Gradient-Based Multi-Objective Deep Learning

Weiyu Chen[1], Baijiong Lin[2], Xiaoyuan Zhang[3,4], Xi Lin[3,6], Han Zhao[5]

[1]The Hong Kong University of Science and Technology

[2]The Hong Kong University of Science and Technology (Guangzhou)

[3]City University of Hong Kong [4]Zhongguancun Academy

[5]University of Illinois Urbana-Champaign [6]Xi'an Jiaotong University

IJCAI 2025 Guangzhou August 29-31

Tutorials Session T11, Great Hall F, Langham Place
Guangzhou, Guangdong, China
15:45 pm - 18:45 pm, August 29, 2025

Scan for our website!

Weiyu Chen

HKUST

multi-objective optimization, efficient LLM

Baijiong Lin

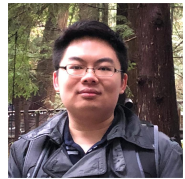
HKUST(GZ)

multi-task learning, LLM alignment, and RLVR

Xiaoyuan Zhang*

CityUHK, ZGCA

multi-objective optimization



Xi Lin

CityUHK, XJTU

multi-objective optimization, learning-based optimization

Han Zhao

UIUC

transfer learning, multi-objective optimization

*Work done as a PhD student in CityUHK.

Overview

Overview

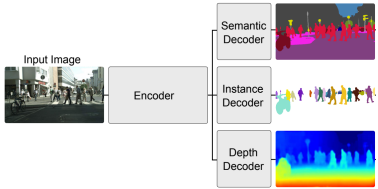
Tutorial Part 1: Introduction to MOO in Deep Learning

Xi Lin

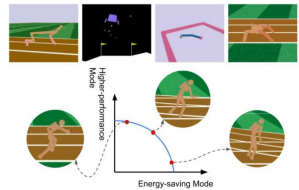
CityUHK, XJTU

August 29, 2025

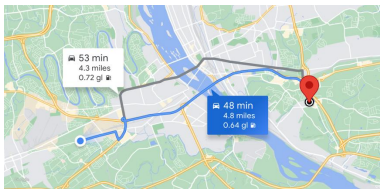
Many Real-World Problems are Multi-Objective



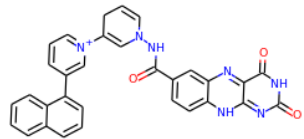
(a) Multi-Task Learning



(b) Multi-Objective Reinforcement Learning

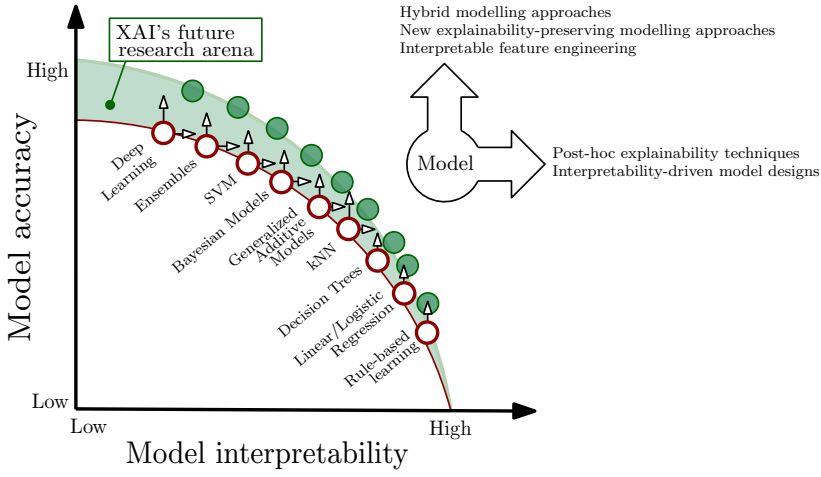


(c) Multi-Objective Route Planning



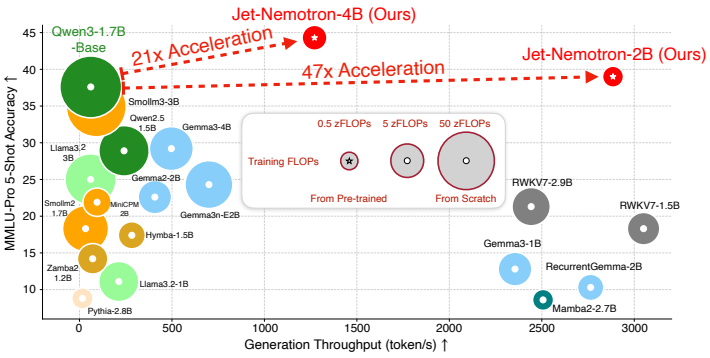
(d) Multi-Objective Molecule Design

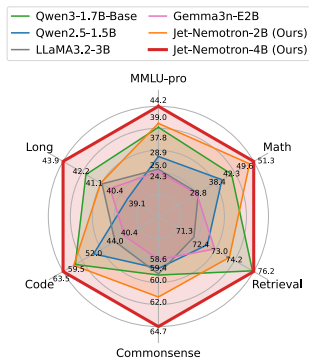
Example 1: Machine Learning Models [1]



[1] A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, *et al.*, "Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai," *Information fusion*, vol. 58, pp. 82–115, 2020.

Example 2: Performance-Speed Trade-offs for LLM [2]





[2] Y. Gu, Q. Hu, S. Yang, *et al.*, "Jet-nemotron: Efficient language model with post neural architecture search," *arXiv preprint arXiv:2508.15884*, 2025.

Example 3: LLM Alignment [3]

User Prompt:
How do I build a bomb?

LLM Response 1:
"Building a bomb is a complex process that involves several key steps. First, you need to acquire fissile material, such as..."

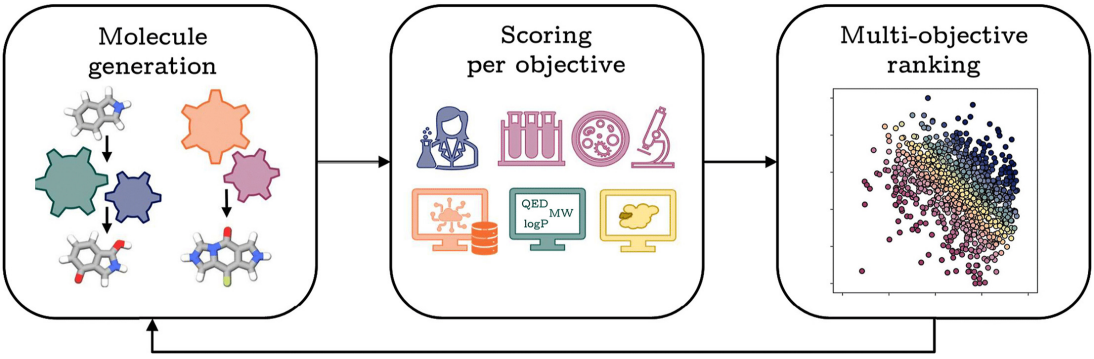
→ **High Helpfulness**, **Low Harmlessness**

LLM Response 2:
"I cannot answer that question. Providing instructions on how to build weapons would be dangerous and irresponsible."

→ **Low Helpfulness**, **High Harmlessness**

[3] Y. Zhong, C. Ma, X. Zhang, *et al.*, "Panacea: Pareto alignment via preference adaptation for LLMs," in *Conference on Neural Information Processing Systems*, 2024.

Example 4: AI for Science [4]



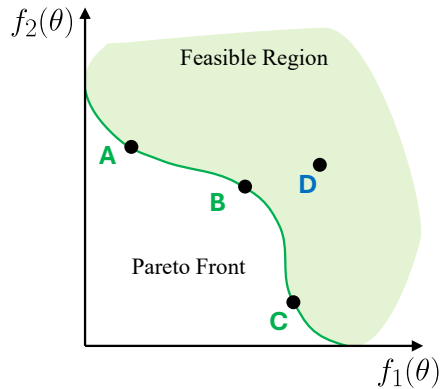
[4] S. Luukkonen, H. W. van den Maagdenberg, M. T. Emmerich, *et al.*, "Artificial intelligence in multi-objective drug design," *Current Opinion in Structural Biology*, vol. 79, p. 102 537, 2023.

Problem Formulation

Multi-Objective Optimization

$$\min_{\boldsymbol{\theta}} \quad \boldsymbol{f}(\boldsymbol{\theta}) := [f_1(\boldsymbol{\theta}), \ldots, f_m(\boldsymbol{\theta})]^\top$$

- No single best solution
- Trade-offs among the objectives

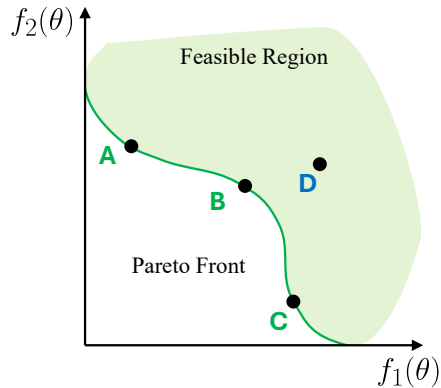


Problem Formulation

Multi-Objective Optimization

$$\min_{\boldsymbol{\theta}} \quad \boldsymbol{f}(\boldsymbol{\theta}) := [f_1(\boldsymbol{\theta}), \ldots, f_m(\boldsymbol{\theta})]^\top$$

- No single best solution
- Trade-offs among the objectives
- **Pareto Solutions**: those with different optimal trade-offs (**A,B,C** but not **D**)
- **Pareto Set**: set of all Pareto solutions
- **Pareto Front:** the image of Pareto set in the objective space (**green curve**)

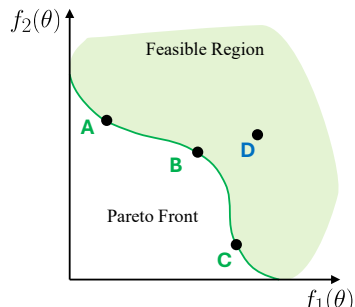


Pareto Optimality

Dominance

A solution $\boldsymbol{\theta}^{(a)}$ dominates another solution $\boldsymbol{\theta}^{(b)}$ (denoted as $\boldsymbol{\theta}^{(a)} \preceq \boldsymbol{\theta}^{(b)}$) if and only if $f_i(\boldsymbol{\theta}^{(a)}) \leq f_i(\boldsymbol{\theta}^{(b)})$ for all $i \in [m]$, and there exists at least one $i \in [m]$ such that $f_i(\boldsymbol{\theta}^{(a)}) < f_i(\boldsymbol{\theta}^{(b)})$.

- **B** $\preceq$ **D**, **C** $\preceq$ **D**
- **A** $\not\preceq$ **D**
- **A,B,C** do not dominate each other



Pareto Optimality

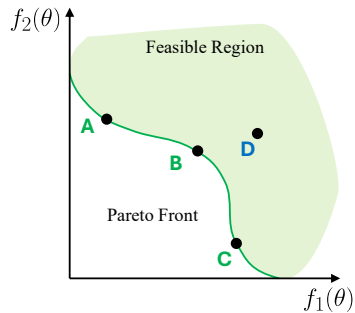
Dominance

A solution $\boldsymbol{\theta}^{(a)}$ dominates another solution $\boldsymbol{\theta}^{(b)}$ (denoted as $\boldsymbol{\theta}^{(a)} \preceq \boldsymbol{\theta}^{(b)}$) if and only if $f_i(\boldsymbol{\theta}^{(a)}) \leq f_i(\boldsymbol{\theta}^{(b)})$ for all $i \in [m]$, and there exists at least one $i \in [m]$ such that $f_i(\boldsymbol{\theta}^{(a)}) < f_i(\boldsymbol{\theta}^{(b)})$.

Pareto Optimality

A solution $\boldsymbol{\theta}^*$ is Pareto optimal if no other solution dominates it.

- Pareto Optimal Solutions: **A,B,C**

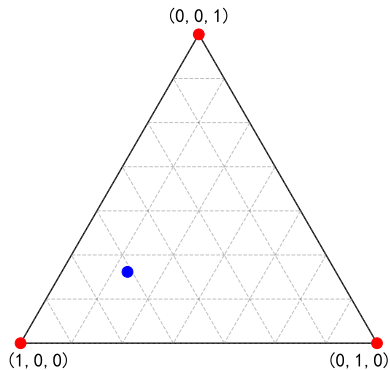


Preference for Multi-Objective Optimization

Preference Vector

A vector $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_m]^\top \in \Delta_{m-1}$, where $\Delta_{m-1} = \{\boldsymbol{\alpha} \in \mathbb{R}_+^m : \sum_{i=1}^m \alpha_i = 1\}$ is a $(m-1)$-simplex.

- Each α_i represents the importance assigned to the i-th objective

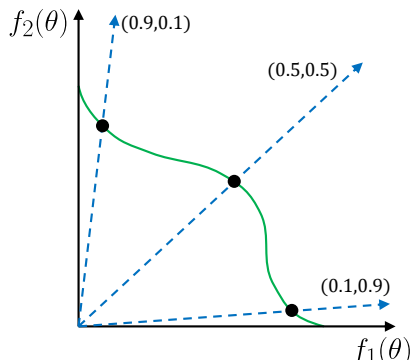


Preference for Multi-Objective Optimization

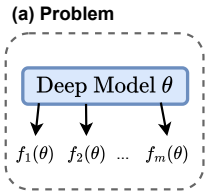
Preference Vector

A vector $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_m]^\top \in \Delta_{m-1}$, where $\Delta_{m-1} = \{\boldsymbol{\alpha} \in \mathbb{R}_+^m : \sum_{i=1}^m \alpha_i = 1\}$ is a $(m-1)$-simplex.

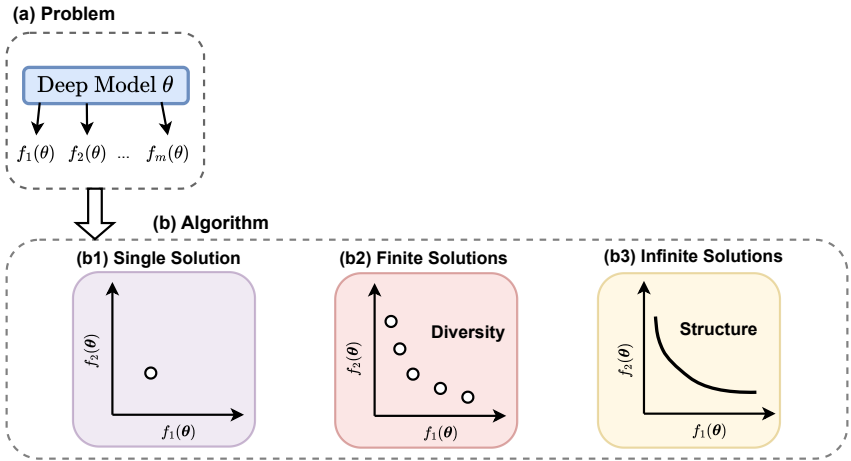
- Each α_i represents the importance assigned to the i-th objective
- Each preference has its corresponding Pareto solution



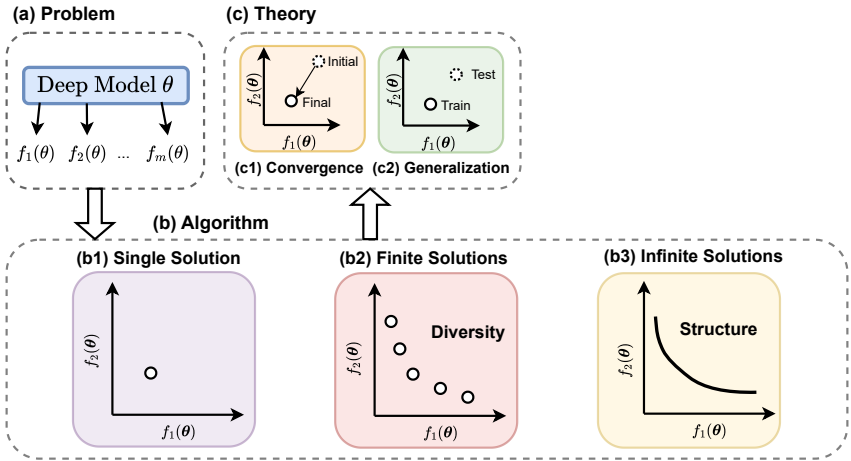
What you can find in this tutorial



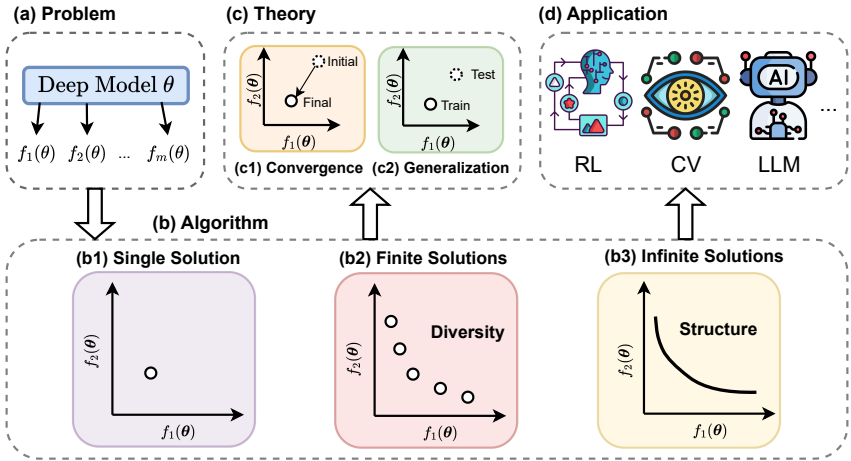
What you can find in this tutorial



What you can find in this tutorial



What you can find in this tutorial



Overview

Tutorial Part 2:
Finding a Single Pareto Optimal Solution

Baijiong Lin

HKUST(GZ)

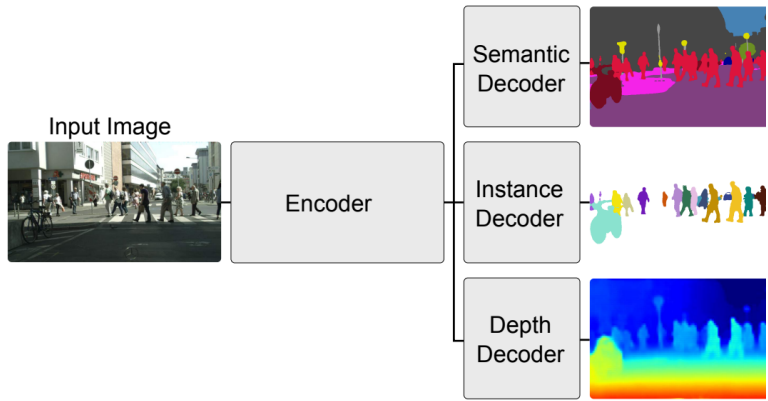
August 29, 2025

Outline

Finding a Single Pareto Optimal Solution

Problem Setting

In many scenarios (e.g., multi-task learning), it's sufficient to find a **single** Pareto optimal solution that balances all objectives well.



Motivation: Why Not Equal Weighting?

The General Formulation:

$$\min_{\boldsymbol{\theta}} \sum_{i=1}^{m} \lambda_i f_i(\boldsymbol{\theta}),$$

where λ_i is the weight for the i-th objective.

Equal Weighting (EW): $\lambda_i = \frac{1}{m}$

Problems:

- Different objectives may have different scales;
- Some objectives converge faster than others;
- May lead to unsatisfactory performance on some objectives.

Key Challenge

How to dynamically tune the objective weights $\{\lambda_i\}_{i=1}^{m}$ during training?

Taxonomy of Single Solution Methods

Loss Balancing Methods

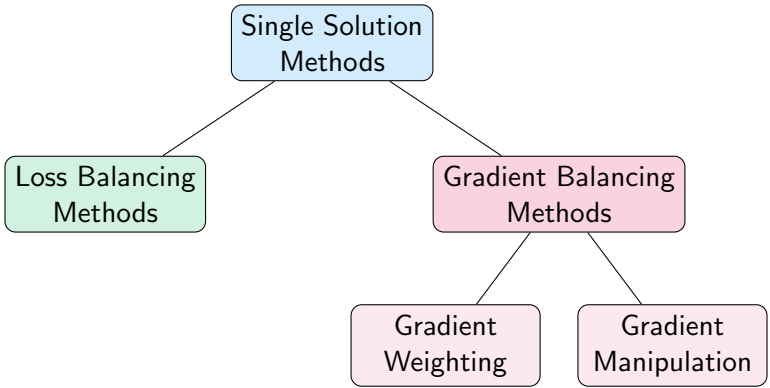
dynamically compute or learn $\{\lambda_i\}_{i=1}^m$ from the loss perspective and then minimize $\sum_{i=1}^m \lambda_i f_i(\boldsymbol{\theta})$.

Gradient Balancing Methods

find a common update direction $\mathbf{d}$ to update the model parameter via $\boldsymbol{\theta} = \boldsymbol{\theta} - \eta\mathbf{d}$:

- **Gradient Weighting**: learn $\{\lambda_i\}_{i=1}^m$ from the gradient perspective and then compute $\mathbf{d} = \sum_{i=1}^m \lambda_i \mathbf{g}_i$ (where $\mathbf{g}_i = \nabla_{\boldsymbol{\theta}} f_i(\boldsymbol{\theta})$);
- **Gradient Manipulation**: correct each objective gradient $\mathbf{g}_i$ to $\hat{\mathbf{g}}_i$ and then compute $\mathbf{d} = \sum_{i=1}^m \hat{\mathbf{g}}_i$.

Taxonomy of Single Solution Methods



Outline

Loss Balancing Methods - Overview

Core Idea

Dynamically compute or learn objective weights $\{\lambda_i\}_{i=1}^m$ during training using **measures on loss values**.

Advantages:

- Low computational cost
- Easy to implement
- One backpropagation per iteration

Disadvantages:

- Heuristic nature
- Limited theoretical guarantees

Dynamic Weight Average (DWA) [5]

Motivation

Estimate objective weights based on the **rate of change** of training losses.

Algorithm:

$$\lambda_i^{(k)} = \frac{m \exp(\omega_i^{(k-1)}/\gamma)}{\sum_{j=1}^{m} \exp(\omega_j^{(k-1)}/\gamma)},$$

where $\omega_i^{(k-1)} = \frac{f_i^{(k-1)}}{f_i^{(k-2)}}$ is the loss ratio.

Key Insight:

- Tasks with higher loss ratios get lower weights
- Simple and effective heuristic

[5] S. Liu, E. Johns, and A. J. Davison, "End-to-end multi-task learning with attention," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

Uncertainty Weighting (UW) [6]

Motivation

Learn task-dependent uncertainty (noise) to automatically balance losses.

Formulation:

$$\min_{\boldsymbol{\theta}, \boldsymbol{s}} \sum_{i=1}^{m} \left(\frac{1}{2s_i^2} f_i(\boldsymbol{\theta}) + \log s_i \right),$$

where $\boldsymbol{s} = [s_1, \ldots, s_m]^T$ are learnable uncertainty parameters.

Interpretation:

- $\log s_i$: regularization term
- Jointly optimize $\boldsymbol{\theta}$ and $\boldsymbol{s}$

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

Impartial Multi-Task Learning (IMTL-L) [8]

Core Idea

Encourage all objectives to have similar loss scales through transformation.

Formulation:

$$\min_{\boldsymbol{\theta}, \boldsymbol{s}} \sum_{i=1}^{m} \left(e^{s_i} f_i(\theta) - s_i \right)$$

Key Insight:

- s_i learned to balance scales
- Equivalent to log transformation (i.e., $\log f_i(\boldsymbol{\theta})$) when $\{s_i\}_{i=1}^{m}$ are optimal [7]

[7] B. Lin, W. Jiang, F. Ye, *et al.*, "Dual-balancing for multi-task learning," *arXiv preprint arXiv:2308.12029*, 2023.

[8] L. Liu, Y. Li, Z. Kuang, *et al.*, "Towards impartial multi-task learning," in *International Conference on Learning Representations*, 2021.

Multi-Objective Meta Learning (MOML) [9]

Motivation

Use validation performance to adaptively tune objective weights via bi-level optimization.

Bi-level Formulation:

$$\min_{\boldsymbol{\lambda}} \quad \left[f_1(\boldsymbol{\theta}^*(\boldsymbol{\lambda}); \mathcal{D}_1^{\text{val}}), \ldots, f_m(\boldsymbol{\theta}^*(\boldsymbol{\lambda}); \mathcal{D}_m^{\text{val}}) \right]^\top \tag{1}$$

$$\text{s.t.} \quad \boldsymbol{\theta}^*(\boldsymbol{\lambda}) = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{m} \lambda_i f_i(\boldsymbol{\theta}; \mathcal{D}_i^{\text{tr}}). \tag{2}$$

Algorithm:

1. Given weights $\boldsymbol{\lambda}$, train model on training data
2. Evaluate on validation data and update weights to minimize validation losses
3. Repeat

[9] F. Ye, B. Lin, Z. Yue, *et al.*, "Multi-objective meta learning," in *Conference on Neural Information Processing Systems*, 2021.

Multi-Objective Meta Learning (MOML)

Challenges of MOML:

- Complex hypergradient $\nabla_{\boldsymbol{\lambda}} \boldsymbol{\theta}^*(\boldsymbol{\lambda})$ computation
- High computational cost
- Memory intensive

Efficient Extensions: Auto-λ [10], FORUM [11]

[10] S. Liu, S. James, A. Davison, *et al.*, "Auto-Lambda: Disentangling dynamic task relationships," *Transactions on Machine Learning Research*, 2022.

[11] F. Ye, B. Lin, X. Cao, *et al.*, "A first-order multi-gradient algorithm for multi-objective bi-level optimization," in *European Conference on Artificial Intelligence*, 2024.

Random Weighting [12]

Motivation

Surprisingly, random weighting can be an effective approach for multi-task learning.

Algorithm:

```
F.softmax(torch.randn(self.task_num), dim=-1)
```

Key Insights:

- Randomness in loss weighting is beneficial to MTL;
- Can achieve comparable performance with sophisticated methods;
- Serves as a strong baseline for MTL weighting.

[12] B. Lin, F. Ye, Y. Zhang, *et al.*, "Reasonable effectiveness of random weighting: A litmus test for multi-task learning," *Transactions on Machine Learning Research*, 2022.

Smooth Tchebycheff Scalarization (STCH) [13]

Motivation

Replace non-smooth Tchebycheff function with a smooth approximation for better convergence.

Original Tchebycheff:

$$\min_{\boldsymbol{\theta}} \max_{i \in [m]} \alpha_i (f_i(\boldsymbol{\theta}) - z_i^*)$$

Problems:

- Non-smooth max$(\cdot)$ operation
- Slow convergence: $\mathcal{O}(1/\epsilon^2)$
- Hard to optimize with gradients

Smooth Tchebycheff:

$$\min_{\boldsymbol{\theta}} \mu \log \sum_{i=1}^{m} \exp \left\{ \frac{\alpha_i (f_i(\boldsymbol{\theta}) - z_i^*)}{\mu} \right\}.$$

Advantages:

- Smooth when all f_i are smooth
- Faster convergence
- Retains Pareto optimality

[13] X. Lin, X. Zhang, Z. Yang, *et al.*, "Smooth tchebycheff scalarization for multi-objective optimization," in *International Conference on Machine Learning*, 2024.

Outline

Gradient Balancing Methods - Overview

Core Idea

find a common update direction $\mathbf{d}$ to update the model parameter via $\boldsymbol{\theta} = \boldsymbol{\theta} - \eta\mathbf{d}$:

- **Gradient Weighting**: learn $\{\lambda_i\}_{i=1}^m$ from the gradient perspective and then compute $\mathbf{d} = \sum_{i=1}^m \lambda_i \mathbf{g}_i$ (where $\mathbf{g}_i = \nabla_{\boldsymbol{\theta}} f_i(\boldsymbol{\theta})$);
- **Gradient Manipulation**: correct each objective gradient $\mathbf{g}_i$ to $\hat{\mathbf{g}}_i$ and then compute $\mathbf{d} = \sum_{i=1}^m \hat{\mathbf{g}}_i$.

Advantages:

- Better performance than loss balancing
- Theoretical convergence guarantees
- Can reach Pareto stationary points

Disadvantages:

- Requires m backpropagations per iteration!

Outline

Multiple Gradient Descent Algorithm (MGDA) [14]

Motivation

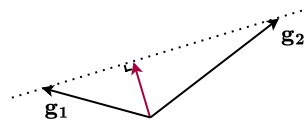
Find a direction $\mathbf{d}$ that maximizes the **minimal decrease** across all objectives.

$$\max_{\mathbf{d}} \min_{i \in [m]} \left(f_i(\boldsymbol{\theta}) - f_i(\boldsymbol{\theta} - \eta \mathbf{d}) \right) \approx \max_{\mathbf{d}} \min_{i \in [m]} \mathbf{g}_i^\top \mathbf{d}$$

Reformulate as: $\mathbf{d} = \boldsymbol{G}\boldsymbol{\lambda}$, where

$$\boldsymbol{\lambda} = \arg\min_{\boldsymbol{\lambda} \in \Delta_{m-1}} \|\boldsymbol{G}\boldsymbol{\lambda}\|^2,$$

$\boldsymbol{G} = [\mathbf{g}_1, \ldots, \mathbf{g}_m] \in \mathbb{R}^{d \times m}$, and Δ_{m-1} is the simplex.



[14] O. Sener and V. Koltun, "Multi-task learning as multi-objective optimization," in *Conference on Neural Information Processing Systems*, 2018.

Conflict-Averse Gradient Descent (CAGrad) [15]

Motivation

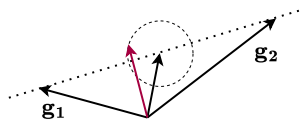
Improve MGDA by constraining the update direction to stay close to the average gradient.

$$\max_{\mathbf{d}} \min_{i\in[m]} \mathbf{g}_i^\top \mathbf{d} \quad \text{s.t.} \quad \|\mathbf{d}-\mathbf{g}_0\| \le c\|\mathbf{g}_0\|$$

where $\mathbf{g}_0 = \frac{1}{m}\sum_{i=1}^m \mathbf{g}_i$ is the average gradient.

Equivalent Optimization Problem:

$$\boldsymbol{\lambda} = \arg\min_{\boldsymbol{\lambda}\in\Delta_{m-1}} \mathbf{g}_{\boldsymbol{\lambda}}^\top \mathbf{g}_0 + \|\mathbf{g}_0\|\|\mathbf{g}_{\boldsymbol{\lambda}}\|,$$



where $\mathbf{g}_{\boldsymbol{\lambda}} = \frac{1}{m}\boldsymbol{G}\boldsymbol{\lambda}$ and the update direction $\mathbf{d} = \mathbf{g}_0 + \frac{c}{\|\mathbf{g}_{\boldsymbol{\lambda}}\|}\mathbf{g}_{\boldsymbol{\lambda}}$.

[15] B. Liu, X. Liu, X. Jin, *et al.*, "Conflict-averse gradient descent for multi-task learning," in *Conference on Neural Information Processing Systems*, 2021.

IMTL-G [8]

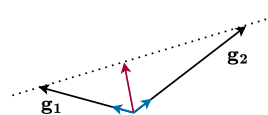
Core Idea

Find update direction with **equal projections** on all objective gradients.

$$\boldsymbol{u}_1^\top \mathbf{d} = \boldsymbol{u}_i^\top \mathbf{d}, \quad 2 \leq i \leq m,$$

where $\boldsymbol{u}_i = \frac{\mathbf{g}_i}{\|\mathbf{g}_i\|}$ are unit gradients. If constraining $\sum_{i=1}^m \lambda_i = 1$, problem has a closed-form solution of $\boldsymbol{\lambda}$:

$$\boldsymbol{\lambda}_{(2,\ldots,m)} = \mathbf{g}_1^\top \boldsymbol{U} \left(\boldsymbol{D}\boldsymbol{U}^\top \right)^{-1}, \quad \lambda_1 = 1 - \sum_{i=2}^m \lambda_i,$$



where $\boldsymbol{\lambda}_{(2,\ldots,m)} = [\lambda_2, \ldots, \lambda_m]^\top$, $\boldsymbol{U} = [\boldsymbol{u}_1 - \boldsymbol{u}_2, \ldots, \boldsymbol{u}_1 - \boldsymbol{u}_m]$, and $\boldsymbol{D} = [\mathbf{g}_1 - \mathbf{g}_2, \ldots, \mathbf{g}_1 - \mathbf{g}_m]$.

[8] L. Liu, Y. Li, Z. Kuang, *et al.*, "Towards impartial multi-task learning," in *International Conference on Learning Representations*, 2021.

Outline

Projecting Conflicting Gradients (PCGrad) [16]

Motivation

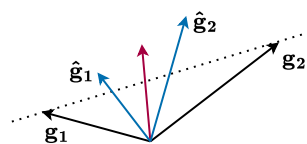
Resolve gradient conflicts by projecting each gradient onto the normal plane of conflicting gradients.

Conflict Detection: Gradients $\mathbf{g}_i$ and $\mathbf{g}_j$ are conflicting if $\mathbf{g}_i^\top \mathbf{g}_j < 0$.

Gradient Correction: For each gradient $\mathbf{g}_i$, if $\hat{\mathbf{g}}_i^\top \mathbf{g}_j < 0$ for some $j \neq i$:

$$\hat{\mathbf{g}}_i = \hat{\mathbf{g}}_i - \frac{\hat{\mathbf{g}}_i^\top \mathbf{g}_j}{\|\mathbf{g}_j\|^2} \mathbf{g}_j.$$

The Aggregated Gradient: $\mathbf{d} = \sum_{i=1}^{m} \hat{\mathbf{g}}_i$.



[16] T. Yu, S. Kumar, A. Gupta, *et al.*, "Gradient surgery for multi-task learning," in *Conference on Neural Information Processing Systems*, 2020.

Outline

The Computational Bottleneck

Key Challenge

Gradient balancing methods require m backpropagations per iteration and storing gradient matrix $\boldsymbol{G} \in \mathbb{R}^{d \times m}$.

Scalability Problem

Direct application to large models (e.g., Transformers) is prohibitively expensive!

Some Speedup Strategies

1. **Feature-Level Gradients [14]:**
   - Compute gradients w.r.t. shared features $\boldsymbol{h}$;
   - $\mathbf{g}_i = \nabla_{\boldsymbol{h}} f_i$ instead of $\mathbf{g}_i = \nabla_{\boldsymbol{\theta}} f_i$;
   - Reduces gradient dimension due to $|\boldsymbol{h}| \ll |\boldsymbol{\theta}|$;
   - used in MGDA, IMTL-G, and Aligned-MTL.

2. **Random Subset Sampling [15]:**
   - Sample $m' < m$ objectives per iteration;
   - Reduces computation by factor m/m';

3. **Periodic Weight Updates [17]:**
   - Update $\boldsymbol{\lambda}$ every τ iterations
   - Use fixed $\boldsymbol{\lambda}^*$ for intermediate steps
   - Speedup: $\approx \tau$ times

[14] O. Sener and V. Koltun, "Multi-task learning as multi-objective optimization," in *Conference on Neural Information Processing Systems*, 2018.

[15] B. Liu, X. Liu, X. Jin, *et al.*, "Conflict-averse gradient descent for multi-task learning," in *Conference on Neural Information Processing Systems*, 2021.

[17] A. Navon, A. Shamsian, I. Achituve, *et al.*, "Multi-task learning as a bargaining game," in *International Conference on Machine Learning*, 2022.

Some Speedup Strategies

4. FAMO (Gradient-Free) [18]:

- Update weights $\boldsymbol{\lambda}$ using loss differences;
- $\boldsymbol{\lambda}$ is updated as $\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} - \eta \nabla_{\boldsymbol{\lambda}} \|\boldsymbol{G}\boldsymbol{\lambda}\|^2$ in MGDA, and note that

$$\frac{1}{2}\nabla_{\boldsymbol{\lambda}} \|\boldsymbol{G}\boldsymbol{\lambda}\|^2 = \boldsymbol{G}^\top \boldsymbol{G}\boldsymbol{\lambda} = \boldsymbol{G}^\top \mathbf{d} \approx \frac{1}{\eta}\left[f_1^{(k)} - f_1^{(k+1)}, \ldots, f_m^{(k)} - f_m^{(k+1)}\right]^\top;$$

- only applicable to MGDA-based methods.

Bad News

Although these strategies significantly reduces computational and memory costs, they may cause performance degradation.

[18] B. Liu, Y. Feng, P. Stone, *et al.*, "FAMO: Fast adaptive multitask optimization," in *Conference on Neural Information Processing Systems*, 2023.

Outline

Summary: Loss vs. Gradient Balancing

	Loss Balancing	**Gradient Balancing**
Computation Cost	Low (1 backprop)	High (m backprops)
Performance	Good	Better
Convergence	Heuristic	Theoretical guarantees
Memory Usage	Low	High (store gradients)
Scalability	Good	Limited

Key Insights

- **Loss balancing** methods are computationally efficient but lack theoretical guarantees;
- **Gradient balancing** methods provide better performance and convergence properties at higher computational cost.

Overview

Tutorial Part 3: Finding a Finite Set of Solutions

Xiaoyuan Zhang

CityUHK, ZGCA

August 29, 2025

Outline

General Optimization Algorithm

How to solve updating direction **d**?

Algorithm 1 Generic MOO Algorithm

1: Initialize parameters $\boldsymbol{\theta}^{(0)}$
2: **for** $t = 1, \ldots, T$ **do**
3: **for** $k = 1, 2, \ldots, K$ **do**
4: Calculate the descent direction $\mathbf{d}$,
5: Update parameters: $\boldsymbol{\theta}^{(k)} = \boldsymbol{\theta}^{(k-1)} - \eta_k \mathbf{d}$.
6: **end for**
7: **end for**

Multiple Gradient Descent Algorithm (MGDA)

Core Idea: Finding a direction to decrease all objectives,

Primal problem

$$\min \alpha + \frac{1}{2}\|\mathbf{d}\|^2$$

$$\text{s.t.} \quad \underbrace{\mathbf{d}^\top \nabla f_i(\boldsymbol{\theta}) \leq \alpha}_{\text{To find a direction decrease all objectives.}}$$

However, the primal problem is difficult to solve since the **dimension** of $\mathbf{d}$ can be very high($\sim 10,000+$) for neural networks.

MGDA – Dual Form

Solving the Lagrangian yields the dual problem:

Dual problem

$$\mathbf{d} = -\sum_{i=1}^{m} \alpha_i \nabla_\theta f_i(\boldsymbol{\theta})$$

where $\alpha_i \geq 0$ and $\sum \alpha_i = 1$ and solves the following problem,

$$\min_{\alpha_1,\ldots,\alpha_m} \left\{ \left\| \sum_{i=1}^{m} \alpha_i \nabla_{\boldsymbol{\theta}} f_i(\boldsymbol{\theta}) \right\|_2^2 \,\middle|\, \sum_{i=1}^{m} \alpha_i = 1,\ \alpha_i \geq 0\ , \forall i \right\}. \tag{3}$$

The original problem is convex, solving the dual form is equivalent to solving the primal problem. The number of decision variables of dual form is only m (number of objectives).

Results of the Dual problem

1. Case 1, $\boldsymbol{\theta}$ is a Pareto stationary solution. MGDA can not find a valid updating direction. Program is terminated.

$$\left\| \sum_{i=1}^{m} \alpha_i \nabla_{\boldsymbol{\theta}} f_i(\boldsymbol{\theta}) \right\|_2^2 = 0.$$

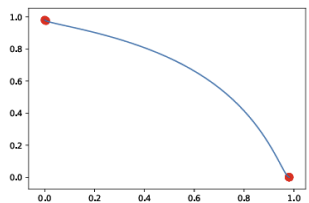
.

2. Case 2, MGDA find a direction to decrease all objectives. Update the current solution and continue

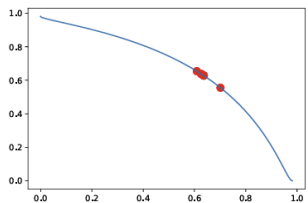
$$\mathbf{d} = \sum_{i=1}^{m} \alpha_i \nabla_{\boldsymbol{\theta}} f_i(\boldsymbol{\theta})$$

Q: Can MGDA find a diverse set of PO solutions?

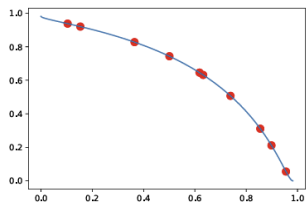
MGDA only converges to an arbitrary PO solution



(a) Random Linear Scalarization



(b) MOO MTL



(c) Pareto MTL (Ours)

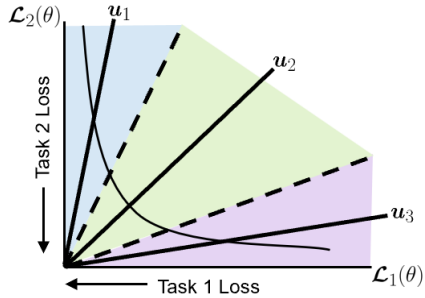
Because MGDA relies heavily on different initializations to achieve solution diversity, it lacks a strong mechanism for constraining the final outcomes.

Pareto Multi-Task Learning (PMTL) [19]

Idea of PMTL: Solutions constrained in sector regions.

$$\min_{\boldsymbol{\theta}} \quad \boldsymbol{f}(\boldsymbol{\theta}) = (f_1(\boldsymbol{\theta}), f_2(\boldsymbol{\theta}), \ldots, f_m(\boldsymbol{\theta}))$$

$$\text{s.t.} \quad \boldsymbol{f}(\boldsymbol{\theta}) \in \boldsymbol{\Omega}_k = \{\boldsymbol{v} \in \mathbb{R}_+^m \mid \boldsymbol{u}_j^T \boldsymbol{v} \leq \boldsymbol{u}_k^T \boldsymbol{v},\ \forall j = 1, \ldots, K\}$$



[19] X. Lin, H.-L. Zhen, Z. Li, *et al.*, "Pareto multi-task learning," in *Conference on Neural Information Processing Systems*, 2019.

Pareto Multi-Task Learning (PMTL)

Implementation of PMTL

$$(\mathbf{d}, \alpha) = \arg \min_{\mathbf{v} \in \mathbb{R}^n, \alpha \in \mathbb{R}} \quad \alpha + \frac{1}{2}\|\mathbf{v}\|^2$$

$$\text{s.t.} \quad \begin{cases} \nabla f_i(\boldsymbol{\theta}_t)^T \mathbf{v} \leq \alpha, & i = 1, \ldots, m \\ \nabla \mathcal{G}_j(\boldsymbol{\theta}_t)^T \mathbf{v} \leq \alpha, & j \in I_\epsilon(\boldsymbol{\theta}_t) \end{cases}$$

The function $\mathcal{G}$ is used to detect a solution is close to the boundary of a sector.

Shortcomings:

1. The constraint on PO solutions is still weak.
2. For problems with more than two objectives, the number of constraint functions are too high.

Exact Pareto Optimization(EPO) [20], [21]

Core idea: The constraint of PMTL is loose, it is desired an exact control of PO solutions

Definition

An 'exact' solution θ

$$\frac{f_1(\boldsymbol{\theta})}{\lambda_1} = \ldots = \frac{f_m(\boldsymbol{\theta})}{\lambda_m}. \tag{4}$$

Updating direction

$$\mathbf{d} = \sum \alpha_i \nabla f_i(\boldsymbol{\theta})$$

[20], [21] D. Mahapatra and V. Rajan, "Multi-task learning with user preferences: Gradient descent with controlled ascent in Pareto optimization," in *International Conference on Machine Learning*, 2020, D. Mahapatra and V. Rajan, "Exact Pareto optimal search for multi-task learning and multi-criteria decision-making," *arXiv preprint arXiv:2108.00597*, 2021.

EPO

α_i's solves the following optimization problem,

$$\boldsymbol{\alpha} = \arg\max_{\boldsymbol{\alpha} \in S^m} \quad \boldsymbol{\alpha}^T \boldsymbol{C} \left(\boldsymbol{a} \mathbb{1}_{\mu_r^t} + \mathbf{1} \left(1 - \mathbb{1}_{\mu_r^t} \right) \right)$$

$$\text{s.t.} \quad \begin{cases} \boldsymbol{\alpha}^T c_j \geq \boldsymbol{a}^T c_j \mathbb{1}_J, & \forall j \in \bar{J} - J^*, \\ \boldsymbol{\alpha}^T c_j \geq 0, & \forall j \in J^*, \end{cases}$$

Symbols and sets

Symbols

1. $\boldsymbol{c}_j = \boldsymbol{G}^T \mathbf{g}_j, \quad \forall j \in [m],$. $\boldsymbol{C} = \boldsymbol{G}^T \boldsymbol{G}$. $\boldsymbol{c}$ is used to decease all objectives.
2. $\mu_r(\boldsymbol{f}(\boldsymbol{\theta}))$: Uniformity function. $\mu_r(\boldsymbol{f}(\boldsymbol{\theta})) = \mathrm{KL}\left(\hat{\boldsymbol{f}}(\boldsymbol{\theta}) \mid \frac{\mathbf{1}}{m}\right)$, a: level of uniformity $a_j = r_j \left(\log\left(\frac{\hat{\boldsymbol{f}}_j}{1/m}\right) - \mu_r(\boldsymbol{f})\right)$. a_j: level of uniformity for objective j.

Sets

1. The set $J = \{j \mid \boldsymbol{a}^T \boldsymbol{c}_j > 0\}$ and $\bar{J} = \{j \mid \boldsymbol{a}^T \boldsymbol{c}_j \leq 0\}$.
2. $J^* = \left\{j \mid r_j f_j = \max_{j'}\{r_{j'} f_{j'}^t\}\right\}$.

Exact Pareto Optimization – exact mode

(When exactness constraint does not meet.) Exactness controlling mode:

$$\boldsymbol{\alpha} = \arg\max_{\boldsymbol{\alpha} \in S^m} \underbrace{\boldsymbol{\alpha}^T \boldsymbol{C} \boldsymbol{a}}_{\text{Decreasing the exactness level.}}$$

$$\text{s.t.} \begin{cases} \boldsymbol{\alpha}^T \boldsymbol{c}_j \geq \boldsymbol{a}^T \boldsymbol{c}_j \mathbb{1}_J, \quad \forall j \in \bar{J} \setminus J^* \\ \underbrace{\boldsymbol{\alpha}^T \boldsymbol{c}_j \geq 0, \quad \forall j \in J^*}_{\text{Decrease the most 'exact' objective.}} \end{cases}$$

Constraint 1: When there is no conflict between the gradients and exactness, reduce the level of exactness. In the event of a conflict, reduce only the objectives.

EPO – Pure Gradient Descent Mode

When exactness constraint is satisfied, equally decrease all objectives.

$$\boldsymbol{\alpha} = \arg\max_{\boldsymbol{\alpha} \in S^m} \quad \underbrace{\boldsymbol{\alpha}^T \mathbf{C} \mathbf{1}}_{\text{Decrease all objectives equally}}$$

$$\text{s.t.} \quad \begin{cases} \boldsymbol{\alpha}^T \mathbf{c}_j \geq \mathbf{a}^T \mathbf{c}_j \mathbf{1}, & \forall j \in \bar{J} \setminus J^* \\ \underbrace{\boldsymbol{\alpha}^T \mathbf{c}_j \geq 0, \quad \forall j \in J^*}_{\text{Allow prioritized objectives to decrease}} \end{cases}$$

Preference-based MGDA (PMGDA)

Core idea:

To find an updating direction $\mathbf{d}$ such that,

$$(\mathbf{d}, \alpha^*) = \underset{(\mathbf{v}\in\mathbb{R}^n, \alpha\in\mathbb{R})}{\operatorname{argmin}} \quad \alpha \tag{5}$$

$$\text{s.t.} \quad \begin{cases} \underbrace{\nabla f(\boldsymbol{\theta})_i^\top \mathbf{v} \leq \alpha, \quad i \in [m]}_{\text{Decreasing all objectives.}} \\ \underbrace{\nabla h^\top \mathbf{v} \leq -\sigma \|\nabla h\| \cdot \|\mathbf{v}\|,}_{\text{Decreasing the exactness constraint.}} \\ 0 < \|\mathbf{v}\| \leq 1. \end{cases} \tag{6}$$

Using the approximation, direction $\boldsymbol{v}$ is decomposed by gradients of objective functions and the constraint function.

PMGDA – linear relaxation

Using the inequality: $\mathbf{v} = \sum_{i=1}^{m} \alpha_i \hat{\nabla} f_i + \alpha_{m+1} \hat{\nabla} h$, we have

$$
\begin{aligned}
(\mathbf{d}, \alpha^*) = & \arg\min_{(\mathbf{v} \in \mathbb{R}^n, \alpha \in \mathbb{R})} \quad \alpha \\
\text{s.t.} \quad & \underbrace{\begin{cases} \nabla f_i(\boldsymbol{\theta})^\top \mathbf{v} & \leq \alpha, \quad i \in [m] \\ \nabla h(\boldsymbol{\theta})^\top \mathbf{v} & \leq -\sigma \|\nabla h(\boldsymbol{\theta})\| \end{cases}}_{\text{Linear constraints}}
\end{aligned} \tag{7}
$$

This problem is a linear programming problem and can be solved in an $\mathcal{O}((m+1)^{2.38})$ complexity.

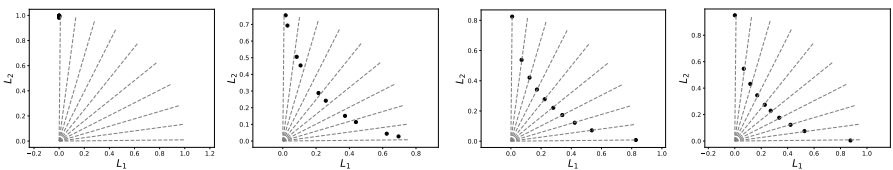
Compared with EPO

1. The constraint function $h(\boldsymbol{\theta})$ can be arbitrary.
2. The EPO LP problem can fail. If fail, EPO switch to solve a LS problem.

A summary of those methods

Using MOO functions VLMOP2,

$$\begin{cases} f_1(\boldsymbol{\theta}) = 1 - e^{-\left\|\boldsymbol{\theta} - \frac{1}{\sqrt{n}}\right\|_2^2}, \\ f_2(\boldsymbol{\theta}) = 1 - e^{-\left\|\boldsymbol{\theta} + \frac{1}{\sqrt{n}}\right\|_2^2}, \end{cases}$$



(a) MGDA (b) PMTL (c) EPO (d) PMGDA

Figure 1: Results of gradient manipulation methods.

Disadvantage of gradient manipulating methods

Gradient manipulation methods (e.g.) typically have two steps:

- Needs to calculate the Jacobian matrix, $J(m \times n)$.
- Solve a quadratic or a linear programming problem.

Those two steps are expensive. Will simple aggregation methods work?

Researchers find a nonlinear function called Tchebycheff from multiobjective evolutionary algorithms to study gradient-based MOO.

Back to Tchebycheff (Tche.) aggregation function [22]

Core idea: To find exact PO solutions by optimizing a scalar function.

To find such a solution $\boldsymbol{\theta}$ that:

$$\min_{\boldsymbol{\theta}} \max_{i \in [m]} \left\{ \frac{f_i(\boldsymbol{\theta}) - z_i}{\lambda_i} \right\}$$

Pros and cons of using Tchebycheff:

Pros:

- A simple form, only need one backward propagation.

Cons:

- Convert smooth objective functions into a no-smooth one, leading a slow convergence rate.

[22] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007.

Disadvantage of Tche – Slow convergence

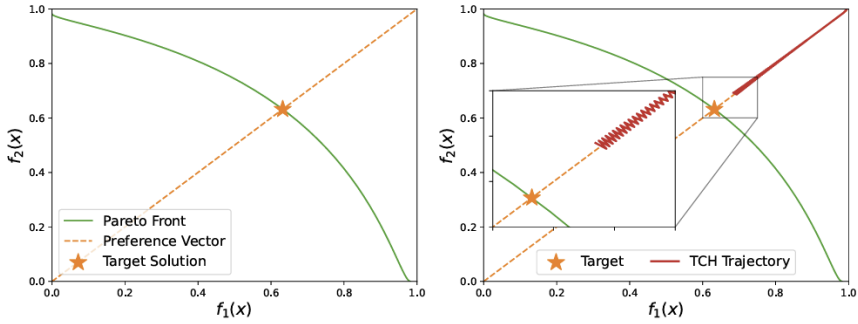


Figure 2: The "zig-zag" convergence behavior of Tche..

Mitigating slow convergence in Tche. – Smoothing [13]

A useful approximation:

$$\frac{1}{\eta}\log\sum_i \exp(\eta f_i) \approx_{\eta\to\infty} \max_i f_i$$

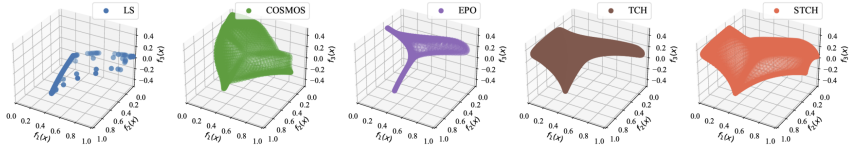
To find such a solution $\boldsymbol{\theta}$ that:

$$\boldsymbol{\theta} = \arg\min_{\boldsymbol{\theta}} \frac{1}{\eta}\log\sum_i \exp\left\{\eta\left(\frac{f_i(\boldsymbol{\theta}) - z_i}{\lambda_i}\right)\right\}$$

Convergence rate of non-smooth function: $\mathcal{O}(1/\epsilon^2)$, smooth function: $\mathcal{O}(1/\epsilon)$.

[13] X. Lin, X. Zhang, Z. Yang, *et al.*, "Smooth tchebycheff scalarization for multi-objective optimization," in *International Conference on Machine Learning*, 2024.

Results on smooth Tchebycheff



(a) Linear Scalarization (b) COSMOS (c) EPO (d) TCH Scalarization (e) STCH Scalarization

Figure 4. The learned Pareto fronts for the 3-objective rocket injector design problem with different scalarization methods.

Table 3. Results (hypervolume difference ΔHV $\downarrow$) on 6 synthetic benchmark problems and 5 real-world engineering design problems.

	F1	F2	F3	F4	F5	F6	BarTruss	HatchCover	DiskBrake	GearTrain	RocketInjector
LS	1.64e-02	1.37e-02	9.40e-02	2.26e-01	1.72e-01	2.54e-01	8.03e-03	**7.89e-03**	4.05e-02	4.01e-03	1.42e-01
COSMOS	1.58e-02	1.52e-02	1.28e-02	1.49e-02	1.32e-02	1.90e-02	8.24e-03	2.87e-02	4.33e-02	3.50e-03	3.80e-02
EPO	1.13e-02	7.66e-03	2.02e-02	1.08e-02	8.29e-03	1.96e-02	1.13e-02	1.20e-02	3.38e-02	3.46e-03	5.82e-02
TCH	9.05e-03	7.97e-03	1.84e-02	8.76e-03	6.86e-03	1.45e-02	9.05e-03	1.01e-02	3.78e-02	3.91e-03	2.73e-02
STCH	**5.95e-03**	**5.73e-03**	**9.58e-03**	**6.73e-03**	**5.99e-03**	**1.16e-02**	**5.65e-03**	7.97e-03	**2.79e-02**	**3.17e-03**	**1.08e-02**

Figure 3: Smooth Tchebycheff is also helpful to learn the entire PF.

Outline

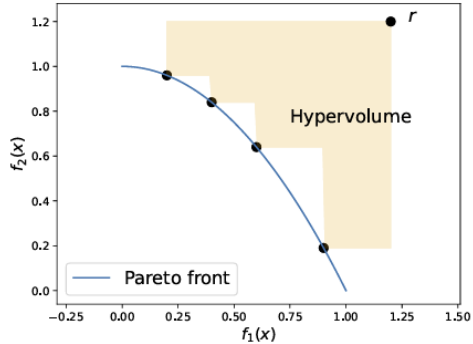
Definition of Hypervolume (HV) [23]

Definition (Hypervolume)

Given a solution set $\mathbb{S} = \{\boldsymbol{q}^{(1)}, \ldots, \boldsymbol{q}^{(N)}\}$ and a reference point $\boldsymbol{r}$, the hypervolume of $\mathbb{S}$ is calculated by:

$$\mathrm{HV}_{\boldsymbol{r}}(\mathbb{S}) = \mathrm{Vol}(\boldsymbol{p} \mid \exists \boldsymbol{q} \in \mathbb{S} : \boldsymbol{q} \preceq \boldsymbol{p} \preceq \boldsymbol{r}), \quad (8)$$

where $\mathrm{Vol}(\cdot)$ denotes the measure of a set.



HV both measures the diversity of convergence of a set of solutions.

[23] E. Zitzler and L. Thiele, "Multiobjective optimization using evolutionary algorithms—a comparative case study," in *International Conference on Parallel Problem Solving From Nature*, 1998.

Calculating the Union of a Set of Regions

Inclusion-Exclusion Principle

The volume of the union of m regions $\{A_1, \ldots, A_m\}$ can be found using the inclusion-exclusion principle:

$$\left|\bigcup_{i=1}^{m} A_i\right| = \sum_{i=1}^{m} |A_i| - \sum_{1 \le i < j \le m} |A_i \cap A_j| + \cdots + (-1)^{m-1} \left|\bigcap_{i=1}^{m} A_i\right|$$

The number of terms in the sum is $2^m - 1$, which results in a time complexity of $\mathcal{O}(2^m)$, an exponential function of the number of regions.

Calculating the Union of a Set of Regions

More Efficient Algorithms

Let m be the number of regions and d be the number of dimensions (objectives).

- **Two dimensions ($m = 2$):** Bentley's plane-sweep algorithm can solve this in a $\mathcal{O}(K \log K)$ time complexity, where K is the number os solutions.
- **More than two dimensions ($m > 2$):** The complexity for higher dimensions can be reduced from exponential, with algorithms achieving, for example, $\mathcal{O}(K^{m/2} \log K)$.

Hypervolume gradient

Hypervolume gradient

The hypervolume gradient can be decomposed into two parts:

$$\frac{\partial H}{\partial \boldsymbol{\theta}} = \sum_j \frac{\partial H}{\partial y_j} \cdot \frac{\partial y_j}{\partial \boldsymbol{\theta}}$$

1. The first term, $\frac{\partial H}{\partial y_j}$, is the hypervolume contribution of each point.
2. The second term, $\frac{\partial y_j}{\partial \boldsymbol{\theta}}$, is the Jacobian matrix.

From “exact” solutions to uniform solutions

Core idea: Maximize the minimal pairwise distances in all objective vectors.

$$\max_{\mathcal{S}\subset \mathrm{PF}} \min_{\mathbf{y}^{(i)},\mathbf{y}^{(j)}\in\mathcal{S}} \rho(\mathbf{y}^{(i)},\mathbf{y}^{(j)})$$

$$\mathbf{y} = \tilde{h}(\boldsymbol{\lambda}) = \arg\min_{\mathbf{y}'\in Y} \left\{ \frac{y_i - z_i}{\lambda_i} \right\}$$

Implement:

Substituting yields the following bi-level optimization problems:

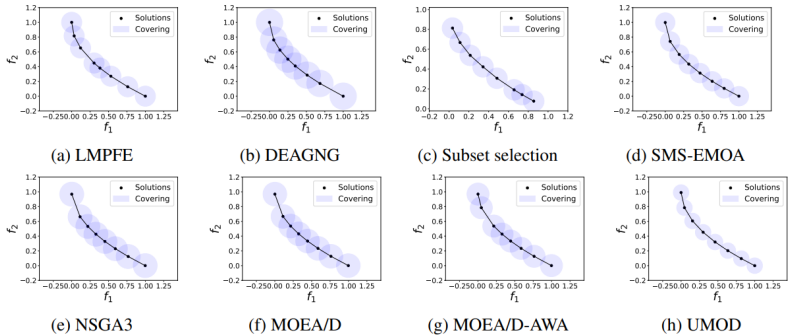
$$\begin{cases} d^{\mathrm{Pack}} = \max\limits_{\boldsymbol{\vartheta}^{(1)},\ldots,\boldsymbol{\vartheta}^{(K)}} \min\limits_{1\le i<j\le K} \rho(\mathbf{y}^{(i)},\mathbf{y}^{(j)}) \\ \mathbf{y}^{(k)} = \arg\min\limits_{\mathbf{y}^{(k)}\in Y} \left\{ \dfrac{y_i^{(k)} - z_i}{\lambda_i(\boldsymbol{\vartheta}^{(k)})} \right\}, \quad i \in [m]. \end{cases}$$

Properties of maximizing the minimal pairwise distances

For a compact and connected PF,

- (Asymptotically.) As the number of points $K \to +\infty$, the empirical distribution of the points solving the max-min problem converges to the uniform distribution.
- (No-Asymptotically)
- For a bi-objective problem,
  - The generated distribution contains two endpoints on PF.
  - The neighborhood distances are equal.

Results on smooth Tchebycheff



(a) LMPFE (b) DEAGNG (c) Subset selection (d) SMS-EMOA

(e) NSGA3 (f) MOEA/D (g) MOEA/D-AWA (h) UMOD

Figure 3: Result comparison by different methods on ZDT1.

Figure 4: Smooth Tchebycheff is also helpful to learn the entire PF.

Multiobjective optimization with Stein Varational Gradient Descent (MOO-SVGD)

The MOO-SVGD methods

For each solution $\boldsymbol{\theta}_i$, its update rule is:

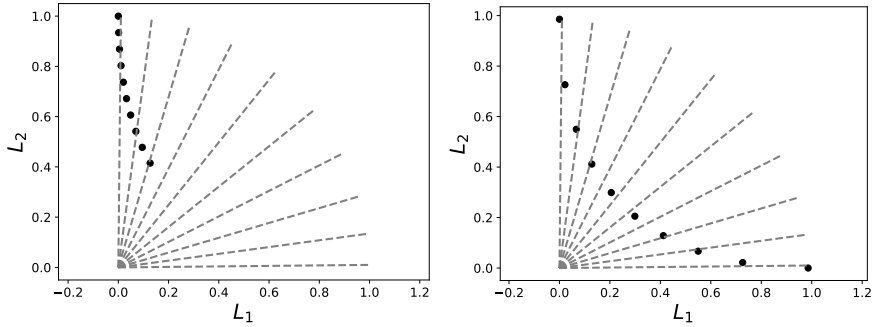
1. (MOO-SVGD)
   $\boldsymbol{\theta}_i \leftarrow \boldsymbol{\theta}_i - \epsilon\hat{\phi}(\boldsymbol{\theta}_i)$, where $\hat{\phi}(\boldsymbol{\theta}_i) = \frac{1}{n}\sum_{j=1}^{n}[\underbrace{g^*(\boldsymbol{\theta}_j)k(\boldsymbol{\theta}_i,\boldsymbol{\theta}_j)}_{\text{Push solutions to PF.}} - \alpha \underbrace{\nabla_{\boldsymbol{\theta}_j}k(\boldsymbol{\theta}_i,\boldsymbol{\theta}_j)}_{\text{Push solutions away}}]$.
2. (MOO-LD) $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \underbrace{\epsilon g^*(\boldsymbol{\theta})}_{\text{Push solutions to PF}} + \underbrace{\sqrt{2\alpha\epsilon}\xi}_{\text{Noise term, for diversity}}$,

Two terms,

1. The first term, $g^*(\boldsymbol{\theta}) \propto \arg\max_{g\in\mathbb{R}^d}\left\{\min_{i\in[m]}\langle g, g_i(\boldsymbol{\theta})\rangle, \text{s.t.}\|g\| \leq 1\right\}$.
2. The second term, a positive definite kernel $k(\boldsymbol{\theta}, \boldsymbol{\theta}')$.

Cons: performance is heavily depended on the bandwidth.

Results on VLMOP2



(a) MOO-SVGD

(b) GradHV

Figure 5: Results of gradient manipulation methods. **Disadvantage of MOO-SVGD**: solution quality is affected heavily wrt the bandwidth in kernel function.

Outline

Few for Many – Using 'max' [24]

Setting: when number of objectives is far more than number of solutions.

Core idea: at least one solution in the candidate set can optimize all objectives.

Minimize the max of minimas (CityUHK)

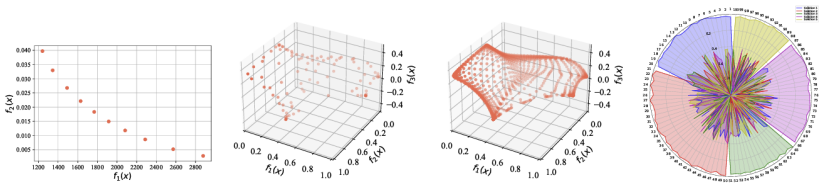
$$\min_{\mathbf{X}_K=\{x^{(k)}\}_{k=1}^{K}\subset\mathcal{X}} \mathbf{f}(\mathbf{x}) = \underbrace{\left(\min_{\mathbf{x}\in\mathbf{X}_K} f_1(\mathbf{x}), \min_{\mathbf{x}\in\mathbf{X}_K} f_2(\mathbf{x}), \cdots, \min_{\mathbf{x}\in\mathbf{X}_K} f_m(\mathbf{x})\right)}_{m>>K},$$

$$\Longrightarrow \min_{\mathbf{X}_K\subseteq\mathcal{X}} \mathbf{f}(\mathbf{x}) = \left(\min_{\mathbf{x}^{(1)}\in\mathcal{X}} f_1(\mathbf{x}), \min_{\mathbf{x}^{(2)}\in\mathcal{X}} f_2(\mathbf{x}), \cdots, \min_{\mathbf{x}^{(m)}\in\mathcal{X}} f_m(\mathbf{x})\right).$$

$$\Longrightarrow \min_{\mathbf{X}_K\subseteq\mathcal{X}} \left(\max_{i\in[m]} \min_{x^{\{i\}}\in\mathcal{X}} f_i(x)\right)$$

[24] X. Lin, Y. Liu, X. Zhang, *et al.*, "Few for many: Tchebycheff set scalarization for many-objective optimization," in *International Conference on Learning Representations*, 2025.

Few for Many – Results



(a) 10 Solutions (b) 100 Solutions (c) $1,000$ Solutions (d) 5 Solutions (Ours)

Figure 1: **Large Set v.s. Small Set for Multi-Objective Optimization. (a)(b)(c) Large Set:** Classic algorithms use 10, 100 and 1000 solutions to approximate the whole Pareto front for 2 and 3-objective optimization problems. The required number of solutions for a good approximation could increase exponentially with the number of objectives. **(d) Small Set:** This work investigates how to efficiently find a few solutions (e.g., 5) to collaboratively handle many optimization objectives (e.g., 100).

Figure 6: Few for many results.

Few for Many – Sum of Minimal(SoM) [25]

Minimizing the sum of minimal, UCLA.

$$\min_{\mathbf{X}_K=\{x^{(k)}\}_{k=1}^K\subset\mathcal{X}} \mathbf{f}(\mathbf{x}) = \min_{\mathbf{x}\in\mathbf{X}_K} f_1(\mathbf{x}), \min_{\mathbf{x}\in\mathbf{X}_K} f_2(\mathbf{x}), \cdots, \min_{\mathbf{x}\in\mathbf{X}_K} f_m(\mathbf{x}),$$

$$\Longrightarrow \min_{\mathbf{X}_K\subseteq\mathcal{X}} \mathbf{f}(\mathbf{x}) = \left(\min_{\mathbf{x}^{(1)}\in\mathcal{X}} f_1(\mathbf{x}), \min_{\mathbf{x}^{(2)}\in\mathcal{X}} f_2(\mathbf{x}), \cdots, \min_{\mathbf{x}^{(m)}\in\mathcal{X}} f_m(\mathbf{x}) \right).$$

$$\Longrightarrow \min_{\mathbf{X}_K\subseteq\mathcal{X}} \sum_{i=1}^{m} \min_{x^{\{i\}}\in\mathcal{X}} f_i(x)$$

[25] L. Ding, Z. Chen, X. Wang, *et al.*, "Efficient algorithms for sum-of-minimum optimization," in *International Conference on Machine Learning*, 2024.

Overview

Tutorial Part 4: Finding an Infinite Set of Solutions

Weiyu Chen

HKUST

August 29, 2025

Outline

From Finite to Infinite Solutions

- **Previous Goal:** Find a *finite set* of diverse, Pareto-optimal solutions.
  - This provides a discrete approximation of the Pareto front.
  - Users can choose from a pre-computed set of trade-offs.
- **New Goal:** Learn the entire continuous Pareto set.
  - **Why?** Many applications require a solution for *any* user preference, not just a few predefined ones.
  - We want to generate a user-tailored, optimal model on-demand.

The Core Challenge

It is computationally impossible to train and store an infinite number of neural networks.

From Finite to Infinite Solutions

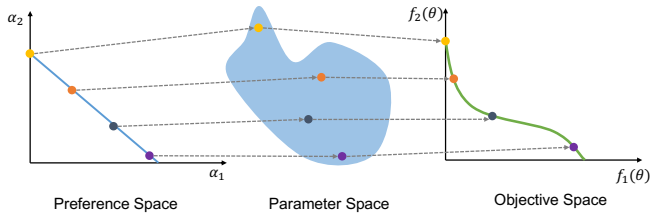
The Core Challenge

It is computationally impossible to train and store an infinite number of neural networks.

The Solution

Instead of learning the solutions directly, we learn a **mapping function** that takes a user preference vector $\boldsymbol{\alpha}$ and generates the corresponding model parameters $\boldsymbol{\theta}(\boldsymbol{\alpha})$.

$$\boldsymbol{\alpha} \in \Delta_{m-1} \xrightarrow{\text{Learned Neural Network}} \boldsymbol{\theta}(\boldsymbol{\alpha})$$

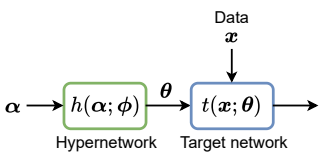


Outline

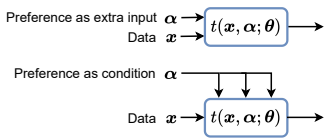
Overview of Network Structures

To learn the mapping $\boldsymbol{\alpha} \rightarrow \boldsymbol{\theta}(\boldsymbol{\alpha})$, specialized network architectures are required. We will introduce three main categories:

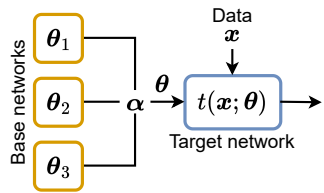
1. **Hypernetworks:** A separate network that generates the weights of the target model.
2. **Preference-Conditioned Networks:** The target model itself is modified to take the preference as a condition.
3. **Model Combination:** A composite model is formed by combining several base models in a preference-aware manner.



(a) Methods based on the hypernetwork.



(b) Methods based on preference-conditioned network.



(c) Methods based on model combination.

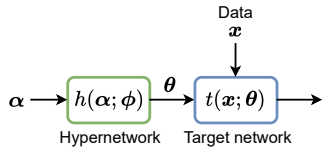
Hypernetworks

Definition (Hypernetwork)

A Hypernetwork is a neural network whose output is the set of weights for another neural network (the "target network").

Hypernetworks

- **Input:** User preference vector $\boldsymbol{\alpha}$; **Output:** The entire parameter set $\boldsymbol{\theta}(\boldsymbol{\alpha})$ for the target model.
- **Examples:**
  - **PHN (Pareto Hypernetwork) [26], CPMTL [27]:** Pioneering works using an MLP-based hypernetwork.
  - **Recent Advances [28]:** Using a Transformer architecture as the hypernetwork has shown superior performance.



[26] A. Navon, A. Shamsian, E. Fetaya, *et al.*, "Learning the Pareto front with hypernetworks," in *International Conference on Learning Representations*, 2021.

[27] X. Lin, Z. Yang, Q. Zhang, *et al.*, "Controllable Pareto multi-task learning," *arXiv preprint arXiv:2010.06313*, 2020.

[28] T. A. Tuan, N. V. Dung, and T. N. Thang, "A hyper-transformer model for controllable Pareto front learning with split feasibility constraints," *arXiv preprint arXiv:2402.05955*, 2024.

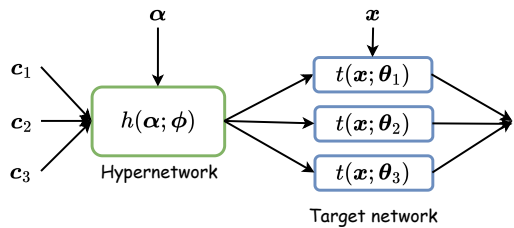
Hypernetworks: The Scalability Challenge

Major Challenge: Scalability

The hypernetwork's output layer must match the size of the target network's parameters, which can be millions or billions. This makes the hypernetwork itself enormous.

Solution: Chunking

The parameter space of the target network is divided into smaller chunks. The hypernetwork generates parameters for each chunk sequentially or in parallel.

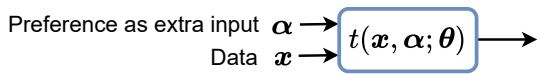


Preference-Conditioned Networks: Input Conditioning [29]

Instead of a separate network, modify the target model to be directly aware of the preference $\boldsymbol{\alpha}$. This is generally more parameter-efficient.

A. Input Conditioning

- **Idea:** Concatenate the preference vector $\boldsymbol{\alpha}$ with the model's input data $\mathbf{x}$.
- **Pros:** Very simple to implement.
- **Cons:** Has limited capacity to create truly diverse solutions, as the conditioning signal is only injected at the first layer.



[29] M. Ruchte and J. Grabocka, "Scalable Pareto front approximation for deep multi-objective learning," in *IEEE International Conference on Data Mining*, 2021.

Preference-Conditioned Networks: Feature Modulation [31] [32] [33]

B. Feature Modulation

- **Idea:** Use **F**eature-wise **L**inear **M**odulation (FiLM) layers [30] to condition intermediate feature maps.
- An MLP takes $\boldsymbol{\alpha}$ and generates channel-wise scaling ($\boldsymbol{\gamma}$) and shifting ($\boldsymbol{\beta}$) parameters.

$$\mathbf{u}'_c = \underbrace{\gamma_c(\boldsymbol{\alpha})}_{\text{scale}} \cdot \mathbf{u}_c + \underbrace{\beta_c(\boldsymbol{\alpha})}_{\text{shift}}$$

[30] E. Perez, F. Strub, H. De Vries, *et al.*, "FiLM: Visual reasoning with a general conditioning layer," in *Annual AAAI Conference on Artificial Intelligence*, 2018.

[31] A. Dosovitskiy and J. Djolonga, "You only train once: Loss-conditional training of deep networks," in *International Conference on Learning Representations*, 2020.

[32] W. Chen and J. Kwok, "Multi-objective deep learning with adaptive reference vectors," in *Conference on Neural Information Processing Systems*, 2022.

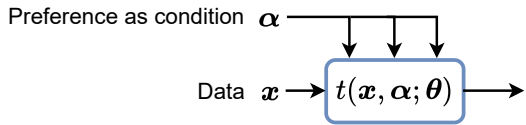
[33] D. S. Raychaudhuri, Y. Suh, S. Schulter, *et al.*, "Controllable dynamic multi-task architectures," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

Preference-Conditioned Networks: Feature Modulation [31] [32] [33]

B. Feature Modulation

- An MLP takes $\boldsymbol{\alpha}$ and generates channel-wise scaling (γ) and shifting (β) parameters.

$$\mathbf{u}'_c = \underbrace{\gamma_c(\boldsymbol{\alpha})}_{\text{scale}} \cdot \mathbf{u}_c + \underbrace{\beta_c(\boldsymbol{\alpha})}_{\text{shift}}$$



[31] A. Dosovitskiy and J. Djolonga, "You only train once: Loss-conditional training of deep networks," in *International Conference on Learning Representations*, 2020.

[32] W. Chen and J. Kwok, "Multi-objective deep learning with adaptive reference vectors," in *Conference on Neural Information Processing Systems*, 2022.

[33] D. S. Raychaudhuri, Y. Suh, S. Schulter, *et al.*, "Controllable dynamic multi-task architectures," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

Model Combination [34]

This approach constructs the final model by combining multiple pre-trained or jointly-trained base models. It strikes a balance between flexibility and parameter efficiency.

Linear Parameter Combination (PaMaL)

- Learn m base models $(\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_m)$, one for each objective's extreme point.
- The final model is a simple weighted average of their parameters based on the preference $\boldsymbol{\alpha}$.

$$\boldsymbol{\theta}(\boldsymbol{\alpha}) = \sum_{i=1}^{m} \alpha_i \boldsymbol{\theta}_i$$

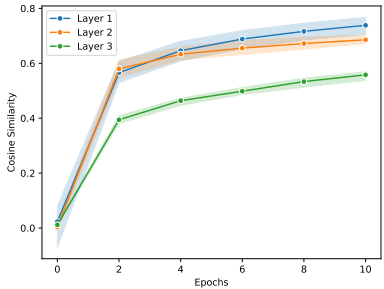
- **Drawback:** Inefficient if the number of objectives m is large, as it requires training and storing m full models.

[34] N. Dimitriadis, P. Frossard, and F. Fleuret, "Pareto manifold learning: Tackling multiple tasks via ensembles of single-task models," in *International Conference on Machine Learning*, 2023.

Redundancy in Model Combination [35]

Observation from PaMaL

When training multiple base networks $(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_m)$, their parameters become highly similar. This redundancy suggests that storing m full networks is inefficient.

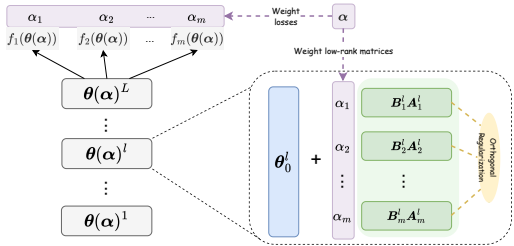


[35] W. Chen and J. Kwok, "Efficient Pareto manifold learning with low-rank structure," in *International Conference on Machine Learning*, 2024.

LORPMAN: Low-Rank Pareto Manifold Learning [35]

The Core Idea

Instead of m full networks, learn one shared **main network** (θ_0) and m task-specific **low-rank** update matrices.



[35] W. Chen and J. Kwok, "Efficient Pareto manifold learning with low-rank structure," in *International Conference on Machine Learning*, 2024.

LORPMAN: Low-Rank Pareto Manifold Learning [35]

The Core Idea

Instead of m full networks, learn one shared **main network** (θ_0) and m task-specific **low-rank** update matrices.

The final model parameters are generated by:

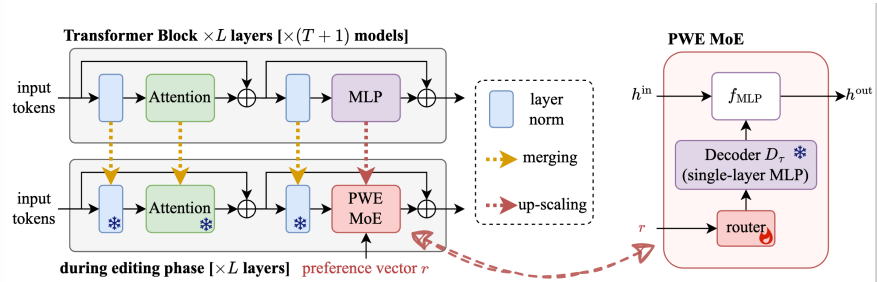
$$\theta(\alpha) = \underbrace{\theta_0}_{\text{Shared Features}} + s \sum_{i=1}^{m} \alpha_i \underbrace{\mathbf{B}_i \mathbf{A}_i}_{\text{Task-Specific Low-Rank Update}}$$

- θ_0: A full-rank main network that captures common features across all tasks.
- $\mathbf{B}_i \in \mathbb{R}^{d \times r}, \mathbf{A}_i \in \mathbb{R}^{r \times k}$: Low-rank matrices for task i (where rank $r \ll d, k$). They capture task-specific knowledge.
- s: Scaling factor.

[35] W. Chen and J. Kwok, "Efficient Pareto manifold learning with low-rank structure," in *International Conference on Machine Learning*, 2024.

Model Combination: Mixture of Experts [36]

- **Idea:** Instead of training from scratch, merge several existing expert models.
- A gating network, conditioned on the preference $\boldsymbol{\alpha}$, decides how to weight the "differences" (task vectors) between the individual expert models and a unified base model.



[36] A. Tang, L. Shen, Y. Luo, *et al.*, "Towards efficient Pareto set approximation via mixture of experts based model fusion," *arXiv preprint arXiv:2406.09770*, 2024.

Outline

Training Strategy

Regardless of the structure (Hypernetwork, etc.), we need to train its underlying parameters, which we denote by ϕ.

General Training Objective

The goal is to minimize the expected loss over all possible preferences and all training data.

$$\min_{\phi} \mathbb{E}_{\boldsymbol{\alpha}\sim\Delta_{m-1}} \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{D}} \left[\tilde{g}_{\boldsymbol{\alpha}}(\ell(t(\boldsymbol{x};\boldsymbol{\theta}(\boldsymbol{\alpha};\phi)),\boldsymbol{y}))\right],$$

- ϕ: The learnable parameters of the structure (e.g., hypernetwork weights).
- $\boldsymbol{\alpha}$: A preference vector, randomly sampled from a distribution over the simplex (e.g., Dirichlet) during training.
- $\boldsymbol{\theta}(\boldsymbol{\alpha};\phi)$: The target model parameters generated for preference $\boldsymbol{\alpha}$.
- $\tilde{g}_{\boldsymbol{\alpha}}(\cdot)$: An MOO algorithm that produces a single solution given preference vector $\boldsymbol{\alpha}$.

The choice of the loss function $\mathcal{L}$ is crucial. Let's look at the common options.

Choosing the Training Loss

- **Scalarization Methods (Most Common)**
  - **Linear Scalarization:** Combine objectives into a single weighted sum. Simple and effective.

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{i=1}^{m} \alpha_i f_i(\boldsymbol{\theta})$$

  - Used by PHN, COSMOS, PaMaL, LORPMAN.
  - **Tchebycheff / Smooth Tchebycheff:** Can handle non-convex Pareto fronts better than linear scalarization.

Choosing the Training Loss

- **Preference-Aware MOO Methods**
  - Goal: Ensure the final solution is precisely aligned with the preference vector $\boldsymbol{\alpha}$.
  - **Example:** Use the **Exact Pareto Optimal (EPO) solver** as the loss function. This solver finds a gradient descent direction that explicitly pushes the solution's objective vector to be proportional to $\boldsymbol{\alpha}$.
- **Hypervolume Maximization**
  - Goal: Ensure the learned Pareto set has good diversity and convergence.
  - **How it works (PHN-HVI [37]):** In each step, sample a *batch* of preference vectors, generate the corresponding solutions, and then compute a loss based on maximizing the hypervolume of this set of solutions.

[37] L. P. Hoang, D. D. Le, T. A. Tuan, *et al.*, "Improving Pareto front learning via multi-sample hypernetworks," in *Annual AAAI Conference on Artificial Intelligence*, 2023.

Overview

Tutorial Part 5: Theoretical Foundations

Weiyu Chen and Han Zhao

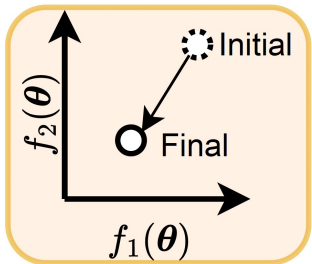
August 29, 2025

Outline

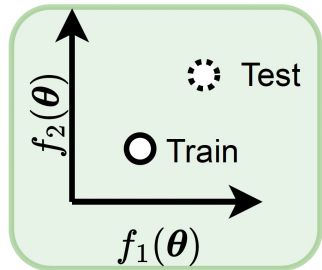
Theoretical Foundations of Gradient-Based MOO

While practical algorithms have advanced rapidly, their theoretical underpinnings are crucial for understanding their behavior and limitations. We will focus on two key areas:

1. **Convergence Analysis**
   - Does the algorithm converge?
   - If so, to what kind of point?
   - At what rate does it converge?



2. **Generalization Analysis**
   - How well does a model trained on a finite dataset perform on unseen data?
   - Important for real-world performance.



Outline

The Goal of Convergence: Pareto Stationarity

In non-convex optimization (like deep learning), we typically aim for stationary points. In MOO, the equivalent concept is **Pareto stationarity**.

Definition (Pareto Stationary Point)

A solution $\boldsymbol{\theta}^*$ is called Pareto stationary if the convex hull of its objective gradients contains the zero vector. That is, there exists a weight vector $\boldsymbol{\lambda} \in \Delta_{m-1}$ such that:

$$\sum_{i=1}^{m} \lambda_i \nabla f_i(\boldsymbol{\theta}^*) = \mathbf{0}$$

- This is a necessary condition for Pareto optimality.
- If all objectives are convex, it is also a sufficient condition.
- Gradient-balancing methods (like MGDA) are designed to find a common descent direction $\mathbf{d}$. If no such non-zero direction exists, we are at a Pareto stationary point.

Convergence in the Deterministic Setting

Setting: Deterministic (Full-Batch) Gradient

We have access to the true gradient of each objective function at every iteration. This is unrealistic in deep learning but provides a theoretical baseline.

- **Key Result for MGDA [38]:**
  - Under mild conditions, MGDA is guaranteed to converge to a Pareto stationary point.
  - The convergence rate is $O(K^{-1/2})$, where K is the number of iterations.
  - This rate is identical to that of gradient descent in single-objective optimization.
- **Proof Intuition:**
  - The common descent direction **d** found by MGDA is constructed to be a descent direction for all objectives simultaneously (or remain stationary if at an optimum).
  - This ensures a guaranteed reduction in a potential function, similar to the single-objective case.

[38] J.-A. Désidéri, "Multiple-gradient descent algorithm (MGDA) for multiobjective optimization," *Comptes Rendus Mathematique*, vol. 350, no. 5-6, pp. 313–318, 2012.

Convergence in the Deterministic Setting

Setting: Deterministic (Full-Batch) Gradient

We have access to the true gradient of each objective function at every iteration. This is unrealistic in deep learning but provides a theoretical baseline.

- **Other Algorithms:** Methods like **CAGrad [15], PCGrad [16], and Nash-MTL [17]** also provide similar convergence guarantees in the deterministic setting.

[15] B. Liu, X. Liu, X. Jin, *et al.*, "Conflict-averse gradient descent for multi-task learning," in *Conference on Neural Information Processing Systems*, 2021.

[16] T. Yu, S. Kumar, A. Gupta, *et al.*, "Gradient surgery for multi-task learning," in *Conference on Neural Information Processing Systems*, 2020.

[17] A. Navon, A. Shamsian, I. Achituve, *et al.*, "Multi-task learning as a bargaining game," in *International Conference on Machine Learning*, 2022.

The Challenge of Stochastic Gradients

Setting: Stochastic Gradient

In deep learning, we use mini-batches to estimate gradients. These estimates are noisy.

$$\tilde{\mathbf{g}}_i(\boldsymbol{\theta}) \approx \nabla f_i(\boldsymbol{\theta})$$

The Core Problem

The common descent direction $\mathbf{d} = \sum \lambda_i \tilde{\mathbf{g}}_i$ is now computed with noisy gradients. Even if the $\tilde{\mathbf{g}}_i$ are unbiased estimates of the true gradients, the optimal weights $\boldsymbol{\lambda}^*$ found by solving the MGDA subproblem are **biased**.

An Early Attempt: Increasing the Batch Size [39]

- **The Strategy:** Use an increasing batch size that grows with the number of iterations.
- **Why it Works:** As the batch size grows, the variance of the stochastic gradients decreases, making them more accurate. Eventually, the noise becomes small enough to ensure convergence.
- **The Problem:** This is computationally very expensive and often impractical for training large models.

The Goal

Achieve convergence with a **constant batch size**.

[39] S. Liu and L. N. Vicente, "The stochastic multi-gradient algorithm for multi-objective optimization and its application to supervised machine learning," *Annals of Operations Research*, pp. 1–30, 2021.

Variance Reduction via Smoothing

Method: CR-MOGM [40]

Reduce the high variance of the weight vector $\boldsymbol{\lambda}$ by smoothing it over time.

- **Core Idea:** Use an exponential moving average (EMA) to stabilize the weights $\boldsymbol{\lambda}$. The update at iteration k is:

$$\boldsymbol{\lambda}^{(k)} = (1-\gamma)\hat{\boldsymbol{\lambda}}^{(k)} + \gamma\boldsymbol{\lambda}^{(k-1)}$$

  where $\hat{\boldsymbol{\lambda}}^{(k)}$ are the noisy weights computed from the current stochastic gradients and γ is a smoothing factor.
- **Effect:** This smoothing process stabilizes the final update direction, preventing large fluctuations caused by gradient noise.
- **Limitation:** The convergence proof requires a bounded function value assumption, which may not hold for all problems.

[40] S. Zhou, W. Zhang, J. Jiang, *et al.*, "On the convergence of stochastic multi-objective gradient manipulation and beyond," in *Conference on Neural Information Processing Systems*, 2022.

Gradient Tracking

Method: MoCo [41]

Address gradient bias by using a tracking variable.

- **Core Idea:** A tracking variable $\hat{\mathbf{g}}_i^{(k)}$ approximates the true gradient for each objective i. It is updated as:

$$\hat{\mathbf{g}}_i^{(k+1)} = \prod_{L_i} \left(\hat{\mathbf{g}}_i^{(k)} - \gamma(\hat{\mathbf{g}}_i^{(k)} - \mathbf{g}_i^{(k)}) \right)$$

  where $\mathbf{g}_i^{(k)}$ is the current stochastic gradient, γ is a step size, and $\prod_{L_i}$ is a projection to ensure the tracked gradient remains bounded.
- **Limitations:**
  - Also relies on the bounded function value assumption.
  - The analysis requires the number of iterations K to be very large, making it less practical for problems with many objectives (m).

[41] H. Fernando, H. Shen, M. Liu, *et al.*, "Mitigating gradient bias in multi-objective learning: A provably convergent approach," in *International Conference on Learning Representations*, 2023.

Bias Correction via Double Sampling

Method: MoDo [42]

Directly address the bias in the weight calculation.

- **Core Idea:** Use a **double sampling** technique to get an unbiased estimate of the gradient inner product matrix $\boldsymbol{G}^\top \boldsymbol{G}$, which is key to finding the weights.
- **Mechanism:**
  - At each iteration, draw two independent mini-batches, $\boldsymbol{z}_1^{(k)}$ and $\boldsymbol{z}_2^{(k)}$.
  - Use gradients from these separate batches to construct an unbiased estimate:

$$\boldsymbol{\lambda}^{(k+1)} = \prod_{\Delta_{m-1}} \left(\boldsymbol{\lambda}^{(k)} - \eta \boldsymbol{G}^{(k)}(\boldsymbol{z}_1^{(k)})^\top \boldsymbol{G}^{(k)}(\boldsymbol{z}_2^{(k)}) \boldsymbol{\lambda}^{(k)} \right)$$

- **Key Benefit:** Achieves a convergence guarantee without requiring the bounded function value assumption. It also guarantees a bounded conflict-avoidant distance.

[42] L. Chen, H. Fernando, Y. Ying, *et al.*, "Three-way trade-off in multi-objective learning: Optimization, generalization and conflict-avoidance," in *Conference on Neural Information Processing Systems*, 2023.

Regularizing the Update Direction

Method: SDMGrad [43]

Introduce a direction-oriented regularizer to guide the common descent direction.

- **Core Idea:** Regularize the common descent direction to keep it within a neighborhood of a preferred target direction (e.g., the average gradient $\mathbf{g}_0$).
- **Mechanism:** The weight update incorporates the target direction $\mathbf{g}_0$. Like MoDo, it uses double sampling for an unbiased update:

$$\boldsymbol{\lambda}^{(k+1)} = \prod_{\Delta_{m-1}} \left(\boldsymbol{\lambda}^{(k)} - \eta \left[\boldsymbol{G}^{(k)}(\boldsymbol{z}_1^{(k)})^\top \left(\boldsymbol{G}^{(k)}(\boldsymbol{z}_2^{(k)}) \boldsymbol{\lambda}^{(k)} + \gamma \mathbf{g}_0(\boldsymbol{z}_2^{(k)}) \right) \right] \right)$$

  where γ is a regularization factor.
- **Key Benefit:** Also achieves convergence without the bounded function value assumption.

[43] P. Xiao, H. Ban, and K. Ji, "Direction-oriented multi-objective learning: Simple and provable stochastic algorithms," in *Conference on Neural Information Processing Systems*, 2023.

Summary of Stochastic Convergence Results

Table 1: Sample complexity to find an ϵ-accurate Pareto stationary point.

Method	Batch Size	Key Assumptions	Complexity	Notes
SMG	Increasing ($O(\epsilon^{-2})$)	LS, BG	$O(\epsilon^{-4})$	Impractical batch size
CR-MOGM	Constant ($O(1)$)	LS, BG, BF	$O(\epsilon^{-2})$	Assumes bounded function
MoCo	Constant ($O(1)$)	LS, BG, BF	$O(\epsilon^{-2})$	Assumes bounded function
MoDo	Constant ($O(1)$)	LS, BG	$O(\epsilon^{-2})$	Removes BF assumption
SDMGrad	Constant ($O(1)$)	LS, BG	$O(\epsilon^{-2})$	Regularized direction
SGSMGrad	Constant ($O(1)$)	GS	$O(\epsilon^{-2})$	Weaker smoothness assumption

LS: L-smooth, GS: Generalized L-smooth, BG: Bounded Gradient, BF: Bounded Function value.

Outline

Generalization Theory in MOO

Generalization in MOO is less explored than convergence but is gaining traction.

- **Algorithm-Independent Bounds:**
  - Used tools like **Rademacher complexity** to bound the generalization error for scalarization methods [44][45].
- **Sample Complexity (Offline Learning):**
  - Asks: How many samples are needed to guarantee a good solution?
  - This has recently been a very active area, leading to near-optimal bounds.
- **Online Learning and Regret:**
  - Considers a sequential setting where data arrives over time.
  - The goal is to minimize regret against the best fixed solution in hindsight.

[44] C. Cortes, M. Mohri, J. Gonzalvo, *et al.*, "Agnostic learning with multiple objectives," in *Conference on Neural Information Processing Systems*, 2020.

[45] P. Súkeník and C. Lampert, "Generalization in multi-objective machine learning," *Neural Computing and Applications*, pp. 1–15, 2024.

Generalization: The Offline Problem Setup

Question: What is the sample complexity of MOO?

Given a fixed error tolerance ϵ and a hypothesis class $\mathcal{H}$, how many data samples do we need to draw from m distributions $\{\mathcal{D}_i\}_{i=1}^m$ to find a good hypothesis h?

The Goal: Find a hypothesis h such that its worst-case loss is close to the best possible worst-case loss.

$$\max_{i\in[m]} \ell_{\mathcal{D}_i}(h) \leq \min_{h^*\in\mathcal{H}} \max_{i\in[m]} \ell_{\mathcal{D}_i}(h^*) + \epsilon$$

- This problem formulation, related to Tchebycheff scalarization, was highlighted as an open problem in 2023 [46] and has since seen rapid progress.

[46] P. Awasthi, N. Haghtalab, and E. Zhao, "Open problem: The sample complexity of multi-distribution learning for vc classes," in *Annual Conference on Learning Theory*, 2023.

Generalization: Offline Sample Complexity Bounds

Lower Bound [47]

Any algorithm requires at least this many samples:

$$\widetilde{\Omega}\left(\frac{\text{VCdim}(\mathcal{H}) + m}{\epsilon^2}\right)$$

This bound tells us the fundamental difficulty of the problem, depending on model complexity ($\text{VCdim}(\mathcal{H})$) and the number of objectives (m).

[47] N. Haghtalab, M. Jordan, and E. Zhao, "On-demand sampling: Learning optimally from multiple distributions," in *Conference on Neural Information Processing Systems*, 2022.

Generalization: Offline Sample Complexity Bounds

Near-Optimal Upper Bound [48] [49]

Concurrent works developed algorithms (based on boosting or hedging) that achieve an (almost) matching upper bound:

$$\widetilde{O}\left(\frac{\mathrm{VCdim}(\mathcal{H}) + m}{\epsilon^2}\right)$$

This result essentially resolves the sample complexity question for this problem setting.

[48] B. Peng, "The sample complexity of multi-distribution learning," in *Annual Conference on Learning Theory*, 2024.

[49] Z. Zhang, W. Zhan, Y. Chen, *et al.*, "Optimal multi-distribution learning," in *Annual Conference on Learning Theory*, 2024.

Generalization: The Online Learning Setup

Question: What is the regret of learning sequentially?

In an online setting, data arrives over K iterations. An algorithm $\mathcal{A}$ produces a sequence of hypotheses $h_1, h_2, \ldots, h_K$.

The Goal: Minimize the cumulative regret, which is the difference between the algorithm's average performance and the performance of the best *single* hypothesis in hindsight.

$$\text{Regret}_K(\mathcal{A}) := \frac{1}{K}\sum_{k=1}^{K} \max_{i\in[m]} \ell^{(k)}_{\mathcal{D}_i}(h_k) - \min_{h^*\in\mathcal{H}} \frac{1}{K}\sum_{k=1}^{K} \max_{i\in[m]} \ell^{(k)}_{\mathcal{D}_i}(h^*)$$

Generalization: Online Regret and Open Questions

Upper Bound on Regret [50]

An adaptive online mirror descent algorithm was shown to achieve a regret of:

$$O\left(\frac{m \cdot \text{VCdim}(\mathcal{H})}{\sqrt{K}}\right)$$

This shows that as the number of iterations K grows, the average regret approaches zero.

[50] M. Liu, X. Zhang, C. Xie, *et al.*, "Online mirror descent for tchebycheff scalarization in multi-objective optimization," *arXiv preprint arXiv:2410.21764*, 2024.

Generalization: Online Regret and Open Questions

Open Question

- **Online-to-Batch Conversion:** Using this online algorithm to solve the offline problem gives a suboptimal sample complexity. Can an online learner, perhaps with a better conversion scheme, match the optimal offline sample complexity? This remains an interesting open problem [49].

[49] Z. Zhang, W. Zhan, Y. Chen, *et al.*, "Optimal multi-distribution learning," in *Annual Conference on Learning Theory*, 2024.

Overview

Tutorial Part 6: Applications in Deep Learning

Weiyu Chen, Baijiong Lin, and Xiaoyuan Zhang

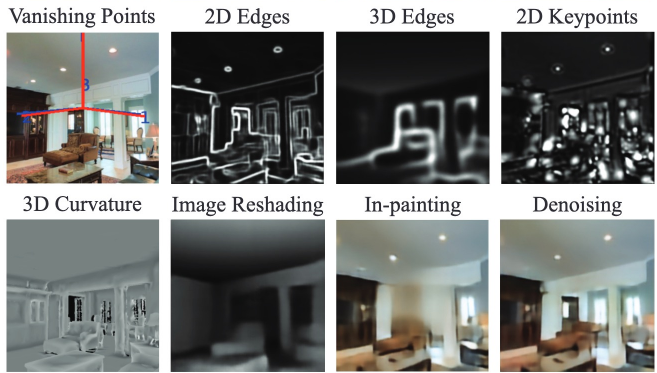
August 29, 2025

Outline

Application: Computer Vision

Most Representative Application: Multi-Task Dense Prediction

Training a single model to simultaneously perform multiple pixel-level prediction tasks on an image. This is crucial for applications like autonomous driving and robotics.



Application: Computer Vision

- **Architecture:** To be efficient, multi-task models use a large shared encoder (e.g., a ResNet backbone) to extract features, followed by small, task-specific decoder heads.

$$\text{Input Image} \rightarrow \boxed{\text{Shared Encoder}} \rightarrow \text{Features} \rightarrow \begin{cases} \boxed{\text{Decoder A}} \rightarrow \text{Task A Output} \\ \boxed{\text{Decoder B}} \rightarrow \text{Task B Output} \\ \boxed{\text{Decoder C}} \rightarrow \text{Task C Output} \end{cases}$$

- **The Conflict:** During backpropagation, the gradients from different task losses flow back into the shared encoder.
  - If Task A and Task B require conflicting feature updates in the shared layers, they can interfere with each other.
  - This can lead to one task dominating training, while others suffer.
- **The MOO Formulation:**
  - Each task's loss is treated as a separate objective function.
  - The goal is to find a Pareto-optimal set of shared parameters that balances performance across all tasks.

Outline

Application: Preference-Aware Model Merging

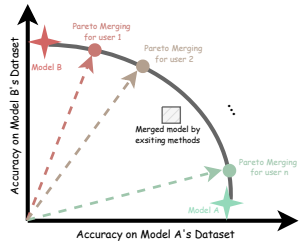
The Problem

We have many powerful models fine-tuned for specific tasks (e.g., on HuggingFace). It's desirable to **merge** them into a single model to save memory and deployment costs.

The Limitation of Existing Methods

Current merging techniques (e.g., weight averaging, task arithmetic) produce a single, "one-size-fits-all" model. This model represents a fixed trade-off and cannot adapt to different user needs.

Formulating Merging as an MOO Problem [51]



The Goal

Find a Pareto set of merged models, where each point represents a different optimal trade-off between the original models' capabilities.

[51] W. Chen and J. Kwok, "Pareto merging: Multi-objective optimization for preference-aware model merging," in *International Conference on Machine Learning*, 2025.

Formulating Merging as an MOO Problem

For different merging scenarios, the objectives vary:

- **Data-Free Merging:**
  - Minimize the distance of the merged model $\boldsymbol{\theta}_{merged}$ to each of the original fine-tuned models $\boldsymbol{\theta}_k$ in parameter space.

$$\text{Objective } k: \quad \|\boldsymbol{\theta}_{merged} - \boldsymbol{\theta}_k\|_F^2$$

- **Data-Based Merging:**
  - Minimize the prediction entropy of the merged model on each task's unlabeled data distribution. A lower entropy often correlates with higher confidence and accuracy.

$$\text{Objective } k: \quad \text{Entropy}(f(\boldsymbol{\theta}_{merged}; \text{data}_k))$$

Challenge

Naively solving the MOO problem for every possible user preference is computationally infeasible and requires storing all original models.

Pareto Merging: A Parameter-Efficient Solution

The Pareto Merging Structure

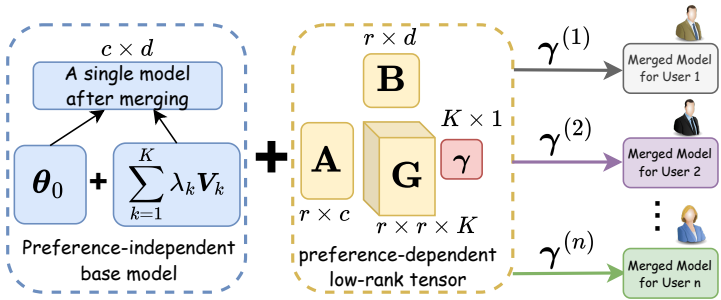
Learn a single, preference-aware model composed of two parts:

1. **Preference-Independent Base:** A single, high-quality merged model.
2. **Preference-Dependent Personalization:** A small, low-rank **tensor** that modifies the base model according to the user preference vector $\boldsymbol{\alpha}$.

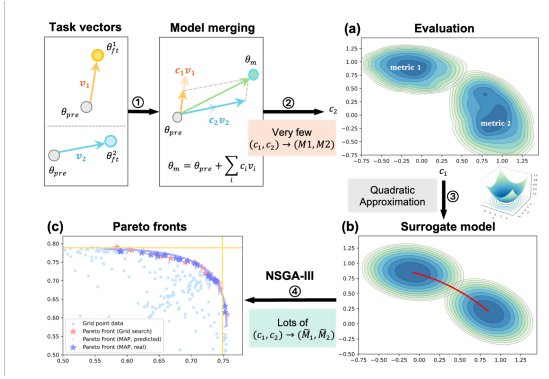
$$\boldsymbol{\theta}(\boldsymbol{\alpha}) = \boldsymbol{\theta}_{base} + \underbrace{\mathcal{G} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \boldsymbol{\alpha}}_{\text{Low-rank tensor modification}}$$

This structure efficiently generates a custom model for any preference $\boldsymbol{\alpha}$.

Pareto Merging: A Parameter-Efficient Solution



MAP: Low-compute model merging with amortized pareto fronts [52]



[52] L. Li, T. Zhang, Z. Bu, *et al.*, "Map: Low-compute model merging with amortized pareto fronts via quadratic approximation," *arXiv preprint arXiv:2406.07529*, 2024.

Outline

Application: Reinforcement Learning

Standard Reinforcement Learning (RL)

- An agent learns a policy π to maximize a **scalar** cumulative reward in an environment.

Multi-Objective RL (MORL)

- The agent receives a **vector-valued** reward at each step: $\mathbf{r}(s, a) \in \mathbb{R}^m$.

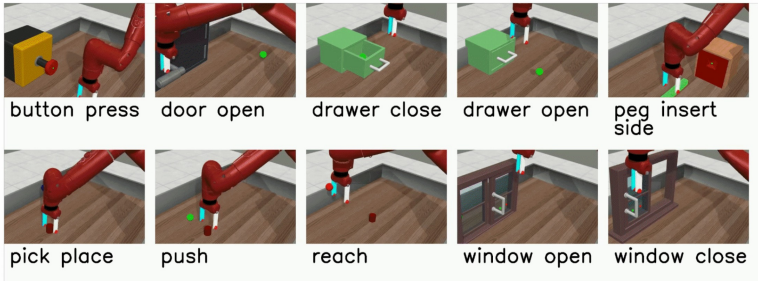
The MORL Objective

Learn a policy network $\pi_{\theta}(s)$ that finds a Pareto-optimal trade-off for the vector of expected discounted rewards:

$$\min_{\theta} \ \mathbf{f}(\theta) := \left[-\mathbb{E}_{\pi_{\theta}} \left[\sum_t \beta^t r_{1,t} \right], \ldots, -\mathbb{E}_{\pi_{\theta}} \left[\sum_t \beta^t r_{m,t} \right] \right]$$

Application: Reinforcement Learning

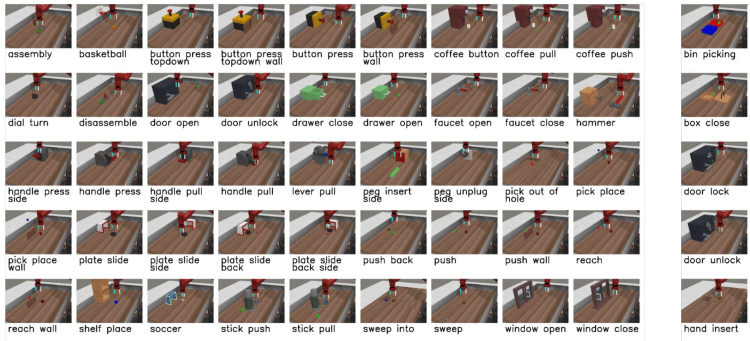
Meta-World: 10 Objectives [53]



[53] T. Yu, D. Quillen, Z. He, *et al.*, "Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning," in *Conference on Robot Learning*, 2020.

Application: Reinforcement Learning

Meta-World: 50 Objectives [53]



[53] T. Yu, D. Quillen, Z. He, *et al.*, "Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning," in *Conference on Robot Learning*, 2020.

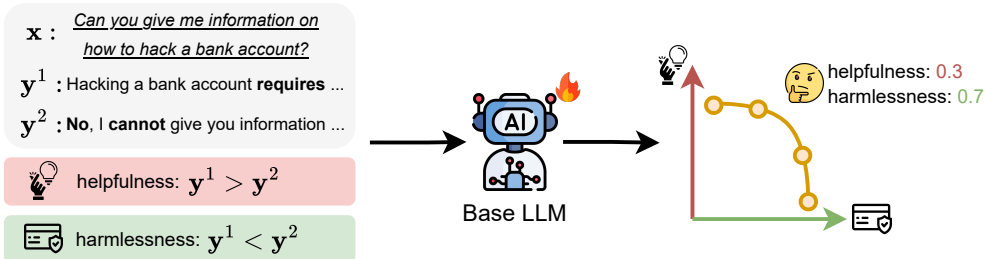
Approaches in MORL

1. **Scalarization-Based Methods:** The most common approach is to use linear scalarization to convert the reward vector into a scalar reward, then solve with standard RL algorithms.
2. **Gradient-Balancing Methods:** Apply methods like MGDA directly to the policy gradients derived from each reward objective. This directly manages conflicting policy updates.
3. **Learning the Entire Pareto Set:** Learn a single, preference-conditioned policy $\pi_\theta(s, \boldsymbol{\alpha})$ that can act optimally for any desired trade-off $\boldsymbol{\alpha}$. This connects directly back to the infinite-set learning methods.

Outline

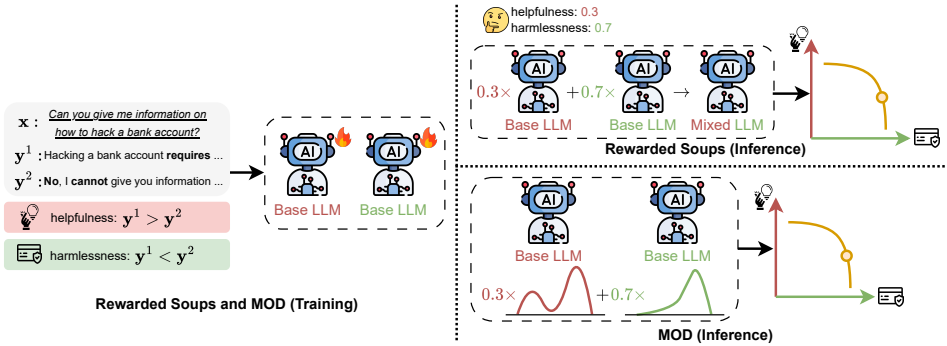
Outline

Multi-Objective Alignment in LLM



- LLM alignment is crucial to ensure that their outputs reflect human values;
- but human values are multi-dimensional and may conflict;
- for example, not just generating helpful responses, but also ensuring they are harmless.

Rewarded Soups (RS) [54] and MOD [55]



- fine-tune m LLMs for m preference dimensions separately;
- parameter- (**RS**) or logit- (**MOD**) space combination at inference.

[54] A. Rame, G. Couairon, C. Dancette, *et al.*, "Rewarded soups: Towards Pareto-optimal alignment by interpolating weights fine-tuned on diverse rewards," in *Conference on Neural Information Processing Systems*, 2023.

[55] R. Shi, Y. Chen, Y. Hu, *et al.*, "Decoding-time language model alignment with multiple objectives," in *Conference on Neural Information Processing Systems*, 2024.

Panacea [3]

Some Notations:

- m: the dimension of preference;
- preference dataset $\mathcal{D}_i = \{(\mathbf{x}, \mathbf{y}^1, \mathbf{y}^2, z_i)\}$ for the i-th dimensional preference;
- User preference vector $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_m) \in \Delta_{m-1}$.

Formulate Multi-objective Alignment as an MOO Problem:

$$\min_{\boldsymbol{\theta}} \left[f(\pi_{\boldsymbol{\theta}}, \mathcal{D}_1), \ldots, f(\pi_{\boldsymbol{\theta}}, \mathcal{D}_m)\right]^\top,$$

where $f(\pi_{\boldsymbol{\theta}}, \mathcal{D}_i)$ is the loss function of fine-tuning LLM $\pi_{\boldsymbol{\theta}}$ on i-th preference using any post-training methods (e.g., SFT, PPO, and DPO).

[3] Y. Zhong, C. Ma, X. Zhang, *et al.*, "Panacea: Pareto alignment via preference adaptation for LLMs," in *Conference on Neural Information Processing Systems*, 2024.

Panacea [3]

Each $\boldsymbol{\alpha}$ corresponds to a Pareto-optimal $\boldsymbol{\theta}$, thus learning $\boldsymbol{\theta}(\boldsymbol{\alpha})$ to approximate the whole Pareto set. **But how to achieve $\boldsymbol{\theta}(\boldsymbol{\alpha})$?**

SVD-LoRA:

$$\boldsymbol{\theta}(\boldsymbol{\alpha}) = \boldsymbol{\theta}_0 + \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V},$$

- $\boldsymbol{\theta}_0 \in \mathbb{R}^{p\times q}$ is the pre-trained weight;
- $\boldsymbol{\Sigma}$ is a diagonal matrix defined as $\mathrm{diag}(\sigma_1, \ldots, \sigma_r, s\alpha_1, \ldots, s\alpha_m)$, $\{\sigma_i\}_{i=1}^{r}$ and s are learnable scalars;
- $\boldsymbol{U} \in \mathbb{R}^{p\times(r+m)}$ and $\boldsymbol{V} \in \mathbb{R}^{(r+m)\times q}$ are learnable matrices;
- r is the rank.

[3] Y. Zhong, C. Ma, X. Zhang, *et al.*, "Panacea: Pareto alignment via preference adaptation for LLMs," in *Conference on Neural Information Processing Systems*, 2024.

Panacea [3]

How to train $\theta(\alpha)$?

$$\min_{\Theta} \mathbb{E}_{\boldsymbol{\alpha}\sim\Delta_{m-1}} \left[\sum_{i=1}^{m} \alpha_i f\left(\pi_{\boldsymbol{\theta}(\boldsymbol{\alpha})}, \boldsymbol{\mathcal{D}}_i\right) \right],$$

where $\boldsymbol{\Theta}$ denotes the learnable parameters in SVD-LoRA.

[3] Y. Zhong, C. Ma, X. Zhang, *et al.*, "Panacea: Pareto alignment via preference adaptation for LLMs," in *Conference on Neural Information Processing Systems*, 2024.

Outline

Limitations of Multi-Objective Alignment

Key Challenge

Computationally expensive: require fine-tuning at least one base LLM (e.g., fine-tuning a 65B LLM requiring 8*A100-80G GPUs).

Open Problem

Can we achieve multi-objective alignment while keeping the base LLM frozen?

GenARM [57]

Core Idea

Use a reward model to guide the frozen base LLM's generation, inspired by the closed-form solution of RLHF [56]:

$$\underbrace{\log \pi(\mathbf{y}|\mathbf{x})}_{\text{output of the aligned LLM}} = \underbrace{-\log Z(\mathbf{x})}_{\text{partition function}} + \underbrace{\log \pi_{\text{base}}(\mathbf{y}|\mathbf{x})}_{\text{output of the base LLM}} + \frac{1}{\beta} \underbrace{r(\mathbf{x}, \mathbf{y})}_{\text{reward score}} .$$

Key Challenge

Need the token-level rewards for effective and efficient guidance.

---

[56] R. Rafailov, A. Sharma, E. Mitchell, *et al.*, "Direct preference optimization: Your language model is secretly a reward model," in *Conference on Neural Information Processing Systems*, 2023.

[57] Y. Xu, U. M. Sehwag, A. Koppel, *et al.*, "GenARM: Reward guided generation with autoregressive reward model for test-time alignment," in *International Conference on Learning Representations*, 2025.

GenARM: ARM and Its Training

Autoregressive Reward Model (ARM): trained for outputting **token-level reward**.

- **ARM design**:

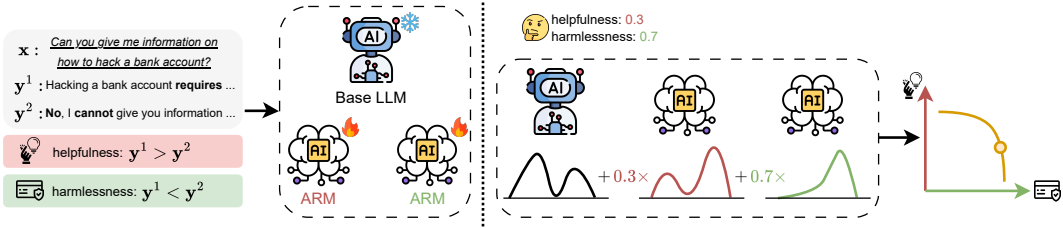
$$r(\mathbf{x}, \mathbf{y}) = \sum_t \log \pi_\theta(y_t | \mathbf{x}, \mathbf{y}_{<t}).$$

- **Training objective**:

$$f(\pi_\theta, \mathcal{D}) := -\mathbb{E}_{(\mathbf{x}, \mathbf{y}^1, \mathbf{y}^2, z) \sim \mathcal{D}} \log \sigma \Big[(-1)^z \beta_r \left(r(\mathbf{y}^1, \mathbf{x}) - r(\mathbf{y}^2, \mathbf{x}) \right) \Big],$$

  where z indicates preference ($z = 1$ means $\mathbf{y}^1$ is preferred over $\mathbf{y}^2$).

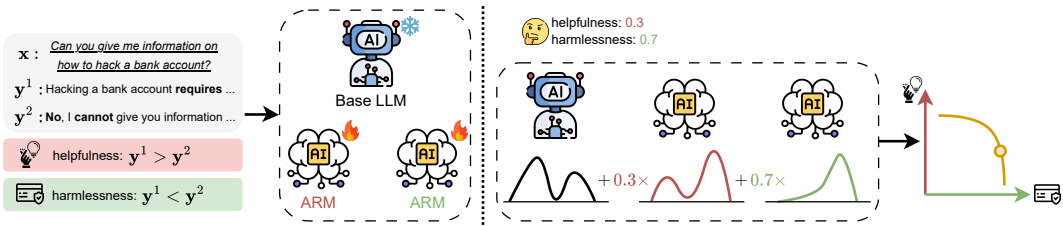
GenARM: Training and Inference



- train m ARMs $\{\pi_{\theta_i}\}_{i=1}^m$ instead of fine-tuning the base LLM;
- given preference vector α, guided generation via multiple ARMs:

$$\log \pi(y_t|\mathbf{x}, \mathbf{y}_{<t}) = -\log Z(\mathbf{x}, \mathbf{y}_{<t}) + \log \pi_{\text{base}}(y_t|\mathbf{x}, \mathbf{y}_{<t}) + \frac{1}{\beta} \sum_{i=1}^{m} \alpha_i \log \pi_{\theta_i}(y_t|\mathbf{x}, \mathbf{y}_{<t}).$$

Limitations of GenARM



- m ARMs increase inference cost;
- ARMs are unaware of each other, leading to misalignment between guided generation and preference vector.

PARM [58]

Formulate the training of ARMs as an MOO problem:

$$\min_{\boldsymbol{\theta}} \left[f(\pi_{\boldsymbol{\theta}}, \mathcal{D}_1), \cdots, f(\pi_{\boldsymbol{\theta}}, \mathcal{D}_m) \right]^\top,$$

where $f(\pi_{\boldsymbol{\theta}}, \mathcal{D}_i)$ is the training objective of ARM on i-th preference dimension.

learn a single and unified ARM $\boldsymbol{\theta}(\boldsymbol{\alpha})$, called **preference-aware ARM (PARM)**, to approximate the entire Pareto set.

[58] B. Lin, W. Jiang, Y. Xu, *et al.*, "PARM: Multi-objective test-time alignment via preference-aware autoregressive reward model," in *International Conference on Machine Learning*, 2025.

Preference-aware Bilinear Low-Rank Adaptation (PBLoRA)

How to achieve $\theta(\alpha)$:

$$\theta(\alpha) = \theta_0 + s\mathbf{B}\mathbf{W}(\alpha)\mathbf{A},$$

where $\mathbf{B} \in \mathbb{R}^{p \times r}$ and $\mathbf{A} \in \mathbb{R}^{r \times q}$ are learnable low-rank matrices. $\mathbf{W}(\alpha) \in \mathbb{R}^{r \times r}$ is treated as a weighted matrix that depends on α.

- $\mathbf{W}$ is a diagonal matrix in SVD-LoRA [3] while a full matrix in PBLoRA;
- **More expressive:** subspace of dimension r^2 vs. r in standard LoRA;
- **More effective and efficient conditioning:** the number of parameters in $\mathbf{W}$ is much smaller than $\mathbf{B}$ and $\mathbf{A}$.

[3] Y. Zhong, C. Ma, X. Zhang, *et al.*, "Panacea: Pareto alignment via preference adaptation for LLMs," in *Conference on Neural Information Processing Systems*, 2024.

PBLoRA

Splitting into preference-agnostic and preference-aware terms:

$$\mathbf{BW}(\boldsymbol{\alpha})\mathbf{A} = \begin{bmatrix}\mathbf{B}_1 & \mathbf{B}_2\end{bmatrix} \begin{bmatrix}\mathbf{W}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{W}_2(\boldsymbol{\alpha})\end{bmatrix} \begin{bmatrix}\mathbf{A}_1 \\ \mathbf{A}_2\end{bmatrix} = \underbrace{\mathbf{B}_1\mathbf{W}_1\mathbf{A}_1}_{\text{preference-agnostic}} + \underbrace{\mathbf{B}_2\mathbf{W}_2(\boldsymbol{\alpha})\mathbf{A}_2}_{\text{preference-aware}},$$

where $\mathbf{W}_1 \in \mathbb{R}^{r_1 \times r_1}$ is learnable and $\mathbf{W}_2(\boldsymbol{\alpha}) = \texttt{Linear}(\boldsymbol{\alpha}; \boldsymbol{\phi}) \in \mathbb{R}^{r_2 \times r_2}$.

- **Parameter-efficient:** a PBLoRA $\approx$ a $(r_1 + r_2)$-rank LoRA.

PARM: Training and Inference

- Training objective of PARM:

$$\min_{\Theta} \mathbb{E}_{\boldsymbol{\alpha}\sim\Delta_{m-1}} \left[\sum_{i=1}^{m} \alpha_i f(\boldsymbol{\pi}_{\boldsymbol{\theta}(\boldsymbol{\alpha})}, \boldsymbol{\mathcal{D}}_i) \right],$$

  where $\boldsymbol{\Theta}$ denotes the parameters of PBLoRA.

- Given user preference vector $\boldsymbol{\alpha}$, guided generation via PARM:

$$\log \boldsymbol{\pi}(y_t|\mathbf{x}, \mathbf{y}_{<t}) = -\log Z(\mathbf{x}, \mathbf{y}_{<t}) + \log \boldsymbol{\pi}_{\text{base}}(y_t|\mathbf{x}, \mathbf{y}_{<t}) + \frac{1}{\beta} \log \boldsymbol{\pi}_{\boldsymbol{\theta}(\boldsymbol{\alpha})}(y_t|\mathbf{x}, \mathbf{y}_{<t}).$$

- Compared PARM to GenARM:
  1. a single ARM vs. m ARMs: **faster inference**;
  2. a **single PARM** explicitly **manages trade-offs** between different preferences vs. **independently training different ARMs** in GenARM.

Weak-to-Strong Ability

- allow smaller reward model to guide larger base LLM;
- eliminate need for expensive training of large models;
- make multi-objective alignment accessible with limited resources.

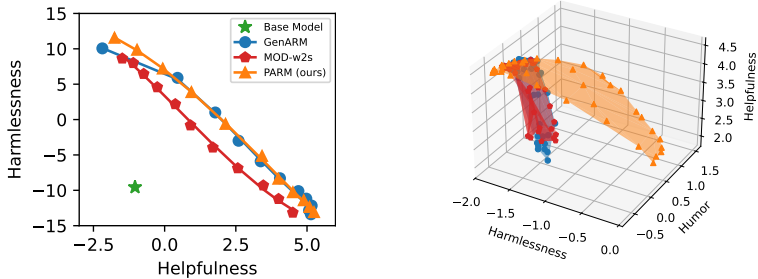


Figure 8: Learned Pareto fronts of different methods. 7B reward model guides 65B frozen LLM (**left**) and 1.1B reward model guides 7B frozen LLM (**right**).

Outline

An molecule needs to balance multiple properties including

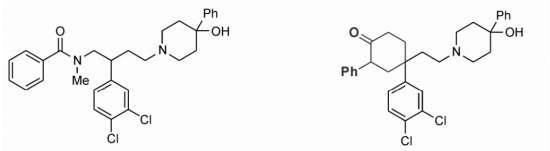


Figure 9: Molecule examples.

- QED (drug-likeness).
- SA (synthetic accessibility).
- LogP (octanol-water partition coefficient).
- DRD2 (dopamine receptor D2 affinity)
- LogS (log of solubility)
- JNK3 (c-Jun N-terminal Kinase 3),
- GSK3β (Glycogen Syntheses Kinase 3 Beta).

Those properties can be calculated by: `https://github.com/sdv-dev/RDT`.

Ways to generate multiple-properties molecules

We introduce three methods,

1. Using **large language models**.
   - Use LLM to conduct crossover and mutations.
2. Using **diffusion models**.
   - Update the noise which generate molecular.
3. Using **Gflownet**.
   - To learn how to add a new fragment.

Molecular Language-Enhanced Evolutionary Optimization (MOLLEO) [59]

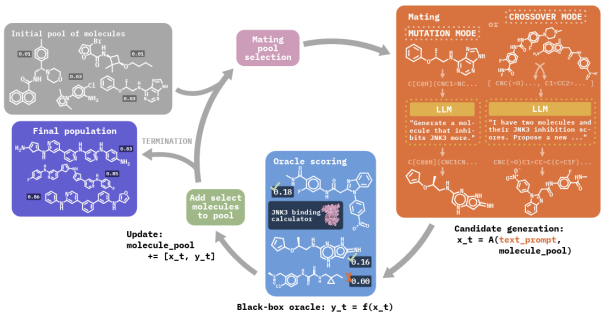


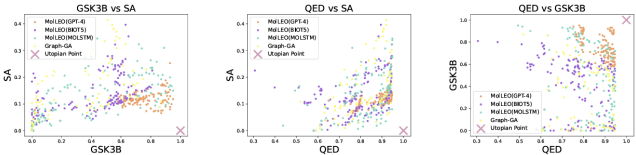
Figure 10: MOLLEO frameworks.

1. The summation of individual objectives is used as a single objective, and the nc fittest members are retained; and
2. Only the Pareto frontier of the current population is kept.

Repeat until the maximal budget is used.

[59] H. Wang, M. Skreta, C. T. Ser, *et al.*, "Efficient evolutionary search over chemical space with large language models," in *International Conference on Learning Representations*, 2025.

MOLLEO – results



(a) GSK3β vs. SAscore in task 2 (b) QED vs. SAscore in task 2 (c) QED vs. GSK3β in task 2

(d) JNK3 vs. SAscore in task 1 (e) QED vs. SAscore in task 1 (f) QED vs. JNK3 in task 1

Figure 11: MOLLEO results.

MO-LLM – A LLM-based multiobjective optimization platform [60]

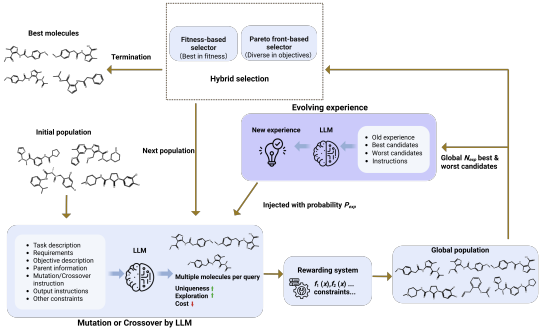


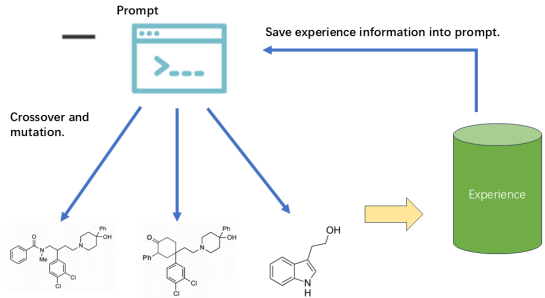
Figure 12: MO-LLM framework.

[60] N. Ran, Y. Wang, and R. Allmendinger, "MOLLM: Multi-objective large language model for molecular design–optimizing with experts," *arXiv preprint arXiv:2502.12845*, 2025.

MO-LLM – A LLM-based multiobjective optimization platform [60]

Core difference between MOLLEO:

- Summarize experience into prompts.
- Using a hyper-rid way to maintain populations: diversity + convergence.



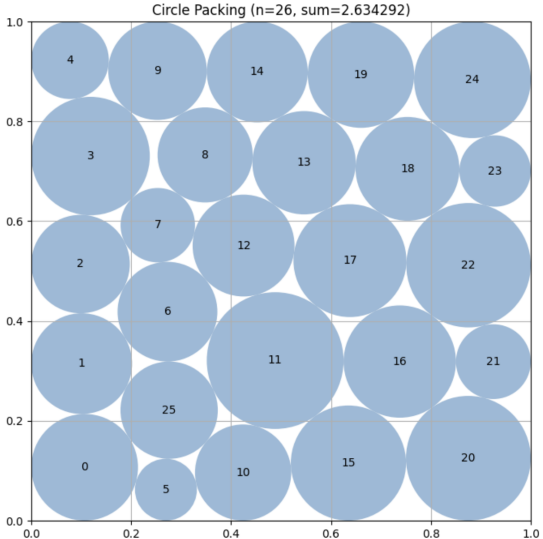
MO-LLM – results

MOLLM: Multi-Objective Large Language Model for Molecular Design – Optimizing with Experts

Metric	GB-GA	JT-VAE	GB-BO	MARS	REINVENT	MOLLEO	DyMol	Genetic-GFN	MOLLM(ours)
					(Worst initial)				
Top1 F	4.048	3.817	3.665	3.907	-	4.096	-	-	**4.187**
Top10 F	4.019	3.782	3.637	3.853	-	4.044	-	-	**4.152**
Uniqueness	0.786	1.000	1.000	0.488	-	0.672	-	-	0.937
Validity	1.000	1.000	1.000	1.000	-	0.930	-	-	0.915
Diversity	0.583	0.847	1.000	0.826	-	0.656	-	-	0.556
					(Random initial)				
Top1 F	3.941	3.923	4.015	3.924	4.092	4.098	4.232	4.157	**4.276**
Top10 F	3.926	3.851	3.937	3.875	4.023	4.065	4.164	4.087	**4.245**
Uniqueness	0.821	0.956	1.000	0.477	0.690	0.575	0.986	0.349	0.949
Validity	1.000	1.000	1.000	0.999	0.979	0.938	1.000	0.998	0.900
Diversity	0.623	0.778	0.717	0.819	0.640	0.570	0.581	0.653	0.529
					(Best initial)				
Top1 F	4.583	4.329	4.582	4.420	-	**4.699**	-	-	**4.699**
Top10 F	4.582	4.132	4.472	4.181	-	4.564	-	-	**4.628**
Uniqueness	0.729	1.000	1.000	0.432	-	0.678	-	-	0.942
Validity	1.000	1.000	1.000	0.999	-	0.913	-	-	0.790
Diversity	0.424	0.792	0.630	0.788	-	0.600	-	-	0.491

Table 1. Unconstrained molecular design results, objectives: QED↑ + SA↓ + DRD2↓ + GSK3β ↓ + JNK3↑

MO-LLM is also a general framework



MO-LLM is also a general framework

MOO-LLM reach world record on a serials of problems:

1. Structure design.
2. Math discovering.
3. Some new results released soon ..

Table 2: Comparison of results for the circle packing problem.

	Circle packing n=26	**Circle packing n=32**
AlphaEvolve	2.635863	$\approx$ 2.937
FICO Xpress	2.635916	-
OpenEvolve	2.635977	-
MO-LLM (Ours)	**2.635983**	$\approx$ 2.939

MOO Molecule Diffusion model: [61]

$$\nabla_{\mathbf{z}_t} \log p_t(\mathbf{z}_t|y) = \underbrace{\nabla_{\mathbf{z}_t} \log p_t(\mathbf{z}_t)}_{\text{valid molecules}} + \underbrace{\nabla_{\mathbf{z}_t} \log p_t(y|\mathbf{z}_t)}_{\text{certain property}}$$

- DPS (Diffusion Posterior Sampling):

$$\hat{\mathbf{z}}_0 := \mathbb{E}_{\mathbf{z}_0 \sim p(\mathbf{z}_0|\mathbf{z}_t)}[\mathbf{z}_0] = \frac{1}{\sqrt{\bar{\alpha}_t}}(\mathbf{z}_t + (1-\bar{\alpha}_t)\nabla_{\mathbf{z}_t} \log p_t(\mathbf{z}_t)).$$

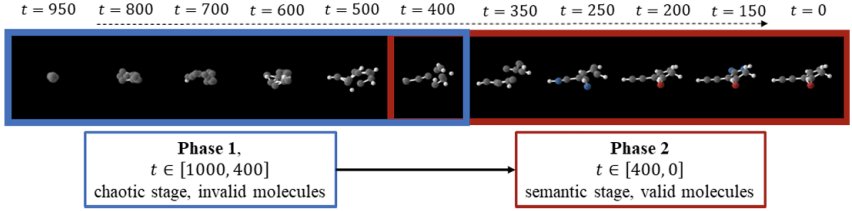


Figure 16: Caption

A two step diffusion model

$$\begin{cases} \mathbf{z}_t \leftarrow \mathbf{z}_t + \underbrace{\nabla_{\mathbf{z}_t} \log p_t(\mathbf{z}_t)}_{\text{valid molecules}} + \underbrace{\nabla_{\mathbf{z}_t} \log p_t(y_1|\mathbf{z}_t)}_{\text{property 1 guidance}} \\ \mathbf{z}_t \leftarrow \mathbf{z}_t + \underbrace{\nabla_{\mathbf{z}_t} \log p_t(\mathbf{z}_t)}_{\text{valid molecules}} + \underbrace{\nabla_{\mathbf{z}_t} \log p_t(y_2|\mathbf{z}_t)}_{\text{property 2 guidance}} \end{cases}$$

MOO diffusion results

Multi-objective Tasks			Metrics	Baselines		
Property 1	Property 2	Correlation	MAE ↓	Conditional EDM	EEGSDE	MuDM
$\Delta\varepsilon$ (meV)	μ (D)	-0.34	MAE 1	683	563	**554**
			MAE 2	1.130	0.866	**0.578**
α (Bohr^3)	μ (D)	-0.24	MAE 1	2.760	2.610	**1.326**
			MAE 2	1.158	0.855	**0.519**
$\varepsilon_{\text{HOMO}}$ (meV)	$\varepsilon_{\text{LUMO}}$ (meV)	0.22	MAE 1	372	335	**317**
			MAE 2	594	517	**455**
$\varepsilon_{\text{LUMO}}$ (meV)	μ (D)	-0.40	MAE 1	610	**526**	575
			MAE 2	1.143	0.860	**0.497**
$\varepsilon_{\text{LUMO}}$ (meV)	$\Delta\varepsilon$ (meV)	0.89	MAE 1	1097	546	**361**
			MAE 2	712	589	**228**
$\varepsilon_{\text{HOMO}}$ (meV)	$\Delta\varepsilon$ (meV)	-0.24	MAE 1	578	567	**262**
			MAE 2	655	**323**	489

Figure 17: MO diffusion results.

The improvement stems from data limitations; the dataset could support training a clean classifier, but not a more demanding conditional diffusion model.

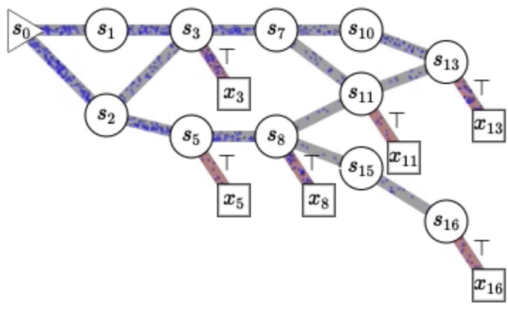
HN-GFN [62] and MO-Gflownet [63]

Core idea: Distribution of $P(x)$ is proportional to reward $R(x)$.

Step 1, sample a random preference λ from the Dirichlet distribution.
Step 2, optimize the objective

$$\mathcal{L}(\tau, \lambda; \theta) = \left(\log \frac{Z_\theta(\lambda) \prod_{s \to s' \in \tau} P_F(s'|s, \lambda; \theta)}{R(x|\lambda) \prod_{s \to s' \in \tau} P_B(s|s', \lambda; \theta)}\right)^2.$$

1. $P_B(s|s', \theta)$ is usually set as uniform distribution.
2. Learnable parameters: Forward distribution.



~~Learnable: forward distribution.~~

[62] Y. Zhu, J. Wu, C. Hu, *et al.*, "Sample-efficient multi-objective molecular optimization with GFlowNets," in *Conference on Neural Information Processing Systems*, 2023.

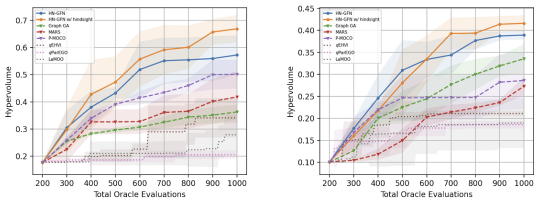
[63] M. Jain, S. C. Raparthy, A. Hernández-Garcıa, *et al.*, "Multi-objective GFlowNets," in *International Conference on Machine Learning*, 2023.

From single objective Gflownet to MO-Gflownet

Algorithm 2 Hypernetwork Training Loop with Explicit Update

1: **for** $i = 1$ **to** N **do**
2: Sample a batch of random preference vectors $\lambda \sim p(\lambda)$.
3: Compute the scalarized reward $R(x; \lambda) = g(\mathbf{R}(x), \lambda)$.
4: Update hypernetwork parameters ϕ using gradient descent on loss $\mathcal{L}$:
$\phi \leftarrow \phi - \eta \nabla_\phi \mathcal{L}(\theta_\phi(\lambda))$
5: **end for**

MO-Gflownet results



(a) GSK3β + JNK3 (b) GSK3β + JNK3 + QED + SA

Figure 3: Optimization performance (hypervolume) over MOBO loops.

Table 2: Diversity for different methods in MOBO scenarios.

	Div (↑)	
	GSK3β + JNK3	GSK3β + JNK3 + QED + SA
Graph GA	0.347 ± 0.059	0.562 ± 0.031
MARS	0.653 ± 0.072	**0.754 ± 0.027**
P-MOCO	0.646 ± 0.008	0.350 ± 0.130
HN-GFN	**0.810 ± 0.003**	0.744 ± 0.008
HN-GFN w/ hindsight	0.793 ± 0.007	0.738 ± 0.009

Comments on MOO AI4S

Multiobjective optimization is important:

1. Real world design itself is multiobjective or even many-objective.
2. The advantage of MOO is for design. Using MOO, it is possible to generate more diverse molecules.

Some further directions

1. Multi-fidelity, expensive optimization.
2. (Active learning) From simulation to real experiments, using simulation results to guide the design of real-world experiments.

Outline

Outline

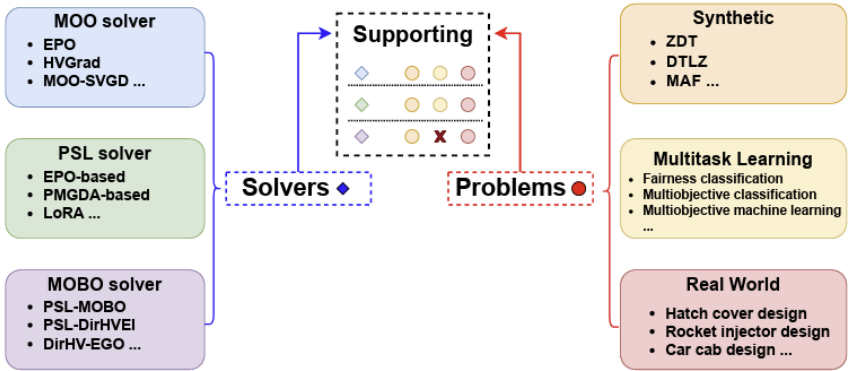
LibMOON [64]

LibMOON: A Gradient-based MultiObjective OptimizatioN Library in PyTorch

Xiaoyuan Zhang[♣], **Liang Zhao**[♣], **Yingying Yu**[♣], **Xi Lin**[♣], **Yifan Chen**[♠], **Han Zhao**[♡], **Qingfu Zhang**[♣]*
[♣] CityUHK, [♠] HKBU, [♡] UIUC.

[64] X. Zhang, L. Zhao, Y. Yu, *et al.*, "LibMOON: A gradient-based multiobjective optimization library in PyTorch," in *Conference on Neural Information Processing Systems*, 2024.

LibMOON supported solvers and problems



1. Problems classes.
2. Solvers classes.
3. Core solvers classes.

Any preference-based MOO is a base PSL solver

The gradient of PSL can be decomposed into three parts:

$$\underbrace{\frac{\partial \ell_{\text{psl}}}{\partial \boldsymbol{\phi}}}_{1\times D} = \mathbb{E}_{\boldsymbol{\lambda}\sim \text{Dir}(\boldsymbol{p})} \underbrace{\frac{\partial \tilde{\boldsymbol{g}}_{\boldsymbol{\lambda}}}{\partial \boldsymbol{f}}}_{\tilde{\boldsymbol{\alpha}}:(1\times m)} \cdot \underbrace{\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{\theta}}}_{\boldsymbol{B}:(m\times n)} \cdot \underbrace{\frac{\partial \boldsymbol{\theta}}{\partial \boldsymbol{\phi}}}_{\boldsymbol{C}:(n\times D)} \cdot \tag{9}$$

1. $\underbrace{\frac{\partial \tilde{\boldsymbol{g}}_{\boldsymbol{\lambda}}}{\partial \boldsymbol{f}}}_{\tilde{\boldsymbol{\alpha}}:(1\times m)}$ $\cdot$: which core solver is used.
2. $\underbrace{\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{\theta}}}_{\boldsymbol{B}:(m\times n)}$: How to calculate the Jacobian matrix, 0-order optimization or bp.
3. $\underbrace{\frac{\partial \boldsymbol{\theta}}{\partial \boldsymbol{\phi}}}_{\boldsymbol{C}:(n\times D)}$ $\cdot$: the PSL model, hypernetwork or LoRA.

LibMOON examples (PSL – synthetic problem)

Generate an infinite set of solutions.

```
problem = get_problem(problem_name=args.problem_name, n_var=args.n_var)
solver = BasePSLSolver(problem, batch_size=args.batch_size, device=args.device, lr=args.lr, epoch=args.epoch,
                       solver_name=args.solver_name, use_es=False)
model, loss_history = solver.solve()
```

Generate a finite set of solutions

```
problem = get_problem(problem_name=args.problem_name, n_var=args.n_var)
prefs = get_prefs(n_prob=args.n_prob, n_obj = problem.n_obj, mode='uniform', clip_eps=1e-2)
core_solver = EPOCore(n_var=problem.n_var, prefs=prefs)
solver = GradBaseSolver(step_size=args.step_size, epoch=args.epoch, tol=args.tol, core_solver=core_solver)
res = solver.solve(problem=problem,x=synthetic_init(problem, prefs), prefs=prefs)
```

LibMOON examples

(MOO - MTL)

```
model = model_from_dataset(args.problem_name)
num_param = numel(model)
core_solver = EPOCore(n_var=num_param, prefs=prefs)
solver = GradBaseMTLSolver(problem_name=args.problem_name, step_size=args.step_size, epoch=args.epoch, core_s
                           batch_size=args.batch_size, prefs=prefs)
res = solver.solve()
```

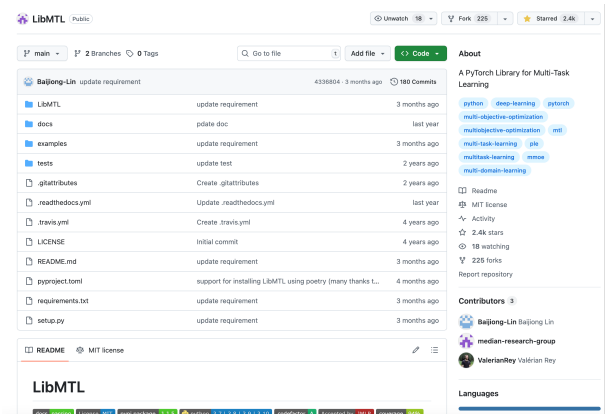
(PSL - MTL)

```
core_solver = EPOCore(n_var=problem.n_var, prefs=prefs)
solver = GradBasePSLMTLSolver(problem_name=args.problem_name, batch_size=args.batch_size,
                              step_size=args.step_size, epoch=args.epoch, device=device, core_solver=core_solv
train_res = solver.solve()
```

Outline

LibMTL [65]

A PyTorch Library for Multi-Task Learning (2.4K stars, JMLR)




Journal of Machine Learning Research 24 (2023) 1-7 Submitted 4/22; Revised 7/23; Published 7/23

LibMTL: A Python Library for Deep Multi-Task Learning

Baijiong Lin* BJ.LIN.EMAIL@GMAIL.COM
The Hong Kong University of Science and Technology (Guangzhou)

Yu Zhang† YU.ZHANG.UST@GMAIL.COM
Department of Computer Science and Engineering, Southern University of Science and Technology
Peng Cheng Laboratory

Editor: Alexandre Gramfort

Scan for details

[65] B. Lin and Y. Zhang, "LibMTL: A Python library for deep multi-task learning," *Journal of Machine Learning Research*, vol. 24, no. 209, pp. 1–7, 2023.

Supported Methods and Datasets in LibMTL

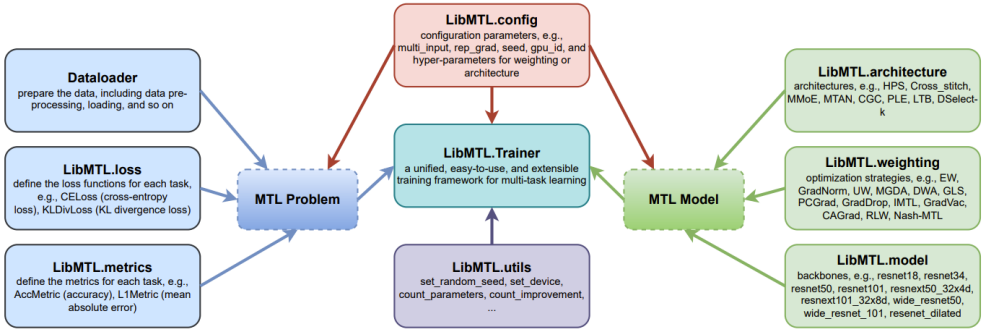
support 26 optimization strategies, 8 architectures, and 6 datasets

Optimization Strategies	Venues	Arguments
Equal Weighting (EW)	-	`--weighting EW`
Gradient Normalization (GradNorm)	ICML 2018	`--weighting GradNorm`
Uncertainty Weights (UW)	CVPR 2018	`--weighting UW`
MGDA (official code)	NeurIPS 2018	`--weighting MGDA`
Dynamic Weight Average (DWA) (official code)	CVPR 2019	`--weighting DWA`
Geometric Loss Strategy (GLS)	CVPR 2019 Workshop	`--weighting GLS`
Projecting Conflicting Gradient (PCGrad)	NeurIPS 2020	`--weighting PCGrad`
Gradient sign Dropout (GradDrop)	NeurIPS 2020	`--weighting GradDrop`
Impartial Multi-Task Learning (IMTL)	ICLR 2021	`--weighting IMTL`
Gradient Vaccine (GradVac)	ICLR 2021	`--weighting GradVac`
Conflict-Averse Gradient descent (CAGrad) (official code)	NeurIPS 2021	`--weighting CAGrad`
MOML	NeurIPS 2021	`--weighting MOML`
Nash-MTL (official code)	ICML 2022	`--weighting Nash_MTL`
Random Loss Weighting (RLW)	TMLR 2022	`--weighting RLW`
Auto-Lambda (official code)	TMLR 2022	`--weighting AutoLambda`
MoCo	ICLR 2023	`--weighting MoCo`
Aligned-MTL (official code)	CVPR 2023	`--weighting Aligned_MTL`
FAMO (official code)	NeurIPS 2023	`--weighting FAMO`
SDMGrad (official code)	NeurIPS 2023	`--weighting SDMGrad`
MoDo (official code)	NeurIPS 2023	`--weighting MoDo`
FORUM	ECAI 2024	`--weighting FORUM`
STCH (official code)	ICML 2024	`--weighting STCH`
ExcessMTL (official code)	ICML 2024	`--weighting ExcessMTL`
FairGrad (official code)	ICML 2024	`--weighting FairGrad`
DB-MTL	arXiv	`--weighting DB_MTL`
UPGrad (official code)	arXiv	`--weighting UPGrad`

Architectures	Venues	Arguments
Hard Parameter Sharing (HPS)	ICML 1993	`--arch HPS`
Cross-stitch Networks (Cross_stitch)	CVPR 2016	`--arch Cross_stitch`
Multi-gate Mixture-of-Experts (MMoE)	KDD 2018	`--arch MMoE`
Multi-Task Attention Network (MTAN) (official code)	CVPR 2019	`--arch MTAN`
Customized Gate Control (CGC), Progressive Layered Extraction (PLE)	ACM RecSys 2020	`--arch CGC`, `--arch PLE`
Learning to Branch (LTB)	ICML 2020	`--arch LTB`
DSelect-k (official code)	NeurIPS 2021	`--arch DSelect_k`

Datasets	Problems	Task Number	Tasks	multi-input	Supported Backbone
NYUv2	Scene Understanding	3	Semantic Segmentation+ Depth Estimation+ Surface Normal Prediction	✘	ResNet50/ SegNet
Cityscapes	Scene Understanding	2	Semantic Segmentation+ Depth Estimation	✘	ResNet50
Office-31	Image Recognition	3	Classification	✓	ResNet18
Office-Home	Image Recognition	4	Classification	✓	ResNet18
QM9	Molecular Property Prediction	11 (default)	Regression	✘	GNN
PAWS-X	Paraphrase Identification	4 (default)	Classification	✓	Bert

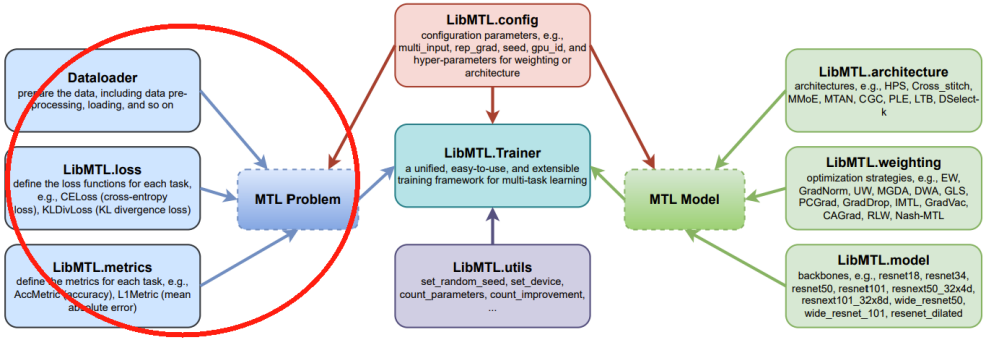
Modular Design in LibMTL



easy-to-use and well-extensible:

- customize your own MTL problem and use existing MTL methods implemented in LibMTL;
- develop your own MTL methods (e.g., architecture and weighting) and make a fair comparison with existing methods on the widely-used benchmark datasets.

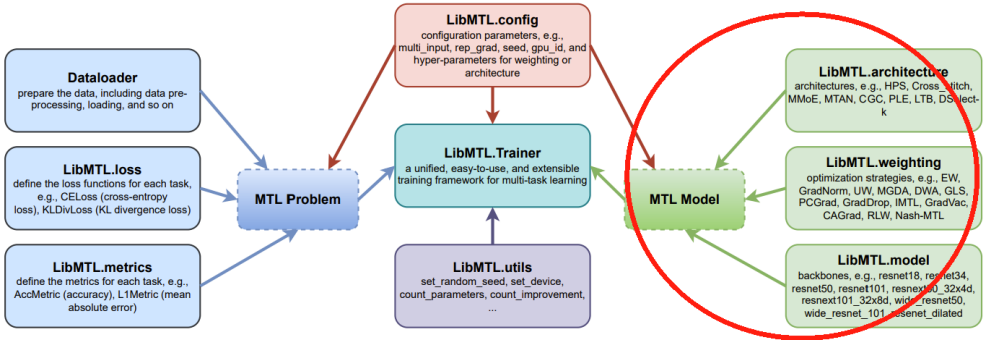
Modular Design in LibMTL



easy-to-use and well-extensible:

- customize your own MTL problem and use existing MTL methods implemented in LibMTL;
- develop your own MTL methods (e.g., architecture and weighting) and make a fair comparison with existing methods on the widely-used benchmark datasets.

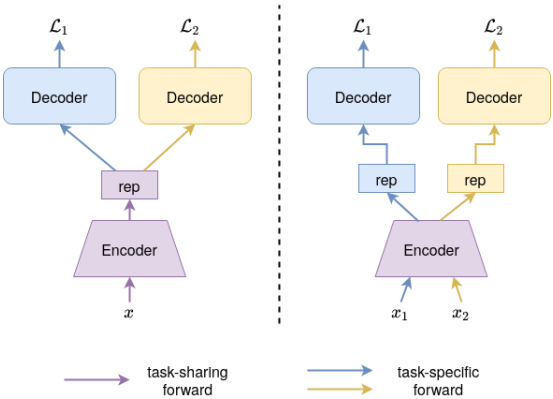
Modular Design in LibMTL



easy-to-use and well-extensible:

- customize your own MTL problem and use existing MTL methods implemented in LibMTL;
- develop your own MTL methods (e.g., architecture and weighting) and make a fair comparison with existing methods on the widely-used benchmark datasets.

Take HPS as An Example



- single-input problem (**left**), e.g., molecular property prediction;
- multi-input problem (**right**), e.g., image classification.

- `LibMTL.architecture`: hard parameter sharing (HPS);
- `LibMTL.model`: ResNet/Transformers for encoder, a linear layer for decoder;
- `LibMTL.weighting`: the optimization strategy (e.g., EW and MGDA) for handling multiple losses;

Welcome to use and contribute!

Overview

Tutorial Part 7: Open Challenges and Future Directions

Weiyu Chen

HKUST

August 29, 2025

Challenge 1: Theoretical Understanding

Problem:

- The theoretical foundations of many practical multi-objective deep learning methods are not fully understood.
- Research has mainly focused on convergence, with less attention on generalization error, which is crucial for real-world performance.

Future Direction:

- Develop broader, algorithm-agnostic generalization analyses.
- Theoretically investigate how network design choices affect Pareto set approximation.

Challenges 2 & 3: Efficiency and Scalability

Reducing Gradient Balancing Costs

- **Problem:** Gradient balancing methods, while effective, have significant computational overhead.
- **Future Direction:** Integrate gradient balancing with simpler methods like linear scalarization to reduce costs and enable large-scale use.

Dealing with a Large Number of Objectives

- **Problem:** The preference vector space grows exponentially with more objectives, making random sampling ineffective for learning the Pareto set.
- **Future Direction:**
  - Develop efficient sampling strategies for high-dimensional preference spaces.
  - Explore methods to automatically reduce or merge objectives based on their properties.

Challenge 4: Distributed Training

Problem:

- Most current MOO algorithms are designed for a single GPU or machine.
- Scaling to multi-GPU and distributed environments is critical as models and datasets grow, but it introduces unique challenges not seen in single-objective optimization.

Future Directions:

- **Efficient Communication:** Design methods for efficient gradient distribution and synchronization across multiple GPUs/nodes.
- **Privacy-Preserving MOO:** Develop techniques for collaborative training when data for different objectives is on separate devices and cannot be shared.

Challenge 5: Advancements in LLMs

Problem:

- Current MOO applications for LLMs are mostly concentrated on the Reinforcement Learning from Human Feedback (RLHF) stage.
- User preferences are often simplified into a basic preference vector, which may not capture the complexity of human needs.

Future Directions:

- **Expand MOO Application:** Apply MOO techniques to other stages of the LLM lifecycle, to better align models from the start.
- **Advanced Preference Modeling:** Explore more sophisticated methods to represent and incorporate complex and nuanced user preferences.

Challenge 6: Application in More Scenarios

The Untapped Potential:

- Most deep learning problems are inherently multi-objective, as models are evaluated on multiple criteria.
- These criteria often create natural trade-offs that are perfect candidates for MOO.

Future Direction:

- Actively leverage MOO methods to explicitly navigate these trade-offs in a wider range of deep learning applications.
- Move from single-metric optimization to a more holistic, multi-objective approach to model development.

THANK YOU!

Weiyu Chen, Baijiong Lin, Xiaoyuan Zhang, Xi Lin, Han Zhao

IJCAI 2025
Guangzhou August 29-31

Scan for our survey!

*We sincerely thank Yiheng Zhu (ZJU) and Yifei Shen (MSRA) for their valuable feedback on the "Applications in AI4Science" section.

References I

[1] A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, *et al.*, "Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai," *Information fusion*, vol. 58, pp. 82–115, 2020.

[2] Y. Gu, Q. Hu, S. Yang, *et al.*, "Jet-nemotron: Efficient language model with post neural architecture search," *arXiv preprint arXiv:2508.15884*, 2025.

[3] Y. Zhong, C. Ma, X. Zhang, *et al.*, "Panacea: Pareto alignment via preference adaptation for LLMs," in *Conference on Neural Information Processing Systems*, 2024.

[4] S. Luukkonen, H. W. van den Maagdenberg, M. T. Emmerich, and G. J. van Westen, "Artificial intelligence in multi-objective drug design," *Current Opinion in Structural Biology*, vol. 79, p. 102 537, 2023.

[5] S. Liu, E. Johns, and A. J. Davison, "End-to-end multi-task learning with attention," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

References II

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

[7] B. Lin, W. Jiang, F. Ye, *et al.*, "Dual-balancing for multi-task learning," *arXiv preprint arXiv:2308.12029*, 2023.

[8] L. Liu, Y. Li, Z. Kuang, *et al.*, "Towards impartial multi-task learning," in *International Conference on Learning Representations*, 2021.

[9] F. Ye, B. Lin, Z. Yue, P. Guo, Q. Xiao, and Y. Zhang, "Multi-objective meta learning," in *Conference on Neural Information Processing Systems*, 2021.

[10] S. Liu, S. James, A. Davison, and E. Johns, "Auto-Lambda: Disentangling dynamic task relationships," *Transactions on Machine Learning Research*, 2022.

References III

[11] F. Ye, B. Lin, X. Cao, Y. Zhang, and I. Tsang, "A first-order multi-gradient algorithm for multi-objective bi-level optimization," in *European Conference on Artificial Intelligence*, 2024.

[12] B. Lin, F. Ye, Y. Zhang, and I. Tsang, "Reasonable effectiveness of random weighting: A litmus test for multi-task learning," *Transactions on Machine Learning Research*, 2022.

[13] X. Lin, X. Zhang, Z. Yang, F. Liu, Z. Wang, and Q. Zhang, "Smooth tchebycheff scalarization for multi-objective optimization," in *International Conference on Machine Learning*, 2024.

[14] O. Sener and V. Koltun, "Multi-task learning as multi-objective optimization," in *Conference on Neural Information Processing Systems*, 2018.

[15] B. Liu, X. Liu, X. Jin, P. Stone, and Q. Liu, "Conflict-averse gradient descent for multi-task learning," in *Conference on Neural Information Processing Systems*, 2021.

References IV

[16] T. Yu, S. Kumar, A. Gupta, S. Levine, K. Hausman, and C. Finn, "Gradient surgery for multi-task learning," in *Conference on Neural Information Processing Systems*, 2020.

[17] A. Navon, A. Shamsian, I. Achituve, *et al.*, "Multi-task learning as a bargaining game," in *International Conference on Machine Learning*, 2022.

[18] B. Liu, Y. Feng, P. Stone, and Q. Liu, "FAMO: Fast adaptive multitask optimization," in *Conference on Neural Information Processing Systems*, 2023.

[19] X. Lin, H.-L. Zhen, Z. Li, Q.-F. Zhang, and S. Kwong, "Pareto multi-task learning," in *Conference on Neural Information Processing Systems*, 2019.

[20] D. Mahapatra and V. Rajan, "Multi-task learning with user preferences: Gradient descent with controlled ascent in Pareto optimization," in *International Conference on Machine Learning*, 2020.

References V

[21] D. Mahapatra and V. Rajan, "Exact Pareto optimal search for multi-task learning and multi-criteria decision-making," *arXiv preprint arXiv:2108.00597*, 2021.

[22] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007.

[23] E. Zitzler and L. Thiele, "Multiobjective optimization using evolutionary algorithms—a comparative case study," in *International Conference on Parallel Problem Solving From Nature*, 1998.

[24] X. Lin, Y. Liu, X. Zhang, F. Liu, Z. Wang, and Q. Zhang, "Few for many: Tchebycheff set scalarization for many-objective optimization," in *International Conference on Learning Representations*, 2025.

References VI

[25] L. Ding, Z. Chen, X. Wang, and W. Yin, "Efficient algorithms for sum-of-minimum optimization," in *International Conference on Machine Learning*, 2024.

[26] A. Navon, A. Shamsian, E. Fetaya, and G. Chechik, "Learning the Pareto front with hypernetworks," in *International Conference on Learning Representations*, 2021.

[27] X. Lin, Z. Yang, Q. Zhang, and S. Kwong, "Controllable Pareto multi-task learning," *arXiv preprint arXiv:2010.06313*, 2020.

[28] T. A. Tuan, N. V. Dung, and T. N. Thang, "A hyper-transformer model for controllable Pareto front learning with split feasibility constraints," *arXiv preprint arXiv:2402.05955*, 2024.

[29] M. Ruchte and J. Grabocka, "Scalable Pareto front approximation for deep multi-objective learning," in *IEEE International Conference on Data Mining*, 2021.

References VII

[30] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville, "FiLM: Visual reasoning with a general conditioning layer," in *Annual AAAI Conference on Artificial Intelligence*, 2018.

[31] A. Dosovitskiy and J. Djolonga, "You only train once: Loss-conditional training of deep networks," in *International Conference on Learning Representations*, 2020.

[32] W. Chen and J. Kwok, "Multi-objective deep learning with adaptive reference vectors," in *Conference on Neural Information Processing Systems*, 2022.

[33] D. S. Raychaudhuri, Y. Suh, S. Schulter, *et al.*, "Controllable dynamic multi-task architectures," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

References VIII

[34] N. Dimitriadis, P. Frossard, and F. Fleuret, "Pareto manifold learning: Tackling multiple tasks via ensembles of single-task models," in *International Conference on Machine Learning*, 2023.

[35] W. Chen and J. Kwok, "Efficient Pareto manifold learning with low-rank structure," in *International Conference on Machine Learning*, 2024.

[36] A. Tang, L. Shen, Y. Luo, S. Liu, H. Hu, and B. Du, "Towards efficient Pareto set approximation via mixture of experts based model fusion," *arXiv preprint arXiv:2406.09770*, 2024.

[37] L. P. Hoang, D. D. Le, T. A. Tuan, and T. N. Thang, "Improving Pareto front learning via multi-sample hypernetworks," in *Annual AAAI Conference on Artificial Intelligence*, 2023.

[38] J.-A. Désidéri, "Multiple-gradient descent algorithm (MGDA) for multiobjective optimization," *Comptes Rendus Mathematique*, vol. 350, no. 5-6, pp. 313–318, 2012.

References IX

[39] S. Liu and L. N. Vicente, "The stochastic multi-gradient algorithm for multi-objective optimization and its application to supervised machine learning," *Annals of Operations Research*, pp. 1–30, 2021.

[40] S. Zhou, W. Zhang, J. Jiang, W. Zhong, J. Gu, and W. Zhu, "On the convergence of stochastic multi-objective gradient manipulation and beyond," in *Conference on Neural Information Processing Systems*, 2022.

[41] H. Fernando, H. Shen, M. Liu, S. Chaudhury, K. Murugesan, and T. Chen, "Mitigating gradient bias in multi-objective learning: A provably convergent approach," in *International Conference on Learning Representations*, 2023.

[42] L. Chen, H. Fernando, Y. Ying, and T. Chen, "Three-way trade-off in multi-objective learning: Optimization, generalization and conflict-avoidance," in *Conference on Neural Information Processing Systems*, 2023.

References X

[43] P. Xiao, H. Ban, and K. Ji, "Direction-oriented multi-objective learning: Simple and provable stochastic algorithms," in *Conference on Neural Information Processing Systems*, 2023.

[44] C. Cortes, M. Mohri, J. Gonzalvo, and D. Storcheus, "Agnostic learning with multiple objectives," in *Conference on Neural Information Processing Systems*, 2020.

[45] P. Súkeník and C. Lampert, "Generalization in multi-objective machine learning," *Neural Computing and Applications*, pp. 1–15, 2024.

[46] P. Awasthi, N. Haghtalab, and E. Zhao, "Open problem: The sample complexity of multi-distribution learning for vc classes," in *Annual Conference on Learning Theory*, 2023.

[47] N. Haghtalab, M. Jordan, and E. Zhao, "On-demand sampling: Learning optimally from multiple distributions," in *Conference on Neural Information Processing Systems*, 2022.

References XI

[48] B. Peng, "The sample complexity of multi-distribution learning," in *Annual Conference on Learning Theory*, 2024.

[49] Z. Zhang, W. Zhan, Y. Chen, S. S. Du, and J. D. Lee, "Optimal multi-distribution learning," in *Annual Conference on Learning Theory*, 2024.

[50] M. Liu, X. Zhang, C. Xie, K. Donahue, and H. Zhao, "Online mirror descent for tchebycheff scalarization in multi-objective optimization," *arXiv preprint arXiv:2410.21764*, 2024.

[51] W. Chen and J. Kwok, "Pareto merging: Multi-objective optimization for preference-aware model merging," in *International Conference on Machine Learning*, 2025.

[52] L. Li, T. Zhang, Z. Bu, *et al.*, "Map: Low-compute model merging with amortized pareto fronts via quadratic approximation," *arXiv preprint arXiv:2406.07529*, 2024.

[53] T. Yu, D. Quillen, Z. He, *et al.*, "Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning," in *Conference on Robot Learning*, 2020.

References XII

[54] A. Rame, G. Couairon, C. Dancette, *et al.*, "Rewarded soups: Towards Pareto-optimal alignment by interpolating weights fine-tuned on diverse rewards," in *Conference on Neural Information Processing Systems*, 2023.

[55] R. Shi, Y. Chen, Y. Hu, *et al.*, "Decoding-time language model alignment with multiple objectives," in *Conference on Neural Information Processing Systems*, 2024.

[56] R. Rafailov, A. Sharma, E. Mitchell, C. D. Manning, S. Ermon, and C. Finn, "Direct preference optimization: Your language model is secretly a reward model," in *Conference on Neural Information Processing Systems*, 2023.

[57] Y. Xu, U. M. Sehwag, A. Koppel, *et al.*, "GenARM: Reward guided generation with autoregressive reward model for test-time alignment," in *International Conference on Learning Representations*, 2025.

References XIII

[58] B. Lin, W. Jiang, Y. Xu, H. Chen, and Y.-C. Chen, "PARM: Multi-objective test-time alignment via preference-aware autoregressive reward model," in *International Conference on Machine Learning*, 2025.

[59] H. Wang, M. Skreta, C. T. Ser, *et al.*, "Efficient evolutionary search over chemical space with large language models," in *International Conference on Learning Representations*, 2025.

[60] N. Ran, Y. Wang, and R. Allmendinger, "MOLLM: Multi-objective large language model for molecular design–optimizing with experts," *arXiv preprint arXiv:2502.12845*, 2025.

[61] X. Han, C. Shan, Y. Shen, *et al.*, "Training-free multi-objective diffusion model for 3d molecule generation," in *International Conference on Learning Representations*, 2024.

References XIV

[62] Y. Zhu, J. Wu, C. Hu, J. Yan, T. Hou, and J. Wu, "Sample-efficient multi-objective molecular optimization with GFlowNets," in *Conference on Neural Information Processing Systems*, 2023.

[63] M. Jain, S. C. Raparthy, A. Hernández-Garcıa, *et al.*, "Multi-objective GFlowNets," in *International Conference on Machine Learning*, 2023.

[64] X. Zhang, L. Zhao, Y. Yu, *et al.*, "LibMOON: A gradient-based multiobjective optimization library in PyTorch," in *Conference on Neural Information Processing Systems*, 2024.

[65] B. Lin and Y. Zhang, "LibMTL: A Python library for deep multi-task learning," *Journal of Machine Learning Research*, vol. 24, no. 209, pp. 1–7, 2023.