MAMO: Multi-task Architecture Learning via Multi-Objective and Gradients Mediative Kernel

Yuzheng Tan[b], **Guangneng Hu**[a,*] **and Shuxin Zhang**[b]

[a]School of Computer Science and Technology, Xidian university, Xi'an, China
[b]State Key Laboratory of Integrated Electromechanical Manufacturing of High-performance Electronic Equipments, Xidian university, Xi'an, China

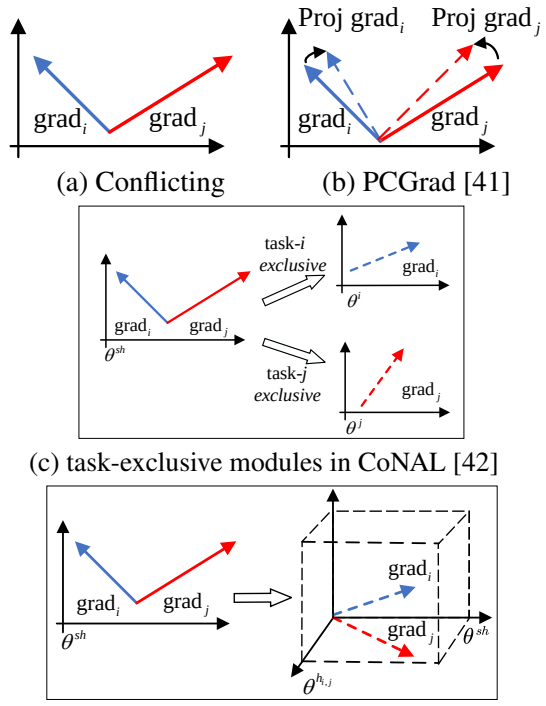
Abstract. Multi-task learning (MTL) is effective in solving multiple related tasks simultaneously by sharing knowledge. However, a key challenge hindering its applications is the task interference problem where different tasks compete with each other, leading to the gradients conflicts during optimization and suffering from negative transfer. One thread is to manipulate task gradients by adjusting conflicting directions, ignoring architecture learning. Another thread is to learn architectures by generating task-exclusive modules, ignoring all-task balances. We address the problem by proposing a novel Multi-task Architecture learning model via Multi-Objective (MAMO) optimization. It achieves the goal in two steps. First, for the competing tasks detected during architecture learning, MAMO automatically generates a new module of gradient mediative kernel (GMK). Second, MAMO finds a Pareto optimal solution that balances all tasks during model parameter learning. MAMO outperforms various MTL baselines on benchmarks with an effective model size. It is model-agnostic and can be integrated into other SOTA methods to promote their performance. Extensive ablation study is conducted to understand the working of MAMO.

1 Introduction

Multi-task learning exploits knowledge shared among related tasks to improve the generalization performance of multiple tasks [1, 43]. MTL offers a dual advantage: it is not only minimizing the overall training cost by learning multiple tasks concurrently, but also bolstering the performance of each individual task through the mutual sharing of knowledge among various tasks. Successful applications vary from computer vision [30, 8, 33, 39, 9] to natural language processing [5, 12, 35] and multimodal learning tasks [29, 28, 40].

Compared to single-task learning, the simultaneous training of multiple tasks can be challenging due to the task interference problem where tasks compete with each other, leading to gradients conflicts (see Fig. 1a). It is difficult to optimize the multitask objective since the task of larger magnitude may dominate the update and has negative effect on other tasks [42, 38]. It goes worse under the heterogeneous domains and distribution shift, suffering from negative transfer where the performance drops on one task conditioned on the learning of other tasks. Overall performance degenerates than that of learning them independently [37, 16]. One thread of mitigating conflicts is to manipulate gradients/ losses to make a balance among competing tasks. Gradient manipulation technique is to adjust gradients directions (see Fig. 1b), drop part of the gradient vectors, and rotate shared features [41, 3, 15]. The reweighting technique is to adaptively re-weight the loss functions by uncertainty, balance the learning pace among tasks, and learn loss weights by dynamic weight average [17, 2, 22].

The occurrence of gradient conflicts shows that not all tasks are strongly related and the fixed human-designed network structures like hard parameter sharing and soft parameter sharing could be



(a) Conflicting (b) PCGrad [41]

(c) task-exclusive modules in CoNAL [42]

(d) gradients mediative kernel (GMK) in our MAMO

Figure 1: Gradients conflicts and different mitigation methods. (a) Two tasks (i, j) have conflicting gradients, i.e., their cosine similarity is negative. (b) PCGrad mitigates the conflict by orthogonal projections to adjust the gradient direction. (c) CoNAL mitigates the conflict by generating task exclusive modules (parameterized with θ^i and θ^j respectively) optimized with a linear weighted sum loss. (d) Our MAMO mitigates the conflict by generating gradients mediative kernel (GMK) modules (parameterized with $\theta^{h_{i,j}}$) optimized with a multi-objective loss (see full MAMO in Fig. 2).

* Corresponding Author. Email: njuhgn@gmail.com

improper for a given MTL problem [23, 27, 10]. Therefore, another thread is to learn multitask architectures to mitigate task conflicts from the root [24, 36, 44]. Branch-based architecture learning methods automate to build a tree-structured network by learning where to branch with task-specific modules and layer-wise replacement [13, 32]. To kill two birds with one stone, a recent CoNAL method [42] was proposed to mitigate gradient conflicts during learning the architectures. The purely specific (task exclusive) modules (see Fig. 1c) are introduced as the candidate search space and CoNAL adaptively switches to task-exclusive module when gradient conflicts are detected during the architecture learning process.

Existing architecture learning methods (e.g. CoNAL) optimize a weighted sum of empirical risks for multiple tasks, called linear scalarization. In fact, the objective of MTL is to find solutions that are not dominated by any others (called Pareto optimal) and the problem of finding such Pareto optimal is called multi-objective optimization [31, 20, 25, 4]. Motivated by theoretical results [14] saying multi-objective optimizers can find Pareto optimal solutions that cannot be obtained via any linear scalarization objective, in this paper, we address the problem from the perspective of multi-objective optimization. To the best of our knowledge, we are the first to learn MTL architectures under multi-objective optimization.

We propose a novel Multi-task Architecture learning method under Multi-objective Optimization (MAMO). It achieves the goal by finding a Pareto optimal solution to balance among competing tasks, and generating mediative kernel modules to mitigate gradient conflicts during architecture learning. Instead of generating task-exclusive modules as that of CoNAL, our introduced gradients mediative kernel (GMK) modules mitigate the conflicts in the new high-dimensional joint space (see Fig. 1d). Operating on the GMK modules, MAMO can adaptively fuse the shared knowledge and dynamically learn to switch to task-specific modules during architecture learning. Besides theoretical-motivated advantages, MAMO also empirically outperforms dozens of strong MTL baselines on two benchmarks (the NYUv2 and Cityscapes datasets) with a reasonable model size. It is model-agnostic and thus can easily plug-and-play into SOTA methods to promote their performance.

The main contributions are summarized in the following:

- We propose a novel MAMO method—to the best of our knowledge, we are the first—to learn Multitask Architectures from the perspective of Multi-objective Optimization, well-motivated by recent theoretical results.
- In MAMO, we introduce the gradients mediative kernel (GMK) modules in the search space of neural architectures. It can adaptively fuse the shared knowledge and dynamically learn to switch to task-specific modules.
- On two MTL benchmarks (NYUv2 and CityScapes), our MAMO outperforms various strong baselines with a reasonable model size. It can be integrated into SOTA methods to improve their performance further.

2 Preliminary

Before introducing our proposed method, we first describe the problem setup of MTL and then introduce the background on Pareto optimality of multi-objective optimization.

2.1 MTL Setup

Consider a multi-task learning problem over an input space $\mathcal{X}$ and a collection of m-sized task spaces $\{\mathcal{Y}^t\}_{t\in[m]}$, and a collection of N-sized data samples $\{\mathbf{x}_i, y_i^1, \ldots, y_i^m\}_{i\in[N]}$ are also given where y_i^t is the label of the t-th task for the i-th data point.

Each task has a parametric hypothesis class as:

$$f^t(\mathbf{x}; \theta^{sh}, \theta^t) : \mathcal{X} \to \mathcal{Y}^t, \tag{1}$$

where θ^{sh} corresponds to shared parameters across all tasks while parameters θ^t are task-specific. The loss function of the t-th task is denoted by:

$$\mathcal{L}^t(\cdot,\cdot) : \mathcal{Y}^t \times \mathcal{Y}^t \to \mathbb{R}^+. \tag{2}$$

2.2 Multi-objective Optimization

Linear scalarization MTL can be formulated as linear scalarization (LS) optimization which uses the following linearly combined weighted summation loss [42]:

$$\min_{\theta^{sh}, \cup_{t\in[m]}\{\theta^t\}} \sum\nolimits_{t\in[m]} w^t \hat{\mathcal{L}}^t(\theta^{sh}, \theta^t), \tag{3}$$

where w^t is the weight for the t-th task and $\hat{\mathcal{L}}^t$ is the empirical loss computed over the data samples using $\mathcal{L}^t$.

Multi-objective MTL can also be formulated as multi-objective optimization which is to optimize a collection of possibly conflicting objectives (reflecting the reality of multiple tasks are competing with each other) denoted by a vector-valued loss [31]:

$$\begin{aligned} &\min_{\theta^{sh}, \cup_{t\in[m]}\{\theta^t\}} \mathbf{L}(\theta^{sh}, \cup_{t\in[m]}\{\theta^t\}) \\ =&\min_{\theta^{sh}, \cup_{t\in[m]}\{\theta^t\}} \left(\hat{\mathcal{L}}^1(\theta^{sh}, \theta^1), \ldots, \hat{\mathcal{L}}^m(\theta^{sh}, \theta^m)\right)^\top. \end{aligned} \tag{4}$$

The goal of multi-objective (MO) optimization is to achieve Pareto optimality which can be solved to local optimality via gradient descent methods, leveraging Karush-Kuhn-Tucker (KKT) conditions [7]:

1. Exist $a^1, \ldots, a^m \geq 0$, such that $\sum_{t\in[m]} a^t = 1$ and

$$\sum\nolimits_{t\in[m]} a^t \nabla_{\theta^{sh}} \hat{\mathcal{L}}^t(\theta^{sh}, \theta^t) = 0; \tag{5}$$

2. For $\forall t \in [m]$, $\nabla_{\theta^t} \hat{\mathcal{L}}^t(\theta^{sh}, \theta^t) = 0$.

The theoretical results [14] say, under some conditions, the linear scalarization is incapable of fully exploring the Pareto front, especially for those Pareto optimal solutions that strike the balanced trade-offs between multiple tasks. Since our goal is to attack the task interference and make a balance among all tasks, we address the MTL architecture learning problem via multi-objective optimization. Later in experiments, we will empirically ablate the impact of LS and MO (see Sec. 4.4.2).

3 The MAMO Approach

In this section, we introduce the proposed MAMO method, including the motivation of gradients mediative kernel (GMK) modules and the strategy of mitigating gradient conflicts during architecture learning. We finally analyze the complexity of the model and computation.

3.1 Search Space

Our MAMO method is to find an architecture that mitigate gradient conflicts for multiple competing tasks in the search space. The core in the search space is the gradients mediative kernel (GMK) modules, each of which could be FC, Conv, or ResNet layer depending on

the given MTL problem. As a result, MAMO contains two kinds of modules: the all-shared module and gradients mediative kernel module, defined as:

- **Definition 3.1** (All-shared module): It is used by all tasks and is updated by the gradients backpropagated from the losses of all tasks during the training.
- **Definition 3.2** (Gradients mediative kernel module): It is used by only part of tasks (they are conflicting detected during the architecture learning) and is updated solely by the gradients backpropagated from the losses of these tasks during the training.

Different from existing architecture learning methods like CoNAL [42] which uses task-exclusive module when gradient conflicts are detected (see Fig. 1(c)), our MAMO mitigates the conflict by generating gradients mediative kernel (GMK) modules (see Fig. 1(d)). One advantage of GMK module is to reserve knowledge from all-shared modules instead of discarding them totally as that of CoNAL. The reason is that although the conflicting gradients will increase the training losses of conflicting tasks and slow down the convergence speed, it may also play a role similar to regularization [19], reducing the risk of overfitting of the conflicting tasks, thereby improving their generalization performance [16].

We now provide a brief explanation on why the addition of GMK module can mitigate the gradient conflicts. Assume two tasks (i, j) are in conflict at some layer in the network, i.e., $\cos(\text{grad}_i, \text{grad}_j) < 0$ or equivalently $\langle \text{grad}_i, \text{grad}_j \rangle < 0$, where grad_i represents the gradient of i-th task at the corresponding conflicting layer. Assume the shared parameters are θ^{sh}, then we have $\langle \text{grad}_i(\theta^{sh}), \text{grad}_j(\theta^{sh}) \rangle = -c$ where c is a positive value since they are conflicting. When operating on our introduced gradients mediative kernel (GMK) module parameterized with $\theta^{h_{i,j}}$, we can divide the gradient vectors into two components consisting of all-shared and GMK parts:

$$\begin{aligned}\langle \text{grad}_i, \text{grad}_j \rangle = &\langle \text{grad}_i(\theta^{sh}),\ \text{grad}_j(\theta^{sh}) \rangle \\ &+ \langle \text{grad}_i(\theta^{h_{i,j}}),\ \text{grad}_j(\theta^{h_{i,j}}) \rangle\end{aligned}$$

Since the first term (regarding all-shared part) on the right hand equals $-c$, we only need to ensure that the second term (regarding GMK part) $\geq c$ to mitigate the gradient conflicts. In fact, we have more flexible operation space after the introducing of GMK modules: we only need to ensure the whole gradient vectors (the term on the left hand) are not conflicting instead of requiring each of them (all-shared part and GMK part) must be divided into $-c$ and c. For example, they can be divided into $-c'$ and c' for tasks (i', j') where $c \neq c'$. The trade-off between sharing knowledge and task-specific learning is dynamically optimized based on GMK modules. Theoretically, introducing GMK modules between conflicting tasks can satisfy this condition and we will introduce the architecture learning algorithm under multi-objective optimization in the next section.

3.2 *Overall architecture*

The overall architecture of MAMO is shown in Fig. 2a. For learning m tasks with L-layer network, MAMO consists of: (ℓ indexes layer and i, j index tasks in the following)

- an all-shared encoder network $\{f_S^{(\ell)}\}$;
- task-specific encoder networks $\{h_{i,j}^{(\ell)}\}$ implementing the gradients mediative kernel modules (note the GMK modules have a sparse structure described later in the Section 3.3);
- task-specific adaptive fusion layers with parameters $\{\beta_{i,j}\}$;
- task-specific decoder networks $\{g_i\}$ for corresponding task heads.

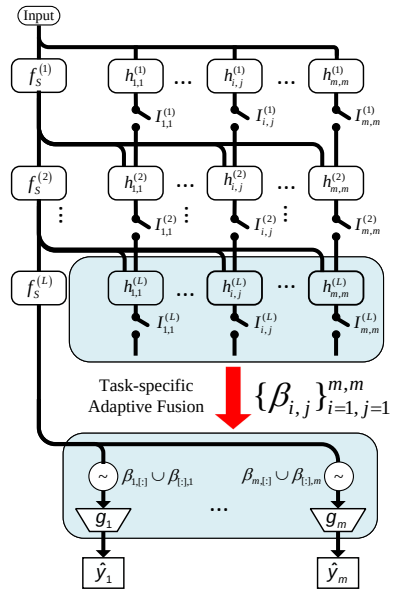
For the all-shared network $\{f_S^{(\ell)}\}$, their outputs is computed by:

$$o_f^{\ell+1} = f_S^{(\ell+1)}(o_f^{\ell}). \tag{6}$$

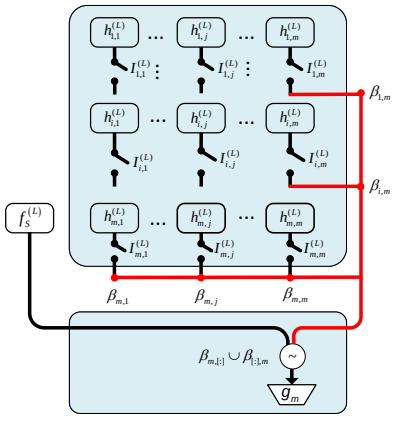
For the task-specific encoder networks $\{h_{i,j}^{(\ell)}\}$ implementing GMK modules, their output is computed by:

$$o_{i,j}^{\ell+1} = h_{i,j}^{(\ell+1)}(o_f^{\ell},\ I_{i,j}^{(\ell)} \cdot o_{i,j}^{\ell}), \tag{7}$$

where the input to GMK consists of both shared knowledge o_f^{ℓ} and task-specific gradients mediative information in preceding layer $o_{i,j}^{\ell}$. The core of architecture search is to determine $I_{i,j}^{(\ell)}$, indicating the gradient conflict detected between tasks (i, j) at the ℓ-th layer and the corresponding mediative module is activated if and only if it is one (details of search strategy described in the next section).



(a) Overall architecture of MAMO



(b) Adaptive fusion layer for the m-th task

Figure 2: Illustration of the proposed MAMO approach. (a) Overall architecture of MAMO: f_S is the shared module, $h_{i,j}$ is the gradients mediative kernel (GMK) module, g_m is the corresponding task head, and the adaptive fusion layer is detailed in (b). (b) The task-specific adaptive fusion module in MAMO: an example for the m-th task where only the m-th row and the m-th column of the GMK matrix are eligible for fusion.

For adaptive task-specific fusion layer $\{\beta_{i,j}\}_{i=1,j=1}^{m,m}$ connecting the encoder networks and the decoder network, it adaptively fuses the shared knowledge and task-specific information. Take the m-th task for example as shown in Fig. 2b, its output is computed by:

$$\hat{y}_m = g_m(o_f^L, \underbrace{\sum_{j\in[m]} \beta_{m,j} \cdot I_{m,j}^{(L)} \cdot o_{m,j}^L}_{\text{the } m\text{-th row}}, \underbrace{\sum_{i\in[m]} \beta_{i,m} \cdot I_{i,m}^{(L)} \cdot o_{i,m}^L}_{\text{the } m\text{-th col}}), \quad (8)$$

where the diagonal elements of the fusion matrix $\{\beta_{i,i}\}$ are dummy since a task is not conflicting with itself.

Algorithm 1 Multi-task Architecture learning via Multi-objective Optimization (MAMO)

Input: $\mathcal{D}$: Training samples, Z: Number of conflict detection
Output: Learned architecture parameters & model parameters

1: Initialize and pre-train all-shared modules;
2: **for** $z = 1$ to Z **do**
3: Sample a mini-batch from $\mathcal{D}$;
// Learning architecture parameters
4: Compute conflicts indicator matrix $\mathcal{I}$ by Eq. (13);
5: Generate gradients mediative kernel (GMK) modules based on the computed indicator;
// Learning model parameters
6: Compute losses and gradients of new KKT conditions shown in Eq. (14);
7: Update model parameters using multiple-gradient descent algorithm (MGDA) [7];
8: **end for**

3.3 Search Strategy

Introducing gradients mediative kernel (GMK) modules can mitigate task conflicting issues. After generating a GMK module, a new Pareto stationary point will be reached. Thereby it may activate new GMK modules along with the optimization going on. Therefore, we detect gradient conflicts during architecture learning to dynamically construct a conflicting indication matrix at each layer.

Formally, at the ℓ-th layer, the gradients cosine matrix $\{C_{i,j}\} \in \mathbb{R}^{m\times m}$ stores the scores of gradient vectors where the element:

$$C_{i,j} = \cos(\text{grad}_i, \text{grad}_j). \quad (9)$$

The proxy conflicting matrix $\{\widetilde{I}_{i,j}\}$ represents the task conflicting situation:

$$\widetilde{I}_{i,j} = \begin{cases} 0, & C_{i,j} \geq 0 \\ 1, & C_{i,j} < 0. \end{cases} \quad (10)$$

Backpropagating all data samples to compute the gradients is computational prohibitive. Therefore, we adopt the mini-batch training trick to accumulate the gradient conflict indications on the fly:

$$Q_{i,j} = \sum_{\text{mini-batch}} \widetilde{I}_{i,j}. \quad (11)$$

Based on the accumulated conflict indications, we convert it to be a zero-one indicator matrix by comparing with an average statistic:

$$Q_{average} = \frac{1}{N_Z} \sum\nolimits_{i,j} Q_{i,j}, \quad (12)$$

where N_Z is the number of non-zero elements in Q. The setting of the indicator matrix is thus by:

$$\mathcal{I}_{i,j} = \begin{cases} 0, & Q_{i,j} < Q_{average} \\ 1, & Q_{i,j} \geq Q_{average}. \end{cases} \quad (13)$$

The reason behind the threshold cutoff based on the average statistic is that the gradients mediative kernel module between two tasks is activated if and only if their conflicts are above the degree of average conflicts among all tasks. The side effect is to learn a sparse structure of the GMK modules during architecture search, reducing the risk of overfitting for the MTL model and achieving an automatic capacity control. We will investigate the effect of threshold cutoff in later experimental ablation study (see Sec. 4.4.4).

3.4 Learning Process

Similar to the learning process of CoNAL [42], our MAMO alternatively optimizes two subproblems. The upper-level subproblem is to determine architecture parameters. The lower-level subproblem is to update model parameters using gradient descent methods. In summary, the learning algorithm for our MAMO model is shown in Algo. 1.

Learning architecture parameters (Lines 4-5, Algo.1) During learning the architectures in the search space, if a gradient conflict is detected between tasks (i, j) at some layer by Eq. (13), then a new GMK module $h_{i,j}$ is generated. The core of architecture search is to determine such conflicts, i.e., $I_{i,j}^{(\ell)}$ in Eq. (7). See details described in the above Section 3.3.

Learning model parameters (Lines 6-7, Algo.1) After the determination of architecture parameters, the problem is reduced as a standard MTL via multi-objective optimization. The new KKT conditions of MAMO with GMK modules are formulated as follows:
1. Exist $a^1, \ldots, a^m \geq 0$, such that $\sum_{t\in[m]} a^t = 1$ and

$$\sum\nolimits_{t\in[m]} a^t \nabla_{\theta^{sh},\theta^{h_{i,j}}} \hat{\mathcal{L}}^t(\theta^{sh}, \theta^{h_{i,j}}, \theta^t) = \sum_{t\in[m]} a^t \big(\nabla_{\theta^{sh}} \hat{\mathcal{L}}^t(\theta^{sh}, \theta^{h_{i,j}}, \theta^t) + \nabla_{\theta^{h_{i,j}}} \hat{\mathcal{L}}^t(\theta^{sh}, \theta^{h_{i,j}}, \theta^t)\big) = 0; \quad (14)$$

2. For $\forall t \in [m]$, $\nabla_{\theta^t} \hat{\mathcal{L}}^t(\theta^{sh}, \theta^{h_{i,j}}, \theta^t) = 0$.

We adopt existing algorithms to learn model parameters, i.e., the multiple-gradient descent algorithm (MGDA) [7, 31].

3.5 Comparison with SOTA Methods

Higher capacity of search space We compare our MAMO with the CoNAL [42], an SOTA model of recent MTL architecture learning

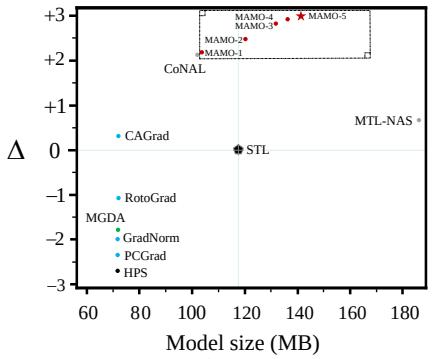


Figure 3: Model performance (Δ) vs model size of various MTL methods on NYUv2 dataset. The reference point is single-task learning (STL). MAMO has different sizes MAMO-{i}, see Sec. 4.4.1.

methods. In the following, m the number of tasks, and L the number of layers (dimension of parameters d omitted due to the same backbone used). For the search space, our MAMO learns the gradients mediative kernel modules $\{h_{i,j}^{(\ell)}\}$ which have space capacity of $\mathcal{O}(m \times m \times L)$, while CoNAL is $\mathcal{O}(m \times L)$. As a result, our MAMO has higher ability to search a better MTL architecture over CoNAL. More importantly, the computation cost of searching the architectures is the same between our MAMO and the CoNAL, i.e., $\mathcal{O}(m^2)$. This is because the CoNAL still has to compute the cosine similarity of pairwise gradient vectors among all tasks, i.e. $\cos(\text{grad}_i, \text{grad}_j)$ for $\forall i \in [m], j \in [m]$, when it performs the conflict-noticing architecture learning process.

Better trade-off performance against size For the complexity of model parameters, it depends on the learned architectures, i.e., how many GMK modules are activated. Fig. 3 shows the overall performance (in terms of Δ defined in Eq. (15)) vs model size (in terms of the number of parameters in mega bytes) of various MTL methods on the NYUv2 dataset. In details, our large MAMO (i.e., MAMO-5, see more Sec. 4.4.1), has a size of around 140M which achieves relative 3.0% improvements. In contrast, the baseline MTL-NAS [11] has larger size (180M) but worse performance (0.66%) than our MAMO. Our proposed MAMO method can learn an effective MTL architecture with a reasonable model size achieving the best performance improvement, see more in Section 4.4.1.

4 Experiments

We evaluate our proposed MAMO method with different groups of MTL approaches on two benchmarks. Then we demonstrate the flexibility of our MAMO by integrating it into SOTA methods to promote their performance further. We study the effect of multi-objective optimization, the impact of search strategy including the number of conflicts detection and the setting for threshold cutoff. Finally, we show the optimization dynamics.

4.1 Experimental Setup

We describe the benchmark datasets, evaluation metrics for MTL overall performance, implementation details for reproducibility, and state-of-the-art methods for comparison in this section.

Datasets Following CoNAL [42], we conducted experiments on two benchmark datasets including CityScapes [6] and NYUv2 [34]. Same with CoNAL, we use the version published in MTAN [22].

Metrics Besides the performance metrics in each task, following [26], we compute an **overall metric** Δ to evaluate an MTL model as the average gain w.r.t. the single task learning (STL) model over all tasks and each task's all evaluation metrics:

$$\Delta = \frac{1}{m} \sum\nolimits_{i \in [m]} (-1)^{s_i} \frac{\text{MTL}_i - \text{STL}_i}{\text{STL}_i} \times 100\% \qquad (15)$$

where MTL_i denotes the performance of the MTL model in terms of the i-th evaluation metric while STL_i is for the STL model. The s_i equals 0 if the metric is the higher the better else 1 otherwise.

Since Δ is an average statistic, it will be largely influenced by an extreme value of some individual task. We compute the **win rate (Win)** as a more robust statistic, accounting for how often MTL_i outperforming STL_i for all $i \in [m]$.

Implementation For fairness, we use the same backbone architecture as CoNAL[1] [42]. Following MTAN [22], we train all models with ADAM [18] using a learning rate 10^{-4} with batch size of 4 on NYUv2 and 16 on CityScapes, and halve the learning rate in 40k iterations. We will release our code with learned architectures checkpoints for benefiting the research community.

[1] https://github.com/yuezhixiong/CoNAL

Table 1: Results on the CityScapes dataset. Best results are in boldface in each scenario. STL is the reference point and the relative performance improvements are computed for MTL methods ("+" indicating MTL better than STL while "-" indicating worse). The last two columns are computed across all tasks and metrics (↑ indicating the higher the better).

Method	Segmentation		Depth		Δ↑	**Win**↑
	mIoU	Pix Acc	Abs Err	Rel Err		
STL	69.60	91.82	0.0125	45.95	0	0
HPS	+0.96	+0.29	-0.80	-7.48	-1.75	50%
GradNorm	-3.18	-1.07	-12.80	+3.11	-3.49	25%
PCGrad	+0.93	+0.20	0	-2.41	-0.31	50%
CAGrad	+0.56	+0.08	-2.39	+1.32	+0.42	75%
RotoGrad	-2.05	-0.49	-8.00	+2.11	-2.10	25%
MGDA	-3.67	-1.33	+2.40	+4.26	-1.38	50%
MTL-NAS	-6.10	+1.76	-15.20	-4.30	-6.84	25%
CoNAL	+0.10	-0.03	+1.60	**+7.18**	+2.21	75%
MAMO	**+1.39**	**+0.33**	**+4.80**	+3.74	**+2.57**	**100%**

4.2 Baselines

We compare with different groups of various methods.

- **i) STL and HPS**: The single-task learning (STL) method trains each task separately. The hard parameter sharing (HPS) architecture uses a shared encoder for all the tasks while reserves task-specific decoders for each task.
- **ii) Gradients manipulation**: GradNorm [2], PCGrad [41], CAGrad [21], RotoGrad [15].
- **iii) Multi-objective optimization**: MGDA [31].
- **iv) Architecture learning**: MTL-NAS [11], CoNAL [42].

4.3 Main Results

4.3.1 Results on Cityscapes

The results on the Cityscapes dataset are shown in Table 1. First, the single task learning (STL) method is a strong baseline and we can find that the manually-designed network architectures (HPS) face the serious negative transfer issue. Second, the architecture learning methods (CoNAL) generally reduce the risk of negative transfer but is still struggling (MTL-NAS). Third, the gradients manipulation methods show a diverse difference where most of them have difficulty in improving overall MTL performance. Fourth, modeling the multitask learning as multi-objective optimization has its potential advantages (MGDA vs GradNorm). Our MAMO achieves consistent improvements over both individual tasks and overall MTL metric.

Notably, since tasks are competing with each other in multi-task learning, a larger improvement in one task usually hurts another task. For example, GradNorm has +3.11 Rel Err improvement on the Depth task, but it has -3.18 mIoU degradation on the Segmentation task, leading to an overall -3.39% degradation over all tasks and metrics. In contrast, our MAMO can mediate task conflicts and achieve an overall 2.57% improvements. Attacking the task interference is our goal in this paper. Overall, the trend is very consistent over all tasks, metrics, and datasets.

Table 2: Result on NYUv2 dataset. Best results are in boldface in each scenario. STL is the reference point and the relative performance improvements are computed for MTL methods ("+" indicating MTL better than STL while "-" indicating worse). The last two columns are computed across all tasks and metrics (↑ indicating the higher the better).

Method	Segmentation		Depth		Surface Normal					$\Delta\uparrow$	**Win↑**
					Angle Distance		Within t°				
	mIoU	Pix Acc	Abs Err	Rel Err	Mean	Median	11.25	22.5	30		
STL	54.07	75.51	0.4074	0.1681	22.11	15.59	38.36	64.42	74.80	0	0
HPS	+0.31	+0.06	+4.61	+6.78	-5.61	-11.33	-7.74	-4.62	-3.13	-2.74	44%
GradNorm	+0.57	+0.13	+5.20	+8.56	-6.19	-9.55	-7.84	-5.26	-3.62	-2.00	44%
PCGrad	-1.24	-0.38	+5.81	+3.80	-5.79	-8.01	-7.14	-4.33	-3.00	-2.25	22%
CAGrad	+0.07	+0.06	-4.29	-5.41	+1.49	+1.98	-1.87	-1.08	-0.68	+0.25	44%
RotoGrad	+0.03	-0.34	+6.72	+6.60	-4.29	-5.83	-8.55	-3.95	-2.40	-1.22	33%
MGDA	-12.83	-5.64	-0.39	+1.84	+0.27	+0.19	+0.15	+0.26	+0.21	-1.77	66%
MTL-NAS	-0.27	-0.39	+5.57	**+8.98**	-1.40	-2.69	-0.54	-0.32	-2.96	+0.66	22%
CoNAL	0	+0.18	+5.49	+7.01	+0.49	+2.50	+2.50	+1.00	+0.46	+2.18	88%
MAMO	**+1.20**	**+0.74**	**+6.70**	+8.80	**+1.22**	**+2.88**	**+2.81**	**+1.50**	**+1.09**	**+3.00**	**100%**

4.3.2 *Results on NYUv2*

The results on the NYUv2 dataset are shown in Table 2. We have the following observations. First, the manually-designed network architectures are largely impacted by the datasets. For example, HPS shows a more severe negative transfer on the NYUv2 dataset than that on the Cityscapes dataset. Second, using a single alone technique like gradients manipulation (GradNorm, PCGrad, and RotoGrad) or architecture search (MTL-NAS) is not enough to improve the MTL performance with a large margin. Third, we may need to combine multiple techniques to achieve a better performance like CoNAL which integrates both gradients manipulation and architecture search in one model. Finally, our MAMO achieves the best overall MTL metric and also shows a strong performance over individual tasks. Note, although MTL-NAS and PCGrad have the same win rate (**Win**) over STL (22%), MTL-NAS is better than STL (+0.66) while PCGrad worse than STL (-2.25) in terms of overall average performance (Δ). In summary, together with the Fig. 3, our MAMO learns an effective architecture with a reasonable model size, achieving the best overall MTL performance on both datasets.

4.4 *Ablation Study*

We first show MAMO achieves a good trade-off performance against model size. We ablate the impact of GMKs and multi-objective optimization in MAMO. We then show that MAMO can be integrated into other SOTA methods to promote their performance. We analyse the choices of search strategy regarding threshold cutoff. Finally, we give optimization curves and computational efficiency of MAMO.

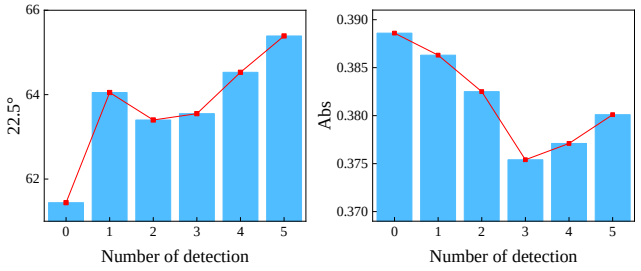


Figure 4: Performance changes with the number of detections (where number 0 corresponds to the HPS method). Left: on the Surface Normal task in terms of the 22.5°↑ metric (the higher the better). Right: on the Depth task in terms of the Abs↓ metric (the lower the better).

Table 3: Number of activated GMKs varying with conflicts detection.

Detection	MAMO-1	MAMO-2	MAMO-3	MAMO-4	MAMO-5
NYUv2	7	4	2	1	1
Cityscapes	2	1	1	1	0

4.4.1 *Trade-off performance against model size*

Number of activated GMKs As shown in Algo. 1, the key to search strategy is the number of conflicts detection performed by MAMO. We first show the number of activated GMKs varying with conflicts detection as shown in Table 3. As more GMKs are generated, the parameter scaling will be getting larger. The severity of conflicts are higher in early detection where seven GMKs are generated in the first conflict detection. Gradients conflict situation gets mitigated along with more GMK modules activated.

Performance against size As the number of detection increases, more gradients mediative kernel (GMK) modules will be automatically generated according to the conflicts indicator, leading to an MAMO that has bigger model capacity to fit the data samples. In details, as shown in Fig. 4 (left), performing only one conflict detection will lead to large performance improvements over the fixed HPS architecture on the Surface Normal task. This demonstrates the necessity of architecture learning. In Fig. 4 (right), the tendency shows the performance of MAMO stably increases along with the detection times rasing from one to three, and then MAMO shows a sign of saturation when the number of detection reaches some threshold point, say four on the Depth task, due to the fact that an over-complex model might be overfitted.

4.4.2 *Impact of multi-objective and GMKs*

Impact of multi-objective MTL can be formulated as both linear scalarization as in Eq. (3) and multi-objective (MO) optimization as in Eq. (4). We ablate the impact of MO by removing it, i.e., replacing MO with LS. As shown in Table 4, optimizing MAMO via multi-objective optimization has advantage over linear scalarization on the NYUv2 dataset. Theoretical result [14] reveals that LS is incapable of fully exploring Pareto front and MO has the potential of finding balanced solutions. Our empirical results show the advantage of MO over LS under multitask architecture learning.

Impact of GMKs As shown in Fig. 1d, our MAMO mitigates the conflict by generating gradients mediative kernel (GMK) modules. We ablate the impact of GMK by removing it, i.e., replacing GMK

Table 4: Impact of multi-objective (MO) and gradients mediative kernel (GMK) of MAMO on the NYUv2 dataset.

Method	Segmentation		Depth		Surface Normal				
					Angle Distance		Within t°		
	mIoU↑	Pix Acc↑	Abs Err↓	Rel Err↓	Mean↓	Median↓	11.25↑	22.5↑	30↑
HPS	54.24	75.56	0.3886	0.1567	23.34	17.98	35.39	61.44	72.46
MAMO	**54.72**	**76.07**	**0.3801**	**0.1533**	**21.83**	**15.14**	**39.44**	**65.39**	**75.62**
w/o MO	54.06	75.65	0.3850	0.1563	21.99	15.20	39.32	65.07	75.15
w/o GMK	53.47	75.55	0.3859	0.1550	22.52	15.81	37.59	63.98	74.40

with task-exclusive modules as in CoNAL [42] (see Fig. 1c). As shown in Table 4, it is necessary for our GMK modules to mitigate the conflicts in the new high-dimensional joint space (see Eq. 3.1) since it can adaptively fuse the shared knowledge and task-specific modules during architecture learning. The trade-off between sharing knowledge and task-specific learning is dynamically optimized based on GMK modules.

4.4.3 *Integrating MAMO with SOTA methods*

Our MAMO is model-agnostic and it can be integrated into other SOTA methods. We show such flexibility in Fig. 5 where MAMO are integrated with CAGrad (left column) and PCGrad (right column) respectively. Both SOTA methods are belonging to the gradients manipulation methods, see Fig. 1. We can see that integrating MAMO with SOTA methods can promote their performance further, demonstrating the necessity of architecture search and multi-objective optimization besides gradient projections.

4.4.4 *Search strategy for indicator threshold cutoff*

As shown in Eq. (13), one key to search strategy is the setting for threshold cutoff to compute the gradients conflicting indicator, which controls the automatic generation of GMK modules and the model capacity. Besides the average statistic $Q_{average}$, we show results of keeping the TopK most conflicting gradients kernel in Fig. 6. The larger the K, the more GMK modules generated, thus the bigger capacity. Since it is tedious to tune the K, we use the average statistic which is automatically computed along with the architecture search.

4.4.5 *Optimization curves and computation efficiency*

Optimization curves We show the optimization dynamics of our MAMO comparing with HPS evaluated on the Normal Surface task

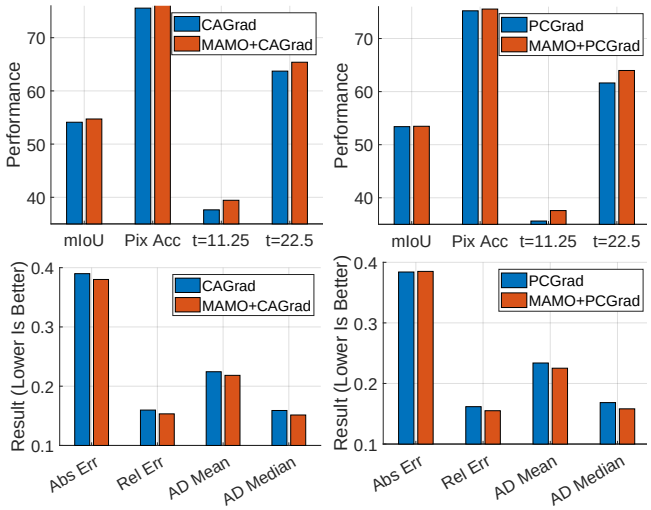


Figure 5: Integrating MAMO into CAGrad and PCGrad on NYUv2 dataset. Upper part: Higher is better, on the Segmentation and Surface Normal (Within t°) tasks. Lower part: Lower is better, on the Depth and Surface Normal (Angle Distance normalized to 1) tasks.

Table 5: Computational efficiency (Speedup↑) vs overall performance on the NYUv2 dataset.

Method	STL	HPS	CAGrad	CoNAL	MAMO
Speedup↑	1.0x	1.61x	0.73x	0.60x	0.56x
Δ↑	0	-2.74	+0.25	+2.18	+3.00

(Within t=11.25°) on the NYUv2 dataset in Fig. 7, and we can see that our MAMO reaches a better Pareto optimal solution than the HPS. Moreover, HPS saturates very soon in the first 50 epoches and then gets stuck in the local optimal. Our MAMO still learns hungry till 250 epoches and converges to a higher performance solution.

Computation efficiency Our MAMO achieves a good trade-off over computational efficiency (Speedup↑) and overall performance (Δ↑) by comparing with typical baselines, as shown in Table 5.

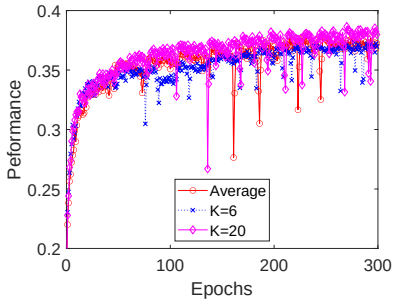


Figure 6: Search setting for threshold cutoff: Average vs TopK. On the Normal Surface task in terms of the Within t=11.25°↑ metric on NYUv2 dataset (the higher the better).

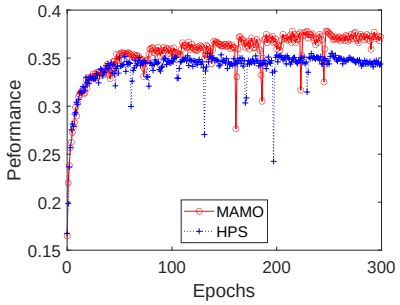


Figure 7: Optimization performance curve. On the Normal Surface task in terms of the Within t=11.25°↑ metric on NYUv2 dataset (the higher the better).

5 Conclusion

We proposed a novel MAMO method to learn Multi-task Architectures from the perspective of Multi-objective Optimization. We introduced the gradients mediative kernel (GMK) modules in the search space and designed an effective and efficient search strategy. The adaptive task-specific layer of MAMO adaptively fused the shared knowledge and dynamically learned to switch to task-specific modules. We demonstrated the effectiveness by comparing with various groups of MTL approaches on challenging benchmarks. We analyzed the complexity of MAMO and it achieved automatic capacity control by comparing the model performance vs model size. We empirically showed that STOA methods can be improved further by integrating with MAMO. In the future, we will extend our MAMO to large number of tasks.

Acknowledgements

This work is supported by National Natural Science Foundation of China (No.52322507 and No.62306220)

References

[1] R. Caruana. Multitask learning. *Machine learning*, 28, 1997.

[2] Z. Chen, V. Badrinarayanan, C.-Y. Lee, and A. Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International conference on machine learning*, 2018.

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

[4] Z. Chen, J. Ge, H. Zhan, S. Huang, and D. Wang. Pareto self-supervised training for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

[5] R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, 2008.

[6] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *IEEE conference on computer vision and pattern recognition*, 2016.

[7] J.-A. Désidéri. Multiple-gradient descent algorithm (mgda) for multiobjective optimization. *Comptes Rendus Mathematique*, 350(5-6), 2012.

[8] C. Ding, Z. Lu, S. Wang, R. Cheng, and V. N. Boddeti. Mitigating task interference in multi-task learning via explicit task routing with non-learnable primitives. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2023.

[9] L. Fei, L. Sun, M. Kudo, K. Kimura, et al. Structured sparse multi-task learning with generalized group lasso. In *Proceedings of the European Conference on Artificial Intelligence*. IOS Press, 2023.

[10] Y. Gao, J. Ma, M. Zhao, W. Liu, and A. L. Yuille. Nddr-cnn: Layerwise feature fusing in multi-task cnns by neural discriminative dimensionality reduction. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2019.

[11] Y. Gao, H. Bai, Z. Jie, J. Ma, K. Jia, and W. Liu. Mtl-nas: Task-agnostic neural architecture search towards general-purpose multi-task learning. In *IEEE Conference on computer vision and pattern recognition*, 2020.

[12] R. Giaquinto, D. Zhang, B. Kleiner, Y. Li, M. Tan, P. Bhatia, R. Nallapati, and X. Ma. Multitask pretraining with structured knowledge for text-to-SQL generation. In A. Rogers, J. Boyd-Graber, and N. Okazaki, editors, *Annual Meeting of the Association for Computational Linguistics*, 2023.

[13] P. Guo, C.-Y. Lee, and D. Ulbricht. Learning to branch for multi-task learning. In *International conference on machine learning*, 2020.

[14] Y. Hu, R. Xian, Q. Wu, Q. Fan, L. Yin, and H. Zhao. Revisiting scalarization in multi-task learning: A theoretical perspective. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[15] A. Javaloy and I. Valera. Rotograd: Gradient homogenization in multitask learning. In *International Conference on Learning Representations*, 2022.

[16] J. Jiang, B. Chen, J. Pan, X. Wang, D. Liu, M. Long, et al. Forkmerge: Mitigating negative transfer in auxiliary-task learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[17] A. Kendall, Y. Gal, and R. Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018.

[18] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *The International Conference for Learning Representations*, 2015.

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

[20] X. Lin, H.-L. Zhen, Z. Li, Q.-F. Zhang, and S. Kwong. Pareto multi-task learning. In *Advances in neural information processing systems*, 2019.

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

[22] S. Liu, E. Johns, and A. J. Davison. End-to-end multi-task learning with attention. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2019.

[23] M. Long, Z. Cao, J. Wang, and P. S. Yu. Learning multiple tasks with multilinear relationship networks. *Advances in neural information processing systems*, 2017.

[24] Y. Lu, A. Kumar, S. Zhai, Y. Cheng, T. Javidi, and R. Feris. Fully-adaptive feature sharing in multi-task networks with applications in person attribute classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017.

[25] 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.

[26] K.-K. Maninis, I. Radosavovic, and I. Kokkinos. Attentive single-tasking of multiple tasks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2019.

[27] I. Misra, A. Shrivastava, A. Gupta, and M. Hebert. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[28] C. Oguz, P. Denis, E. Vincent, S. Ostermann, and J. van Genabith. Find-2-find: Multitask learning for anaphora resolution and object localization. In H. Bouamor, J. Pino, and K. Bali, editors, *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2023.

[29] G. Pantazopoulos, M. Nikandrou, A. Parekh, B. Hemanthage, A. Eshghi, I. Konstas, V. Rieser, O. Lemon, and A. Suglia. Multitask multimodal prompted training for interactive embodied task completion. In H. Bouamor, J. Pino, and K. Bali, editors, *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2023.

[30] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le. Regularized evolution for image classifier architecture search. In *AAAI conference on artificial intelligence*, 2019.

[31] O. Sener and V. Koltun. Multi-task learning as multi-objective optimization. *Advances in neural information processing systems*, 2018.

[32] G. Shi, Q. Li, W. Zhang, J. Chen, and X.-M. Wu. Recon: Reducing conflicting gradients from the root for multi-task learning. In *The International Conference on Learning Representations*, 2023.

[33] H. Shi, S. Ren, T. Zhang, and S. J. Pan. Deep multitask learning with progressive parameter sharing. In *IEEE International Conference on Computer Vision*, 2023.

[34] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *12th European Conference on Computer Vision*, 2012.

[35] T. Sun, Z. He, Q. Zhu, X. Qiu, and X. Huang. Multitask pre-training of modular prompt for Chinese few-shot learning. In A. Rogers, J. Boyd-Graber, and N. Okazaki, editors, *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2023.

[36] S. Vandenhende, S. Georgoulis, B. De Brabandere, and L. Van Gool. Branched multi-task networks: Deciding what layers to share. In *British Machine Vision Conference*, 2020.

[37] T. Xie, K. Wang, S. Lu, Y. Zhang, K. Dai, X. Li, J. Xu, L. Wang, L. Zhao, X. Zhang, et al. Co-net: Learning multiple point cloud tasks at once with a cohesive network. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023.

[38] E. Yang, J. Pan, X. Wang, H. Yu, L. Shen, X. Chen, L. Xiao, J. Jiang, and G. Guo. Adatask: A task-aware adaptive learning rate approach to multi-task learning. In *AAAI Conference on Artificial Intelligence*, 2023.

[39] F. Ye, X. Wang, Y. Zhang, and I. W. Tsang. Multi-task learning via time-aware neural ode. In *International Joint Conference on Artificial Intelligence*, 2023.

[40] S. Yu. Mcssme: Multi-task contrastive learning for semi-supervised singing melody extraction from polyphonic music. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.

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

[42] Z. Yue, Y. Zhang, and J. Liang. Learning conflict-noticed architecture for multi-task learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023.

[43] Y. Zhang and Q. Yang. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering*, 34(12), 2021.

[44] J. Zhao, W. Lv, B. Du, J. Ye, L. Sun, and G. Xiong. Deep multi-task learning with flexible and compact architecture search. *International Journal of Data Science and Analytics*, 2021.