OPT2025: 17th Annual Workshop on Optimization for Machine Learning

HyperPALoRA: Parameter-Efficient Pareto Hypernetworks via Preference-Based Diverse Low-Rank Adaptations

Ashmita Bhattacharya AZB6481@PSU.EDU
Malyaban Bal MJB7906@PSU.EDU
Pennsylvania State University

Abstract

Multi-task learning (MTL) is often cast as multi-objective optimization, where Pareto Front Learning (PFL) seeks a continuum of task trade-offs with a single model. Existing PFL methods, especially Pareto Hypernetworks (PHNs), capture complex relationships between task tradeoffs and solution space but struggle with scalability, memory, and convergence. In this work, we propose a parameter-efficient PFL framework that augments a shared backbone with low-rank adaptations (LoRA) generated by a single preference-conditioned hypernetwork. First, instead of predicting full target-network weights, our hypernetwork outputs a single preference-aligned LoRA, sharply reducing preference-dependent parameters and improving PHN scalability. Second. unlike prior LoRA-based approaches that linearly combine per-task LoRAs and thus only realize convex trade-offs, our PHN-based formulation naturally represents non-convex Pareto fronts, as verified on classical non-convex benchmarks. Third, to combat the limited diversity of prior PFL methods, we introduce a contrastive, preference-aware loss that keeps neighboring preferences similar while separating distant ones, yielding a well-spread set of Pareto-optimal solutions. Experiments on standard MTL benchmarks show that our method matches or outperforms state-of-the-art PFL baselines while offering substantially improved parameter efficiency compared to PHN-based PFL methods.

1. Introduction

Multi-task learning (MTL) [4, 47] leverages shared representations across related tasks but remains challenging due to conflicting objectives and differing task scales. Some works mitigate gradient conflicts during optimization [2, 31, 56], while others cast MTL as multi-objective optimization (MOO) [30, 49], aiming to discover Pareto-optimal trade-offs. Building on this view, several studies [30, 37, 42] model task interactions via MOO to obtain Pareto-optimal solutions. However, these methods typically produce only a single solution per fixed preference or a finite set of discrete solutions, limiting flexibility to adapt to user-defined preferences at test time.

Prior works [42, 44, 46, 54] approximate the Pareto front continuously using preference-conditioned mechanisms, either via preference-augmented features or preference-driven parameter adaptations. Navon et al. [41] introduced Pareto Front Learning (PFL) with hypernetworks [17], where a preference-conditioned Pareto Hypernetwork (PHN) generates the Pareto front by producing optimal target-model parameters per preference. However, even with proposed extensions, PHNs often exceed the size of the target network, limiting practicality. We address this by scaling PHNs through preference-conditioned low-rank adaptations (LoRA) of the target network, rather than generating full network weights.

Continuous Pareto approximations have recently been modeled as linear combinations of single-task base networks conditioned on preference vectors [13]. While sometimes effective, this scales

poorly—requiring one base network per task, with parameters growing linearly and shared structure underused. To improve efficiency, recent work adds task-specific low-rank adaptations (LoRA) [20] atop a shared backbone [5, 14], boosting parameter efficiency and generalization. However, these methods are still constrained hypernetwork variants, limited to fixed linear combinations of base networks and thus struggle to capture non-linear task dependencies and complex trade-offs.

In this work, we leverage hypernetworks to generate preference-conditioned low-rank adaptations (LoRA) of a shared base network to generate complex, potentially non-convex Pareto fronts. This avoids generating full target networks or separate per-task LoRAs, keeping parameter overhead scalable as the number of tasks grows. To promote diversity, we introduce a contrastive loss that enforces similarity among solutions for nearby preferences while encouraging dissimilarity for distant ones, yielding well-distributed Pareto-optimal solutions. Our primary contributions are as follows:

- We propose **PHN-LoRA**, a parameter-efficient Pareto Front Learning (PFL) method with hypernetworks leveraging low rank adaptations. PHN-LoRA utilizes a hypernetwork to generate task-preference-aligned, per-layer low-rank adapters, making it adaptable to any target network architecture while maintaining parameter efficiency.
- We introduce a contrastive, preference-aware loss that pulls together solutions for nearby preferences and pushes apart those for distant ones, producing well-spread, functionally diverse Pareto fronts and significantly improving hypervolume.
- PHN-LoRA can inherently represent non-convex Pareto fronts while remaining highly parameter-efficient. In contrast, existing LoRA-based approaches that rely on convex combinations of per-task LoRAs are fundamentally limited to convex trade-offs.
- Across multiple benchmarks and target architectures, we show that PHN-LoRA matches or outperforms state-of-the-art PFL baselines, while significantly reducing memory footprint and improving parameter efficiency and scalability compared to traditional PHN-based approaches.

2. Proposed Methodology: PHN-LoRA

A hypernetwork is a meta-model that generates a target network's parameters. In our setting, it takes a task-preference vector as input and outputs low-rank adaptation (LoRA) matrices for the layers of a shared backbone, enabling flexible, parameter-efficient adaptation.

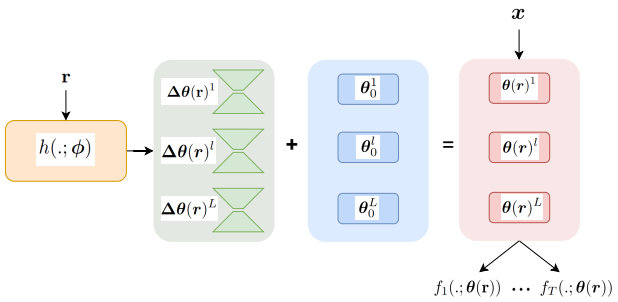

Figure 1: PHN-LoRA architecture: a preference ray is input to generate a single preference-aligned LoRA per target layer. The adapted target produces multi-task outputs, and resulting losses are backpropagated to train PHN-LoRA.

Formally, let $\mathcal{S}_T = \{\mathbf{r} \in \mathbb{R}_+^T \mid \sum_{t=1}^T r_t = 1\}$ denote the T-dimensional preference simplex, and let $h(\cdot;\boldsymbol{\phi}) : \mathcal{S}_T \to \mathbb{R}^n$ be the hypernetwork, where n is the total dimension of the concatenated LoRA parameters generated for all layers of the target network. The hypernetwork maps a preference vector $\mathbf{r} \in \mathcal{S}_T$ to low-rank parameter updates $\Delta\boldsymbol{\theta}(\mathbf{r})$. Given shared backbone parameters

$\boldsymbol{\theta}_0$, the adapted parameters for preference $\mathbf{r}$ are:

$$\boldsymbol{\theta}(\mathbf{r}) = \boldsymbol{\theta}_0 + s\,\Delta\boldsymbol{\theta}(\mathbf{r}) = \boldsymbol{\theta}_0 + s\,h(\mathbf{r};\boldsymbol{\phi}),$$

where $s > 0$ is a scaling factor that controls the strength of the adaptation. The target network $g(\cdot;\boldsymbol{\theta}(\mathbf{r}))$ maps an input $\mathbf{x}$ to task-specific outputs

$$g(\mathbf{x};\boldsymbol{\theta}(\mathbf{r})) = \big(f_1(\mathbf{x};\boldsymbol{\theta}(\mathbf{r})),\ldots,f_T(\mathbf{x};\boldsymbol{\theta}(\mathbf{r}))\big),$$

where $f_t(\cdot)$ denotes the predictor for task t.

Following [41], preference vectors $\mathbf{r} \in \mathcal{S}_T$ are sampled from the simplex and encoded with a small MLP. The hypernetwork h is an MLP with multiple heads, each outputting LoRA parameters for a specific target layer (Figure 1). In multi-task learning with T objectives, we seek a mapping from preferences to Pareto-optimal loss vectors $\mathbf{L} = [\mathcal{L}_1,\ldots,\mathcal{L}_T] \in \mathbb{R}^T$. The embedded preference guides h to generate $\Delta\boldsymbol{\theta}(\mathbf{r})$, adapting the model to the desired trade-off. During training, sampled preferences are used to optimize h so that $\boldsymbol{\theta}(\mathbf{r})$ lies near the corresponding Pareto front; at inference, any preference can be used to generate aligned LoRA weights. Our framework is compatible with various Pareto front learning strategies [41].

Linear Scalarization This common MOO approach minimizes a weighted loss vector $\boldsymbol{L} \in \mathbb{R}^T$ using preference vector $\boldsymbol{r}$. The base target network and hypernetwork are optimized via:

$$\mathcal{L}_{LS}(\boldsymbol{\theta}_0,\boldsymbol{\phi}) = \mathbb{E}_{\boldsymbol{x},\boldsymbol{y}\sim p_D,\boldsymbol{r}}\left[\sum_{i=1}^{T} r_i\,\mathcal{L}_i\big(\boldsymbol{x},y_i,\boldsymbol{\theta}_0 + s\boldsymbol{\Delta\theta}(\boldsymbol{r};\boldsymbol{\phi})\big)\right], \tag{1}$$

where $\boldsymbol{x}$ and $\boldsymbol{y}$ are inputs and targets, and $\boldsymbol{r}$ is the sampled preference.

Exact Pareto Optimization Linear scalarization only recovers convex regions of the Pareto front, and the Pareto-optimal point for a ray may not coincide with the scalarized minimum [41]. Exact Pareto Optimization (EPO) instead computes the true Pareto-optimal point:

$$\mathcal{L}_{EPO}(\boldsymbol{\theta}_0,\boldsymbol{\phi}) = \mathbb{E}_{\boldsymbol{x},\boldsymbol{y}\sim p_D,\boldsymbol{r}}\left[\sum_{i=1}^{m} \beta_i\,\mathcal{L}_i\big(\boldsymbol{x},y_i,\boldsymbol{\theta}_0 + s\boldsymbol{\Delta\theta}(\boldsymbol{r};\boldsymbol{\phi})\big)\right],$$
$$\text{where}\quad \boldsymbol{\beta} \leftarrow \text{EPO}(\boldsymbol{\theta}_0,\boldsymbol{\Delta\theta}(\boldsymbol{\phi},\boldsymbol{r}),\boldsymbol{L},\boldsymbol{r}). \tag{2}$$

2.1. Additional Loss Functions and Sampling of Preferences

Following [42], we adopt a multi-sample hypernetwork training strategy. For each mini-batch $(\boldsymbol{x},\boldsymbol{y})$, a set of preference rays $\{\boldsymbol{r}^i\}_{i=1}^p$ is sampled, and the hypernetwork h generates LoRA parameters $\boldsymbol{\Delta\theta}(\boldsymbol{r}^i;\boldsymbol{\phi})$. For each ray, we compute task-specific losses $\boldsymbol{L}^i = [\mathcal{L}_1^i,\ldots,\mathcal{L}_T^i]^\top \in \mathbb{R}^T$. The training objective $Q(\boldsymbol{\phi};\boldsymbol{x},\boldsymbol{y})$ is defined over the set of loss vectors $\{\boldsymbol{L}^i\}_{i=1}^p$. During training, we optimize two complementary objectives, with tailored preference-ray sampling strategies aligned to each.

Hypervolume Increasing Loss ($\mathcal{L}_{HVI}$) To maximize the Pareto Hypervolume (HV), we compute it over the set of loss (or accuracy) vectors $\boldsymbol{L}_{set} = \{\boldsymbol{L}^i\}_{i=1}^p$. For accurate HV estimation, we sample evenly spaced preference rays in the objective simplex. The objective is:

$$\mathcal{L}_{HVI}(\boldsymbol{\theta}_0,\boldsymbol{\phi}) = \mathbb{E}_{\boldsymbol{x},\boldsymbol{y}\sim p_D}\left[-HV\big(\boldsymbol{L}_{set}(\boldsymbol{\Delta\Theta}_{\boldsymbol{\phi}};\boldsymbol{x},\boldsymbol{y})\big)\right], \tag{3}$$

where increasing HV guarantees Pareto improvement due to its monotonicity. Here, $\boldsymbol{\Delta\Theta}_{\boldsymbol{\phi}} = \{\Delta\boldsymbol{\theta}(\mathbf{r}^i)\}_{i=1}^p = \{h(\mathbf{r}^i;\boldsymbol{\phi})\}_{i=1}^p$ denotes the collection of preference-conditioned parameter updates.

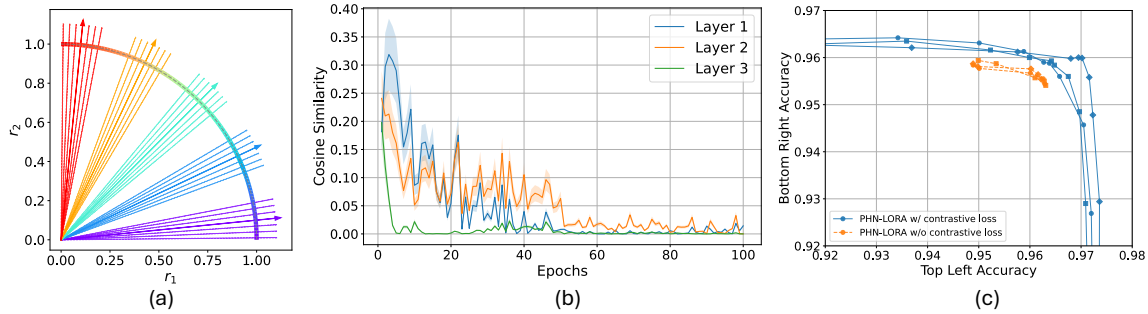

Figure 2: Effect of contrastive loss: (a) preference sampling generates evenly spaced rays with positives in each neighborhood; (b) contrastive loss reduces cosine similarity between LoRA parameters of distant rays on MultiMNIST (shaded: 95% CI over 3 seeds); (c) resulting Pareto fronts are more functionally diverse compared to clustered fronts without the loss.

Diversity Increasing Loss ($\mathcal{L}_{DI}$) As noted in [5, 13], cosine similarity between task-specific networks often grows during training, leading to reduced diversity—a trend also observed in PHNs. To counter this, we design a contrastive loss that encourages similarity of LoRA parameters for nearby rays and dissimilarity for distant ones. Given a ray $\boldsymbol{r} \in \mathbb{R}^m$, positives are generated via perturbations:

$$\boldsymbol{r}_+^i = \text{clamp}(\boldsymbol{r} + \epsilon_i, 0, 1), \quad \epsilon_i \sim \mathcal{N}(0, \sigma^2), \quad \sigma = 0.05,$$

The operator $\text{clamp}(\cdot)$ applies element-wise clipping to a specified range and σ controls the locality of the neighborhood in preference space. Negatives are all other sampled rays $\boldsymbol{r}^j, j \neq i$. We use $\mathcal{P}$ and $\mathcal{N}$ to denote the sets of positive and negative pairs (or indices) used in the contrastive loss, respectively. The contrastive objective is:

$$\begin{aligned}\mathcal{L}_{DI} = \mathbb{E}_{\boldsymbol{r}}\Bigg[\frac{1}{|\mathcal{P}|}\sum_{\boldsymbol{r}^+\in\mathcal{P}}\sum_{l=1}^{L}\Big(1-\cos\left(\boldsymbol{\Delta\theta}^l(\boldsymbol{r}),\boldsymbol{\Delta\theta}^l(\boldsymbol{r}^+)\right)\Big)^2 \\ + \frac{1}{|\mathcal{N}|}\sum_{\boldsymbol{r}^-\in\mathcal{N}}\sum_{l=1}^{L}\Big(\cos\left(\boldsymbol{\Delta\theta}^l(\boldsymbol{r}),\boldsymbol{\Delta\theta}^l(\boldsymbol{r}^-)\right)\Big)^2\Bigg].\end{aligned} \tag{4}$$

3. Experiments

We first evaluate our method on illustrative MOO problems with known non-convex Pareto fronts, and then on multi-task benchmarks including *MultiMNIST*, *Census*, and *UTKFace*, using architectures from simple MLPs to complex CNNs such as LeNets and ResNets. Additional experimental results are provided in Appendix C, implementation details (architectures, baselines, and hyperparameters) in Appendix D, and further ablations in Appendix E.

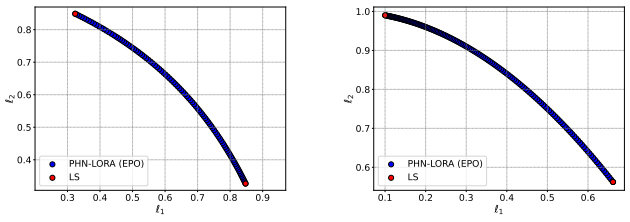

Figure 3: Illustrative examples of known Pareto fronts. PHN-LoRA with EPO (blue) captures non-convex fronts, while LS-based PFL methods, including weight-ensembling [5, 13, 14], capture only convex regions (red).

Table 1: HV values obtained on *MultiMNIST* and *Census* (averaged over three random seeds). The standard deviation is shown in parentheses. For PHN and PHN-HVI, we report the number of parameters in the hypernetwork.

Method	**MultiMNIST**		**Census**	
	HV	**# Parameters**	**HV**	**# Parameters**
PHN	0.900 (0.0068)	2.793M	0.649 (0.0007)	12.251M
PHN-HVI	0.906 (0.0020)	2.793M	0.650 (0.0008)	12.251M
PHN-LoRA	**0.9375 (0.0025)**	0.739M	**0.655 (0.0005)**	0.270M

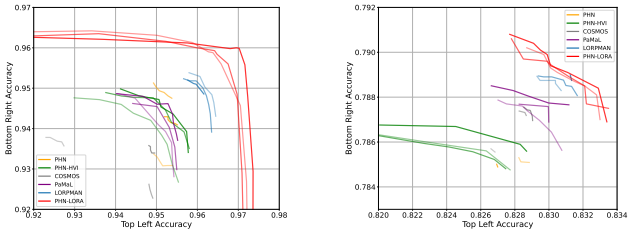

Figure 4: Test performance of PHN-LoRA on MultiMNIST and Census. Three seeds are plotted for each method. PHN-LoRA generates wider and better Pareto fronts that other PFL methods (top right is optimal).

Illustrative Examples: Known Pareto Fronts We first evaluate our method on illustrative multi-objective optimization (MOO) problems with known Pareto fronts. The first is the MOO problem of [9] with $\boldsymbol{\theta} \in \mathbb{R}^{100}$ and a non-convex Pareto front in $\mathbb{R}^2$. Lacking a natural matrix structure, we replace the 100-dimensional adaptation $\boldsymbol{\Delta\theta}$ with a compact LoRA representation $\boldsymbol{\Delta\theta}_{\text{LoRA}} \in \mathbb{R}^r$, $r \ll d$, which is then expanded to reconstruct the full parameter update. Here, $r = 50$ and the adaptation is replicated to obtain the 100-dimensional vector. As shown in Figure 3, PHN-LoRA with the EPO solver recovers the full Pareto front, unlike scalarization and weight-ensembling methods [5, 13, 14], which capture only a subset due to linear combinations of task-specific LoRA models. PHN-LoRA also successfully recovers the non-convex Pareto front of ZDT2 over a 30-dimensional parameter space.

Classification Tasks: MultiMNIST and Census We evaluate PHN-LoRA on **MultiMNIST** [13] using a LeNet backbone, and on **Census** [24] for predicting age and education level with an MLP backbone. Models are trained for 20 epochs, freezing the base network after 5. Evaluation is performed on 11 evenly spaced rays using window size $b = 5$ for the multi-sample hypervolume and contrastive losses. We tune λ_{HVI}, λ_{DI}, and scaling factor $s \in \{1, 2, 4, 6\}$ on the validation set, and set the LoRA rank to 8 for all layers. As shown in Figure 4, the contrastive loss substantially enhances functional diversity in Pareto-optimal solutions and yields significantly higher hypervolume than PHN-based baselines.

Classification and Regression: UTKFace We evaluate PHN-LoRA on the UTKFace dataset [57], which includes three tasks: age prediction, gender classification, and race classification. ResNet-18 [18] serves as the shared base network. Prior works excluded PHN-based methods due to the hypernetwork's size reaching nearly a billion parameters. PHN-LoRA, however, enables learning the Pareto front with a comparable number of parameters. Training is performed for 100 epochs, freezing the base network after epoch 80 (Table 8). A scaling factor of $s = 2$ yields the highest

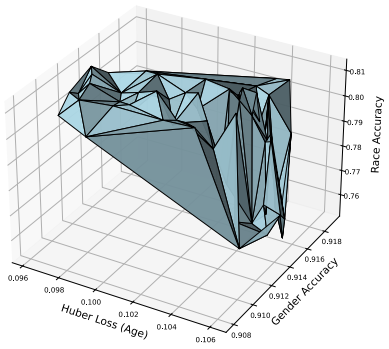

Figure 5: Pareto front in UTKFace.

Method	HV	# Params
COSMOS	0.281 (0.003)	11.2M
PaMaL	0.304 (0.003)	33.6M
PHN-LoRA	**0.3066 (0.003)**	18.0M

Table 2: HV and number of parameters in UTKFace (averaged over three random seeds).

validation hypervolume. We evaluate ranks $r \in \{2, 4, 8\}$, with rank 4 yielding the best performance, as shown in Figure 5 and Table 2.

PHN-LoRA Scalability As shown in Figure 6, PHN-LoRA scales linearly with LoRA rank yet outperforms hypernetwork-based baselines such as PHN and PHN-HVI while using less than 30% of the parameters on MultiMNIST. This highlights PHN-LoRA as a highly parameter-efficient and effective alternative to PHN, as also evidenced in Appendix Table 12. Additional scalability can be achieved by generating LoRA only for selected layers [15] or by applying techniques such as chunking [17, 43], where target parameters are conditioned on segments of the input embedding.

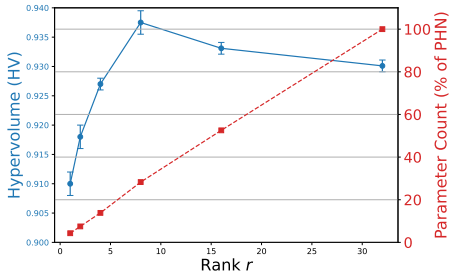

Figure 6: Effect of LoRA rank on hypervolume and parameter efficiency.

Effect of Contrastive Loss $\mathcal{L}_{DI}$. Table 3 shows the effect of introducing the contrastive loss, which promotes diversity in the hypernetwork-generated LoRA parameters across tradeoffs. This leads to improved coverage of the Pareto front and higher hypervolume. This is also evidenced in the Figure 2 (c), where the Pareto fronts generated utilizing the contrastive loss consists of well-distributed solutions. Additional ablation studies are shown in the Appendix Section E.

Table 3: Effect of contrastive loss $\mathcal{L}_{DI}$ (averaged over six random seeds) on *MultiMNIST*.

	HV
w/o $\mathcal{L}_{DI}$	0.9230 (0.0017)
w/ $\mathcal{L}_{DI}$	**0.9375 (0.0025)**

4. Conclusions

In this paper, we propose PHN-LoRA, a scalable and parameter-efficient hypernetwork-based approach for learning continuous Pareto fronts. By generating a single task-preference-aligned LoRA for the target network, PHN-LoRA eliminates the need to learn separate LoRA modules for each task. To enhance solution diversity, we introduce a contrastive loss that encourages better-distributed Pareto fronts, leading to improved hypervolume. PHN-LoRA serves as a strong, and efficient alternative to existing PHN-based methods, with potential for further scalability through techniques such as layer-wise rank adaptation or selective layer LoRA generation.

Acknowledgements

This research was supported by the National Science Foun- dation through the Scipe AI in Civil Engineering Fellowship (Award No. 2321040).

References

[1] Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *Association for Computational Linguistics (ACL)*, 2021. URL https://aclanthology.org/2021.acl-long.568.

[2] Devansh Arpit, Thanh-Tung Nguyen, Behzad Tabibian, and Bernhard Schölkopf. Rotograd: Dynamic gradient homogenization for multi-task learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=LL116CJl4H.

[3] Antoine Audibert, Massih-Reza Amini, Konstantin Usevich, and Marc Clausel. Low-rank updates of pre-trained weights for multi-task learning. In *Findings of the Association for Computational Linguistics*, pages 7544–7554, 2023.

[4] Rich Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997. doi: 10.1023/A:1007379606734. URL https://doi.org/10.1023/A:1007379606734.

[5] Weiyu Chen and James T Kwok. Efficient pareto manifold learning with low-rank structure. *arXiv preprint arXiv:2407.20734*, 2024.

[6] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2018. URL http://arxiv.org/abs/1711.02257v4.

[7] Zhao Chen, Jiquan Ngiam, Yanping Huang, Thang Luong, Henrik Kretzschmar, Yuning Chai, and Dragomir Anguelov. Just pick a sign: Optimizing deep multitask models with gradient sign dropout. In *Advances in Neural Information Processing Systems*, 2020. URL http://arxiv.org/abs/2010.06808v1.

[8] Roberto Cipolla, Yarin Gal, and Alex Kendall. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. URL http://arxiv.org/abs/1705.07115v3.

[9] Carlos Manuel Mira da Fonseca. *Multiobjective Genetic Algorithms with Application to Control Engineering Problems*. Phd thesis, University of Sheffield, 1995.

[10] Kalyanmoy Deb and Himanshu Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints. *IEEE Transactions on Evolutionary Computation*, 18(4):577–601, 2013. doi: 10.1109/TEVC.2013.2281535. URL https://doi.org/10.1109/TEVC.2013.2281535.

[11] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002. doi: 10.1109/4235.996017. URL https://doi.org/10.1109/4235.996017.

[12] Jean-Antoine Désidéri. Multiple-gradient descent algorithm (mgda) for multiobjective optimization. *Comptes Rendus. Mathématique*, 350(5–6):313–318, 2012. doi: 10.1016/j.crma.2012.03.014.

[13] Nikolaos Dimitriadis, Pascal Frossard, and François Fleuret. Pareto manifold learning: Tackling multiple tasks via ensembles of single-task models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 8015–8052. PMLR, 2023. URL https://proceedings.mlr.press/v202/dimitriadis23a.html.

[14] Nikolaos Dimitriadis, Pascal Frossard, and François Fleuret. Pareto low-rank adapters: Efficient multi-task learning with preferences. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025. URL https://arxiv.org/abs/2407.08056.

[15] Alexey Dosovitskiy and Josip Djolonga. You only train once: Loss-conditional training of deep networks. In *International Conference on Learning Representations (ICLR)*, 2020. URL https://openreview.net/forum?id=HyxY6JHKwr.

[16] Wenfeng Feng, Chuzhan Hao, Yuewei Zhang, Yu Han, and Hao Wang. Mixture-of-loras: An efficient multitask tuning method for large language models. In *Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING)*, 2024. URL https://aclanthology.org/2024.lrec-main.994.

[17] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778. IEEE, 2016.

[19] Long P. Hoang, Dung D. Le, Tran Anh Tuan, and Tran Ngoc Thang. Improving pareto front learning via multi-sample hypernetworks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 7875–7883, 2023. doi: 10.1609/aaai.v37i7.25953. URL https://doi.org/10.1609/aaai.v37i7.25953.

[20] E. J. Hu, Y. Shen, H. Wallach, A. Mishra, G. Kohn, A. Gu, R. Salakhutdinov, and Z. Ma. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2205.04806*, 2022. URL https://arxiv.org/abs/2205.04806.

[21] Chengsong Huang, Qian Liu, Bill Yuchen Lin, Chao Du, Tianyu Pang, and Min Lin. Lorahub: Efficient cross-task generalization via dynamic lora composition. In *R0-FoMo: Robustness of Few-shot and Zero-shot Learning in Large Foundation Models*, 2023. URL https://openreview.net/forum?id=lyRpY2bpBh.

[22] Peter J. Huber. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 35(1):73–101, 1964. doi: 10.1214/aoms/1177703732. URL https://projecteuclid.org/euclid.aoms/1177703732.

[23] Joshua D. Knowles, David W. Corne, and Mark Fleischer. Bounded archiving using the lebesgue measure. In *Proceedings of the Congress on Evolutionary Computation (CEC)*, pages 2490–2497, 2003.

[24] Ron Kohavi. Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 202–207, 1996.

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

[26] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2021. URL http://arxiv.org/abs/2104.08691v2.

[27] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Association for Computational Linguistics (ACL)*, 2021. URL https://aclanthology.org/2021.acl-long.353.

[28] Baijiong Lin, Feiyang Ye, Yu Zhang, and Ivor W. Tsang. 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.

[29] Xi Lin, Zhiyuan Yang, Qingfu Zhang, and Sam Kwong. Controllable pareto multi-task learning, 2021. URL http://arxiv.org/abs/2010.06313v2. arXiv:2010.06313v2 [cs.LG], version 2 released Feb. 15, 2021.

[30] Xin Lin, Yiren Zhao, Zhun Deng, Yang Liu, Tianle Cai, Zhenguo Li, and Jun Zhu. Pareto multi-task learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. URL https://arxiv.org/abs/1906.10673.

[31] Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. Conflict-averse gradient descent for multi-task learning. In *Advances in Neural Information Processing Systems*, volume 34, pages 14338–14350, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/9d27fdf2477ffbff837d73ef7ae23db9-Abstract.html.

[32] Liyang Liu, Yi Li, Zhanghui Kuang, Jing-Hao Xue, Yimin Chen, Wenming Yang, Qingmin Liao, and Wayne Zhang. Towards impartial multi-task learning. In *International Conference on Learning Representations (ICLR)*, 2020.

[33] Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. Dora: Weight-decomposed low-rank adaptation. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2024. URL https://openreview.net/forum?id=3d5CIRG1n2.

[34] Shikun Liu, Stephen James, Andrew J. Davison, and Edward Johns. Auto-λ: Disentangling dynamic task relationships. *Transactions on Machine Learning Research*, May 2022. ISSN 2835-8856. URL https://openreview.net/forum?id=KKeCMim5VN.

[35] Xingchao Liu, Xin Tong, and Qiang Liu. Profiling pareto front with multi-objective stein variational gradient descent. In *Advances in Neural Information Processing Systems*, volume 34, pages 14721–14733, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/7bb16972da003e87724f048d76b7e0e1-Abstract.html.

[36] Pingchuan Ma, Tao Du, and Wojciech Matusik. Efficient continuous pareto exploration in multi-task learning. In *Proceedings of the International Conference on Machine Learning*, pages 6522–6531, 2020. URL http://arxiv.org/abs/2006.16434v2.

[37] Debabrata Mahapatra and Vaibhav Rajan. Multi-task learning with user preferences: Gradient descent with controlled ascent in pareto optimization. In *Proceedings of the 37th International Conference on Machine Learning*, pages 6597–6607. PMLR, 2020. URL https://proceedings.mlr.press/v119/mahapatra20a.html.

[38] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch networks for multi-task learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3994–4003, 2016. doi: 10.1109/CVPR.2016.433. URL https://doi.org/10.1109/CVPR.2016.433.

[39] Hyeon-Woo Nam, Ye-Bin Moon, and Tae-Hyun Oh. Fedpara: Low-rank hadamard product for communication-efficient federated learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022. URL https://openreview.net/forum?id=d71n4ftoCBy.

[40] Aviv Navon, Idan Achituve, Haggai Maron, Gal Chechik, and Ethan Fetaya. Auxiliary learning by implicit differentiation, 2020. URL https://arxiv.org/abs/2007.02693. arXiv:2007.02693v3.

[41] Aviv Navon, Aviv Shamsian, Gal Chechik, and Ethan Fetaya. Learning the pareto front with hypernetworks. In *Proceedings of the 9th International Conference on Learning Representations (ICLR)*, 2021. URL https://openreview.net/forum?id=NjF772F4ZZR.

[42] Aviv Navon, Aviv Shamsian, Idan Achituve, Haggai Maron, Kenji Kawaguchi, Gal Chechik, and Ethan Fetaya. Multi-task learning as a bargaining game. In *Proceedings of the 39th International Conference on Machine Learning*, pages 16289–16310. PMLR, 2022. URL https://proceedings.mlr.press/v162/navon22a.html.

[43] J. Oswald, C. Henning, J. Sacramento, and Benjamin F. Grewe. Continual learning with hypernetworks. *arXiv preprint arXiv:1906.00695*, abs/1906.00695, 2020. URL https://arxiv.org/abs/1906.00695.

[44] Simone Parisi, Matteo Pirotta, and Marcello Restelli. Multi-objective reinforcement learning through continuous pareto manifold approximation. *Journal of Artificial Intelligence Research*, 57:187–227, 2016. URL https://jair.org/index.php/jair/article/view/11026.

[45] Michael Ruchte and Josif Grabocka. 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.

[46] Michael Ruchte and Josif Grabocka. Scalable pareto front approximation for deep multi-objective learning. In *Proceedings of the 2021 IEEE International Conference on Data Mining (ICDM)*, pages 1306–1311. IEEE, 2021. doi: 10.1109/ICDM51629.2021.00162. URL https://arxiv.org/abs/2103.13392.

[47] Sebastian Ruder. An overview of multi-task learning in deep neural networks, 2017. URL https://arxiv.org/abs/1706.05098.

[48] Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. Latent multi-task architecture learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 4822–4829, 2019. URL https://aaai.org/ocs/index.php/AAAI/AAAI19/paper/view/17327.

[49] Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. In *Advances in Neural Information Processing Systems 31*, pages 525–536. Curran Associates, Inc., 2018. URL https://papers.nips.cc/paper/2018/hash/432aca3a1e345e339f35a30c8f65edce-Abstract.html.

[50] Chunlin Tian, Zhan Shi, Zhijiang Guo, Li Li, and Cheng Zhong Xu. Hydralora: An asymmetric lora architecture for efficient fine-tuning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. URL https://openreview.net/forum?id=qEpi8uWX3N.

[51] Mojtaba Valipour, Mehdi Rezagholizadeh, Ivan Kobyzev, and Ali Ghodsi. Dylora: Parameter-efficient tuning of pre-trained models using dynamic search-free low-rank adaptation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*, 2023. URL https://aclanthology.org/2023.eacl-main.239.

[52] Xinyu Wang, Tao Chen, Qiang Ge, Hengshuang Xia, Rui Bao, Rong Zheng, Qi Zhang, Tao Gui, and Xiaojun Huang. Orthogonal subspace learning for language model continual learning. In *Findings of the Association for Computational Linguistics*, pages 10658–10671, 2023.

[53] Yifan Wang, Yuhong Lin, Xiaoyu Zeng, and Guoping Zhang. Multilora: Democratizing lora for better multi-task learning. *arXiv preprint arXiv:2311.11501*, 2023.

[54] Runzhe Yang, Xingyuan Sun, and Karthik Narasimhan. A generalized algorithm for multi-objective reinforcement learning and policy adaptation. In *Advances in Neural Information Processing Systems*, volume 32, pages 14610–14621, 2019. URL https://papers.nips.cc/paper/2019/hash/4a46fbfca3f1465a27b210f4bdfe6ab3-Abstract.html.

[55] Yaming Yang, Dilxat Muhtar, Yelong Shen, Yuefeng Zhan, Jianfeng Liu, Yujing Wang, Hao Sun, Weiwei Deng, Feng Sun, Qi Zhang, et al. Mtl-lora: Low-rank adaptation for multi-task learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 22010–22018, 2025.

[56] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/f4a855dbd5067e3af9f9e3ec3389164a-Abstract.html.

[57] Zhifei Zhang, Yang Song, and Hairong Qi. Age progression/regression by conditional adversarial autoencoder. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017.

Appendix A. Related Work

Multi-Task Learning and Multi-Objective Optimization Multi-task learning (MTL) aims to solve multiple tasks jointly, typically via architectural changes [38, 48] or optimization strategies that balance shared and task-specific representations [6, 8, 31, 32, 56]. Optimization-based approaches balance task contributions in the loss [8, 28, 40] or gradient space [6, 7, 31, 34], but they converge to a single tradeoff and cannot adapt to user-defined preferences at inference. We adopt this optimization view, using a shared encoder–decoder architecture [4], and propose learning the full Pareto front in a single model based on task tradeoffs.

Multi-task learning (MTL) can be formulated as a Multi-Objective Optimization (MOO) problem, where the Pareto front represents various tradeoffs (preferences) across multiple tasks or objectives. While genetic algorithms have been widely used for small-scale MOO problems due to their ability to maintain diverse solutions [10, 11], these approaches are not directly compatible with deep learning frameworks. To address this, [49] introduced the use of the gradient-based MOO algorithm MGDA [12] in deep neural networks. Methods such as [30, 37] focused on learning discrete Pareto fronts, where each solution is trained independently to represent a specific task tradeoff. More advanced approaches [35, 36] have also been proposed for constructing discrete Pareto fronts, consisting of a finite set of solutions, but these also typically require training the model for each seed.

Pareto Front Learning and Hypernetworks Continuous Pareto front approximations can be achieved by learning a mapping from preference space to Pareto-optimal solutions—a process known as Pareto Front Learning (PFL). Hypernetworks have been effectively used for this purpose [17, 19, 29, 41], capturing complex tradeoffs across the front. However, they often face scalability issues due to their size relative to the target network. While recent work attempts to address this [15, 45], it often sacrifices diversity or performance. We address scalability by training a hypernetwork to generate compact adapter layers aligned with task-specific tradeoffs, preserving both efficiency and expressiveness.

Low Rank Adaptation (LoRA) Low-Rank Adaptation (LoRA), a leading parameter-efficient fine-tuning (PEFT) method [26, 27], adapts large pre-trained models to downstream tasks by updating only low-rank decompositions of weight matrices [20], leveraging the observation that fine-tuning occurs in low-dimensional subspaces [1]. This reduces trainable parameters, computational cost, and memory usage, while preserving pre-trained weights and mitigating overfitting. LoRA has been effective in transfer, continual, and multi-task learning [21, 33, 39, 50, 51]. In multi-task settings, prior work either learns a single task-specific LoRA [3] or multiple LoRAs to capture task-specific features [16, 52, 53, 55]. Recent MOO approaches construct Pareto fronts via convex combinations of task-specific LoRAs using linear hypernetworks [5, 13, 14]. However, these methods necessitate

learning separate LoRAs for each task and depend on linear hypernetworks to map the preference space to the solution space, limiting their scalability and representational capacity. In contrast, this work proposes a hypernetwork-based approach that learns a single, preference-aligned LoRA, enabling the efficient modeling of complex, non-convex Pareto fronts while addressing the scalability limitations of prior hypernetwork-based MOO/MTL techniques.

Appendix B. Problem Formulation

Preliminaries Consider a dataset $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$ with samples $x^{(i)} \in \mathcal{X}$, labels $y^{(i)} \in \mathcal{Y}$, and T tasks, each associated with a loss $\mathcal{L}_t$. The objective is to minimize the vector loss $\mathbf{L} = [\mathcal{L}_1, \ldots, \mathcal{L}_T]$ using a multi-task architecture comprising an encoder g and task-specific decoders f_t. In multi-task learning, no single solution is optimal for all tasks simultaneously.

Definition 1 (Pareto Optimality). Let $\mathcal{L}_t(\mathbf{y}_t, f_t(x, \theta)) = \mathcal{L}_t(\theta)$ denote the loss for task t, where $\theta \in \Theta$ represents the model parameters. A solution $\theta \in \Theta$ is said to dominate another solution $\theta' \in \Theta$ if:

$$\mathcal{L}_t(\theta) \leq \mathcal{L}_t(\theta') \quad \forall t \in [T] \quad \text{and} \quad \mathcal{L}(\theta) \neq \mathcal{L}(\theta').$$

A solution $\theta^* \in \Theta$ is Pareto optimal if there does not exist another solution $\theta' \in \Theta$ that dominates θ^*. The set of all Pareto optimal solutions is called the *Pareto set* $\mathcal{P}_S \subseteq \Theta$. Its corresponding image in the objective space, $\mathcal{P}_F = \{\mathbf{L}(\theta) : \theta \in \mathcal{P}_S\}$, is referred to as the *Pareto front*. Given a preference simplex $\mathbb{S}^T = \{\boldsymbol{r} \in \mathbb{R}_+^T \mid r_i \geq 0, \sum_{i=1}^{T} r_i = 1\}$, the goal of Pareto Front Learning (PFL) is to generate weights θ corresponding to a preference vector $\boldsymbol{r}$. This is achieved via a monotonic mapping $h : \mathbb{S}^T \to \Theta$, ensuring that increasing a task's importance decreases its loss. The aim is to discover a Pareto subspace, a low-dimensional representation of $\mathcal{P}_F$, in a single training run.

Appendix C. Additional Experiments and Results

Here, we show additional results on the multi-task learning datasets studied in the paper. We also show an additional multi-task learning dataset- MultiMNIST-3.

C.1. MultiMNIST

We report the mean of the best results along with standard deviations, computed across three random seeds, for the *Top-Left* and *Bottom-Right* classification tasks in `MultiMNIST` in Table 4. Some additional baseline results are shown in Table 5.

C.2. MultiMNIST-3

We evaluate the performance of PHN-LoRA on the three-task variant of MultiMNIST, referred to as MultiMNIST-3 [13]. The dataset is constructed following the same protocol described in [13]. MultiMNIST-3 is a synthetic benchmark derived from MNIST, to incorporate three objectives-predicting three digits on an image. Each image is generated by randomly selecting three digits (each of size 28×28) and placing them at the top-left, top-right, and bottom-middle locations of a 42×42 grid. In regions where overlaps occur, pixel-wise maxima are computed to preserve the most prominent features. The resulting composite image is subsequently resized to 28×28 pixels.

Table 6 presents a quantitative comparison of PHN-LoRA against established baselines. To visualize the tradeoffs across the three classification tasks, we plot a parallel coordinates diagram

Table 4: Summary of top-performing results on `MultiMNIST`.

	Top-Left	**Bottom-Right**
LS	95.47±0.08	94.45±0.43
UW	95.70±0.34	94.51±0.32
MGDA	95.57±0.11	94.33±0.13
DWA	95.52±0.07	94.48±0.36
PCGrad	95.51±0.07	94.56±0.37
IMTL	95.78±0.17	94.40±0.16
CAGrad	95.55±0.12	94.17±0.47
NashMTL	95.84±0.16	94.78±0.27
RLW	95.41±0.19	94.00±0.34
GradDrop	95.40±0.14	94.24±0.43
AutoL	95.94±0.39	94.57±0.40
RotoGrad	95.92±0.25	94.48±0.44
PHN	96.04±0.20	94.91±0.46
COSMOS	94.08±0.50	93.90±0.35
PAMAL	96.15±0.27	95.24±0.19
PaLoRA	96.55±0.13	95.39±0.24
PHN-LoRA (ours)	**97.20±0.18**	**96.38±0.22**

Table 5: HV values obtained on *MultiMNIST* and *Census* (averaged over three random seeds). The standard deviation is shown in parentheses. For PHN and PHN-HVI, we report the number of parameters in the hypernetwork.

Method	**MultiMNIST**		**Census**	
	HV	**# Parameters**	**HV**	**# Parameters**
COSMOS	0.888 (0.0077)	0.028M	0.652 (0.0014)	0.122M
PaMaL	0.905 (0.0010)	0.055M	0.654 (0.0006)	0.242M
LORPMAN	0.918 (0.0018)	0.046M	**0.656 (0.0004)**	0.133M
PHN	0.900 (0.0068)	2.793M	0.649 (0.0007)	12.251M
PHN-HVI	0.906 (0.0020)	2.793M	0.650 (0.0008)	12.251M
PHN-LoRA	**0.9375 (0.0025)**	0.739M	0.655 (0.0005)	0.270M

of the non-dominated solutions in Figure 7. Prior to plotting, the task accuracies are normalized to the $[0, 1]$ range to ensure uniform scaling across axes. Each gray line represents a Pareto-optimal solution, while the top-performing solutions for each task are highlighted using distinct colors.

Appendix D. Experimental Details

D.1. Baselines

We compare against state-of-the-art Pareto Front Learning (PFL) approaches, including COSMOS [45], hypernetwork-based methods such as PHN [41] and PHN-HVI [19], the weight-ensemble method PaMaL [13], and the LoRA-based ensemble method LORPMAN [5]. Performance is mea-

Table 6: Performance comparison across three tasks in `MultiMNIST-3`. Results are reported as mean ± standard deviation over multiple runs.

	Task 1	**Task 2**	**Task 3**
STL	96.97 ± 0.06	96.10 ± 0.17	96.40 ± 0.22
LS	96.26 ± 0.20	95.49 ± 0.14	95.87 ± 0.37
UW	96.48 ± 0.08	95.92 ± 0.30	95.70 ± 0.16
MGDA	96.50 ± 0.14	96.00 ± 0.25	95.71 ± 0.35
PCGrad	96.45 ± 0.06	95.96 ± 0.15	95.88 ± 0.01
IMTL	96.58 ± 0.22	96.01 ± 0.29	95.76 ± 0.20
Graddrop	96.25 ± 0.36	95.92 ± 0.24	96.01 ± 0.15
CAGrad	96.70 ± 0.13	95.90 ± 0.26	95.66 ± 0.06
RLW	96.06 ± 0.46	95.93 ± 0.15	96.10 ± 0.04
Nash-MTL	96.85 ± 0.08	95.95 ± 0.23	96.18 ± 0.13
Auto-λ	96.60 ± 0.17	96.14 ± 0.14	96.03 ± 0.01
RotoGrad	94.80 ± 0.75	92.79 ± 0.87	94.77 ± 0.38
PaMaL	96.85 ± 0.43	95.72 ± 0.22	96.27 ± 0.32
PHN-LoRA (ours)	**97.97 ± 0.35**	**97.12 ± 0.28**	**97.55 ± 0.30**

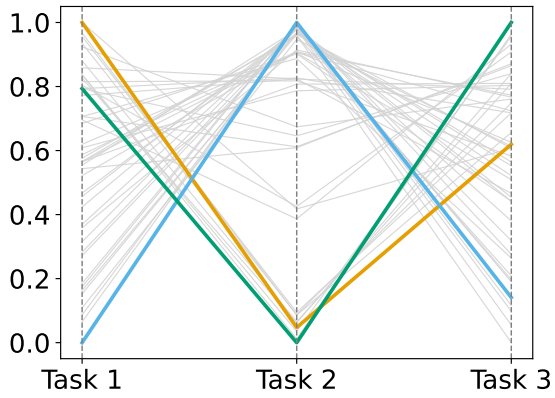

Figure 7: Pareto front in `MultiMNIST-3` in parallel coordinate plot. Each line represents a solution on the Pareto front. The highlighted lines correspond to maximum accuracies obtained on the three distinct tasks.

sured using the Hypervolume (HV) metric [23], widely adopted to evaluate both quality and diversity of the obtained Pareto front (PF). Following [13], we sample 11 solutions in the two-objective case and 66 solutions in the three-objective case for evaluation.

D.2. Hypernetwork Architecture

The hypernetwork architecture is implemented as a multi-layer perceptron (MLP) comprising three hidden layers, each with a hidden dimensionality of $w = 100$, followed by multiple linear output heads, as proposed in [41]. Each output head generates the LoRA parameters A and B for a

corresponding layer in the target network, such as LeNet for `MultiMNIST` or ResNet-18 for `UTKFace`. Notably, the size of each head is significantly smaller than the full parameterization of the target layer.

For a target layer of dimension $M \times N$, the PHN-LoRA framework outputs low-rank adaptation matrices $A \in \mathbb{R}^{M \times r}$ and $B \in \mathbb{R}^{r \times N}$, where $r \ll \min(M, N)$ is the LoRA rank. Consequently, the total number of trainable hypernetwork parameters reduces from $w \times M \times N$ (as in standard PHN) to $w \times (M \times r + r \times N)$ in PHN-LoRA. By appropriately selecting a low rank r, this results in substantial parameter savings—often by several orders of magnitude—compared to conventional hypernetwork-based approaches.

This parameter efficiency is especially evident in our experiments with LeNet and ResNet-18, where PHN-LORA yields compact hypernetworks while maintaining competitive performance. All multi-task learning models are trained using the Adam optimizer.

D.3. Experimental Configurations

D.3.1. MULTIMNIST

Network Structure Following [13], the base network is composed of a shared feature extractor (shared bottom) and two task-specific output heads. The shared bottom adopts the classic LeNet architecture [25], consisting of two convolutional layers followed by a fully connected layer. Each task-specific head comprises two fully connected layers.

For PHN-LORA, we employ the hypernetwork architecture described above. The input preference vector $\boldsymbol{r}$ is passed through a three-layer multilayer perceptron (MLP) to generate a ray embedding. For each layer of the target network, a dedicated linear head maps this embedding to the corresponding low-rank LoRA parameters A and B, which are used to adapt the target network weights.

Parameter Settings We follow the parameter configuration outlined by [13] for all baseline methods. For LORPMAN, hyperparameters are set according to the protocol in [5].

For PHN-LORA, we adopt the same hypernetwork configuration as proposed in [41]. Specifically, the window size is set to $b = 5$, the regularization coefficients for the hypervolume and diversity terms are set to $\lambda_{HVI} = \lambda_{DI} = 0.1$, and the scaling factor is fixed at $s = 4$. The LoRA rank for each adapted layer is set to $r = 8$, based on the best mean performance on the validation set. All models are trained using the Adam optimizer with a learning rate of 10^{-3} and a batch size of 256.

D.3.2. CENSUS

Network Structure Following [13], the base network consists of a shared representation module (shared bottom) and two task-specific output heads. The shared bottom is implemented as a single-layer multilayer perceptron (MLP), while each task-specific head comprises a single fully connected layer. PHN-LORA used for this dataset mirrors the same architecture as employed in the `MultiMNIST` experiments.

Parameter Settings We adopt the parameter configuration proposed by [13] for all the baselines. For PHN-LORA, we retain the original hyperparameter settings as described in the source work. Specifically, the window size is set to $b = 5$, the regularization coefficients for the hypervolume and diversity terms are set to $\lambda_{HVI} = \lambda_{DI} = 0.1$, and the scaling factor is fixed at $s = 1$. The LoRA

rank for each adapted layer is set to $r = 4$, based on the best mean performance on the validation set. All models are trained using the Adam optimizer with a learning rate of 10^{-3} and a batch size of 256.

D.3.3. MULTIMNIST-3

The network architecture follows the same design as in the `MultiMNIST` experiments, consisting of a shared encoder and three task-specific decoders. The hyperparameter settings are also consistent with those used in `MultiMNIST`, with the window size set to $b = 11$.

D.3.4. UTKFACE

Network Structure Following the setup in [13], the base network comprises a shared feature extractor and three task-specific heads. The shared bottom is implemented using a ResNet-18 architecture [18], which encodes common representations across tasks. Each task-specific head consists of a single fully connected layer responsible for producing predictions for its corresponding task. For PHN-LORA, the hypernetwork follows the same architecture as used in the `MultiMNIST` experiments.

Parameter Settings We follow the parameter configuration described in [13] for the baselines. Specifically, the window size is set to $b = 6$, the regularization coefficients for the hypervolume and diversity terms are set to $\lambda_{HVI} = \lambda_{DI} = 0.1$, and the scaling factor is fixed at $s = 2$. The LoRA rank for each adapted layer is set to $r = 4$, based on the best mean performance on the validation set. All models are optimized using the Adam optimizer with a learning rate of 10^{-3} and a batch size of 64.

D.4. Preference Sampling

For the multi-sample regularization, we deterministically sample the preference space to evenly cover the objective space. For the two-task setting, we generate preference vectors by uniformly sampling b scalar values from the interval $[0, 1]$ using `torch.linspace(0, 1, b)`. Each scalar λ defines a preference vector $[\lambda, 1 - \lambda]$ in the 2-dimensional simplex. For the three-task case, we employ the `meshzoo` library to generate an evenly distributed set of preference vectors over the simplex.

D.5. Hypervolume

The Hypervolume (HV) indicator [23] is a widely used metric in multi-objective optimization (MOO), providing a quantitative measure of the quality of a solution set in terms of its coverage of the objective space. Given a set of solutions $\mathcal{P}$ and a reference point $\boldsymbol{r}$, the HV is defined as:

$$HV(\mathcal{P}; \boldsymbol{r}) = \Lambda\left(\{\boldsymbol{q} \in \mathbb{R}^m \mid \exists \boldsymbol{p} \in \mathcal{P} : \boldsymbol{p} \succeq \boldsymbol{q} \succeq \boldsymbol{r}\}\right), \tag{5}$$

where Λ denotes the Lebesgue measure, and $\boldsymbol{p} \succeq \boldsymbol{q}$ indicates that $\boldsymbol{p}$ dominates $\boldsymbol{q}$ component-wise. The HV represents the volume of the objective space that is weakly dominated by the solutions in $\mathcal{P}$ and bounded below by the reference point $\boldsymbol{r}$.

For all datasets except *UTKFace* [57], we adopt a reference point of $[0, 0, \ldots, 0]$, where 0 denotes the smallest possible accuracy (i.e., worst-case performance) across tasks. In the case of *UTKFace*, the first objective corresponds to the Huber loss [22] rather than accuracy. To align with the loss scale and ensure a meaningful HV computation, we set the reference point to $[0.5, 0, 0]$, where 0.5 approximates the largest observed Huber loss across the experiments.

Algorithm 1: PHN-LoRA

Data: Base target parameters $\boldsymbol{\theta}_0$, hypernetwork parameters $\boldsymbol{\phi}$, window size b, batch size q, regularization coefficients $\lambda_{HVI}, \lambda_{DI}$, scaling factor s

Result: Optimized parameters $\boldsymbol{\theta}_0$ and $\boldsymbol{\phi}$

while *not converged* **do**

Sample a minibatch $\xi = \{(\boldsymbol{x}_i^1, \ldots, \boldsymbol{x}_i^m, \boldsymbol{y}_i^1, \ldots, \boldsymbol{y}_i^m)\}_{i=1}^q$;

Sample preference rays $\boldsymbol{r}^1, \ldots, \boldsymbol{r}^b$ evenly over the preference space;

for *each ray* $\boldsymbol{r}^j$ **do**

Generate LoRA adaptations for each layer: $\boldsymbol{\Delta\theta}^l(\boldsymbol{r}^j) = h^l(\boldsymbol{r}^j; \boldsymbol{\phi}), l = 1, \ldots, L$;

Compute adapted target parameters: $\boldsymbol{\theta}^l = \boldsymbol{\theta}_0^l + s\,\boldsymbol{\Delta\theta}^l(\boldsymbol{r}^j), l = 1, \ldots, L$;

Compute task-specific loss: $\mathcal{L}_{LS}$ or $\mathcal{L}_{EPO}$ based on the objective (see Eq. (2) or Eq. (3));

Compute contrastive loss $\mathcal{L}_{DI}$ for $\boldsymbol{r}^j$ based on Eq. (5);

end

Compute Hypervolume increasing loss $\mathcal{L}_{HVI}$ (see Eq. (4));

Compute total loss:

$$\mathcal{L} = \sum_{j=1}^{b} \sum_{i=1}^{m} \alpha_i^j f_i(\boldsymbol{\theta}(\boldsymbol{\alpha}^j); \xi) + \lambda_{HVI}\mathcal{L}_{HVI} + \lambda_{DI}\mathcal{L}_{DI}$$

if *current epoch* $<$ *freeze epoch* **then**

Gradient step on $\boldsymbol{\theta}_0$ and $\boldsymbol{\phi}$;

else

Gradient step only on $\boldsymbol{\phi}$;

end

end

Table 7: **MultiMNIST**: Ablation on LORA rank r and scaling factor s. Table shows mean hypervolume achieved on `MultiMNIST` for various combinations of LoRA rank r and scaling factor s. Results are averaged over three random seeds.

	$s=1$	$s=2$	$s=4$	$s=6$	$s=8$
$r=1$	0.9021	0.9089	0.9100	0.8967	0.8901
$r=2$	0.9022	0.9091	0.9180	0.9011	0.8852
$r=4$	0.9033	0.9152	0.9270	0.9103	0.9061
$r=8$	0.9122	0.9271	**0.9375**	0.9231	0.9201
$r=16$	0.9078	0.9256	0.9331	0.9309	0.9329

Table 8: Effect of freeze epoch on *UTKFace* (3 seeds).

Freeze Epoch	HV
20	0.3010 (0.001)
40	0.3015 (0.001)
60	0.3045 (0.002)
80	**0.3066** (0.003)
100	0.3044 (0.003)

Table 9: Effect of scaling factor s on *UTKFace* (3 seeds).

s	HV
0.1	0.3010 (0.001)
0.5	0.3012 (0.001)
1	0.3033 (0.001)
1.5	0.3042 (0.002)
2	**0.3066** (0.003)

Appendix E. Ablation Studies

Here, we demonstrate the effect of the different loss components along effects of the parameters i) rank (r) of the LoRA, ii) scaling factor (s) for the LoRA component, and iii) freeze epoch.

E.1. LORA Rank (r) and Scaling factor (s)

We conduct an ablation study on the LoRA rank r and scaling factor s for the PHN-LoRA method on the `MultiMNIST` benchmark. Specifically, we evaluate combinations of ranks $r \in \{1, 2, 4, 8, 16\}$ and scaling factors $s \in \{1, 2, 4, 6, 8\}$. For each (r, s) configuration, the model is trained using 3 random seeds, and we report the mean hypervolume in Table 7.

E.2. Freeze Epoch

Table 8 shows that hypervolume reaches its maximum value in the UTKFace dataset when the base network is frozen after 80 epochs. PHN-LoRA achieves the best hypervolume on the UTKFace dataset for $s = 2$ as shown in Table 9.

E.3. Multi-Sample Regularizations

We conduct an ablation study to examine the influence of the two multi-sample regularization terms introduced in Section 4.2—the hypervolume-promoting loss $\mathcal{L}_{HVI}$ and the diversity-promoting loss $\mathcal{L}_{DI}$. Specifically, we evaluate how varying the window size b, along with the regulariza-

tion coefficients λ_{HVI} and λ_{DI}, affects the performance of PHN-LoRA on the `MultiMNIST` benchmark.

To isolate the effect of each term, we construct two separate evaluations. In Table 10, we report the mean hypervolume obtained across different combinations of b and λ_{HVI}, while keeping $\lambda_{DI} = 0$. Conversely, Table 11 presents the results for varying b and λ_{DI}, with $\lambda_{HVI} = 0$. We report the results (across three seeds) by taking the combination of $b = 5$, $\lambda_{HVI} = 0.1$ and $\lambda_{DI} = 0.1$ in Table 1.

Table 10: **MultiMNIST**: Ablation on window size b and regularization coefficient λ_{HVI}. Mean hypervolume on `MultiMNIST` for different window sizes b and hypervolume regularization coefficients λ_{HVI}, averaged over three random seeds for $\lambda_{DI} = 0$.

b	$\lambda_{HVI} = 0.0$	0.1	0.5	1.0	2.0
2	0.9051	0.9139	0.9132	0.9031	0.9052
3	0.9067	0.9155	0.9199	0.9154	0.9132
4	0.9126	0.9208	0.9207	0.9163	0.9142
5	0.9223	**0.9301**	0.9264	0.9176	0.9191

Table 11: **MultiMNIST**: Ablation on window size b and regularization coefficient λ_{DI}. Mean hypervolume on `MultiMNIST` for different window sizes b and diversity regularization coefficients λ_{DI}, averaged over three random seeds for $\lambda_{HVI} = 0$.

b	$\lambda_{DI} = 0.0$	0.1	0.5	1.0	2.0
2	0.9038	0.9157	0.9141	0.9150	0.9098
3	0.9163	0.9206	0.9183	0.9181	0.9152
4	0.9258	0.9266	0.9244	0.9225	0.9223
5	0.9289	**0.9355**	0.9321	0.9290	0.9241

Table 12: Comparison of hypernetwork parameter counts between PHN and PHN-LoRA across different target network architectures. PHN-LoRA parameter counts correspond to the optimal rank r, selected based on the highest validation performance.

Model (Dataset)	**PHN (Params)**	**PHN-LoRA (Params)**	**Reduction Ratio**
MLP (Census)	12.251M	0.270M ($r = 4$)	45.4
LeNet (MultiMNIST)	2.793M	0.739M ($r = 8$)	3.8
ResNet-18 (UTKFace)	1.100B	18.0M ($r = 4$)	61.1