Pareto Manifold Learning: Tackling multiple tasks via ensembles of single-task models

Nikolaos Dimitriadis [1] **Pascal Frossard** [1] **François Fleuret** [2]

Abstract

In Multi-Task Learning (MTL), tasks may compete and limit the performance achieved on each other, rather than guiding the optimization to a solution, superior to all its single-task trained counterparts. Since there is often not a unique solution optimal for all tasks, practitioners have to balance tradeoffs between tasks' performance, and resort to optimality in the Pareto sense. Most MTL methodologies either completely neglect this aspect, and instead of aiming at learning a Pareto Front, produce one solution predefined by their optimization schemes, or produce diverse but discrete solutions. Recent approaches parameterize the Pareto Front via neural networks, leading to complex mappings from tradeoff to objective space. In this paper, we conjecture that the Pareto Front admits a linear parameterization in parameter space, which leads us to propose *Pareto Manifold Learning*, an ensembling method in weight space. Our approach produces a continuous Pareto Front in a single training run, that allows to modulate the performance on each task during inference. Experiments on multi-task learning benchmarks, ranging from image classification to tabular datasets and scene understanding, show that *Pareto Manifold Learning* outperforms state-of-the-art single-point algorithms, while learning a better Pareto parameterization than multi-point baselines.

1. Introduction

In Multi-Task Learning (MTL), multiple tasks are learned concurrently within a single model, striving towards infusing inductive bias that will help outperform the single-task baselines. Apart from the promise of superior performance and some theoretical benefits (Ruder, 2017), such as generalization properties for the learned representation, modeling multiple tasks jointly has practical benefits as well, e.g., lower training and inference times and memory requirements. However, building machine learning models presents a multifaceted host of decisions for multiple and often competing objectives, such as model complexity, runtime and generalization. Conflicts arise since optimizing for one metric often leads to the deterioration of other(s). A single solution satisfying optimally all objectives rarely exists and practitioners must balance the inherent trade-offs.

The notion of tradeoffs is formally defined as *Pareto optimality*. In contrary to single-task learning, where one metric governs the comparison between methods (e.g., top-1 accuracy in `ImageNet`), multiple models can be optimal in MTL; e.g., model X yields superior performance on task $\mathcal{A}$ compared to model Y, but the reverse holds true for task $\mathcal{B}$; thus, there is not a single better model among the two. Intuitively, improvement on an individual task performance can come only at the expense of another task.

In this paper, we develop a novel method, *Pareto Manifold Learning*, which casts MTL problems as learning an ensemble of single-task predictors by interpolating among (ensemble) members during training. By operating in the convex hull of the members' weight space, each single-task model infuses and benefits from representational knowledge to and from the other members. During training, the losses are weighted in tandem with the interpolation, i.e., a monotonic relationship is imposed between the degree of a single-task predictor participation and the weight of the corresponding task loss. Consequently, the ensemble as a whole engenders a (weight) subspace that explicitly encodes tradeoffs and results in a continuous parameterization of the Pareto Front. We identify challenges in guiding the ensemble to such subspaces, designated *Pareto subspaces*, and propose solutions regarding balancing the loss contributions, and regularizing the Pareto properties of the subspaces and adapting the interpolation sampling distribution.

Our method is based on a novel geometrical perspective; multiple Pareto stationary points lie in close proximity and are connected by simple paths whose parameterization

[1]Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland [2]University of Geneva, Geneva, Switzerland. Correspondence to: Nikolaos Dimitriadis <nikolaos.dimitriadis@epfl.ch>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

produces a monotonic mapping in objective space. This is motivated by the recent advancements in single task machine learning that have explored the geometry of the loss landscape and shown experimentally that local optima are connected by simple paths, even linear ones in some cases (Wortsman et al., 2021; Garipov et al., 2018; Frankle et al., 2020; Draxler et al., 2018). We assume that, when the problem has multiple objectives, it acquires a new dimension relating to the number of tasks. Concretely, there are multiple loss landscapes and a solution that satisfies users' performance requirements must lie in the intersection of low loss valleys (for all tasks).

Experimental results validate that the proposed method is able to discover *Pareto Subspaces*, and outperforms baselines on multiple benchmarks. Our training scheme offers two main advantages. First, enforcing low loss for all tasks on a linear subspace implicitly penalizes curvature, which has been linked to generalization (Chaudhari et al., 2017), benefitting all tasks' performance. Second, the algorithm produces in a single training run and with minimal additional complexity a subspace of Pareto Optimal solutions, rather than a single model, enabling practitioners to handpick during inference the solution that offers the tradeoff that best suits their needs. The source code is available at `https://github.com/nik-dim/pamal`.

Overall, our main contributions are the following:

- we offer a geometrical view on the problem of Pareto Front Learning and show that multiple functionally diverse solutions can exist in a straight path in weight space (Section 3),
- we propose a novel algorithm, Pareto Manifold Learning, that employs weight ensembles to infuse inductive bias to the optimization trajectory regarding the monotonicity dictated by Pareto optimality (Section 4),
- We validate our approach on several benchmarks and show that it outperforms baselines, while producing a more reliable mapping from desired preference to objective space compared to other Pareto Front Approximation techniques (Section 5).

2. Related Work

Multi-Task Learning Learning multiple tasks in the Deep Learning setting (Ruder, 2017; Crawshaw, 2020) is usually approached by architectural methodologies (Misra et al., 2016; Ruder et al., 2019), where the architectural modules are combined in several layers to govern the joint representation learning, or optimization approaches (Cipolla et al., 2018; Chen et al., 2018), where the architecture is standardized to be an encoder-decoder(s), for learning the joint and task-specific representations, respectively, and the focus shifts to the descent direction for the shared parameters. We focus on the more general track of optimization methodologies fixing the architectural structure to Shared-Bottom (Caruana, 1997). The various approaches focus on finding a suitable descent direction for the shared parameters. The optimization methods can be broadly categorized into *loss-* and *gradient-balancing* (Liu et al., 2020). For the former, the goal is to appropriately weigh the losses, e.g., via task-dependent homoscedastic uncertainty (Cipolla et al., 2018), by enforcing task gradient magnitudes to have close norms (Chen et al., 2018). The latter class of methodologies manipulate the gradients so that they satisfy certain conditions; projecting the gradient of a (random) task on the normal plane of another so that gradient conflict is avoided (Yu et al., 2020), enforcing the common descent direction to have equal projections for all task gradients (Liu et al., 2020), casting the gradient combination as a bargaining game (Navon et al., 2022). While the aforementioned methodologies focus on the *Single Input-Multiple Outputs* (SIMO) setting, Multi-Task Learning can also be studied under the Multiple Input-Multiple Output prism (Long et al., 2017; Shen et al., 2021). In this case, the challenge lies in the dearth of training data and the goal also includes the characterization of task relatedness.

Multi-Task Learning for Pareto Optimality Sener & Koltun (2018) were the first to view the search for a common descent direction under the Pareto optimality prism and employ the Multiple Gradient Descent Algorithm (MGDA) (Désidéri, 2012) in the Deep Learning context. However, MGDA does not account for task preferences (Lin et al., 2019), and biases solutions towards the task with the smallest gradient magnitude (Liu et al., 2020). By solving a slightly different formulation, Lin et al. (2019) are able to systematically introduce task trade-offs and produce a *discrete* Pareto Front. However, each point requires a different training run. Ma et al. (2020) propose an orthogonal approach for Pareto stationary points; after a model is fitted with any MTL method, a separate phase seeks other Pareto stationary points in its vicinity. But training still needs to occur for every seed point, the separate phase overhead grows linearly with the number of additional models, and the Pareto Front is not continuous across seed points in *parameter space*. Navon et al. (2021) and Lin et al. (2021) employ hypernetworks to continuously approximate the Pareto Front in a single run, which introduces additional design choices and suffers from limited scalability, due to the hypernetwork requiring multiple times the number of parameters of the target network. Ruchte & Grabocka (2021) address the scalability issues by augmenting the feature space with the desired trade-off, which sacrifices either functional diversity or optimality. In both cases, the connection between desired tradeoff and network weights is obfuscated by the complex dynamics of a forward pass by a full neural network, and may not comply to the

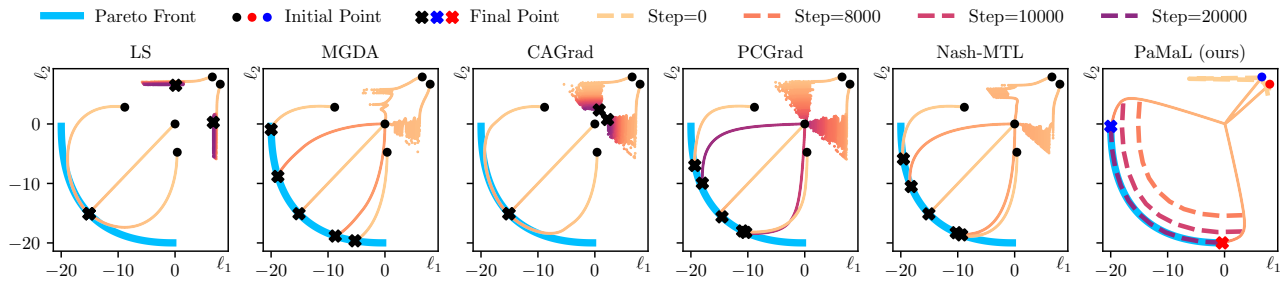


Figure 1: Illustrative example following (Yu et al., 2020; Navon et al., 2022). We present the optimization trajectories in loss space starting from different initializations (black bullets) leading to final points (crosses). Color reflects the iteration number when the corresponding value is achieved. To highlight that our method (PaMaL) deals in pairs of models, we use blue and red to differentiate them. Dashed lines show intermediate results of the discovered subspace. While baselines may not reach the Pareto Front or display bias towards specific solutions, PaMaL discovers the entire Pareto Front *in a single run* and shows superior functional diversity.

monotonicity constraints of Pareto optimal sets of solutions. For a fixed training budget, it may be more beneficial to ignore the user preference and search for one weight configuration to dominate all. Our approach is based on weight ensembles which can be seen as a particular case of linear hypernetworks; instead of generating weights, a convex combination of stored parameters is performed. This change of perspective infuses geometrical inductive bias and allows for more reliable and scalable Pareto Front Learning.

Ensemble Learning and Mode Connectivity Apart from MTL, our algorithm is methodologically tied to prior work in the geometry of the single-task neural network optimization landscapes. The authors in (Garipov et al., 2018; Draxler et al., 2018) independently and concurrently showed that for two local optima $\boldsymbol{\theta}_1^*, \boldsymbol{\theta}_2^*$ produced by separate training runs (but same initializations) there exist nonlinear paths, defined as *connectors* by Wortsman et al. (2021), where the loss remains low. The connectivity paths can be extended to include linear in the case of the training runs sharing some part of the optimization trajectory (Frankle et al., 2020). These findings can be leveraged to train a neural network subspace by enforcing linear connectivity among the subspace endpoints (Wortsman et al., 2021). Linear mode connectivity encourages flatness and, therefore, is linked with methods explicitly enforcing flat minima (Chaudhari et al., 2017; Foret et al., 2021; Dinh et al., 2017; Jiang et al., 2020). These approaches are applicable when designing a single objective, e.g. average of losses in Multi-Task Learning, but do not allow for the infusion of Pareto properties and the inclusion of tradeoffs. Izmailov et al. (2018) produce flat minima by averaging multiple weight vectors discovered during the optimization trajectory, so that the final model lies in the middle of the low-loss basin. Wortsman et al. (2022) perform weight ensembling with fine-tuned models produced via different hyperparameter configurations. Apart from the recent weight ensembling works, output ensembling has been one of the staples of machine learning literature. Lakshminarayanan et al. (2017) utilize deep ensembles for uncertainty prediction but inference scales linearly with the number of ensemble members. Wen et al. (2020) improve on the computational complexity of output ensembles by sharing the bulk of the parameters among members and differentiating them via rank-1 matrices, while Havasi et al. (2021) employ a multi-input multi-output network by accommodating independent subnetworks for each ensemble and allowing a single-forward pass ensemble prediction. However, this results in subnetworks with incompatible architecture which does not allow for a continuous approximation of the Pareto Front.

3. Problem Formulation

Notation We use bold font for vectors $\boldsymbol{x}$, capital bold for matrices $\boldsymbol{X}$ and regular font for scalars x. T is the number of tasks and m is the number of ensemble members. Each task $t \in [T]$ has a loss $\mathcal{L}_t$. The overall multi-task loss is $\boldsymbol{L} = \left[\mathcal{L}_1, \dots, \mathcal{L}_T\right]^\top$. $\boldsymbol{w} \in \Delta_T \subset \mathbb{R}^T$ is the weighting scheme for the tasks, i.e., the overall loss is calculated as $\mathcal{L} = \boldsymbol{w}^\top \boldsymbol{L} = \sum_{t=1}^T w_t \mathcal{L}_t$. Each member $k \in [m]$ is associated with parameters $\boldsymbol{\theta}_k \in \mathbb{R}^N$ and weighting $\boldsymbol{w} \in \Delta_T$.

Preliminaries Our goal lies in solving an unconstrained vector optimization problem of minimizing $\boldsymbol{L}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = [\mathcal{L}_1(y_1, \hat{y}_1), \dots, \mathcal{L}_T(y_T, \hat{y}_T)]^\top$, where $\mathcal{L}_i$ corresponds to the objective function for the i^{th} task, e.g., cross-entropy loss in case of classification. Constructing an optimal solution for all tasks is often unattainable due to competing objectives. Hence, an alternative notion of optimality is used, as described in Theorem 3.1.

Definition 3.1 (Pareto Optimality)**.** Consider two points $\boldsymbol{x}$

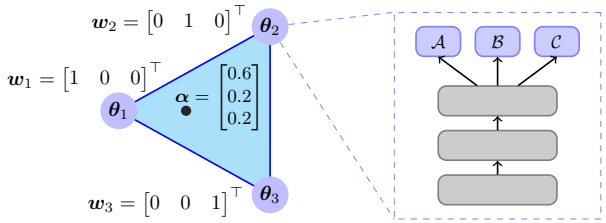


Figure 2: A representation of parameter space for $T = 3$ tasks. Each node corresponds to a tuple of parameters and weighting scheme $(\boldsymbol{\theta}_v, \boldsymbol{w}_v) \in \mathbb{R}^N \times \Delta_T$. The blue dashed frame shows the model, e.g., shared-bottom architecture, implemented by the parameters $\boldsymbol{\theta}_v$ of each node. For each training step, we sample $\boldsymbol{\alpha} \in \Delta_T$ and construct the weight combination $\boldsymbol{\theta} = \boldsymbol{\alpha}^\top \boldsymbol{\Theta} = 0.6 \cdot \boldsymbol{\theta}_1 + 0.2 \cdot \boldsymbol{\theta}_2 + 0.2 \cdot \boldsymbol{\theta}_3$.

and $\boldsymbol{y}$ in the parameter space. A point $\boldsymbol{x}$ dominates a point $\boldsymbol{y}$ if $\mathcal{L}_t(\boldsymbol{x}) \leq \mathcal{L}_t(\boldsymbol{y})$ for all tasks $t \in [T]$ and $\boldsymbol{L}(\boldsymbol{x}) \neq \boldsymbol{L}(\boldsymbol{y})$. Then, a point $\boldsymbol{x}$ is called Pareto optimal if there exists no point $\boldsymbol{y}$ that dominates it. The set of Pareto optimal points forms the Pareto front $\mathcal{P}_{\boldsymbol{L}}$.

The vector loss function is scalarized by the vector $\boldsymbol{w} \in [0,1]^T$ to form the overall objective $\boldsymbol{w}^\top \boldsymbol{L}$. Without loss of generality, we assume that $\boldsymbol{w}$ lies in the T-dimensional simplex Δ_T by imposing the constraint $\|\boldsymbol{w}\| = \sum_{t=1}^T w_t = 1$. This formulation permits to think of the vector of weights as an encoding of task preferences, e.g., for two tasks letting $\boldsymbol{w} = [0.8, 0.2]$ results in attaching more importance to the first task. Overall, the MTL problem can be formulated within the Empirical Risk Minimization (ERM) framework for preference vector $\boldsymbol{w}$ and dataset $\mathcal{D} = \{(\boldsymbol{x}, \boldsymbol{y})\}_{i=1}$ as:

$$\min_{\boldsymbol{\theta}} \quad \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{D}} \left[\boldsymbol{L}\left(\boldsymbol{f}\left(\boldsymbol{x};\boldsymbol{\theta}\right), \boldsymbol{y}\right)\right] \tag{1}$$

Our overall goal is to discover a low-dimensional parameterization in weight space that yields a (continuous) Pareto Front in functional space. This desideratum leads us to the following definition:

Definition 3.2 (Pareto Subspace)**.** Let T be the number of tasks, $\mathcal{X}$ the input space, $\mathcal{Y}$ the multi-task output space, $\mathcal{R} \subset \mathbb{R}^N$ the parameter space, $f : \mathcal{X} \times \mathcal{R} \to \mathcal{Y}$ the function implemented by a neural network, and $\boldsymbol{L} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^T_{>0}$ be the vector loss. Let $\{\boldsymbol{\theta}_t \in \mathcal{R} : t \in [T]\}$ be a collection of network parameters and $\mathcal{S}$ the corresponding convex envelope, i.e., $\mathcal{S} = \left\{\sum_{t=1}^T \alpha_t \boldsymbol{\theta}_t : \sum_{t=1}^T \alpha_t = 1 \text{ and } \alpha_t \geq 0, \forall t\right\}$. For the dataset $\mathcal{D} = (\mathcal{D}_{\mathcal{X}}, \mathcal{D}_{\mathcal{Y}})$, the subspace $\mathcal{S}$ is called Pareto if its mapping to functional space via the network architecture f forms a Pareto Front $\mathcal{P} = \boldsymbol{L}(f(\mathcal{D}_{\mathcal{X}}; \mathcal{S}), \mathcal{D}_{\mathcal{Y}}) = \{\boldsymbol{l} : \boldsymbol{l} = \boldsymbol{L}(f(\mathcal{D}_{\mathcal{X}}; \boldsymbol{\theta}), \mathcal{D}_{\mathcal{Y}}), \quad \forall \boldsymbol{\theta} \in \mathcal{S}\}$.

4. Method

We seek to find a collection of m neural networks, of identical architecture, whose linear combination in *weight space* forms a continuous Pareto Front in *objective space*. Model i corresponds to a tuple of network parameters $\boldsymbol{\theta}_i$ and task weighting $\boldsymbol{w}_i$ and implements the function $\boldsymbol{f}(\cdot; \boldsymbol{\theta}_i)$. We impose connectivity among models by modeling the subspace in the convex hull of the ensemble members. Section 4.1 presents the core of the algorithm, and in Section 4.2 we discuss various improvements that address MTL challenges.

4.1. Pareto Manifold Learning

Let $\boldsymbol{\Theta} = \left[\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_m\right]^\top$ be an $m \times N$ matrix storing the parameters of all models, $\boldsymbol{W} = \left[\boldsymbol{w}_1, \ldots, \boldsymbol{w}_m\right]^\top$ be a $m \times T$ matrix storing the task weighting of ensemble members. By designing the subspace as a simplex, the objective now becomes:

$$\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{D}} \left[\mathbb{E}_{\boldsymbol{\alpha}\sim \mathrm{P}} \left[\boldsymbol{\alpha}^\top \boldsymbol{W} \boldsymbol{L}\left(\boldsymbol{f}\left(\boldsymbol{x}; \boldsymbol{\alpha}\boldsymbol{\Theta}\right), \boldsymbol{y}\right)\right]\right] \tag{2}$$

where P is the sampling distribution placed upon the simplex. In the case where the ensemble members are single-task predictors ($\boldsymbol{w}$ is one-hot) and the number of tasks coincides with the number of ensemble members ($m = T$), the matrix of task weightings $\boldsymbol{W}$ is an identity matrix and Equation 2 simplifies to $\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{D}} \left[\mathbb{E}_{\boldsymbol{\alpha}\sim\mathrm{P}} \left[\boldsymbol{\alpha}^\top \boldsymbol{L}\left(\boldsymbol{f}\left(\boldsymbol{x}; \boldsymbol{\alpha}\boldsymbol{\Theta}\right), \boldsymbol{y}\right)\right]\right] = \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{D}} \left[\mathbb{E}_{\boldsymbol{\alpha}\sim\mathrm{P}} \left[\sum_{t=1}^T \alpha_t \mathcal{L}_t \left(\boldsymbol{f}\left(\boldsymbol{x}; \sum_{t=1}^T \alpha_t \boldsymbol{\theta}_t\right), \boldsymbol{y}\right)\right]\right]$. By using the same weighting for both the losses and the ensemble interpolation, we explicitly associate models and task losses with a one-to-one correspondence, infusing preference towards one task rather than the other and guiding the learning trajectory to a subspace that encodes such tradeoffs.

Algorithm 1 presents the full training procedure for this ensemble of neural networks, containing modifications discussed in subsequent sections. Figure 1 showcases the algorithm in a toy example. Consider an ensemble parameterized by $\boldsymbol{\Theta} = \begin{bmatrix} \boldsymbol{\theta}_1 & \cdots & \boldsymbol{\theta}_T \end{bmatrix}^\top$. Concretely, at each training step with inputs $\mathbf{x}$ and targets $\mathbf{y}$ a random $\boldsymbol{\alpha}$ is sampled and the corresponding convex combination of the networks is constructed $\boldsymbol{\theta} = \boldsymbol{\alpha}^\top \boldsymbol{\Theta}$ (line 7). This procedure is shown in Figure 2. The batch is forwarded through the constructed network and the vector loss is scalarized by $\boldsymbol{\alpha}$ as well, as in line 8. The procedure is repeated W times at each batch (see Section 4.2) and a regularization term penalizing non-Pareto stationary points is added (line 11).

Claim 4.1. Let $\{\boldsymbol{\theta}_t^* \in \mathcal{R} : t \in [T]\}$ be the optimal ensemble parameters retrieved at the end of training by Algorithm 1 and let $\mathcal{S}$ be the their convex hull. Then $\mathcal{S}$ is a Pareto Subspace.

Note that we have chosen a convex hull parameterization of

Algorithm 1: `ParetoManifoldLearning`

Input: vector loss function $\boldsymbol{L}$, train set $\mathcal{D}$, matrix of model parameters $\boldsymbol{\Theta} = \left[\boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_T\right]^\top$, distribution parameters $\boldsymbol{p}$, window $W \in \mathbb{N}$, regularization coefficient $\lambda > 0$, network f

1 Initialize each $\boldsymbol{\theta}_v$ independently
2 **for** *batch* $(\boldsymbol{x}, \boldsymbol{y}) \subseteq \mathcal{D}$ **do**
3 $\mathcal{V} \leftarrow \varnothing$
4 **for** $i \in \{1, 2, \ldots, W\}$ **do**
5 sample $\boldsymbol{\alpha}_i \sim \text{Dir}(\boldsymbol{p})$
6 $\mathcal{V} \leftarrow \mathcal{V} \cup \boldsymbol{\alpha}_i$
7 $\boldsymbol{\theta}_i \leftarrow \boldsymbol{\alpha}_i^\top \boldsymbol{\Theta}$ `// construct network in convex hull of ensemble members`
8 $\boldsymbol{L}(\boldsymbol{\alpha}_i) = \left[\mathcal{L}_1(\boldsymbol{\alpha}_i) \quad \cdots \quad \mathcal{L}_T(\boldsymbol{\alpha}_i)\right] \leftarrow$ `criterion`$\left(f(\boldsymbol{x}; \boldsymbol{\theta}_i), \boldsymbol{y}\right)$ `// compute losses`
9 **end**
10 construct multi-forward graphs $\mathcal{G}_t = (\mathcal{V}, \mathcal{E}_t), \forall t$
11 $\mathcal{R} \leftarrow \sum_{t=1}^{T} \log\left(\frac{1}{|\mathcal{E}_t|} \sum_{(\boldsymbol{\alpha}_i, \boldsymbol{\alpha}_j) \in \mathcal{E}_t} e^{[\mathcal{L}_t(\boldsymbol{\alpha}_i) - \mathcal{L}_t(\boldsymbol{\alpha}_j)]_+}\right)$ `// multiforward regularization, Section 4.2`
12 $\mathcal{L}_{\text{total}} \leftarrow \sum_{i=1}^{W} \boldsymbol{\alpha}_i^\top \boldsymbol{L}(\boldsymbol{\alpha}_i) + \lambda \cdot \mathcal{R}$
13 Backpropagate $\mathcal{L}_{\text{total}}$ and Gradient *descent* on $\boldsymbol{\Theta}$
14 **end**

the weight space, but there are other options, such as Bezier curves or other nonlinear paths (Wortsman et al., 2021; Draxler et al., 2018). However, the universal approximation theorem implies no loss of generality for our design choice and Theorem 4.2 attests to the existence of such parameterizations. In practice, Theorem 4.1 is validated by uniformly sampling the discovered subspace and the definition of a *Pareto Subspace* is relaxed to conform to the nonconvex settings of Deep Learning, i.e., points are called Pareto optimal if the characterization holds in an open neighborhood rather than globally.

Theorem 4.2. *Given a compact $A \subset \mathbb{R}^D$ and a family of continuous mappings $f_n : A \to \mathbb{R}^{D'}, n = 1, \ldots, N$, for any $\epsilon > 0$, there exists a ReLU multi-layer perceptron f with two different weight parameterizations $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$, such that $\forall n \in \{1, \ldots, N\}, \exists \alpha \in [0, 1], \forall \boldsymbol{x} \in A$,*

$$|f_n(\boldsymbol{x}) - f\left(\boldsymbol{x}; \alpha\boldsymbol{\theta} + (1 - \alpha)\,\boldsymbol{\theta}'\right)| \leq \epsilon.$$

The proof is given in Section A.3. Theorem 4.2 grounds the geometrical intuition; due to overparameterization and the universal approximation theorem the Pareto Front admits a linear paramaterization.

4.2. Regularization and balancing

Loss and gradient balancing schemes A common challenge in MTL is the case where tasks have different loss scales, e.g., consider datasets with regression and classification tasks such as `UTKFace` (Zhang et al., 2017). Then, using the same weighting $\boldsymbol{\alpha}$ for both the losses and the weight ensembling, as presented in Equation 2, the easiest tasks are favored and the important property of scale invariance is neglected. To prevent this, the loss weighting needs to be adjusted. Hence, we propose simple balancing schemes: one loss and one gradient balancing scheme, whose effect is to warp the space of loss weightings. While gradient balancing schemes are applied on the shared parameters, loss balancing also affects the task-specific decoders, rendering the methodologies complementary. To avoid cluttering, balancing schemes are not presented in Algorithm 1.

In terms of loss balancing, we use a lightweight scheme of adding a normalization coefficient to each loss term which depends on past values. Concretely, let $W \in \mathbb{Z}_+$ be a positive integer and $\mathcal{L}_m(\tau_0)$ be the loss of task m in step τ_0. Then, the regularization coefficient is $\overline{\mathcal{L}}(\tau_0; W) = \frac{1}{W}\sum_{\tau=1}^{W} \mathcal{L}_m(\tau_0 + 1 - \tau)$ for $\tau_0 \geq W$ resulting in the overall loss $\mathcal{L}_{total} = \boldsymbol{\alpha}_{\tau_0}^\top \widehat{\boldsymbol{L}} = \sum_{t=1}^{T} \alpha_t \frac{\mathcal{L}_t(\tau_0)}{\overline{\mathcal{L}_m}(\tau_0; W)}$. For gradient balancing, let $\boldsymbol{g}_t$ be the gradient of task $t \in [T]$ w.r.t. the shared parameters. Previously, the update rule occurred with the overall gradient $\boldsymbol{g}_{total} = \boldsymbol{\alpha}^\top \boldsymbol{G} = \boldsymbol{\alpha}^\top \left[\boldsymbol{g}_1 \quad \ldots \quad \boldsymbol{g}_T\right]$. We impose a unit ℓ_2-norm for gradients and perform the update with $\widetilde{\boldsymbol{g}}_{total} = \boldsymbol{\alpha}^\top \widetilde{\boldsymbol{G}} = \boldsymbol{\alpha}^\top \left[\widetilde{\boldsymbol{g}}_1 \quad \ldots \quad \widetilde{\boldsymbol{g}}_T\right]$ where $\widetilde{\boldsymbol{g}}_t = \frac{\boldsymbol{g}_t}{\|\boldsymbol{g}_t\|_2}$.

Improving stability by Multi-Forward batch regularization Consider two different weightings $\boldsymbol{\alpha}_1$ and $\boldsymbol{\alpha}_2 \in \Delta_{T-1}$. Without loss of generality $[\boldsymbol{\alpha}_1]_0 = \alpha_1 > [\boldsymbol{\alpha}_2]_0 = \alpha_2$. Then, ideally, the interpolated model closer to the ensemble member for task 1 has the lowest loss on that task, i.e., we would want the ordering $\mathcal{L}_1(\boldsymbol{\alpha}_1) < \mathcal{L}_1(\boldsymbol{\alpha}_2)$, and, equivalently for the other tasks. Furthermore, if $\boldsymbol{\alpha} = \left[1 - \epsilon, \epsilon/T-1, \ldots, \epsilon/T-1\right]$, only one member essentially reaps the benefits of the gradient update and moves the ensemble towards weight configurations more suitable for one task but, perhaps deleterious for the remaining ones. Thus, we propose repeating the forward pass W times for different random weightings $\{\boldsymbol{\alpha}_i\}_{i \in [W]}$, allowing the advancement of all ensemble members concurrently in a coordinated way (line 10). By performing multiple forward passes for various weightings, we achieve a lower discrepancy sequence and reduce the variance of such pernicious updates.

We also include a regularization term, which penalizes the wrong orderings and encourages the subspace to have Pareto properties, as in line 12. Let $\mathcal{V}$ be the set of interpolation weighs sampled in the current batch $\mathcal{V} = \{\boldsymbol{\alpha}_w = (\alpha_{w,1}, \alpha_{w,2}, \ldots, \alpha_{w,T}) \in \Delta_{T-1}\}_{w \in [W]}$. Then each task defines the *directed* graph $\mathcal{G}_t = (\mathcal{V}, \mathcal{E}_t)$ where $\mathcal{E}_t = \{(\boldsymbol{\alpha}_i, \boldsymbol{\alpha}_j) \in \mathcal{V} \times \mathcal{V} : \alpha_{i,t} < \alpha_{j,t}\}$. The resulting

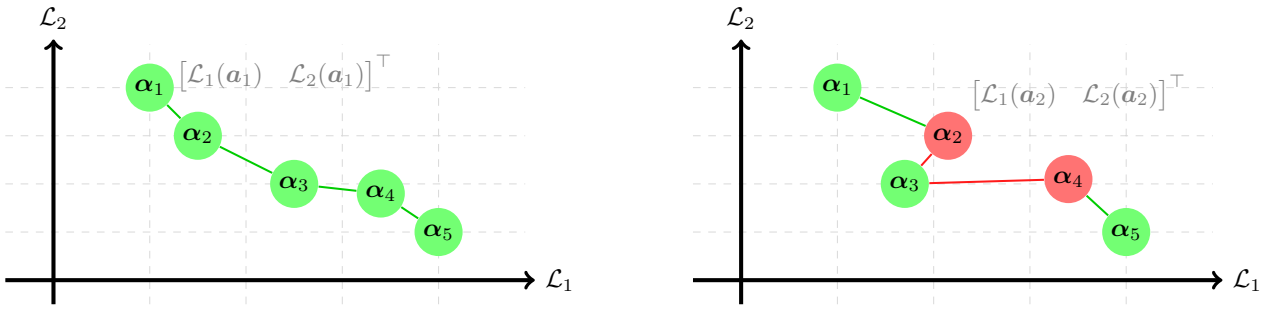


Figure 3: Visual explanation of multiforward regularization, presented in Equation 3. The subfigures depict the loss values for various weightings $\boldsymbol{\alpha}_i = [\alpha_{i,1}, \alpha_{i,2}]$. Optimal lies in the origin. We assume that $\alpha_{1,1} > \cdots > \alpha_{5,1}$. Green color corresponds to Pareto optimality. (Left) all sampled weightings are in the Pareto Front and the regularization term is zero. (Right) The red points are not optimal and, therefore, the regularization term penalizes the violations of the monotonicity constraints for the appropriate task loss: $\boldsymbol{\alpha}_2$ and $\boldsymbol{\alpha}_4$ violate the $\mathcal{L}_1$ and $\mathcal{L}_2$ orderings w.r.t. $\boldsymbol{\alpha}_3$, since $\alpha_{2,1} > \alpha_{3,1} \nRightarrow \mathcal{L}_1(\boldsymbol{\alpha}_2) < \mathcal{L}_1(\boldsymbol{\alpha}_3)$ and $\alpha_{4,2} > \alpha_{3,2} \nRightarrow \mathcal{L}_2(\boldsymbol{\alpha}_4) < \mathcal{L}_2(\boldsymbol{\alpha}_3)$.

regularization is defined as:

$$\mathcal{L}_{reg} = \sum_{t=1}^{T} \log \left(\frac{1}{|\mathcal{E}_t|} \sum_{(\boldsymbol{\alpha}_i, \boldsymbol{\alpha}_j) \in \mathcal{E}_t} e^{[\mathcal{L}_t(\boldsymbol{\alpha}_i) - \mathcal{L}_t(\boldsymbol{\alpha}_j)]_+} \right) \quad (3)$$

The current formulation of the edge set penalizes heavily the connections from vertices with low values. For this reason, we only keep one outgoing edge per node, defined by the task lexicographic order, resulting in the graph $\mathcal{G}_t^{\text{LEX}} = (\mathcal{V}, \mathcal{E}_t^{\text{LEX}})$ and $|\mathcal{E}_t^{\text{LEX}}| = W - 1, \forall t \in [T]$. Note that the regularization is convex as the sum of *log-sum-exp* terms. If no violations occur, the regularization term is zero. Figure 3 offers a visual explanation of the proposed regularization.

The role of sampling Another component of Algorithm 1 is the sampling imposed on the convex hull parameterization. During training, the sampling distribution dictates the loss weighting used and, hence, modulates the degree of task learning. A natural choice is the Dirichlet distribution $\text{Dir}(\boldsymbol{p})$ where $\boldsymbol{p} \in \mathbb{R}_{>0}^T$ are the concentration parameters, since its support is the T-dimensional simplex Δ_T. For $\boldsymbol{p} = p\mathbf{1}_T$, the distribution is symmetric; for $p < 1$ the sampling is more concentrated near the ensemble members, for $p > 1$ it is near the centre and for $p = 1$ it corresponds to the uniform distribution. In contrast, for $p_1 \neq p_2$ the distribution is skewed. In our experiments, we use symmetric Dirichlet distributions with $p \geq 1$ to guide the ensemble to representations best suited for MTL.

5. Experiments

We evaluate our method on several datasets, such as `MultiMNIST`, `Census`, `MultiMNIST-3`, `UTKFace` and `CityScapes`, and various architectures, ranging from MultiLayer Perceptrons (MLPs) to Convolutional Neural Networks (CNNs) and Residual Networks (ResNets). Each ensemble member is initialized independently. In all experiments, the learning rate for our method is m-fold the learning rate of the baselines to counteract the fact that the backpropagation step scales the gradients by m^{-1} in expectation. The detailed settings used for each dataset and additional experiments are provided in the appendix. Our overarching objective is to construct continuous weight subspaces which map to Pareto Fronts in the functional space. However, our method produces a continuum of results rather than a single point, rendering tabular presentation cumbersome. For this reason, (a) for tables we present the best-of-(sampled)-subspace results, (b) we experiment on numerous two-task datasets where plots convey the results succinctly, (c) present qualitative results on three-task datasets.

Baselines We explore various algorithms from the literature: 1. Single-Task Learning (STL), 2. Linear Scalarization (LS) which minimizes the average loss $\frac{1}{T}\sum_{t=1}^{T} \mathcal{L}_t$, 3. Uncertainty Weighting (UW, Cipolla et al. 2018), 4. Multiple-gradient descent algorithm (MGDA, Sener & Koltun 2018), 5. Dynamic Weight Averaging (DWA, Liu et al. 2019), 6. Projecting Conflicting Gradients (PCGrad, Yu et al. 2020), 7. Impartial MTL (IMTL, Liu et al. 2020), 8. Just Pick a Sign (Graddrop, Chen et al. 2020), 9. Conflict-Averse Gradient Descent (CAGrad, Liu et al. 2021), 10. Random Loss Weighting (RLW, Lin et al. 2022). 11. Bargaining MTL (Nash-MTL, Navon et al. 2022), 12. Auto-Lambda (Auto-λ, Liu et al. 2022) and 13. RotoGrad (Javaloy & Valera 2022). When applicable, we also explore methodologies that perform Pareto Front Approximation (PFA) in a single training run; as in 14. Pareto HyperNetwork (PHN, Navon et al. 2021), 15. Conditioned One-shot Multi-Objective Search (COSMOS, Ruchte & Grabocka 2021).

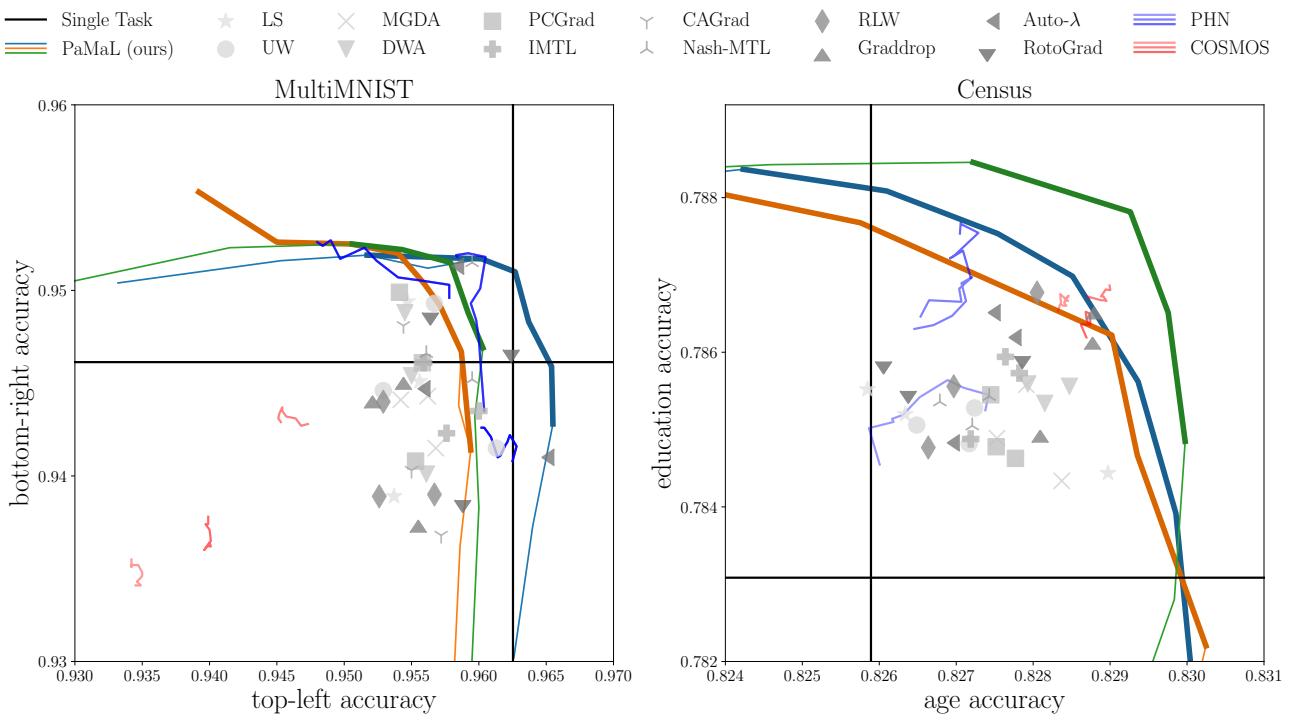
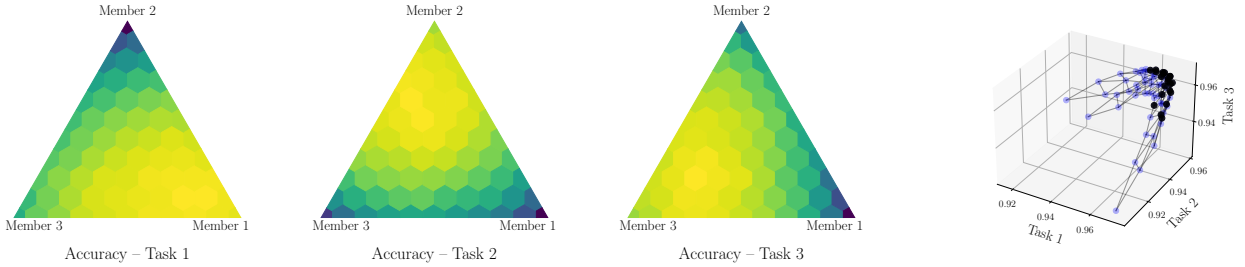


Figure 4: Experimental results on `MultiMNIST` and `Census`. Top right is optimal. Three random seeds per method. Solid lines correspond to our method (PaMaL) and thick lines to the Pareto Front. We have used a different color for each seed of PaMaL. Baselines are shown in shades of gray: scatter plot for MTL baselines, black lines for single task and blue/red lines for multiple-solution methods. In both datasets, Pareto Manifold Learning discovers subspaces with diverse and Pareto-optimal solutions and outperforms the baselines.

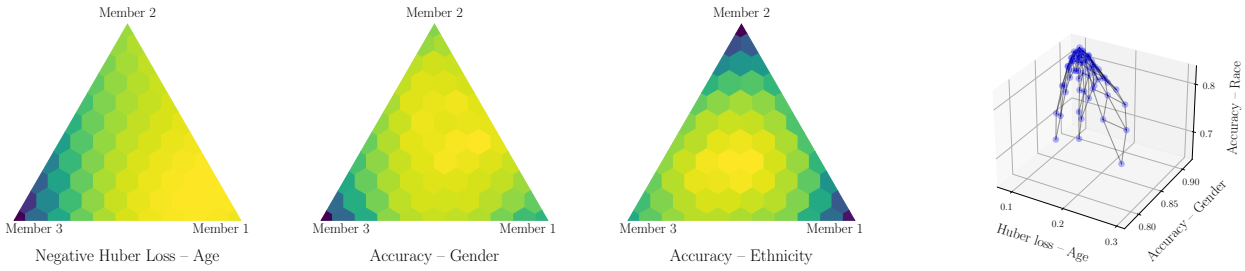
5.1. Classification on `MultiMNIST` and `Census`

We investigate the effectiveness of Pareto Manifold Learning on digit classification using a LeNet model with a shared-bottom architecture and on the tabular dataset `Census` (Kohavi, 1996) for the task combination of predicting age and education level using a Multi-Layer Perceptron. The ensemble consists of two members with single task weightings. To gauge the performance of the models lying in the linear segment between the nodes, we test the performance on the validation set on the ensemble members as well as for $m = 9$ models uniformly distributed across the edge. We use this evaluation/plotting scheme throughout the experiments. We ablate the effect of multi-forward training on Section B.1; we use a grid search on window $W \in \{2, 3, 4, 5\}$ and strength $\lambda \in \{0, 2, 5, 10\}$ along with the base case of $(W, \lambda) = (1, 0)$ and present in the main text the setting that achieves the highest mean (across seeds) HyperVolume score on the validation set. Figure 4 shows the results on both datasets using multi-forward regularization with window $W = 4$ and strength $\lambda = 0$ for `MultiMNIST` and $W = 2$, $\lambda = 5$ for `Census`. We observe that most baselines exhibit limited functional diversity; their predefined optimization schemes lead the differently seeded/initialized training runs to final models with similar performance (same markers are clustered in the plots). This lack of functional diversity, as well as inability to consistently outperform the Linear Scalarization baseline, are also noted by Kurin et al. (2022); Xin et al. (2022). In contrast, all Pareto Manifold Learning seeds find subspaces with diverse functional solutions. This statement is quantitatively translated to higher HyperVolume compared to the baselines, shown in Table 9 of the appendix, and can be attributed to the observation that Equation 2 generalizes the Linear Scalarization method.

Analysis Task symmetry characterizes `MultiMNIST`; both digits are drawn from the same distribution, resulting in equal pace learning. However, for `Census`, tasks differ in statistics and, yet, the proposed method recovers a Pareto subspace with diverse solutions. For both datasets, we perform extensive tuning on the Pareto Front Approximation methods, i.e., PHN and COSMOS, in Section C.1. For COSMOS, the mapping from user preference to network weights is overall invalid in both datasets. Similarly, PHN produces functionally limited solutions that do not conform to the objective of Pareto optimality, while requiring $\sim 100\times$ more parameters. These results can be attributed to two factors. First, the original papers experimented with an am-



(a) `MultiMNIST-3`: Accuracy Heatmap and Pareto Front for all tasks.



(b) `UTKFace`: Objective Heatmap and Pareto Front for all tasks.

Figure 5: Application of Pareto Manifold Learning on datasets with 3 tasks. Each triangle depicts the performance on a task, using color, as a function of the interpolation weighting, i.e. each hexagon corresponds to a different weighting $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \alpha_3] \in \Delta_3$. The closer the interpolated member is to a single-task predictor, the higher the performance on the corresponding task. The 3D plot, on the right, show the performance of the model in the multi-objective space.

ple budget of ≥ 100 epochs, disproportionate to the dataset complexity, and reported *loss* curves instead of accuracy, neglecting the generalization gap between them. Second, the mapping from trade-off to target model via a Hypernetwork in PHN or input augmentation in COSMOS result in complex dynamics, since an ϵ-step in the user preference is translated to different set of weights by the full forward pass of neural network. In contrast, our approach is grounded in geometrical insights and the connection is traced back to a simple linear interpolation. In Figure 19 of the appendix, we supplement the ablation study for PHN and COSMOS by Spearman correlation as a proxy for monotonicity of ranking in the Pareto prism and show that these baselines must sacrifice performance in order to achieve the promise of functional diversity.

5.2. Beyond Pairs of Classification Tasks: `MultiMNIST-3` and `UTKFace`

We expand the experimental validation to triplets of tasks, consider regression and more complex architectures, graduating from MLPs and CNNs to ResNets (He et al., 2016). For three tasks, we create a 2D grid of equidistant points spanning the three single-task predictors. If n is the number of interpolated points between two (out of three) members, the grid has $\binom{n+1}{2}$ points. We use $n = 11$, resulting in 66 points. For visual purposes, neighboring points are connected. For three tasks, it would be visually cluttering to present the discovered subspaces with multiple seeds and baselines. Hence, we opt for a more qualitative discussion here and present quantitative findings in the appendix.

`MultiMNIST-3` First, we construct an equivalent of `MultiMNIST` for 3 tasks. Digits are placed on top-left, top-right and bottom-centre. Figure 5a shows the results on `MultiMNIST-3`. As argued previously, `MNIST` variants are characterized by task symmetry and Figure 5a reflects this. For this reason, we do not employ any balancing scheme. The 3D plot in conjunction with the simplices reveal that the method has the effect of gradual transfer of learned representation from one member to the other, and offers a succinct visual confirmation of Theorem 4.1.

`UTKFace` The `UTKFace` dataset (Zhang et al., 2017) has more than 20,000 face images and three tasks: predicting age (modeled as regression using Huber loss - similar to (Ma et al., 2020)), classifying gender and ethnicity. The introduction of a regression task implies that losses have vastly different scales, which dictates the use of balancing schemes, as discussed in Section 4.2. We apply the proposed gradient-balancing scheme and present the results in Figure 5b. For visual unity and to remain in the theme of "higher is better", the *negative* Huber loss is plotted. Despite the increased complexity, both in terms of network archi-

tecture and dataset, and the existence of a regression task, the proposed method discovers a *Pareto Subspace*. Additional experiments and qualitative results are provided in Section B.3. Figure 5, and in more detail Section A.2, show that (most of) the subspace engenders high performance and, implicitly low loss, for each task separately. Hence, the approach discovers flat regions which are linked to generalization (Foret et al., 2021). There is a also dynamic transition in the weighted (multi-task) loss landscape w.r.t. weight $\boldsymbol{\alpha}$, which leads to the desired Pareto properies.

5.3. Scene understanding

We also explore the applicability of Pareto Manifold Learning for `CityScapes` (Cordts et al., 2016), a scene understanding dataset containing high-resolution images of urban street scenes. Our experimental configuration is drawn from (Liu et al., 2019; Yu et al., 2020; Liu et al., 2021; Navon et al., 2022) with some modifications. Concretely, we address two tasks: semantic segmentation and depth regression. We use a SegNet architecture (Badrinarayanan et al., 2017) trained for 100 epochs with Adam optimizer (Kingma & Ba, 2015) of initial learning rate 10^{-4}, which is halved after 75 epochs. The images are resized to 128×256 pixels. We use 500 of the 2975 training images for validation, and report the test results in Table 1. We use gradient balancing, window $W = 3$ and $\lambda = 1$, while the concentration parameter of the Dirichlet distribution is set to $p = 7$, helping convergence. Additional results are presented in Section C.4. In Depth Estimation and out of MTL methods, Pareto Manifold Learning is optimal along with MGDA (lower is better). In Semantic Segmentation (higher is better), however, MGDA performs poorly and is clearly dominated by all methods, while our approach is outperforming most baselines and offers a balanced solution. Compared to the multi-solution baselines, COSMOS showcases task bias performing poorly on Depth Estimation, while PHN is ommitted altogether due to not scaling to large networks. It is remarkable that, despite our goal of discovering *Pareto subspaces*, the proposed method is dominating most of the state-of-the-art algorithms, attesting to the flexibility of the weight ensembles in Multi-Task Learning.

Table 1: Test performance on *CityScapes*. 3 random seeds per method. For Pareto Manifold Learning, we report the mean (across seeds) best results from the final subspace. Methods are divided into single-task, single-solution MTL, multi-solution MTL and proposed method.

	Segmentation		Depth	
	mIoU ↑	Pix Acc ↑	Abs Err ↓	Rel Err ↓
STL	70.96	92.12	**0.0141**	**38.644**
LS	70.12	91.90	0.0192	124.061
UW	70.20	91.93	0.0189	125.943
MGDA	66.45	90.79	**0.0141**	53.138
DWA	70.10	91.89	0.0192	127.659
PCGrad	70.02	91.84	0.0188	126.255
IMTL	70.77	92.12	0.0151	74.230
Graddrop	70.07	91.93	0.0189	127.146
CAGrad	69.23	91.61	0.0168	110.139
RLW	68.79	91.52	0.0213	126.942
Nash-MTL	**71.13**	**92.23**	0.0157	78.499
RotoGrad	69.92	91.85	0.0193	127.281
Auto-λ	70.47	92.01	0.0177	116.959
COSMOS	69.78	91.79	0.0539	136.614
PaMaL(ours)	70.35	91.99	**0.0141**	54.520

6. Conclusion

In this paper, we proposed a weight-ensembling method tailored to Multi-Task Learning; multiple single-task predictors are trained in conjunction to produce a subspace formed by their convex hull, and endowed with desirable Pareto properties. We experimentally show on a diverse suite of benchmarks that the proposed method is successful in discovering *Pareto subspaces* and outperforms or is on par with state-of-the-art MTL methods. An interesting future direction is to perform a hierarchical weight ensembling, sharing progressively more of the lower layers, given that the features learned at low depth are similar across tasks. An alternative exploration venue is to connect our method to the challenge of task affinity (Fifty et al., 2021; Standley et al., 2020) via a geometrical lens of the loss landscape.

Acknowledgments

The work of Nikolaos Dimitriadis was supported by Swisscom (Switzerland) AG. We would like to thank Guillermo Ortiz-Jiménez, Apostolos Modas, Clément Vignac, Prabhu Teja, and the anonymous reviewers for their valuable feedback.

References

Badrinarayanan, V., Kendall, A., and Cipolla, R. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (12):2481–2495, 2017.

Caruana, R. Multitask Learning. *Machine Learning*, (1):41–75, 1997. URL https://doi.org/10.1023/A:1007379606734.

Chaudhari, P., Choromanska, A., Soatto, S., LeCun, Y., Baldassi, C., Borgs, C., Chayes, J. T., Sagun, L., and Zecchina, R. Entropy-SGD: Biasing Gradient Descent into Wide Valleys. In *International Conference on Learning Representations*, 2017. URL http://arxiv.org/abs/1611.01838v5.

Chen, Z., Badrinarayanan, V., Lee, C.-Y., and Rabinovich, A. GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks. In *International Conference on Machine Learning*, 2018. URL http://arxiv.org/abs/1711.02257v4.

Chen, Z., Ngiam, J., Huang, Y., Luong, T., Kretzschmar, H., Chai, Y., and Anguelov, D. Just pick a sign: Optimizing deep multitask models with gradient sign dropout. In *Advances in Neural Information Processing Systems*, 2020.

Cipolla, R., Gal, Y., and Kendall, A. Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. URL http://arxiv.org/abs/1705.07115v3.

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B. The Cityscapes Dataset for Semantic Urban Scene Understanding. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. URL http://arxiv.org/abs/1604.01685v2.

Crawshaw, M. Multi-Task Learning with Deep Neural Networks: A Survey. *arXiv:2009.09796 [cs, stat]*, 2020. URL http://arxiv.org/abs/2009.09796v1.

Désidéri, J.-A. Multiple-Gradient Descent Algorithm (MGDA) for Multiobjective Optimization. *Comptes Rendus Mathematique*, (5-6):313–318, 2012.

Dinh, L., Pascanu, R., Bengio, S., and Bengio, Y. Sharp minima can generalize for deep nets. In *International Conference on Machine Learning*, 2017. URL http://arxiv.org/abs/1703.04933v2.

Draxler, F., Veschgini, K., Salmhofer, M., and Hamprecht, F. A. Essentially No Barriers in Neural Network Energy Landscape. In *International Conference on Machine Learning*, 2018. URL http://arxiv.org/abs/1803.00885v5.

Fifty, C., Amid, E., Zhao, Z., Yu, T., Anil, R., and Finn, C. Efficiently Identifying Task Groupings for Multi-Task Learning. In *Advances in Neural Information Processing Systems*, 2021.

Foret, P., Kleiner, A., Mobahi, H., and Neyshabur, B. Sharpness-Aware Minimization for Efficiently Improving Generalization. In *International Conference on Learning Representations*, 2021. URL http://arxiv.org/abs/2010.01412v3.

Frankle, J., Dziugaite, G. K., Roy, D. M., and Carbin, M. Linear Mode Connectivity and the Lottery Ticket Hypothesis. In *International Conference on Machine Learning*, 2020. URL http://arxiv.org/abs/1912.05671v4.

Garipov, T., Izmailov, P., Podoprikhin, D., Vetrov, D. P., and Wilson, A. G. Loss Surfaces, Mode Connectivity, and Fast Ensembling of DNNs. In *Advances in Neural Information Processing Systems*, 2018.

Ha, D., Dai, A. M., and Le, Q. V. HyperNetworks. In *International Conference on Learning Representations*, 2017. URL http://arxiv.org/abs/1609.09106v4.

Havasi, M., Jenatton, R., Fort, S., Liu, J. Z., Snoek, J., Lakshminarayanan, B., Dai, A. M., and Tran, D. Training independent subnetworks for robust prediction. In *International Conference on Learning Representations*, 2021. URL http://arxiv.org/abs/2010.06610v2.

He, K., Zhang, X., Ren, S., and Sun, J. Deep Residual Learning for Image Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. URL http://arxiv.org/abs/1512.03385v1.

Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D. P., and Wilson, A. G. Averaging Weights Leads to Wider Optima and Better Generalization. In *Conference on Uncertainty in Artificial Intelligence*, 2018. URL http://arxiv.org/abs/1803.05407v3.

Javaloy, A. and Valera, I. RotoGrad: Gradient homogenization in multitask learning. In *International Conference on Learning Representations*, 2022. URL http://arxiv.org/abs/2103.02631v3.

Jiang, Y., Neyshabur, B., Mobahi, H., Krishnan, D., and Bengio, S. Fantastic Generalization Measures and Where to Find Them. In *International Conference on Learning Representations*, 2020. URL http://arxiv.org/abs/1912.02178v1.

Kingma, D. P. and Ba, J. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*, 2015. URL http://arxiv.org/abs/1412.6980v9.

Kohavi, R. Scaling up the Accuracy of Naive-Bayes Classifiers: A Decision-Tree Hybrid. In *International Conference on Knowledge Discovery and Data Mining*, 1996.

Kurin, V., De Palma, A., Kostrikov, I., Whiteson, S., and Mudigonda, P. K. In defense of the unitary scalarization for deep multi-task learning. In *Advances in Neural Information Processing Systems*, 2022.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and Scalable Predictive Uncertainty Estimation Using Deep Ensembles. In *Advances in Neural Information Processing Systems*, 2017.

Lin, B., Feiyang, Y., Zhang, Y., and Tsang, I. Reasonable effectiveness of random weighting: A litmus test for multi-task learning. *Transactions on Machine Learning Research*, 2022. URL http://arxiv.org/abs/2111.10603v2.

Lin, X., Zhen, H.-L., Li, Z., Zhang, Q., and Kwong, S. Pareto Multi-Task Learning. In *Advances in Neural Information Processing Systems*, 2019.

Lin, X., Yang, Z., Zhang, Q., and Kwong, S. Controllable Pareto Multi-Task Learning. (arXiv:2010.06313), 2021. URL http://arxiv.org/abs/2010.06313v2.

Liu, B., Liu, X., Jin, X., Stone, P., and Liu, Q. Conflict-Averse Gradient Descent for Multi-task Learning. In *Advances in Neural Information Processing Systems*, 2021.

Liu, L., Li, Y., Kuang, Z., Xue, J.-H., Chen, Y., Yang, W., Liao, Q., and Zhang, W. Towards Impartial Multi-task Learning. In *International Conference on Learning Representations*, 2020.

Liu, S., Johns, E., and Davison, A. J. End-To-End Multi-Task Learning With Attention. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019. URL http://arxiv.org/abs/1803.10704v2.

Liu, S., James, S., Davison, A. J., and Johns, E. Auto-lambda: Disentangling dynamic task relationships. *Transactions on Machine Learning Research*, 2022. URL http://arxiv.org/abs/2202.03091v2.

Long, M., Cao, Z., Wang, J., and Yu, P. S. Learning multiple tasks with multilinear relationship networks. In *Advances in Neural Information Processing Systems*, 2017.

Ma, P., Du, T., and Matusik, W. Efficient Continuous Pareto Exploration in Multi-Task Learning. In *International Conference on Machine Learning*. PMLR, 2020. URL http://arxiv.org/abs/2006.16434v2.

Mahapatra, D. and Rajan, V. Multi-Task Learning with User Preferences: Gradient Descent with Controlled Ascent in Pareto Optimization. In *International Conference on Machine Learning*, 2020.

Misra, I., Shrivastava, A., Gupta, A., and Hebert, M. Cross-Stitch Networks for Multi-task Learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. URL http://arxiv.org/abs/1604.03539v1.

Navon, A., Shamsian, A., Fetaya, E., and Chechik, G. Learning the Pareto Front with Hypernetworks. In *International Conference on Learning Representations*, 2021. URL http://arxiv.org/abs/2010.04104v2.

Navon, A., Shamsian, A., Achituve, I., Maron, H., Kawaguchi, K., Chechik, G., and Fetaya, E. Multi-Task Learning as a Bargaining Game. In *International Conference on Machine Learning*, 2022. URL http://arxiv.org/abs/2202.01017v2.

Ruchte, M. and Grabocka, J. Scalable Pareto Front Approximation for Deep Multi-Objective Learning. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, 2021. URL http://arxiv.org/abs/2103.13392v2.

Ruder, S. An Overview of Multi-Task Learning in Deep Neural Networks. *arXiv:1706.05098 [cs, stat]*, 2017. URL http://arxiv.org/abs/1706.05098v1.

Ruder, S., Bingel, J., Augenstein, I., and Søgaard, A. Latent Multi-Task Architecture Learning. In *AAAI Conference on Artificial Intelligence*, 2019.

Sener, O. and Koltun, V. Multi-Task Learning as Multi-Objective Optimization. In *Advances in Neural Information Processing Systems*, 2018.

Shen, J., Zhen, X., Worring, M., and Shao, L. Variational multi-task learning with Gumbel-softmax priors. In *Advances in Neural Information Processing Systems*, 2021.

Standley, T., Zamir, A. R., Chen, D., Guibas, L. J., Malik, J., and Savarese, S. Which Tasks Should Be Learned Together in Multi-task Learning? In *International Conference on Machine Learning*, 2020. URL http://arxiv.org/abs/1905.07553v4.

Wen, Y., Tran, D., and Ba, J. BatchEnsemble: an alternative approach to efficient ensemble and lifelong learning. In *International Conference on Learning Representations*, 2020. URL http://arxiv.org/abs/2002.06715v2.

Wortsman, M., Horton, M., Guestrin, C., Farhadi, A., and Rastegari, M. Learning Neural Network Subspaces. In *International Conference on Machine Learning*, 2021. URL http://arxiv.org/abs/2102.10472v3.

Wortsman, M., Ilharco, G., Gadre, S. Y., Roelofs, R., Lopes, R. G., Morcos, A. S., Namkoong, H., Farhadi, A., Carmon, Y., Kornblith, S., and Schmidt, L. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International Conference on Machine Learning*, 2022. URL http://arxiv.org/abs/2203.05482v3.

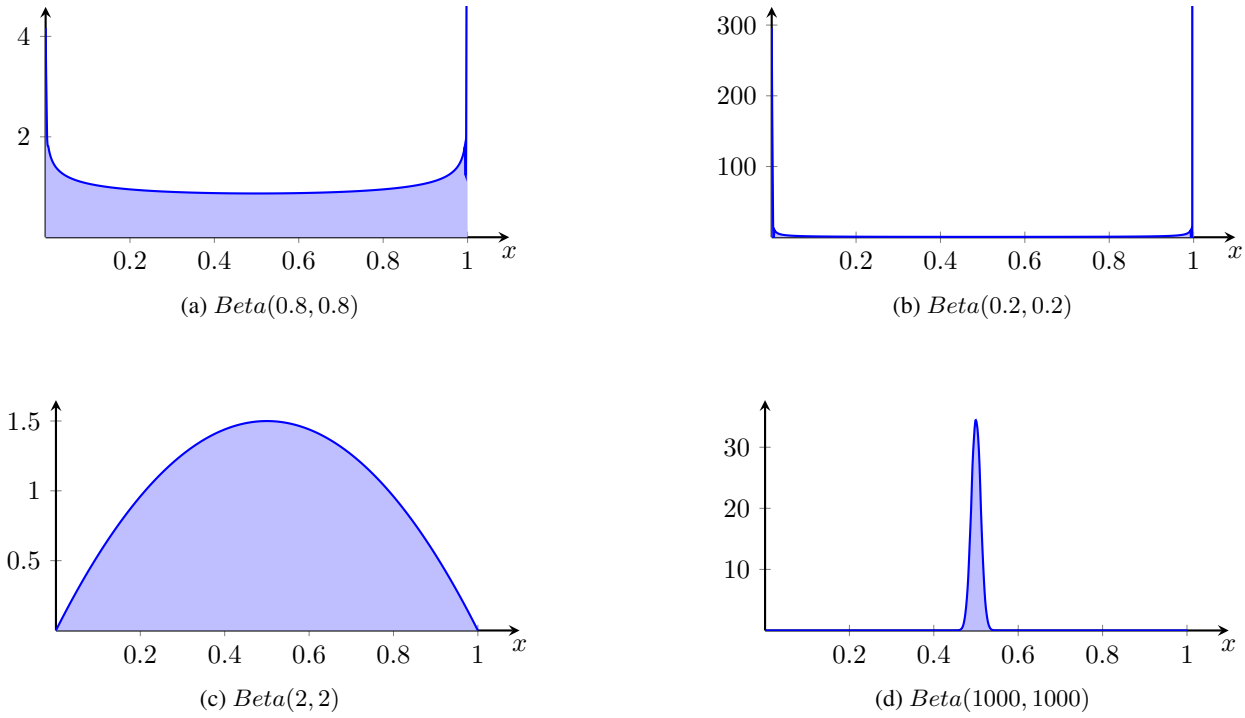
Xin, D., Ghorbani, B., Gilmer, J., Garg, A., and Firat, O. Do Current Multi-Task Optimization Methods in Deep Learning Even Help? In *Advances in Neural Information Processing Systems*, 2022.

Yu, T., Kumar, S., Gupta, A., Levine, S., Hausman, K., and Finn, C. Gradient Surgery for Multi-Task Learning. In *Advances in Neural Information Processing Systems*, 2020.

Zhang, Zhifei, Song, Y., and Qi, H. Age Progression/Regression by Conditional Adversarial Autoencoder. In *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2017.

Appendix Overview

As a reference, we provide the following table of contents solely for the appendix.



(a) $Beta(0.8, 0.8)$ (b) $Beta(0.2, 0.2)$

(c) $Beta(2, 2)$ (d) $Beta(1000, 1000)$

Figure 6: Dirichlet distribution in the case of two tasks. Top row: $p < 1$ and the distribution is more concentrated towards the ensemble members. Bottom row: $p > 1$ and the distribution focuses more on the midpoint which corresponds to all tasks having the same weight. Right column: extreme choices $p \to 0$ or $p \to \infty$. Left column: milder choices.

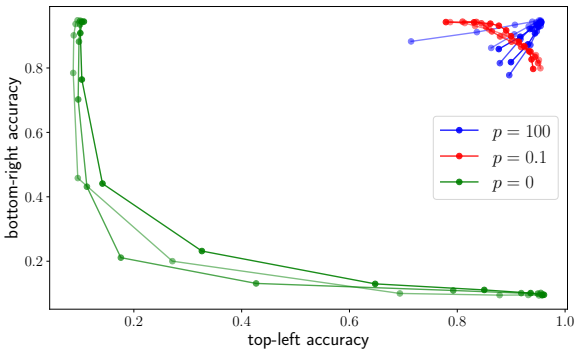
A. Discussion

A.1. Effect of sampling on the Pareto properties of the discovered subspace

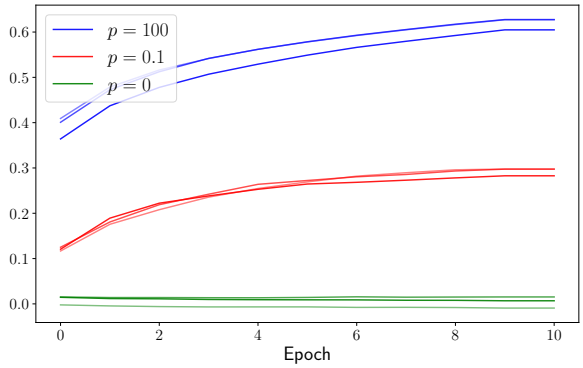
This appendix expands on Section 4.2 and, specifically, presents in greater detail the intuition behind the sampling distribution's parameters. Let $\boldsymbol{p} \in \mathbb{R}_+^T$ be the parameters of the Dirichlet distribution. Assuming no prior knowledge on the tasks, e.g., task difficulties or affinities, a symmetric distribution is used by setting $\boldsymbol{p} = p\mathbf{1}_T$. This design choice results in three cases:

- $p = 1$: the distribution is uniform on the simplex. Intuitively this means that all tasks are equally important and we care about the diversity of solutions for all tradeoffs (reflected in the linear scalarization weights).
- $p \in (0, 1)$: the distribution is more concentrated towards the ensemble members, as in the top row of Figure 6. Assume an extreme case of two tasks and $p = 0$. Then the distribution degenerates to a Bernoulli distribution. Effectively, at each iteration one of the ensemble members is selected and its weights are updated, which will result in two separate and independent single-task predictors with no common representation infused about the other task. Then, linearly interpolating in weight space will result in models with random predictions for both tasks, since the training procedure has not focused in retrieving a Pareto Subspace.

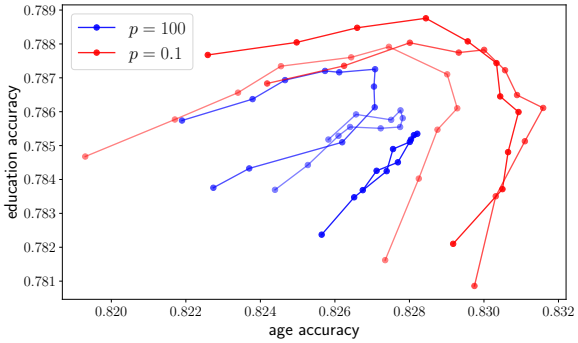
  For milder cases (e.g. $p = 0.7$) , we observed that the models in the middle of the linear interpolation suffered in performance which can be attributed to the fact that the sampling focused more on single-task rather than multi-task representations and performance.
- $p > 1$. Then the distribution is more concentrated towards the midpoint of the simplex, as in the bottom row of Figure 6. Assume an extreme case of two tasks and $p \to \infty$. Then, the distribution becomes deterministic and outputs equal weights for all tasks. The randomly and independently initialized ensemble members will collapse to each other, resulting in duplicate ensemble members. Similarly, for very large values (e.g. $p = 100$), the functional diversity of the ensemble will suffer since the weights produced by the distribution will be almost equal for all tasks, resulting in a milder version of the aforementioned phenomenon. In contrast, we found that small values such as $p = 2$ or $p = 3$ can



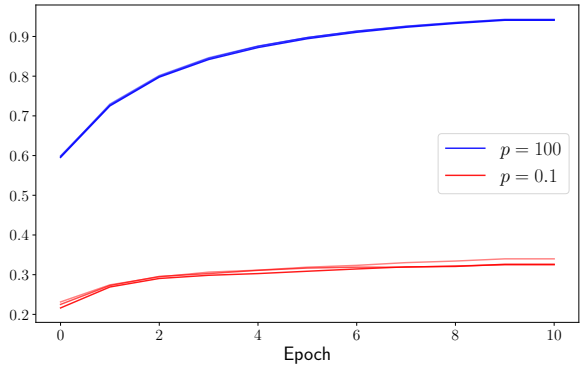
(a) `MultiMNIST`: Experimental results using three random seeds per method.



(b) `MultiMNIST`: Cosine similarities of ensemble members.



(c) `Census`: Experimental results using three random seeds per method.



(d) `Census`: Cosine similarities of ensemble members.

Figure 7: Experimental results on `MultiMNIST` and `Census` varying the concentration parameters $\boldsymbol{p} = p\mathbf{1}_T$ of the sampling distribution. Three seeds depicted in shades of the same colors for the various p.

help convergence since they put more emphasis towards common representation (compared to $p = 1$), but may limit functional diversity.

Figure 7 presents experimental results on `MultiMNIST` and `Census` for various concentration parameters $p \in \{0, 0.1, 100\}$ of the Dirichlet distribution. Let $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ be the parameters of the ensemble members. For $p = 0$, the ensemble consists of two single-task predictors with no multitask learning representational knowledge, since their interpolation meets a low accuracy/high loss barrier. We omit the case of $p = 0$ for `Census` for visual clarity. This lack of common representation is evident in the cosine similarities as well, where for $p = 0$ $\cos(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2) \approx 0$. On the other hand, for $p = 0.1$, common representations are infused into the ensemble and the experimental results show that the test performance is characterized by diversity. However, this comes at the expense of the interpolated models at the middle of the line segment, where the performance is suboptimal compared to $p = 100$ for `MultiMNIST`. This behavior is also illustrated in the cosine similarities, where for $p = 100$ the ensemble weights $\boldsymbol{\alpha}$ are in an ϵ-ball around the midpoint causing the independently initialized models to progressively collapse. For `Census`, we also observe that this collapsing leads to very high cosine similarity $\cos(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2) > 0.9$ and the ensemble is suboptimal compared to $p = 0.1$.

A.2. Connection between Pareto Optimality and multiple valley intersections

In this section, we investigate the connection between the intersection of multiple loss landscapes, pareto optimality and the effect of the proposed algorithm Pareto Manifold Learning. We use the illustrative example, presented in Figure 1. Let $\boldsymbol{\Theta}$ be the parameter space of the model and $\mathcal{L}_t : \boldsymbol{\Theta} \to \mathbb{R}, t \in \{1, 2\}$, be the losses of the problem. For $\alpha \in [0, 1]$ and $\boldsymbol{\theta} \in \boldsymbol{\Theta}$, the overall objective is $\mathcal{L}(\boldsymbol{\theta}, \alpha) = \alpha\mathcal{L}_1(\boldsymbol{\theta}) + (1 - \alpha)\mathcal{L}_2(\boldsymbol{\theta})$.

Figure 8 presents the overall loss objective as α varies from 0 to 1. For the extreme values of the range, the loss landscape is inherently single-task. The subspace discovered by the method is depicted in blue, while a black 'x' is used for the corresponding interpolated model, i.e., it corresponds to $\mathcal{L}(\alpha\boldsymbol{\theta}_1 + (1-\alpha)\boldsymbol{\theta}_2, \alpha)$. In other words, the proposed method tracks the optimum in parameter space as the overall objective evolves and the various loss landscapes are weighted accordingly. While an acceptable multi-task solution lies in the intersection of low loss landscapes, Pareto Manifold Learning focuses on the aforementioned dynamic scenario of loss weighting.

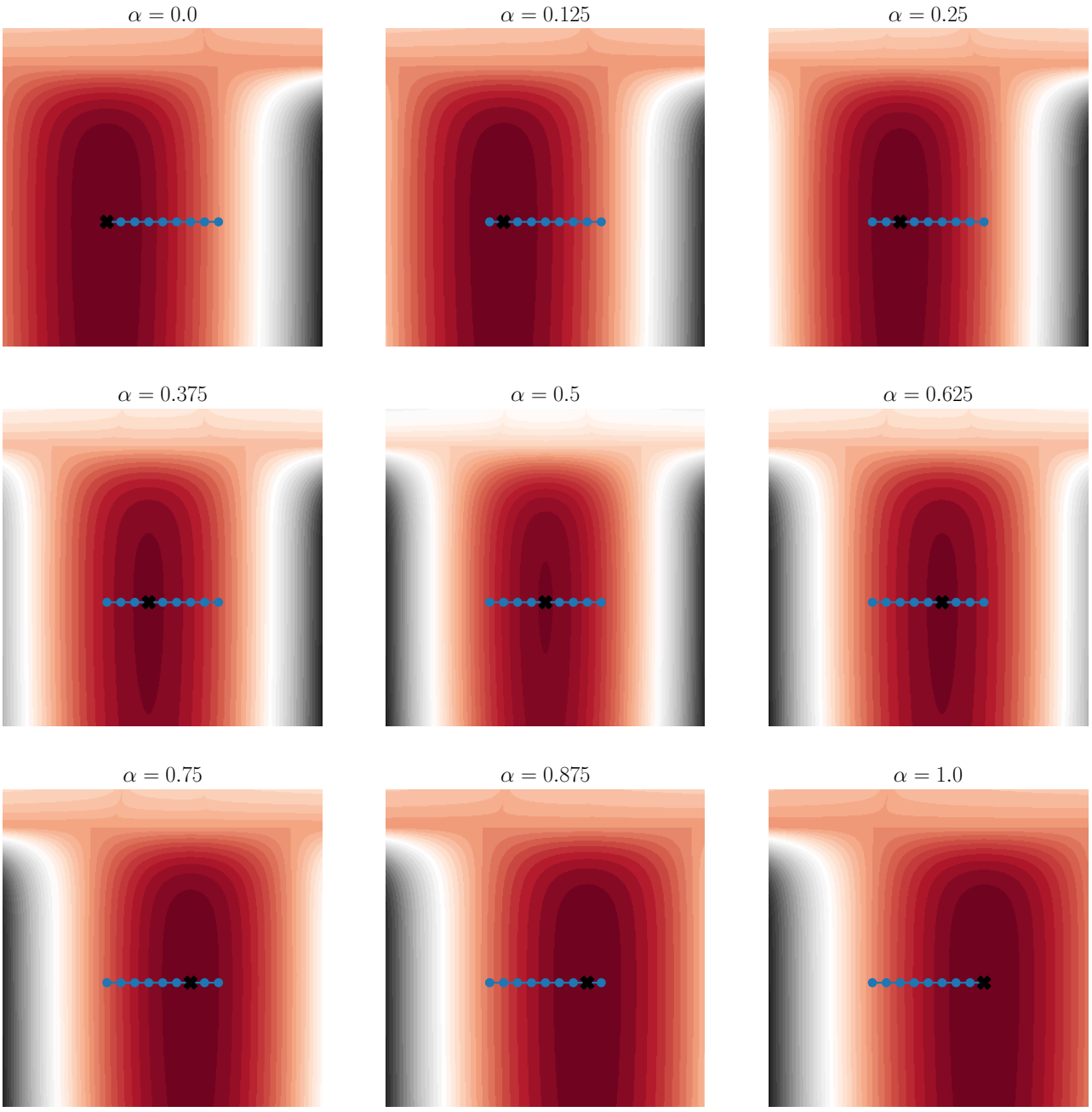


Figure 8: *Illustrative example*: (Overall) loss surface as a function of the model's weights. The overall objective is $\mathcal{L}(\boldsymbol{\theta}, \alpha) = \alpha\mathcal{L}_1(\boldsymbol{\theta}) + (1-\alpha)\mathcal{L}_2(\boldsymbol{\theta})$ and is shown for various values of α. The Pareto subspace discovered by the proposed method is depicted in blue. 'X' shows the solution of the method for the corresponding α.

A.3. Proof of Theorem 4.2

The theorem below shows that given a family of mappings, we can approximate them [arbitrarily accurately] with linear interpolation of two perceptrons in parameter space.

Theorem *Given a compact $A \subset \mathbb{R}^D$ and a family of continuous mappings $f_n : A \to \mathbb{R}^{D'}, n = 1, \dots, N$, for any $\epsilon > 0$, there exists a ReLU multi-layer perceptron f with two different weight parameterizations $\boldsymbol{\theta}$ and $\boldsymbol{\theta'}$, such that*

$$\forall n \in \{1, \dots, N\}, \exists \alpha \in [0, 1], \forall \boldsymbol{x} \in A, |f_n(x) - f\left(\boldsymbol{x}; \alpha\boldsymbol{\theta} + (1-\alpha)\,\boldsymbol{\theta'}\right)| \leq \epsilon$$

Proof. Let σ be the ReLU non-linearity $x \mapsto \max(0, x)$.

From the universal representation theorem, there exists $Q \in \mathbb{N}, \boldsymbol{M} \in \mathbb{R}^{(D+1)\times Q}, \boldsymbol{B} \in \mathbb{R}^Q, \boldsymbol{M'} \in \mathbb{R}^{Q\times D'}$ such that with the one hidden layer perceptron

$$\begin{aligned} g : A \times [0, 1] &\to \mathbb{R}^{D'} \\ \boldsymbol{z} &\mapsto \boldsymbol{M'}\sigma(\boldsymbol{M}\boldsymbol{z} + \boldsymbol{B}), \end{aligned}$$

we have

$$\forall \boldsymbol{x} \in A, \forall n \in \{1, \dots, N\}, \left| f_n(\boldsymbol{x}) - g\left(x_1, \dots, x_D, \frac{n-1}{N-1}\right)\right| \leq \epsilon.$$

Let

$$\boldsymbol{R} = \begin{pmatrix} 1 & 0 & 0 & 0 & & 0 \\ -1 & 0 & 0 & 0 & & 0 \\ 0 & 1 & 0 & 0 & & 0 \\ 0 & -1 & 0 & 0 & \dots & 0 \\ 0 & 0 & 1 & 0 & & 0 \\ 0 & 0 & -1 & 0 & & 0 \\ & \vdots & & & & \vdots \\ 0 & 0 & 0 & 0 & & 1 \\ 0 & 0 & 0 & 0 & \dots & -1 \\ 0 & 0 & 0 & 0 & & 0 \end{pmatrix} \quad \text{and} \quad \boldsymbol{S} = \begin{pmatrix} 1 & -1 & 0 & 0 & 0 & 0 & & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & & 0 & 0 & 0 \\ & & \vdots & & & & & & \vdots & \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & & 0 & 0 & 1 \end{pmatrix},$$

and let $\boldsymbol{U_k} = (\underbrace{0, \dots, 0}_{\times 2D}, k)$.

Then, with $\boldsymbol{x} \in \mathbb{R}^D$, we have

$$\forall \alpha \geq 0, \quad \boldsymbol{S}\sigma(\boldsymbol{R}x + \alpha\boldsymbol{U_1} + (1-\alpha)\boldsymbol{U_0}) = (x_1, \dots, x_D, \alpha).$$

So with $\boldsymbol{\theta} = (\boldsymbol{R}, \boldsymbol{U_1}, \boldsymbol{MS}, \boldsymbol{B}, \boldsymbol{M'})$ and $\boldsymbol{\theta'} = (\boldsymbol{R}, \boldsymbol{U_0}, \boldsymbol{MS}, \boldsymbol{B}, \boldsymbol{M'})$ and

$$f(\boldsymbol{x}; \boldsymbol{r}, \boldsymbol{u}, \boldsymbol{m}, \boldsymbol{b}, \boldsymbol{m'}) = \boldsymbol{m'}\sigma(\boldsymbol{m}\sigma(\boldsymbol{r}\boldsymbol{x} + \boldsymbol{u}) + \boldsymbol{b}),$$

then

$$\begin{aligned} f(x; \alpha\boldsymbol{\theta} + (1-\alpha)\boldsymbol{\theta'}) &= f(\boldsymbol{x}; \boldsymbol{R}, \alpha\boldsymbol{U_1} + (1-\alpha)\boldsymbol{U_0}, \boldsymbol{MS}, \boldsymbol{B}, \boldsymbol{M'}) \\ &= \boldsymbol{M'}\sigma(\boldsymbol{MS}\sigma(\boldsymbol{R}\boldsymbol{x} + \alpha\boldsymbol{U_1} + (1-\alpha)\boldsymbol{U_0}) + \boldsymbol{B}) \\ &= g(\boldsymbol{S}\sigma(\boldsymbol{R}\boldsymbol{x} + \alpha\boldsymbol{U_1} + (1-\alpha)\boldsymbol{U_0})) \\ &= g(x_1, \dots, x_D, \alpha) \end{aligned}$$

□

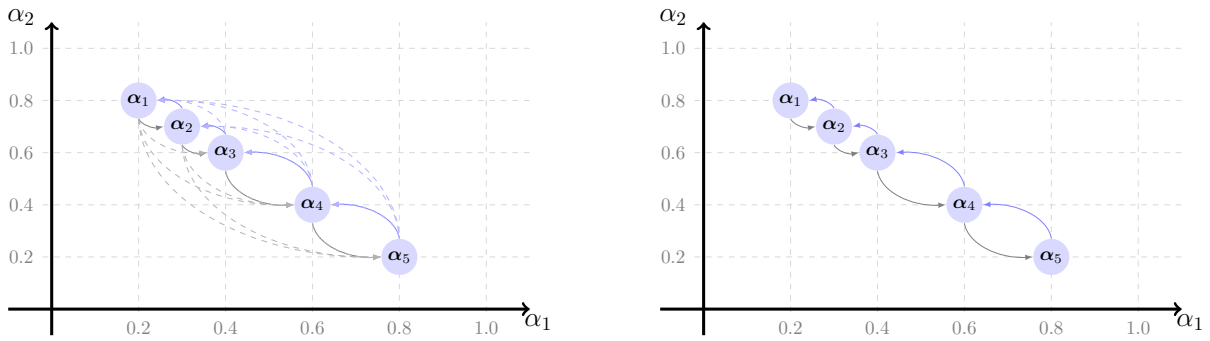


Figure 9: Multi-Forward Graph: case of two tasks. We assume a window of $W = 5$. The nodes lie in the line segment $\alpha_2 + \alpha_1 = 1$, $\alpha_1, \alpha_2 \in [0, 1]$. (Left) Full graph and dashed edges will be removed. (Right) Final graph.

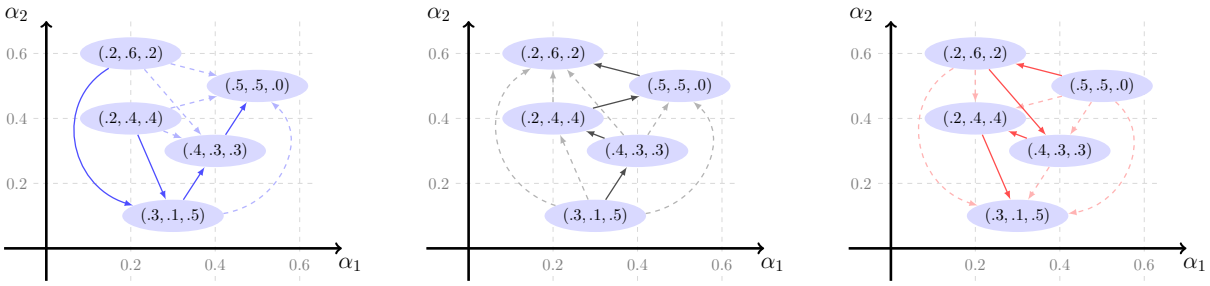


Figure 10: Multi-Forward Graph for three tasks. Left, middle and right present the case of the first, second and third task, respectively. Each node is noted by its weighting, summing up to 1. Edges are drawn if the two nodes obey the total ordering imposed by the task. Dashed edges are omitted from the final graph.

B. Ablation Studies

B.1. Ablation on Multi-Forward Regularization

Multi-Forward regularization, introduced in Section 4.2, penalizes the ensemble if the interpolated models' losses (sampled within a batch) are not in accordance with the tradeoff imposed by the corresponding interpolation weights. Simply put, the closer we sample to the member corresponding to task 1, the lower the loss should be on task 1. The same applies to the other tasks. Figure 3 presents the case of two tasks, where the idea of the regularization is outlined in loss space. For completeness, we present the underlying graph construction for the cases of two and three tasks in Figure 9 and Figure 10, respectively. The nodes of the graphs are associated with the sampled weightings and the edges for the graph $\mathcal{G}_t$ of task t are drawn w.r.t. the corresponding partial ordering. If the loss ordering is violated for a given edge, a penalty term is added.

We ablate the effect multi-forward training and the corresponding regularization have on performance. We explore the `MultiMNIST` and `Census` datasets using the same experimental configurations as in the main text. We are interested in:

- W: number of $\boldsymbol{\alpha}$ re-samplings per batch. This parameter is also referred as *window*.
- λ: the regularization strength as presented in Algorithm 1. For $\lambda = 0$, no regularization is applied but the subspace is still sampled W times and the total loss takes into account all the respective interpolated models.

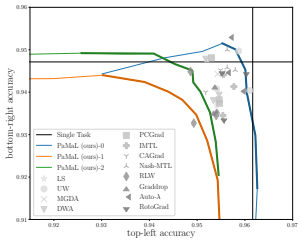
Figure 11 and Table 2 present the results for `MultiMNIST`. Figure 12 and Table 3 present the results for `Census`. It is important to note that `MultiMNIST` is symmetric, while `Census` is not. As a result, the features learned for each single-task predictor are helpful to one another and the case of $\lambda = 0$, i.e., no regularization and only multi-forward training, is beneficial for `MultiMNIST` but not for `Census`. Intuitively, both digit classification tasks have the same difficulty and posterior distribution, which produces few violations of monotonicity constraints and renders the regularization less applicable. On the other hand, severe regularization such as $\lambda = 10$ can be harmful and hinder training. More details in table and figure captions.

Table 2: `MultiMNIST`: Ablation on multi-forward training and regularization, presented in Section 4.2. Validation performance in terms of HyperVolume (HV) metric. Higher is better, except for standard deviation (std). The visual complement of the table appears in Figure 11. For each configuration, we track the Hypervolume across three random seeds and present Mean HV, max HV and standard deviation. We annotate with bold the best per column. In the main text, we report the best result in terms of mean HV, i.e., $W = 4$ and $\lambda = 0$.

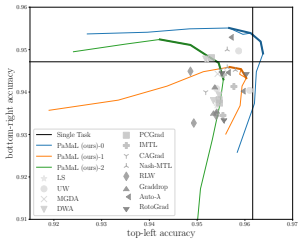
		Seed - 0	Seed - 1	Seed - 2	Mean HV	Max HV	std
$W = 2$	$\lambda = 0$	0.9205	0.9083	0.9100	0.9129	0.9205	0.0054
	$\lambda = 2$	0.9121	0.9105	0.9037	0.9088	0.9121	0.0036
	$\lambda = 5$	0.9132	0.9016	0.8979	0.9043	0.9132	0.0065
	$\lambda = 10$	0.8766	0.8932	0.8470	0.8723	0.8932	0.0191
$W = 3$	$\lambda = 0$	0.9215	0.9141	0.9111	0.9156	0.9215	0.0044
	$\lambda = 2$	0.9176	0.9150	0.9122	0.9149	0.9176	0.0022
	$\lambda = 5$	0.9155	0.9138	0.9140	0.9144	0.9155	**0.0008**
	$\lambda = 10$	0.9122	0.9050	0.8962	0.9045	0.9122	0.0066
$W = 4$	$\lambda = 0$	**0.9220**	0.9187	0.9143	**0.9184**	**0.9220**	0.0032
	$\lambda = 2$	0.9213	0.9149	**0.9157**	0.9173	0.9213	0.0028
	$\lambda = 5$	0.9158	0.9139	0.9132	0.9143	0.9158	0.0011
	$\lambda = 10$	0.9177	0.9022	0.9102	0.9100	0.9177	0.0063
$W = 5$	$\lambda = 0$	0.9131	0.9180	0.9156	0.9156	0.9180	0.0020
	$\lambda = 2$	0.9158	**0.9203**	0.9146	0.9169	0.9203	0.0024
	$\lambda = 5$	0.9138	0.9082	0.9140	0.9120	0.9140	0.0027
	$\lambda = 10$	0.9165	0.9158	0.9121	0.9148	0.9165	0.0019

Table 3: `Census`: Ablation on multi-forward training and regularization, presented in Section 4.2. Validation performance in terms of HyperVolume (HV) metric. Higher is better, except for standard deviation (std). The visual complement of the table appears in Figure 12. For each configuration, we track the Hypervolume across three random seeds and present Mean HV, max HV and standard deviation. We annotate with bold the best per column. In the main text, we report the best result in terms of mean HV, i.e., $W = 2$ and $\lambda = 5$.

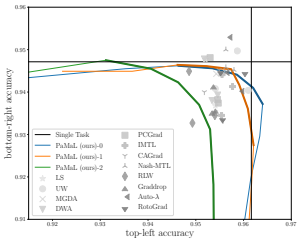
		Seed - 0	Seed - 1	Seed - 2	Mean HV	Max HV	std
$W = 2$	$\lambda = 0$	0.6517	0.6530	0.6532	0.6526	0.6532	0.0006
	$\lambda = 2$	0.6575	0.6564	0.6560	0.6566	0.6575	0.0006
	$\lambda = 5$	**0.6577**	**0.6574**	**0.6590**	**0.6581**	**0.6590**	0.0007
	$\lambda = 10$	0.6548	0.6557	0.6554	0.6553	0.6557	**0.0004**
$W = 3$	$\lambda = 0$	0.6517	0.6496	0.6501	0.6505	0.6517	0.0009
	$\lambda = 2$	0.6540	0.6523	0.6544	0.6536	0.6544	0.0009
	$\lambda = 5$	0.6552	0.6539	0.6536	0.6542	0.6552	0.0007
	$\lambda = 10$	0.6574	0.6567	0.6566	0.6569	0.6574	**0.0004**
$W = 4$	$\lambda = 0$	0.6488	0.6516	0.6504	0.6503	0.6516	0.0011
	$\lambda = 2$	0.6492	0.6522	0.6504	0.6506	0.6522	0.0012
	$\lambda = 5$	0.6499	0.6514	0.6525	0.6513	0.6525	0.0011
	$\lambda = 10$	0.6529	0.6549	0.6558	0.6545	0.6558	0.0012
$W = 5$	$\lambda = 0$	0.6497	0.6502	0.6484	0.6494	0.6502	0.0008
	$\lambda = 2$	0.6478	0.6497	0.6495	0.6490	0.6497	0.0009
	$\lambda = 5$	0.6492	0.6509	0.6489	0.6497	0.6509	0.0009
	$\lambda = 10$	0.6507	0.6538	0.6508	0.6518	0.6538	0.0014



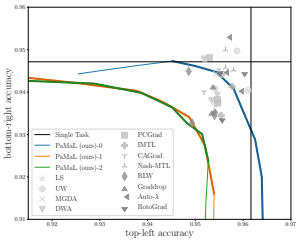
(a) $W = 1$ and $\lambda = 0$



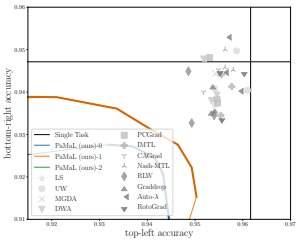
(b) $W = 2$ and $\lambda = 0$



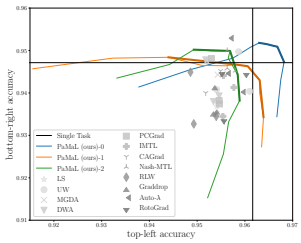
(c) $W = 2$ and $\lambda = 2$



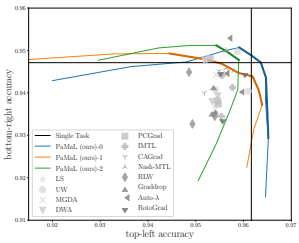
(d) $W = 2$ and $\lambda = 5$



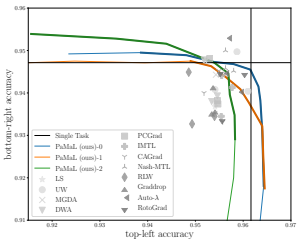
(e) $W = 2$ and $\lambda = 10$



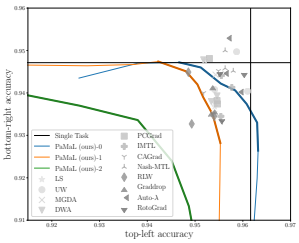
(f) $W = 3$ and $\lambda = 0$



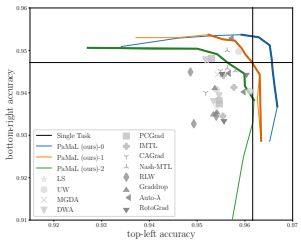
(g) $W = 3$ and $\lambda = 2$



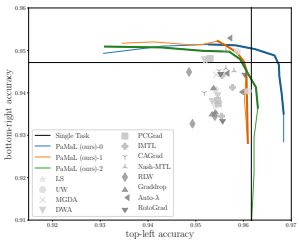
(h) $W = 3$ and $\lambda = 5$



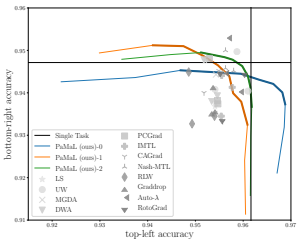
(i) $W = 3$ and $\lambda = 10$



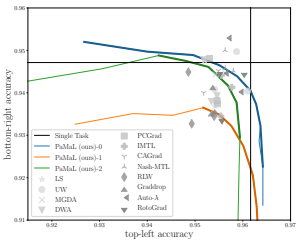
(j) $W = 4$ and $\lambda = 0$



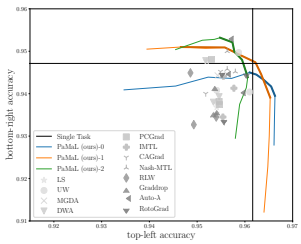
(k) $W = 4$ and $\lambda = 2$



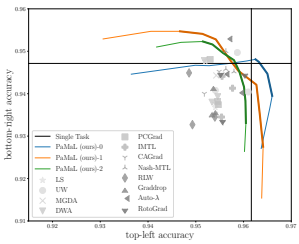
(l) $W = 4$ and $\lambda = 5$



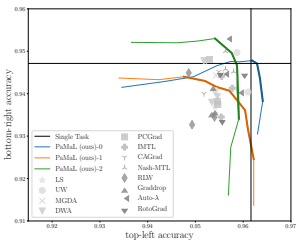
(m) $W = 4$ and $\lambda = 10$



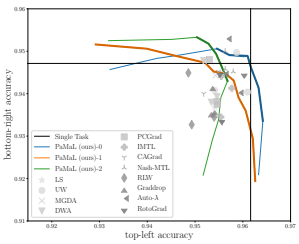
(n) $W = 5$ and $\lambda = 0$



(o) $W = 5$ and $\lambda = 2$

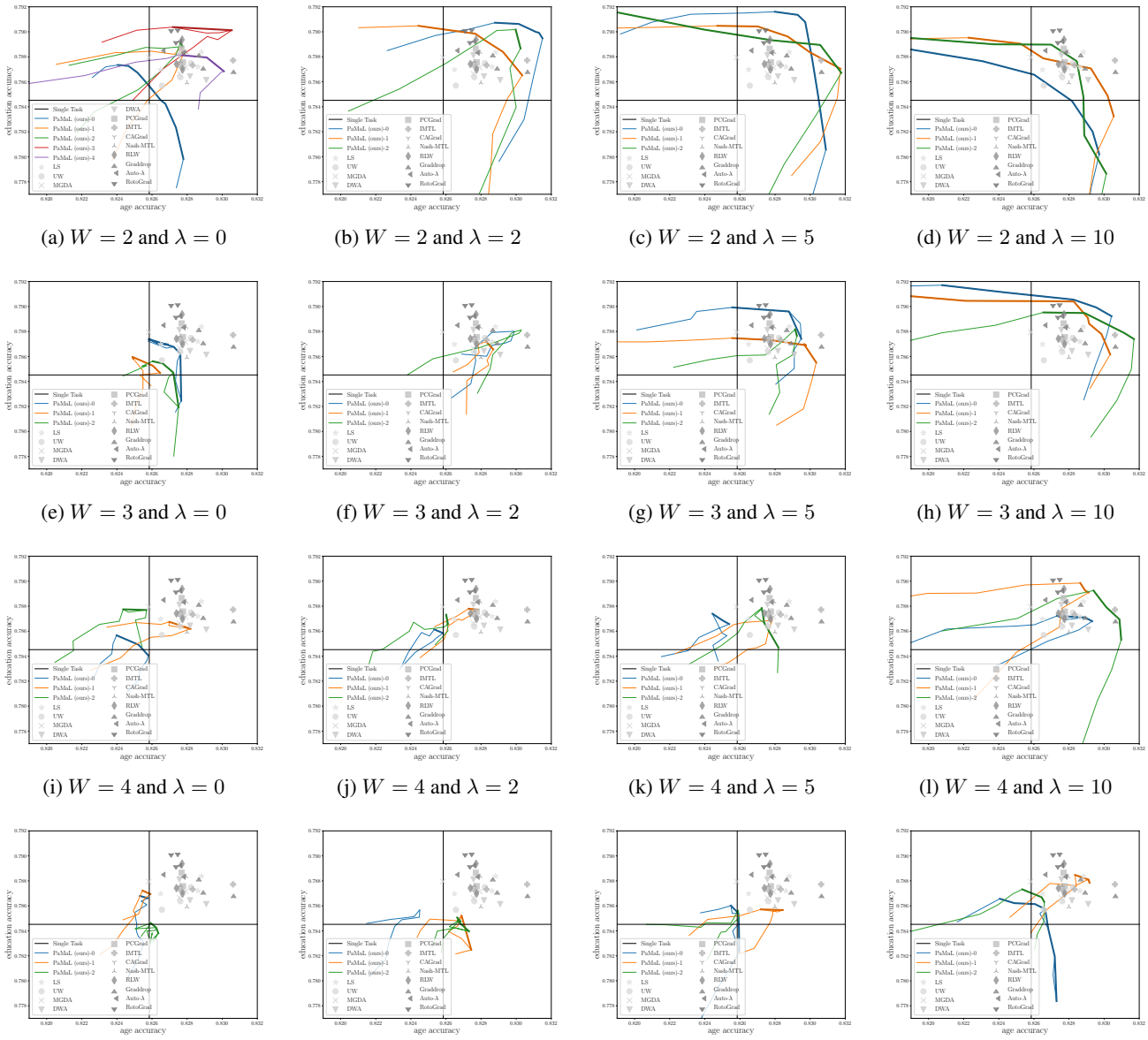


(p) $W = 5$ and $\lambda = 5$



(q) $W = 5$ and $\lambda = 10$

Figure 11: `MultiMNIST`: Effect of multi-forward on the window W and the regularization coefficient λ on the *validation* dataset. The case of no multi-forward ($W = 1$) is presented in the first row. Multi-forward regularization for higher W values is beneficial. Intuitively, attaching serious weight on the regularization $\lambda \in \{5, 10\}$ while sampling few times $W \in \{2, 3\}$ leads to suboptimal performance since the update step focuses on an uninformed regularization term. The accompanying quantitative analysis appears in Table 2.



(a) $W = 2$ and $\lambda = 0$ (b) $W = 2$ and $\lambda = 2$ (c) $W = 2$ and $\lambda = 5$ (d) $W = 2$ and $\lambda = 10$

(e) $W = 3$ and $\lambda = 0$ (f) $W = 3$ and $\lambda = 2$ (g) $W = 3$ and $\lambda = 5$ (h) $W = 3$ and $\lambda = 10$

(i) $W = 4$ and $\lambda = 0$ (j) $W = 4$ and $\lambda = 2$ (k) $W = 4$ and $\lambda = 5$ (l) $W = 4$ and $\lambda = 10$

(m) $W = 5$ and $\lambda = 0$ (n) $W = 5$ and $\lambda = 2$ (o) $W = 5$ and $\lambda = 5$ (p) $W = 5$ and $\lambda = 10$

Figure 12: `Census`: Effect of multiforward on the window W and the regularization coefficient λ. The axes are shared across plots. Compared to `MultiMNIST`, applying multiforward on the *asymmetric* `Census` dataset can improve accuracies and help significantly outperform the baselines. However, widening the window W (e.g., last row for $W = 5$) can be hindering, since larger regularization coefficients are needed. The accompanying quantitative analysis appears in Table 3.

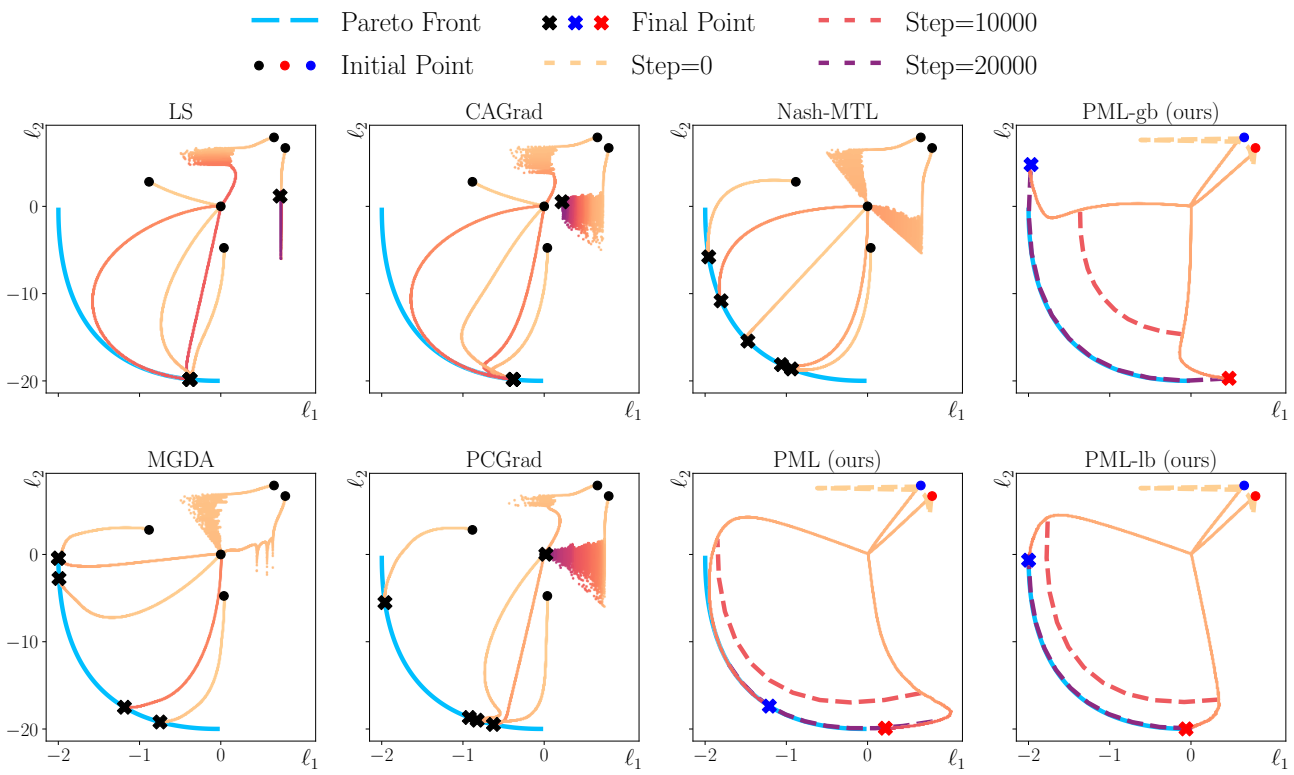


Figure 13: Optimization trajectories in objective space in the case different loss scales. Similar to Figure 1, 5 initializations are shown for baselines and a pair of initializations for Pareto Manifold Learning (PaMaL), in color for clarity. Dashed lines show the evolution of the mapping in loss space for the subspace at the current step. We also show the initial subspace (step= 0). All baselines, except Nash-MTL, and MGDA to a lesser degree, are characterized by trajectories focused on a subset of the Pareto Front, namely minimizing the task with high loss magnitude. The same observation applies to naïvely applying the proposed algorithm PaMaL, because using the same weighting for both the interpolation *and* the losses attaches too much importance on the task with large loss magnitude. However, simple balancing schemes palliate this issue; gradient balancing (PaMaL-gb) discovers a superset of the Pareto Front and loss balancing (PaMaL-lb) discovers the exact Pareto Front.

B.2. Illustrative example: ablation on loss/gradient balancing schemes

The details of the illustrative example are provided in this section. We use the configuration presented by Navon et al. (2022), which was introduced with slight modifications by Liu et al. (2021) and Yu et al. (2020). Specifically, let $\boldsymbol{\theta} = (\theta_1, \theta_2) \in \mathbb{R}^2$ be the parameter vector and $\boldsymbol{L} = (\tilde{\ell}_1, \tilde{\ell}_2)$ be the vector objective defined as follows:

$$\tilde{\ell}_1(\boldsymbol{\theta}) = c_1(\boldsymbol{\theta}) f_1(\boldsymbol{\theta}) + c_2(\boldsymbol{\theta}) g_1(\boldsymbol{\theta}) \quad \text{and} \quad \tilde{\ell}_2(\boldsymbol{\theta}) = c_1(\boldsymbol{\theta}) f_2(\boldsymbol{\theta}) + c_2(\boldsymbol{\theta}) g_2(\boldsymbol{\theta})$$

where

$$\begin{aligned}
f_1(\boldsymbol{\theta}) &= \log\left(\max\left(\left|0.5\left(-\theta_1 - 7\right) - \tanh\left(-\theta_2\right)\right|, 5e-6\right)\right) + 6, \\
f_2(\boldsymbol{\theta}) &= \log\left(\max\left(\left|0.5\left(-\theta_1 + 3\right) - \tanh\left(-\theta_2\right) + 2\right|, 5e-6\right)\right) + 6, \\
g_1(\boldsymbol{\theta}) &= \left(\left(-\theta_1 + 7\right)^2 + 0.1 \cdot \left(-\theta_2 - 8\right)^2\right)/10 - 20, \\
g_2(\boldsymbol{\theta}) &= \left(\left(-\theta_1 - 7\right)^2 + 0.1 \cdot \left(-\theta_2 - 8\right)^2\right)/10 - 20, \\
c_1(\boldsymbol{\theta}) &= \max\left(\tanh\left(0.5\theta_2\right), 0\right) \quad \text{and} \quad c_2(\boldsymbol{\theta}) = \max\left(\tanh\left(-0.5\theta_2\right), 0\right)
\end{aligned}$$

We use the experimental setting outlined by Navon et al. (2022) with minor modifications, i.e., Adam optimizer with a

learning rate of $2e-3$ and training lasts for $50K$ iterations. The overall objectives are $\ell_1 = c \cdot \tilde{\ell}_1$ and $\ell_2 = \tilde{\ell}_2$ where we explore two configurations for the scalar c, namely $c \in \{0.1, 1\}$. For $c = 1$, the two tasks have losses at the same scale. For $c = 0.1$, the difference in loss scales makes the problem more challenging and the algorithm used should be characterized by scale invariance in order to find diverse solutions spanning the entirety of the Pareto Front. The initialization points are drawn from the following set $\{(-8.5, 7.5), (0.0, 0.0), (9.0, 9.0), (-7.5, -0.5), (9, -1.0)\}$. In the case of Pareto Manifold Learning with two ensemble members there are $5^2 = 25$ initialization pairs. In the main text we use the initialization pair with the worst initial objective values.

Figure 13 presents the results for the case of different loss scales, i.e., $c = 0.1$. We plot various baselines and three versions of the proposed algorithm, Pareto Manifold Learning or PaMaL in short. We focus on the effect of the balancing schemes, introduced in Section 4.2, resulting in the use of no balancing scheme (denoted as PaMaL), the use of gradient balancing (denoted as PaMaL-gb) and the use of loss balancing (denoted as PaMaL-lb). We dedicate two figures for each version of the algorithm and we present all 25 initialization pairs for completeness. Figure 14 corresponds to no balancing scheme in the case of equal loss scales $c = 1.0$, i.e., they complement Figure 1 of the main text. The subsequent figures focus on the case of unequal loss scales where $c = 0.1$; Figure 15 corresponds to no balancing scheme, Figure 16 corresponds to the use of gradient balancing, Figure 17 corresponds to the use of loss balancing. The first figures of each pair show the trajectories for each initialization pair, with markers for initial and final positions. The other figures of each pair dispense of the visual clutter and focus on the subspace discovered in the final step of training, which is plotted with dashed lines along with the analytical Pareto Front in solid light blue. Hence, they provide a succinct overview of whether the method was able or not to discover the (entire) Pareto Front.

For $c = 1.0$, the proposed method is able to retrieve the exact Pareto Front with no balancing scheme for most initialization pairs. In three cases (out of 25), the method fails. In our experiments, we found that allowing longer training times or higher learning rates resolve the remaining cases. For $c = 0.1$, the problem is more challenging and the vanilla version of the algorithm results in a subset of the analytical Pareto Front. This subset is consistent across initialization pairs, excluding the ones the method fails, and focuses on the task with higher loss magnitude. Applying gradient balancing, shown in Figure 16, allows the method to retrieve (a superset of) the Pareto Front for all initialization pairs. Similarly, loss balancing, shown in Figure 17, results in the exact Pareto Front. Hence, the inclusion of balancing schemes endows scale invariance in the proposed algorithm. Balancing schemes are used for the more challenging datasets, such as `CityScapes`.

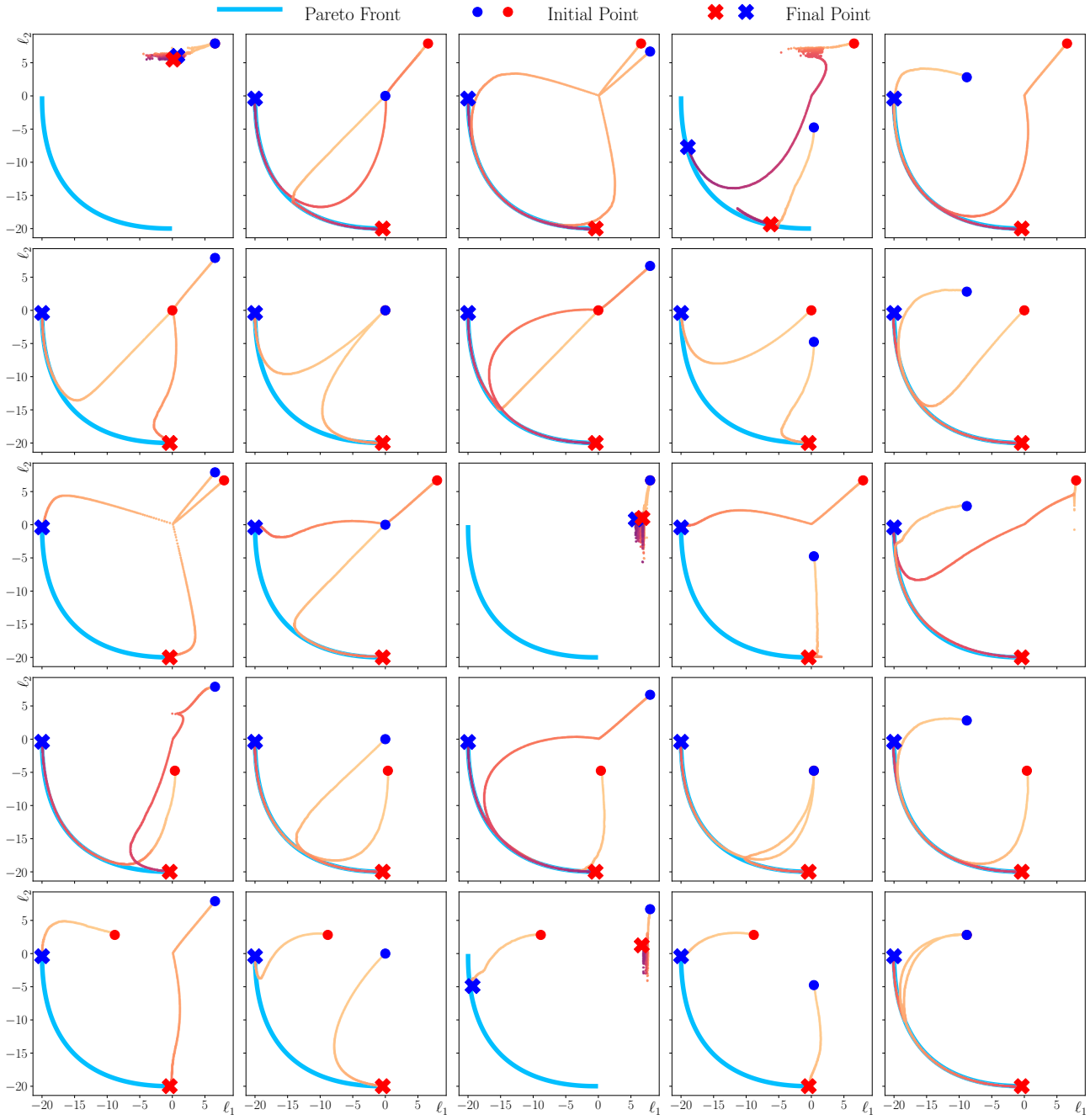


Figure 14: *Illustrative example*. Optimization trajectories in objective space for all initialization pairs in the case of equal loss scales ($c = 1.0$) and application of the proposed method with no balancing scheme. Blue and red markers show each ensemble member's loss value, dots and "X"s correspond to the initial and final step, accordingly. In all but four cases, Pareto Manifold Learning retrieves the entirety of the Pareto Front. Allowing longer training times or higher learning rates solves the remaining initialization pairs.

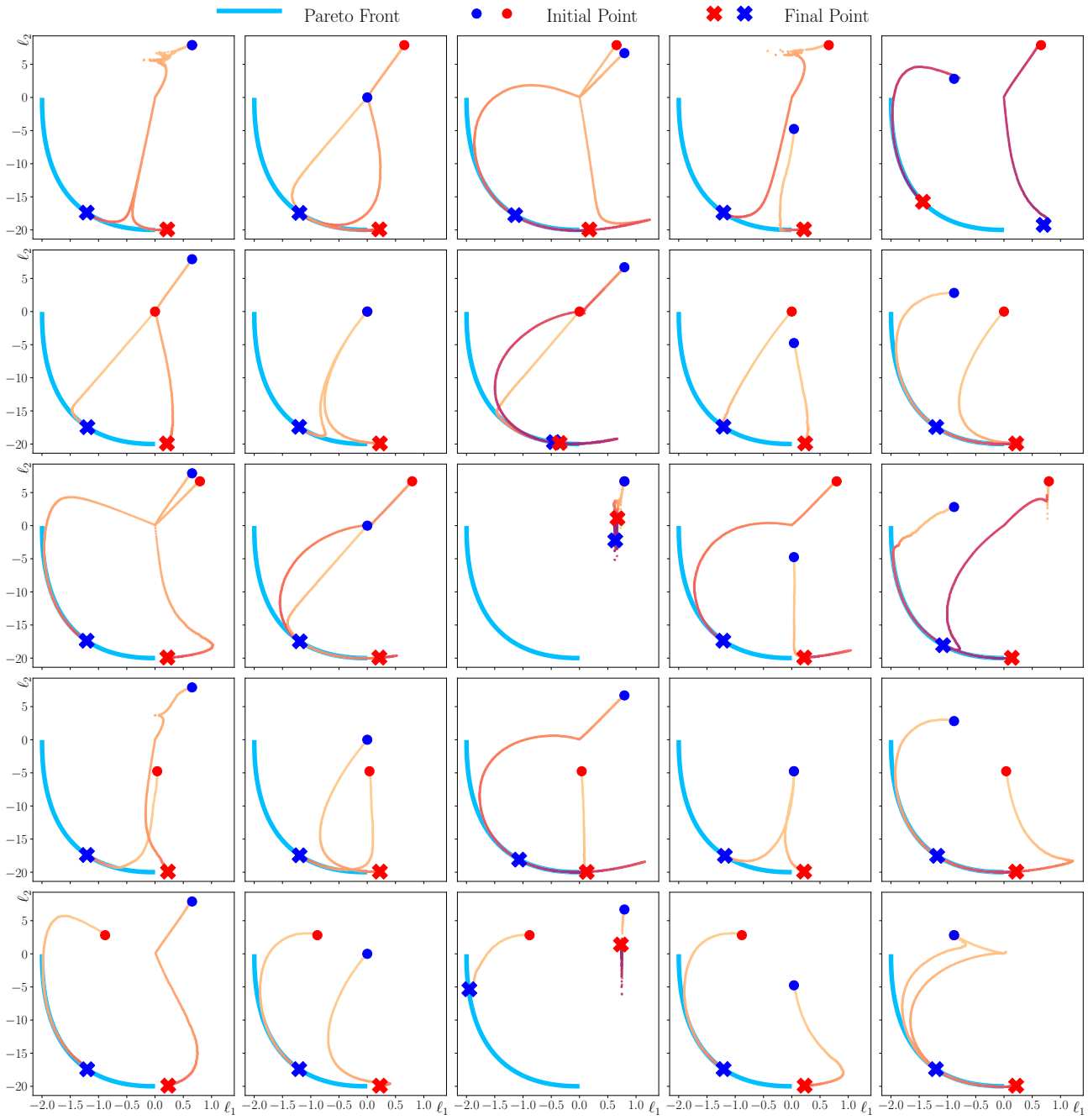


Figure 15: *Illustrative example*. Optimization trajectories in objective space for all initialization pairs in the case of unequal loss scales ($c = 0.1$) and application of the proposed method with no balancing scheme. Blue and red markers show each ensemble member's loss value, dots and "X"s correspond to the initial and final step, accordingly. For the vast majority of initialization pairs, the lack of balancing scheme guides the ensemble to a subset of the Pareto Front, influenced by the task with higher loss magnitude.

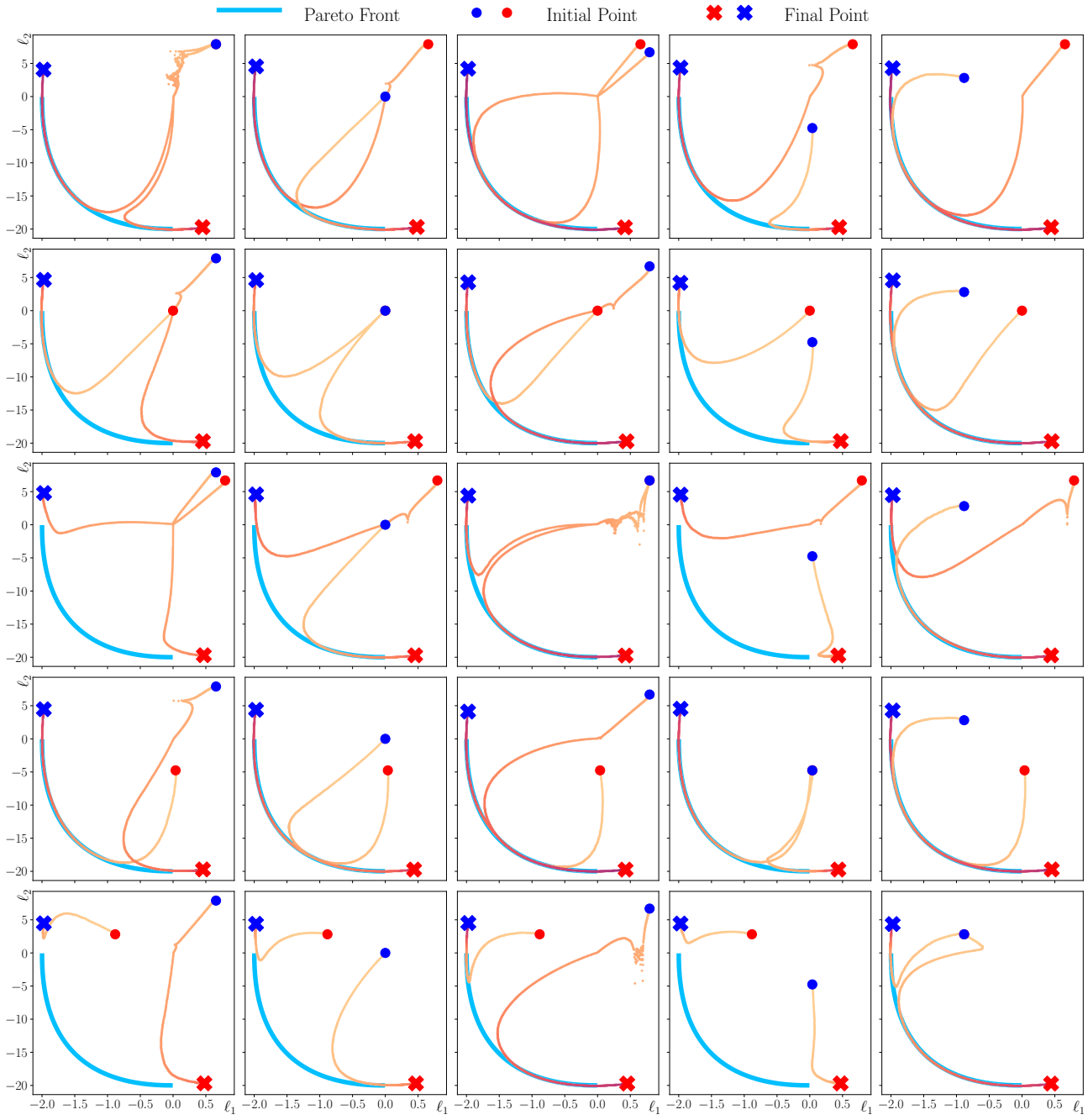


Figure 16: *Illustrative example*. Optimization trajectories in objective space for all initialization pairs in the case of unequal loss scales ($c = 0.1$) and application of the proposed method with gradient balancing scheme. Blue and red markers show each ensemble member's loss value, dots and "X"s correspond to the initial and final step, accordingly. The proposed method discovers a subspace whose mapping in objective space results in a superset of the Pareto Front.

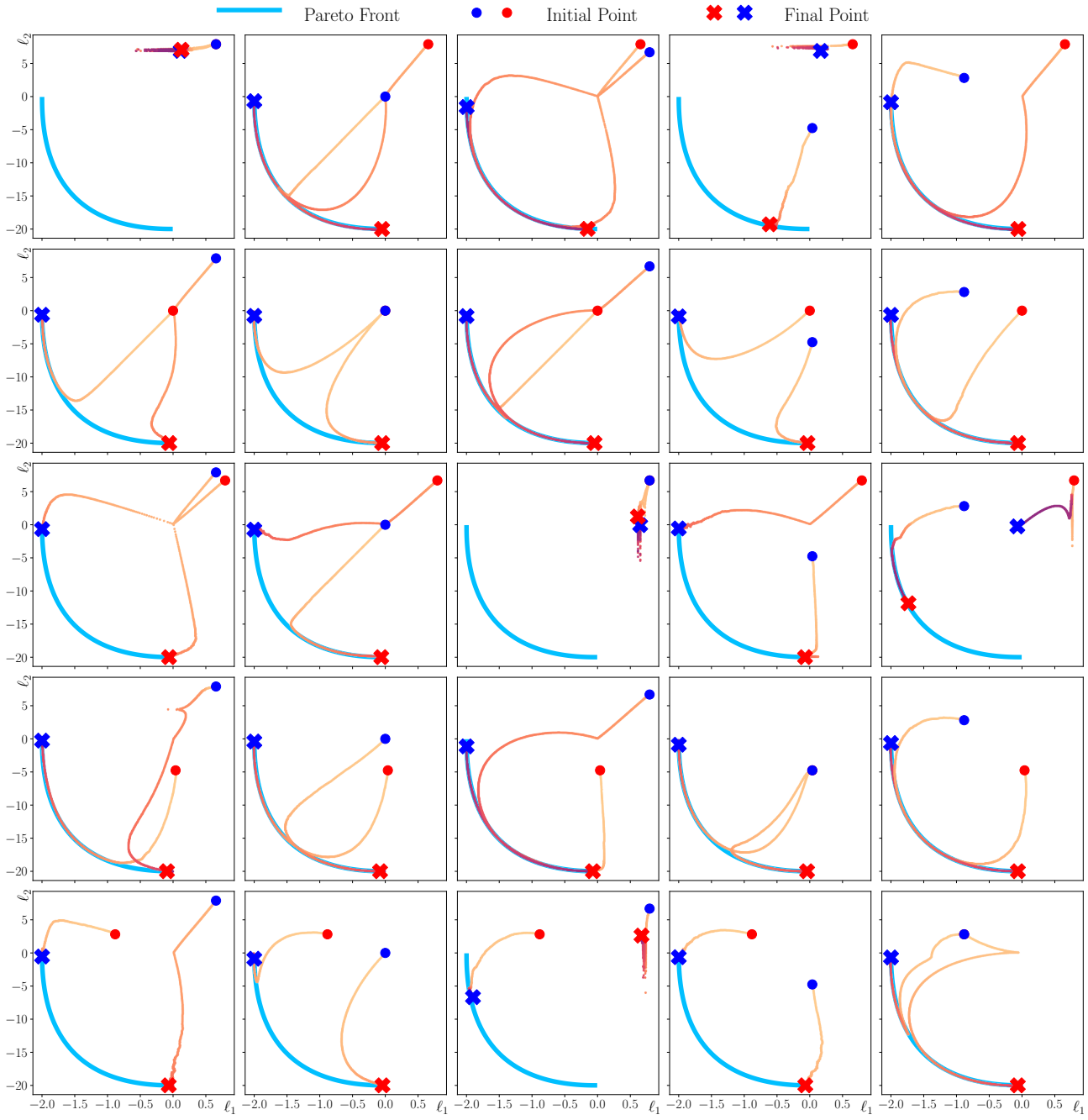


Figure 17: *Illustrative example*. Optimization trajectories in objective space for all initialization pairs in the case of unequal loss scales ($c = 0.1$) and application of the proposed method with loss balancing scheme. Blue and red markers show each ensemble member's loss value, dots and "X"s correspond to the initial and final step, accordingly. For all but five cases, the proposed method discovers a subspace whose mapping in objective space results in the exact Pareto Front.

Table 4: UTKFace: Mean Accuracy and standard deviation of accuracy (over 3 random seeds). For the proposed method (PaMaL), we report the mean and standard deviation of the best performance from the interpolated models in the sampled subspace. No multi-forward training is applied. We present Pareto Manifold Learning with no balancing scheme, with gradient balancing (g) and loss balancing (l).

	Age ↓	Gender ↑	Ethnicity ↑
STL	0.091 ± 0.001	89.80 ± 0.38	81.23 ± 0.22
LS	0.100 ± 0.008	90.43 ± 0.76	80.49 ± 1.64
UW	0.092 ± 0.003	$\mathbf{91.39 \pm 0.08}$	$\mathbf{81.63 \pm 0.22}$
MGDA	0.091 ± 0.006	90.71 ± 0.22	77.29 ± 0.44
PCGrad	0.102 ± 0.008	90.36 ± 1.56	79.96 ± 2.94
IMTL	0.110 ± 0.029	91.16 ± 0.19	80.47 ± 0.96
Graddrop	0.140 ± 0.059	89.43 ± 2.59	77.59 ± 5.75
CAGrad	0.089 ± 0.001	90.84 ± 0.38	81.28 ± 0.53
RLW	0.097 ± 0.002	90.81 ± 0.12	81.50 ± 0.19
Nash-MTL	0.106 ± 0.019	90.36 ± 0.60	78.98 ± 2.14
Auto-λ	0.091 ± 0.003	90.84 ± 0.35	81.58 ± 0.06
COSMOS	0.107 ± 0.003	89.68 ± 0.40	79.39 ± 0.59
PHN	0.106 ± 0.001	90.49 ± 0.34	79.99 ± 0.23
PAMAL-g(W=1, p=1)	0.094 ± 0.001	90.65 ± 0.20	80.03 ± 0.27
PAMAL-l(W=3, p=2)	0.099 ± 0.003	90.62 ± 0.27	80.82 ± 0.97
PAMAL-g(W=3, p=2)	$\mathbf{0.083 \pm 0.001}$	90.93 ± 0.25	80.78 ± 0.29

B.3. UTKFace: ablation on the effect of loss/gradient balancing schemes

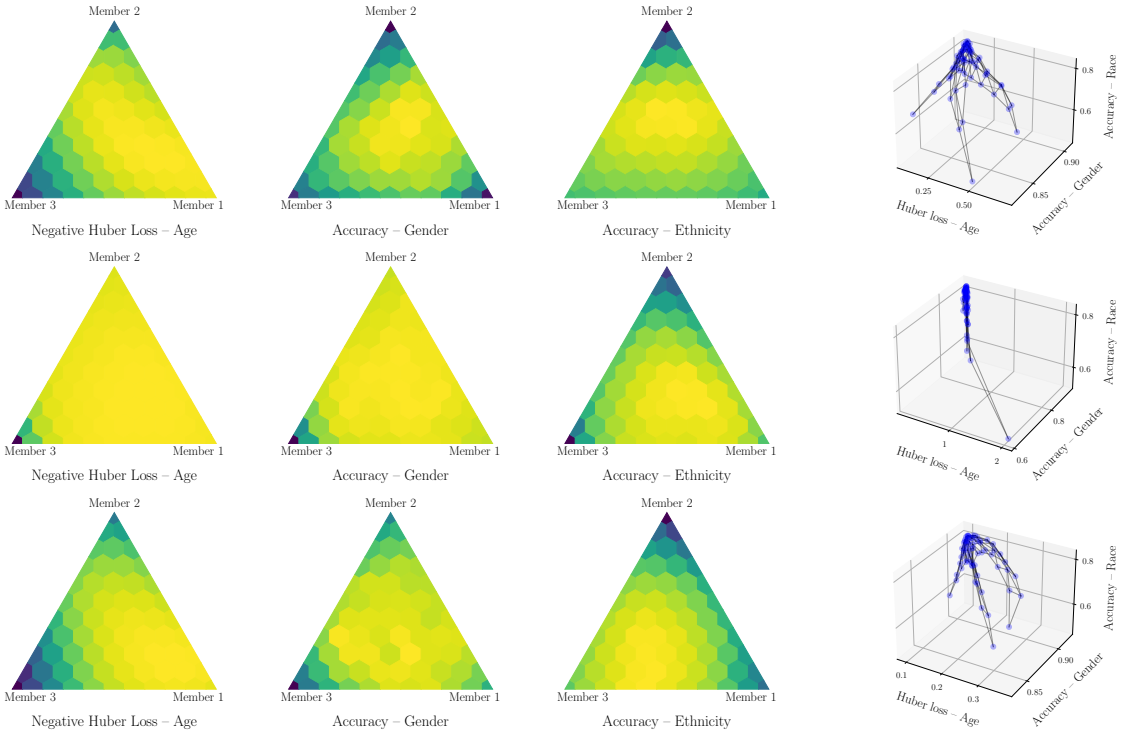
This section serves as supplementary to Section 5.2. Table 4 compares the performance of the baselines and the proposed method. We experiment without balancing schemes and with gradient-balancing, and present the results in Figure 18. Together with the quantitative results, we observe that for datasets with varying task difficulties, scales, etc. the lack of balancing can be impeding. On the other hand, its inclusion makes the subspace functionally diverse and boosts overall performance. For instance, Huber loss on the task of age prediction is significantly improved.

B.4. Hyperparameter optimization for PHN and COSMOS

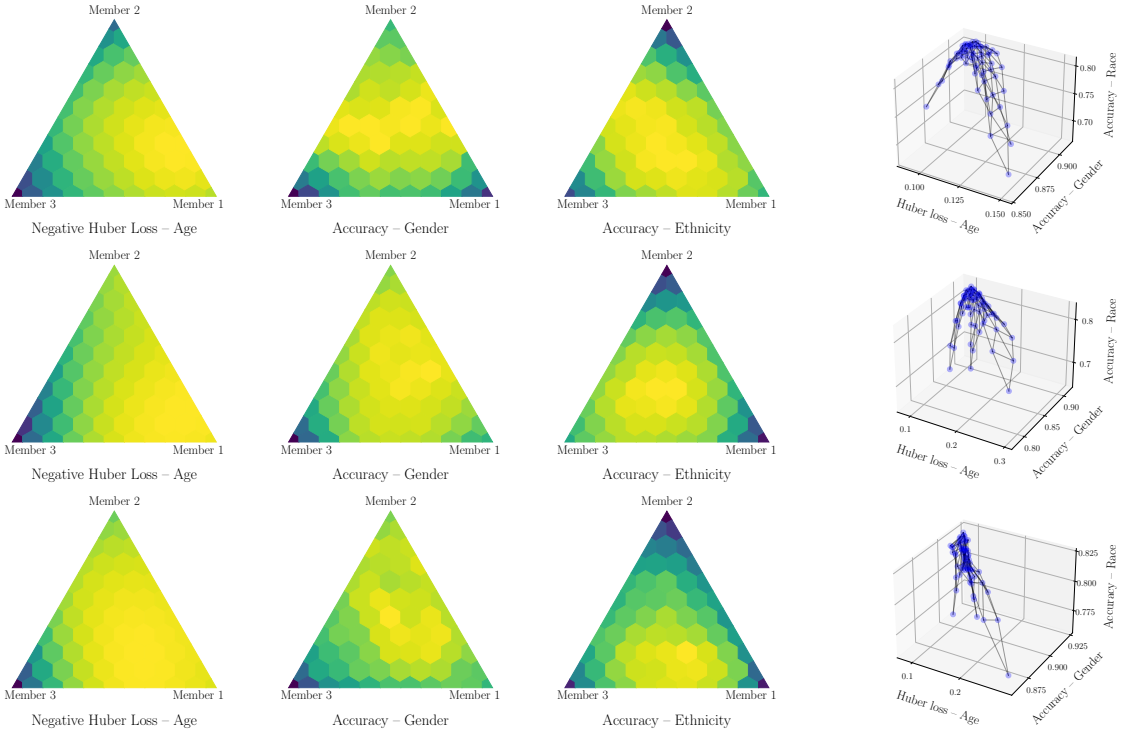
PHN - Pareto HyperNetwork (Navon et al., 2021) The method has one hyperparameter: p for the sampling of the Dirichlet distribution and two solvers: Linear and EPO (Mahapatra & Rajan, 2020). The method also requires the definition of the architecture of the HyperNetwork. Following the authors' implementation we use a MLP with 2 hidden layers of $w = 100$ dimensions each. The output layer of the MLP has as many neurons as the target network, e.g. a LeNet for MultiMNIST experiments or a ResNet18 for UTKFace. Essentially, this means that the HyperNetwork requires at least w times the number of the target network parameters. The models for UTKFace and CityScapes have approximately 11M and 17M parameters, leading to the HyperNetwork to have 1.1B and 1.7B parameters. Hence, due to computational reasons, the PHN baseline for these benchmarks applies chunking (Ha et al., 2017). For CityScapes we were unable to retrieve good results for PHN using chunking and, hence we omit it from the main text and from additional experiments in the appendix.

For the concentration parameter p we ablate over the values $\{0.001, 0.1, 0.2(\text{used in the original paper}), 0.5, 1, 2\}$ and over the two solvers, i.e., Linear and EPO. For MultiMNIST, the original paper used 100 epochs and complex schedulers with a lower learning rate of 10^{-4}. We limit the training budget to 10 epochs for all baselines and omit scheduling for MultiMNIST. However, we also consider the lower learning rate in the ablation. The full hyperparameter sweep is presented in Table 7 for MultiMNIST and Table 6 for Census.

For each hyperparameter configuration, three seeds are considered and the best configuration is selected by the best average (across seeds) HyperVolume score that also satisfies the criterion that the average (across seeds) Spearman correlation over the two task performances is lower than a threshold. Consider the case of two classification tasks, where performance is gauged by validation accuracy. Then, a solution is examined over the different tradeoffs $\mathcal{A} = \{(\alpha, 1-\alpha) : \alpha \in [0, 1]\}$, and, performance of one task should monotonically increase as α is increased and vice versa for the other task. In this optimal scenario, the Spearman correlation would be -1. The reason for this additional criterion is that the optimal hyperparameter configurations produced degenerate solutions, as explained visually in Figure 19. The aforementioned ablation applies to



(a) Linear Scalarization



(b) Gradient Balancing

Figure 18: `UTKFace` results with Gradient-Balancing Scheme for all three seeds. Each triangle shows the 66 points in the convex hull and color is used for the performance on the associated task. The 3d plot shows the mapping of the subspace to the multi-objective space. For datasets with tasks of varying loss scales, applying gradient balancing improves functional diversity and performance, as shown in Table 4.

Table 5: Hyperparameter search for COSMOS on `MultiMNIST`. Results refer to the validation set. Higher HyperVolume is better. Lower Spearman correlation is better. The accuracy columns refer to the maximum accuracy sampled by conditioning the network with ten different user preferences.

α	λ	lr	acc top-left	acc bottom-right	HV	Spearman
0.1	1.0	0.001	0.9440	0.9249	0.8731	-0.9838
	2.0	0.001	0.9406	0.9219	0.8666	-0.9960
	5.0	0.001	0.9400	0.9236	0.8667	-0.9919
	8.0	0.001	0.9402	0.9219	0.8647	-0.9950
1.0	1.0	0.001	0.9339	0.9194	0.8587	-0.8121
	2.0	0.001	0.9294	0.9173	0.8515	-0.8061
	5.0	0.001	0.9279	0.9055	0.8380	-0.8828
	8.0	0.001	0.9293	0.9059	0.8395	-0.9636
2.0	1.0	0.001	0.9422	0.9277	0.8741	-0.1328
	2.0	0.001	0.9304	0.9179	0.8530	-0.8424
	5.0	0.001	0.9285	0.9159	0.8487	-0.9434
	8.0	0.001	0.9240	0.9120	0.8412	-0.9030
5.0	1.0	0.001	0.9378	0.9326	0.8746	0.0866
	2.0	0.001	0.9280	0.9209	0.8544	-0.9279
	5.0	0.001	0.9230	0.9162	0.8443	-0.8797
	8.0	0.001	0.9242	0.9028	0.8338	-0.7078

`MultiMNIST` and `Census`.

COSMOS - Conditioned One-shot Multi-Objective Search (Ruchte & Grabocka, 2021) The method has two hyperparameters: p for the sampling of the Dirichlet distribution and λ as the coefficient for the proposed regularization. We use the PHN search space for α, consider $\lambda \in \{0.1, 1, 2, 5, 8\}$ and learning rate $\in \{10^{-3}, 10^{-4}\}$. The full hyperparameter sweep is presented in Table 5 for `MultiMNIST` and Table 8 for `Census`.

Table 6: Hyperparameter search for PHN on `Census`. Results refer to the validation set. Higher HyperVolume is better. Lower Spearman correlation is better. The accuracy columns refer to the maximum accuracy sampled by conditioning the network with ten different user preferences.

α	solver	lr	acc age	acc education	HV	Spearman
0.001	EPO	1e-4	0.8277	0.7883	0.6524	0.2294
		1e-3	0.8250	0.7869	0.6492	0.9753
	linear	1e-4	0.8283	0.7883	0.6530	0.0226
		1e-3	0.8271	0.7861	0.6502	0.0000
0.010	EPO	1e-4	0.8264	0.7860	0.6495	0.1478
	linear	1e-4	0.8274	0.7879	0.6519	-0.7723
		1e-3	0.8268	0.7874	0.6510	-0.5617
0.100	EPO	1e-4	0.8277	0.7898	0.6537	0.1538
	linear	1e-4	0.8272	0.7895	0.6530	0.2244
		1e-3	0.8274	0.7883	0.6523	0.1743
0.200	EPO	1e-4	0.8292	0.7840	0.6501	-0.1337
	linear	1e-4	0.8282	0.7833	0.6487	0.1567
0.500	EPO	1e-4	0.8287	0.7867	0.6520	-0.0834
		1e-3	0.8274	0.7856	0.6501	-0.1058
	linear	1e-4	0.8270	0.7854	0.6495	0.1374
1.000	EPO	1e-4	0.8291	0.7877	0.6531	-0.0442
		1e-3	0.8270	0.7848	0.6490	-0.3510
	linear	1e-4	0.8279	0.7870	0.6515	-0.0356
2.000	EPO	1e-4	0.8297	0.7876	0.6535	-0.5062
	linear	1e-4	0.8266	0.7872	0.6507	0.6878

Table 7: Hyperparameter search for PHN on `MultiMNIST`. Results refer to the validation set. Higher HyperVolume is better. Lower Spearman correlation is better. The accuracy columns refer to the maximum accuracy sampled by conditioning the network with ten different user preferences.

α	λ	lr	acc top-left	acc bottom-right	HV	Spearman
0.001	EPO	0.001	0.9570	0.9290	0.8890	-0.9464
	linear	0.001	0.9555	0.9440	0.9019	-0.9838
0.010	EPO	0.001	0.9615	0.8874	0.8532	-0.2705
	linear	0.001	0.9587	0.9381	0.8978	-1.0000
0.100	EPO	0.001	0.9446	0.9257	0.8743	-0.9636
	linear	0.001	0.9547	0.9406	0.8978	-0.9960
0.200	EPO	0.001	0.9502	0.9411	0.8941	-0.9394
	linear	0.001	0.9468	0.9256	0.8763	-0.9919
0.500	EPO	0.001	0.9552	0.9436	0.9012	-0.9818
	linear	0.001	0.9585	0.9495	0.9101	-0.7858
1.000	EPO	0.001	0.9500	0.9369	0.8900	-0.8858
	linear	0.001	0.9579	0.9429	0.9032	-0.0016
2.000	EPO	0.001	0.9548	0.9438	0.9011	-0.7765
	linear	0.001	0.9570	0.9384	0.8981	-0.6209

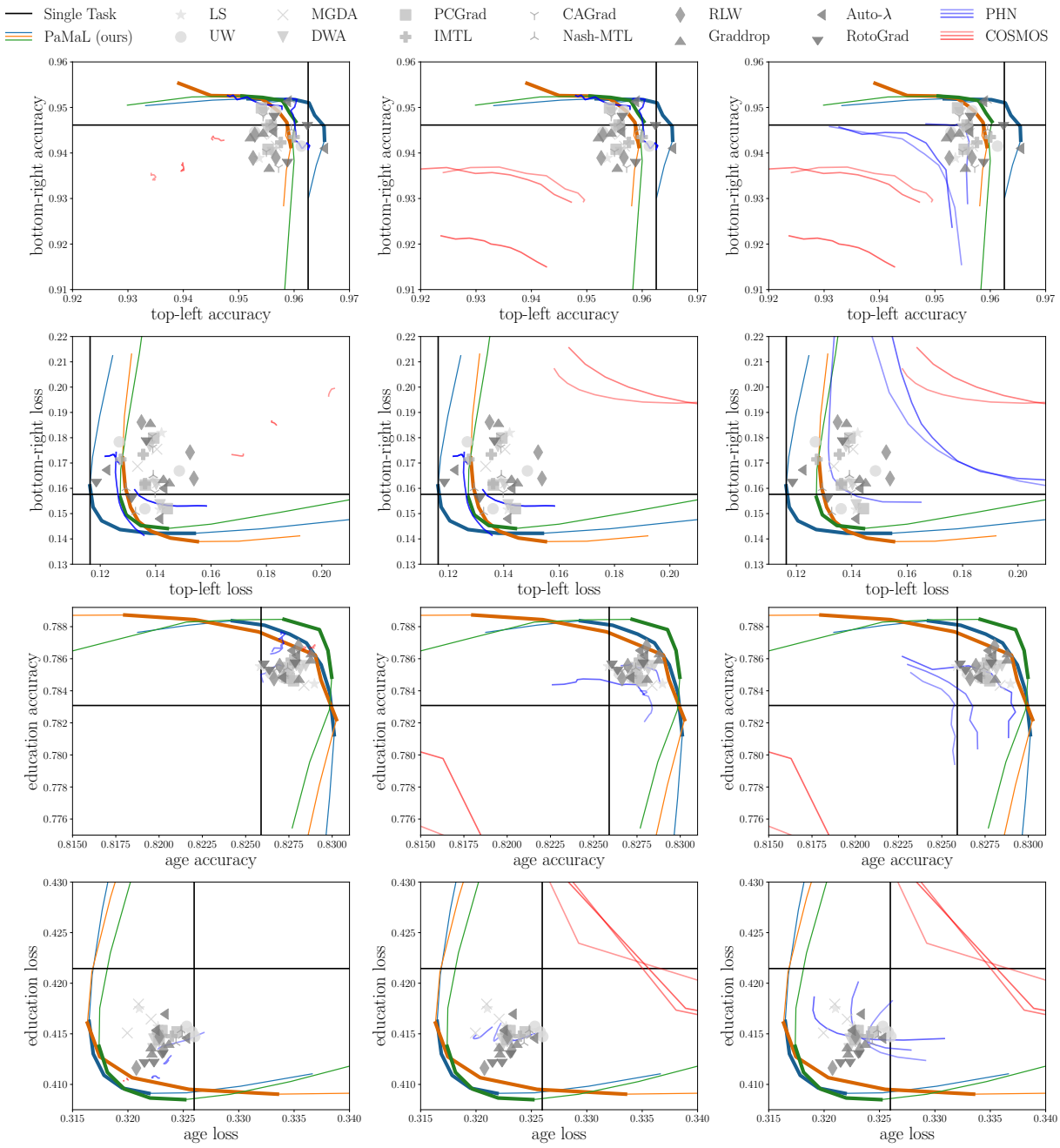


Figure 19: Effect of the Spearman correlation (S) as a secondary criterion on the PHN and COSMOS baselines. The other methods are fixed per row. Zoom for details. (Left) no additional constraint, (Middle) $S < -0.5$, (Right) $S < -0.6$. (First row) `MultiMNIST` accuracy, (Second row) `MultiMNIST` loss, (Third row) `Census` accuracy, (Fourth row) `Census` loss. The exclusion of the Spearman threshold criterion leads to degenerate solutions (left column) that are not in the spirit of the original method; either a single point lies in the Pareto Front or the mapping from desired trade-off to network weights is obfuscated. Additionally, loss curves are generally "smoother" than accuracy curves, attesting to the generalization gap.

Table 8: Hyperparameter search for COSMOS on `Census`. Results refer to the validation set. Higher HyperVolume is better. Lower Spearman correlation is better. The accuracy columns refer to the maximum accuracy sampled by conditioning the network with ten different user preferences.

		lr=1e-3				lr=1e-4			
α	λ	acc age	acc education	HV	Spearman	acc age	acc education	HV	Spearman
0.01	0.1	0.8284	0.7902	0.6546	0.3204	0.8281	0.7891	0.6535	0.5075
	1.0	0.8258	0.7872	0.6500	-0.2898	0.8254	0.7892	0.6514	0.4072
	2.0	0.8118	0.7874	0.6370	-0.9960	0.8190	0.7884	0.6447	-0.9919
	5.0	0.8105	0.7871	0.6298	-1.0000	0.8075	0.7878	0.6301	-1.0000
0.10	0.1	0.8287	0.7892	0.6540	-0.0411	0.8285	0.7887	0.6534	0.0948
	1.0	0.8257	0.7891	0.6516	-0.9667	0.8254	0.7894	0.6515	0.0287
	2.0	0.8196	0.7883	0.6448	-0.9960	0.8181	0.7883	0.6441	-0.9677
	5.0	0.8201	0.7892	0.6405	-0.9838	0.8083	0.7879	0.6314	-0.9960
0.20	0.1	0.8286	0.7887	0.6535	-0.0451	0.8284	0.7889	0.6536	-0.0185
	1.0	0.8276	0.7886	0.6526	-0.5289	0.8267	0.7895	0.6527	-0.0051
	2.0	0.8238	0.7902	0.6502	-0.9677	0.8206	0.7881	0.6461	-0.9475
	5.0	0.8247	0.7907	0.6492	-0.9394	0.8137	0.7876	0.6356	-0.9950
0.50	0.1	0.8291	0.7880	0.6533	0.3141	0.8293	0.7892	0.6545	0.3629
	1.0	0.8283	0.7877	0.6524	-0.2757	0.8267	0.7897	0.6528	-0.4077
	2.0	0.8255	0.7892	0.6508	-0.9596	0.8223	0.7882	0.6476	-0.9717
	5.0	0.8277	0.7900	0.6505	-0.9556	0.8180	0.7879	0.6387	-0.9798
1.00	0.1	0.8285	0.7891	0.6537	0.5783	0.8293	0.7891	0.6544	0.3761
	1.0	0.8279	0.7878	0.6522	-0.6916	0.8272	0.7897	0.6532	-0.1814
	2.0	0.8290	0.7886	0.6537	-0.3253	0.8239	0.7885	0.6496	-0.3459
	5.0	0.8283	0.7898	0.6533	-0.8384	0.8218	0.7874	0.6422	-0.8909
2.00	0.1	0.8280	0.7888	0.6532	-0.2172	0.8294	0.7896	0.6549	-0.0839
	1.0	0.8290	0.7875	0.6528	-0.8392	0.8271	0.7900	0.6534	-0.3673
	2.0	0.8276	0.7891	0.6530	-0.8489	0.8248	0.7888	0.6506	0.3466
	5.0	0.8286	0.7896	0.6531	-0.7423	0.8234	0.7870	0.6441	-0.8061

C. Additional Experiments

C.1. Details on experimental configurations

MultiMNIST MultiMNIST is a synthetic dataset derived form the samples of MNIST. Since there is no publicly available version, we create our own by the following procedure. For each MultiMNIST image, we sample (with replacement) two MNIST images (of size 28×28) and place them top-left and bottom-right on a 36×36 grid. This grid is then resized to 28×28 pixels. The procedure is repeated 60000 times, 10000 and 10000 times for training, validation and test datasets. We use a LeNet shared-bottom architecture. Specifically, the encoder has two convolutional layers with 10 and 20 channels and kernel size of 5 followed by Maxpool and a ReLU nonlinearity each. The final layer of the encoder is fully connected producing an embedding with 50 features. The decoders are fully connected with two layers, one with 50 features and the output layer has 10. We use Adam optimizer (Kingma & Ba, 2015) with learning rate 10^{-3}, no scheduler and the batch size is set to 256. Training lasts 10 epochs.

Census The original version of the Census (Kohavi, 1996) dataset has one task: predicting whether a person's income exceeds \$50000. The dataset becomes suitable for Multi-Task Learning by turning one or several features to tasks (Lin et al., 2019). We focus on the task combination of predicting age and education level, similar to Ma et al. (2020). The model has a Multi-Layer Perceptron shared-bottom architecture. The encoder has one layer with 256 neurons, followed by a ReLU nonlinearity, and two decoders with 2 output neurons each (since the tasks are binary classification). Training lasts 10 epochs. We use Adam optimizer learning rate of 10^{-3}.

MultiMNIST-3 The configuration of MultiMNIST is used. The model has three decoders. Training lasts 20 epochs.

UTKFace The UTKFace dataset has more than 20,000 face images of dimensions 200×200 pixels and 3 color channels. The dataset has three tasks: predicting age (modeled as regression using Huber loss - similar to (Ma et al., 2020)), classifying gender and ethnicity (modeled as classification tasks using Cross-Entropy loss). Images are resized to 64×64 pixels, age is normalized and a 80/20 train/test split is used. We use a shared-bottom architecture; the encoder is a ResNet18 (He et al., 2016) model without the last fully connected layer. The decoders (task-specific layers) consist of one fully-connected layer, where the output dimensions are 1, 2 and 5 for age (modeled as regression), gender (binary classification) and ethnicity (classification with 5 classes). Training lasts 100 epochs, batch size is 256 and we use Adam optimizer with a learning rate of 10^{-3}. No scheduler is used.

CityScapes Our experimental configuration is very similar to prior work, namely (Liu et al., 2019; Yu et al., 2020; Liu et al., 2021; Navon et al., 2022). All images are resized to 128×256. The tasks used are coarse semantic segmentation and depth regression. The task of semantic segmentation has 7 classes, whereas the original has 19. We use a SegNet architecture (Badrinarayanan et al., 2017) and train the model for 100 epochs with Adam optimizer (Kingma & Ba, 2015) of an initial learning rate 10^{-4}. We employ a scheduler that halves the learning rate after 75 epochs.

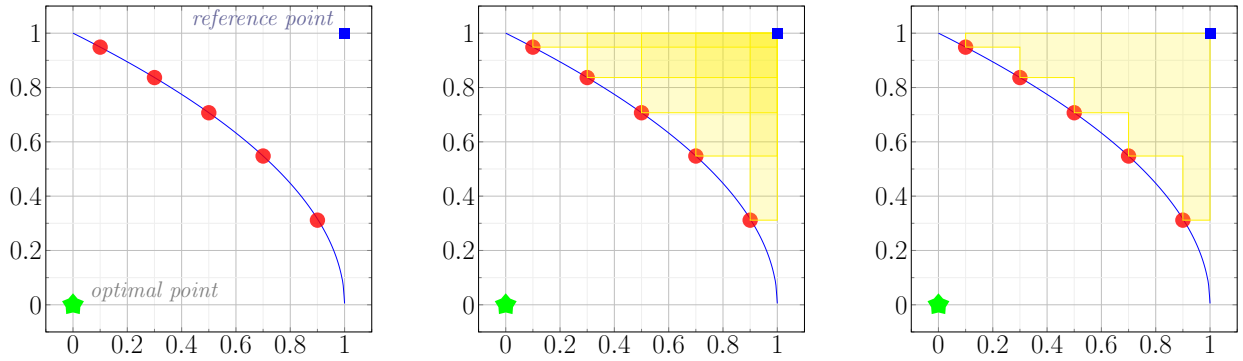


Figure 20: Visual Explanation of Hypervolume. The metric captures the union of axis-aligned rectangles defined by the reference point (star) and the corresponding sample points (red circles). This example showcases loss and the perfect oracle lies in the origin. The point $(1, 1)$ is used for reference. Hence, higher hypervolume implies that the objective space is better explored/covered.

C.2. HyperVolume analysis on `MultiMNIST` and `Census`

HyperVolume is a metric widely used in multi-objective optimization that captures the quality of exploration. A visual explanation of the metric is given in Figure 20. Table 9 presents the results of Figure 4 of the main text in a tabular form. We present the best three results per column (higher is better) to succinctly and visually show that all Pareto Manifold Learning seeds outperform the baselines.

Table 9: Tabular complement to Figure 4. Classification accuracy for both tasks and HyperVolume (HV) metric (higher is better). Three random seeds per method. For baselines, we show the mean accuracy and HV (across seeds). For PaMaL, we show the results per seed; HV and max accuracies for the subspace yielded by that seed. We use underlined bold, solely bold and solely underlined font for the best, second best and third best results. We observe that the best results are concentrated in the rows concerning the proposed method (PaMaL). Note that the use of three decimals leads to ties.

	MultiMNIST			Census		
	Task 1	Task 2	HV	Task 1	Task 2	HV
LS	0.955	0.944	0.907	0.827	0.785	0.651
UW	0.957	0.945	0.913	0.827	0.785	0.65
MGDA	0.956	0.943	0.904	0.828	0.785	0.651
DWA	0.955	0.945	0.907	0.828	0.785	0.651
PCGrad	0.955	0.946	0.908	0.828	0.785	0.65
IMTL	0.958	0.944	0.908	0.828	0.786	0.651
Nash-MTL	0.958	0.948	0.913	0.827	0.785	0.65
RLW	0.954	0.941	0.903	0.827	0.786	0.651
Graddrop	0.954	0.942	0.903	0.829	0.786	0.652
Auto-λ	0.959	0.946	0.918	0.827	0.786	0.651
RotoGrad	0.959	0.945	0.913	0.827	0.786	0.651
PML - 0	**0.968**	0.951	**0.92**	**0.83**	**0.789**	**0.655**
PML - 1	0.961	**0.953**	0.916	**0.83**	**0.789**	**0.655**
PML - 2	**0.964**	**0.953**	**0.919**	0.829	0.788	0.653

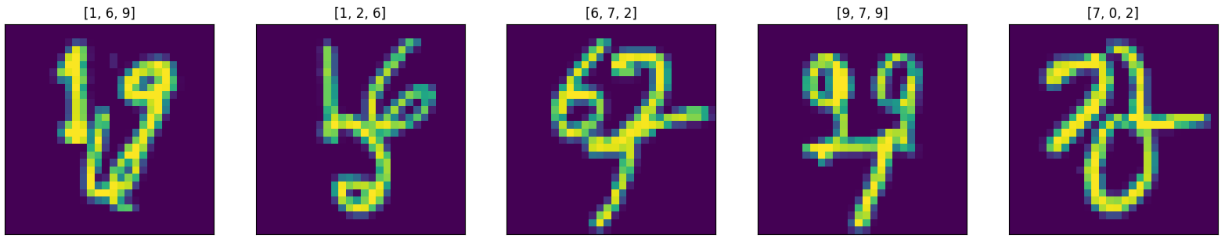


Figure 21: Examples of samples and corresponding labels for the MultiMNIST-3 dataset.

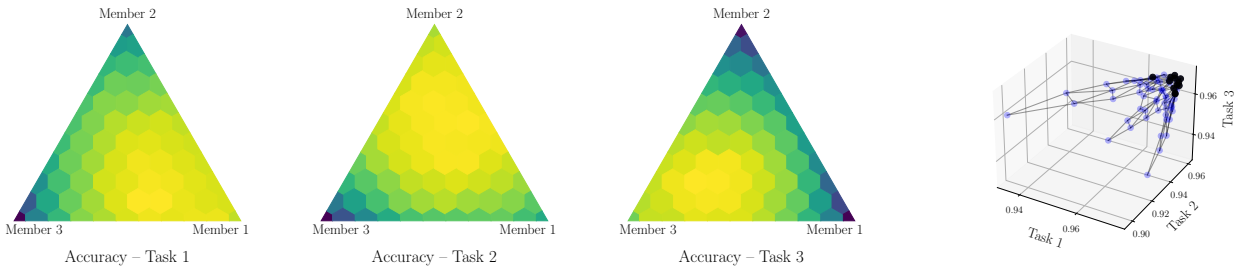
C.3. MultiMNIST-3 quantitative results

This section serves as supplementary to Section 5.2 of the main text. MultiMNIST-3 is a synthetic dataset generated by MNIST samples in a manner similar to the creation of the MultiMNIST dataset, which is ubiquitous in the Multi-Task Learning literature. Specifically, each MultiMNIST-3 sample is created with the following procedure. Three randomly sampled digits of size 28×28 are placed in the top-left, top-right and bottom middle pixels of a 42×42 grid. For the pixels where the initial digits overlap, the maximum value is selected. Finally, the image is resized to 28×28 pixels. Figure 21 shows some examples of the dataset, which consists of three digit classification tasks.

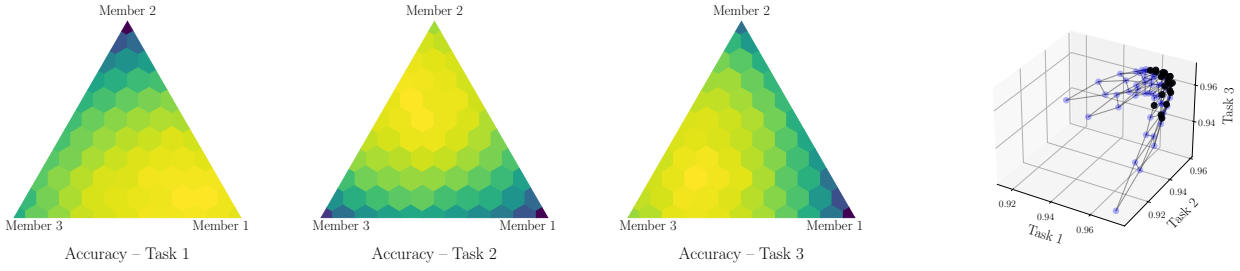
Table 10 compares the performance of baselines and the proposed method while Figure 22 presents visually the performance achieved on the discovered subspace.

Table 10: MultiMNIST-3: Mean Accuracy and standard deviation of accuracy (over 3 random seeds). For the proposed method (PaMaL), we report the mean and standard deviation of the best performance from the interpolated models in the sampled subspace. No balancing schemes and regularization are applied. Bold is used for the best performing multi-task method.

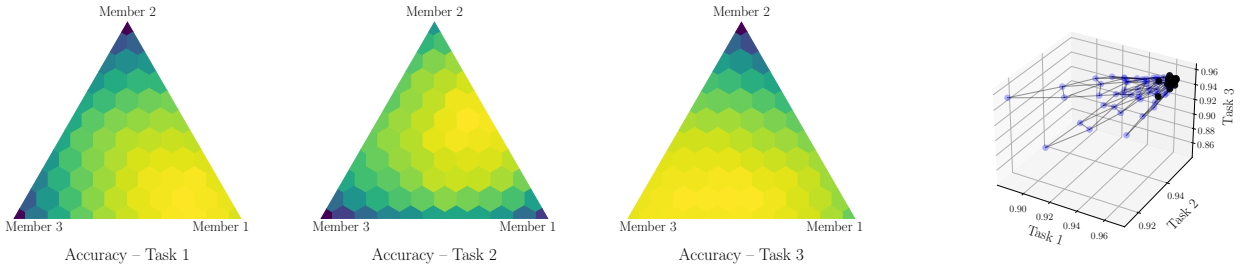
	Task 1	Task 2	Task 3
STL	96.97 ± 0.06	96.10 ± 0.17	96.40 ± 0.22
LS	96.26 ± 0.20	95.48 ± 0.14	95.87 ± 0.37
UW	96.48 ± 0.08	95.42 ± 0.30	95.77 ± 0.06
MGDA	96.50 ± 0.20	94.80 ± 0.22	95.71 ± 0.08
PCGrad	96.45 ± 0.06	95.39 ± 0.15	95.88 ± 0.01
IMTL	96.58 ± 0.22	95.18 ± 0.12	96.08 ± 0.31
Graddrop	96.25 ± 0.36	95.32 ± 0.24	95.61 ± 0.15
CAGrad	96.70 ± 0.13	95.20 ± 0.26	95.66 ± 0.06
RLW	96.06 ± 0.40	94.89 ± 0.18	95.68 ± 0.26
Nash-MTL	$\mathbf{96.85 \pm 0.08}$	95.25 ± 0.23	96.18 ± 0.13
Auto-λ	96.60 ± 0.17	95.16 ± 0.14	96.04 ± 0.18
RotoGrad	94.80 ± 0.75	92.79 ± 0.87	94.77 ± 0.38
PaMaL (ours)	$\mathbf{96.85 \pm 0.43}$	$\mathbf{95.72 \pm 0.22}$	$\mathbf{96.27 \pm 0.32}$



(a) Seed 0



(b) Seed 1



(c) Seed 2

Figure 22: `MultiMNIST-3` results for all three seeds. Each triangle shows the 66 points in the convex hull and color is used for the performance on the associated task. The 3d plot shows the mapping of the subspace to the multi-objective space. No balancing scheme is used.

Table 11: Test performance on *CityScapes*. See text for description of Settings I and II. 3 random seeds per method. For Pareto Manifold Learning, we report the mean (across seeds) best results from the final subspace. Methods are divided into single-task, single-solution MTL, multi-solution MTL and proposed method.

	Setting I				Setting II			
	Segmentation		Depth		Segmentation		Depth	
	mIoU ↑	Pix Acc ↑	Abs Err ↓	Rel Err ↓	mIoU ↑	Pix Acc ↑	Abs Err ↓	Rel Err ↓
STL	71.79	92.60	0.0135	32.786	70.96	92.12	0.0141	38.644
LS	70.94	92.29	0.0192	117.658	70.12	91.90	0.0192	124.061
UW	70.97	92.24	0.0188	118.168	70.20	91.93	0.0189	125.943
MGDA	69.23	91.77	0.0138	51.986	66.45	90.79	0.0141	53.138
DWA	70.87	92.23	0.0190	113.565	70.10	91.89	0.0192	127.659
PCGrad	71.14	92.32	0.0185	117.797	70.02	91.84	0.0188	126.255
IMTL	71.54	92.47	0.0151	65.058	70.77	92.12	0.0151	74.230
Graddrop	71.28	92.41	0.0182	124.645	70.07	91.93	0.0189	127.146
CAGrad	70.23	92.06	0.0173	100.162	69.23	91.61	0.0168	110.139
RLW	69.94	91.94	0.0195	119.667	68.79	91.52	0.0213	126.942
Nash-MTL	72.07	92.61	0.0147	62.980	71.13	92.23	0.0157	78.499
RotoGrad	70.41	92.03	0.0134	48.366	69.92	91.85	0.0193	127.281
Auto-λ	71.08	92.24	0.0173	118.959	70.47	92.01	0.0177	116.959
COSMOS	70.37	92.07	0.0317	107.575	69.78	91.79	0.0539	136.614
PaMaL(W=3, p_0=7)	71.13	92.31	0.0138	50.985	70.35	91.99	0.0141	54.520

C.4. `CityScapes` additional results

The `CityScapes` dataset has 2975 training images and no publicly available test set. The validation set is used for test. We refer to the original validation set as test set to avoid confusion. As far as we understand, prior works in Multi-Task Learning do not discuss any splitting of the training set to accommodate a validation set. Hence, it is unclear how hyperparameters are set. For this reason, we evaluate on two settings:

- Setting I: no validation set. All 2975 images are used for training.
- Setting II: Use 500 out of 2975 images for validation. The validation set is used to tune hyperparameters.

The test set is the same in both settings. In the main text, we report the results for Setting II. Table 11 presents the results for both settings. For clarity, PaMaL (ours) uses the same hyperparameters for both settings. While the increase in number of training samples leads to a quantitative boost in performance, the results are qualitatively similar. Specifically, MGDA still performs optimally (out of MTL methods) in Depth Estimation but performs poorly for Segmentation. COSMOS exhibits task bias towards Segmentation. On the other hand, the proposed method produces balanced solutions.