We have included in Appendix E.7 the computational resources that we have used.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our paper should not cause any predictable harm or have potential harmful consequences. We have spent substantial efforts in making it beneficial to the community. For instance, we provide a library to experiment with Jacobian descent that we documented and tested extensively.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: Our work is mostly theoretical, so we do not think that it has direct societal impacts.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: To the best of our knowledge, our work presents no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

Justification: We have always credited in our code the original implementation from which code was taken and adapted, when applicable. We have cited the corresponding papers in our main text, and we have respected the terms of their licenses. In particular, for MIT licenses, we have duly copied the license at the top of the concerned files.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: We spent a substantial amount of time to provide a high quality documentation of our library. We hope that it will help the community to experiment with the concepts that we introduced in our work.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: Our work did not involve any crowdsourcing nor human subject.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: Our work did not involve any human subject.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: We only used LLMs for some minor reformulations when writing, and to help phrase some of the revisions made in response to the reviewers' suggestions.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.