Pattern Recognition 158 (2025) 110950

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Dual-mask: Progressively sparse multi-task architecture learning

Jiejie Zhao [a], Tongyu Zhu [b,*], Leilei Sun [b], Bowen Du [a,e], Haiquan Wang [d], Lei Huang [c]

[a] Zhongguancun Laboratory, Beijing, China
[b] CCSE, School of Computer Science and Engineering, Beihang University, Beijing 100191, China
[c] CCSE, Institute of Artificial Intelligence, Beihang University, Beijing 100191, China
[d] School of Software, Beihang University, Beijing, China
[e] School of Transportation Science and Engineering, Beihang University, Beijing, China

ARTICLE INFO

Keywords:
Multi-task learning
Neural network
Sparse network

ABSTRACT

Multi-task architecture learning has achieved significant success by learning optimal sharing architectures for different tasks. However, previous works to learn branched architectures for different tasks can sometimes lead to unsatisfying multi-task performance, as not all detailed branches are relevant to a specific task. Task-relevant architectures can be sparse, including only partial channels or layers in the entire architecture (i.e., a sub-network). In addition, most previous works rely on a heuristic architecture selection procedure that could not support continuous architecture optimization. To this end, in this paper, we propose *dual-mask*, a progressively sparse multi-task architecture learning method. Starting with a task-free architecture, it identifies the informative features along two-level, channels and layers, for each task, while suppressing conflicting or noisy parts in a differentiable manner, so that better task-specific sub-networks are captured. Specifically, the channel and layer selection modules produce respective hybrid binary and real value masks, designed to pick salient channels and layers for each task, respectively. To jointly optimize masks with model parameters, we propose an importance-guided relaxation method for solving the stochastic binary optimization problem, after which the interference or noise parts can be pruned by masks. Additionally, a progressive training strategy with continuation is provided that gradually sparsity the task-specific sub-networks. Experiments show that *dual-mask* achieves superior performance than SOTA multi-task methods.

1. Introduction

In recent years, there has seen unprecedented growth in the use of Multi-Task Learning (MTL) combined with deep Convolutional Neural Networks (CNNs) [1,2]. Deep CNNs represent the powerful modeling capability of hierarchical features, yet demanding massive amounts of data, memory, and computational resources. Compared with the single-task paradigm, MTL reduces inference latency and increases data efficiency via training multiple tasks simultaneously, which is critical for devices with limited computational budget. Furthermore, with certain shared representations, MTL models have the potential to enhance the performance of deep CNNs. This has allowed MTL to be applied to a variety of computer vision tasks, such as face detection [3,4].

Within deep CNNs, a significant factor influencing the performance of MTL models is the selection of sharing architectures, which determines how parameters are shared across tasks. A well-designed sharing architecture should capture both commonalities across tasks and task-specific inductive bias. However, there are an enormous number of possible options to tune an advanced MTL architecture, especially for considerable tasks and layers. Traditional MTL methods rely on hand-designed architectures, which may fall into a sub-optimal result, and meanwhile costs huge efforts in the trial-and-error process. In addition, poorly designed sharing architectures may lead to task interference or fail to fully exploit commonalities among tasks, both of which reduce the performance of MTL models.

To tackle this problem, different solutions have been proposed. At first, the typical way is to establish the parameter sharing mechanism by assigning a separate network to each task and designing various connections between the tasks [5]. Such methods leave more room for task specialization, resulting in better performance. However, these methods suffer from the problem of task interference, as such dense MTL networks provide continuous and dense sharing architectures that are susceptible to interference raised by distantly related tasks. Recently, inspired by the success of Neural Architecture Search (NAS) and pruning techniques in various computer vision tasks, there are

* Correspondence to: No. 37 Xueyuan Road, Haidian District, Beijing, 100191, PR China.
E-mail addresses: zhaojiejie@zgclab.edu.cn (J. Zhao), zhutongyu@buaa.edu.cn (T. Zhu), leileisun@buaa.edu.cn (L. Sun), dubowen@buaa.edu.cn (B. Du), whq@buaa.edu.cn (H. Wang), huangleiai@buaa.edu.cn (L. Huang).

https://doi.org/10.1016/j.patcog.2024.110950
Received 13 August 2022; Received in revised form 26 August 2024; Accepted 26 August 2024
Available online 28 August 2024

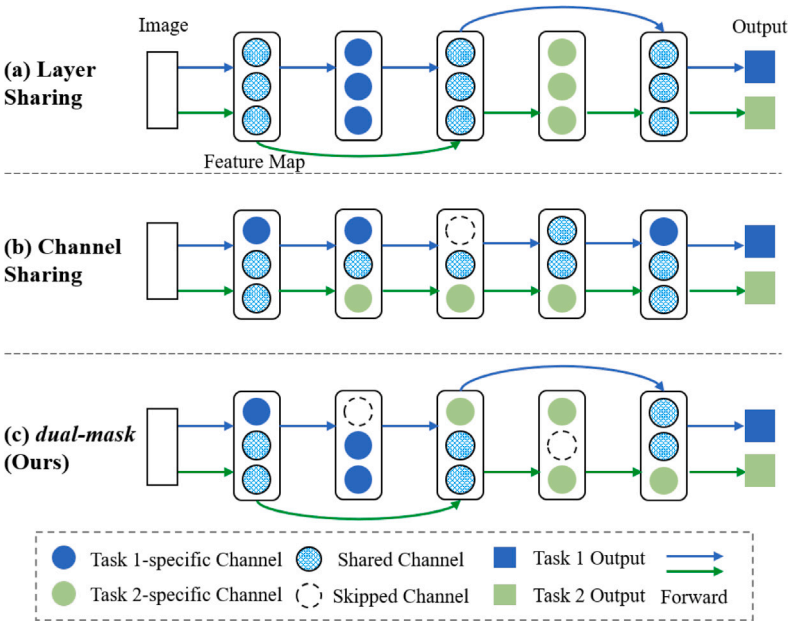

Fig. 1. MTL models with different sparse sharing schemes. (a) Layer sharing scheme selects layers to execute for each task adaptively. (b) Channel sharing scheme partitions the channels among tasks heuristically. (c) Our method synergistically selects the optimal channels and layers for each task in a differentiable manner.

two kinds of methods addressing this problem by leveraging discrete and sparse sharing architectures, enabling more flexible sharing across tasks. The first one aims to learn the layer sharing architectures as well as model parameters via NAS methods, such as gradient descent [6] and so on (e.g., AdaShare [7], see Fig. 1(a)). The other one selects the channel sharing architectures heuristically based on task-conditional networks. It executes separated forward passes for each task utilizing a tailored selection strategy of channels (e.g., MR [8], see Fig. 1(b)).

Despite these methods achieving promising results, they still have some limitations. First, previous works have solely exploited redundancy from a single perspective within MTL networks [7,8]. However, for each task, reducing channel and layer redundancies is crucial for suppressing the conflicting or noisy features [9]. Secondly, there is yet no efficient way to learn diversified and finer-grained (channel-level) sparse sharing architectures for MTL. Specifically, as the granularity of sharing refines from coarse to fine, the number of alternative architecture designs grows, which poses challenges to existing NAS methods. Furthermore, the architecture selection schemes for different tasks are diverse. For example, the significance of channels can vary for a given task. However, binary architectures used in existing NAS methods do not have enough capacity to model such sharing schemes. Finally, to handle the problem of task interference, most existing fine-grained sharing architecture learning methods force MTL models to be sparse at the beginning, which sacrifice inductive bias among tasks.

To tackle the above-mentioned challenges, we propose *dual-mask* (Fig. 1(c)), a progressively sparse multi-task architecture learning method. This method gradually learns optimal and sparse task-specific sub-networks from a dense network in a differentiable manner, enabling the finding of the flexible and efficient feature sharing architectures. It exploits the task-relevant architectures from two principal aspects, channels and layers. To achieve this, the channel and layer selection modules produce respective hybrid binary and real value masks, designed to pick salient channels and layers for each task, respectively. To resolve the non-differentiability issue when training sparse masks, a simple yet effective importance-guided relaxation method is proposed, which jointly learns the architecture parameters (i.e., masks) and the model parameters. Next, we design a framework to integrate these two modules with existing task-conditional based MTL networks. Moreover, we design a progressive training strategy with continuation that gradually increases mask sparsity, thereby increasing inductive bias across tasks. Finally, experiments show that *dual-mask* outperforms SOTA MTL baselines. Our method can be easily integrated into existing deep CNNs. Thus, it also benefits extensive computer vision tasks built on CNNs to find the optimal sharing architecture. The major contributions of our work can be summarized as follows:

- A novel progressively sparse multi-task architecture learning method, namely *dual-mask*, is proposed, which selects the salient channels and layers from a dense network for each task in a differentiable manner, enabling the findings of the flexible and efficient feature sharing architectures.
- A simple yet effective importance-guided relaxation method is proposed to learn sparse, high-quality masks for tasks. It enables the simultaneous learning of both hybrid binary and real-value masks and model parameters.
- A progressive training strategy with continuation is designed to induce mask sparsity gradually instead of forcing the mask to be sparse at the beginning, which benefits the MTL model by increasing inductive bias across tasks.

The rest of the paper is organized accordingly: Section 2 introduces existing research related to our work. Section 3 elaborates on our proposed method. Section 4 conducts the experiments to evaluate our method. Section 5 summarizes this paper.

2. Related works

In this section, we briefly discuss the related studies, including multi-task learning and sparse networks, and point out the limitations of existing studies.

2.1. Multi-task learning

Early Multi-Task Learning (MTL) methods focused on studying parameter sharing by shallow and linear models [10], such as LASSO framework. They usually lack the ability to represent complex nonlinear relationships among different tasks. Recently, deep learning-based MTL methods have attracted a lot of attention due to their powerful representation ability. Within the deep neural networks, the popular MTL methods can be typically classified into two distinct types: hard

parameter sharing schemes and soft parameter sharing schemes. In hard parameter sharing schemes, all tasks share bottom layers and learn task-specific top layers separately [11–13]. In soft parameter sharing schemes, each task has its own network with a mechanism for feature sharing between networks [5]. An alternative strand of MTL methods is based on task-conditional networks, which used masks to activate a task-specific parameter subset of a shared network, and then performed a separate forward pass for each task [14]. TR [14] is a typical task-conditional MTL network with hard binary masks that are randomly initialized and fixed during training. Compared with soft and hard sharing methods, such methods not only scale well with more tasks, but also could mitigate task interference, making them highly applicable to various real-world setups [15]. Therefore, we proceed along this direction to propose our novel method.

Most of the aforementioned methods use hand-crafted network architectures that might lead to sub-optimal solutions. Recent several works have attempted to find the advanced MTL network architectures automatically rather than heuristically. For example, NAS-MTL [16], AdaMerging [17], and CoNAL [18] are MTL frameworks that consist multiple task-specific backbones, and their goal is to find the layer-wise feature sharing architecture. However, these methods struggle to scale effectively as the number of tasks and the depth of the backbones increase. Vandenhende et al. [19] proposed a method to learn tree-structured MTL network architectures through a task affinity-guided greedy optimization method. AdaShare [7] learns to choose different subsets of layers for tasks in a shared network through a task-specific policy. It allows for the learning of much more flexible feature sharing architectures beyond tree-like architectures [19], as demonstrated effective in [20]. In addition, it uses an end-to-end learning method that is more effective than the task affinity-guided greedy search method. Unlike recent task-conditional based MTL methods that use fixed masks during training, recent works have introduced promising alternatives. Notable works include MT-SFG [21], MR [8], SE-MTL [22], and MPPS [23]. SE-MTL [22] learns soft parameter partitions with task-specific squeeze-and-excitation modulation. MR [8] employs a dynamic mask updating scheme to maximize the inductive bias across tasks.

In stark contrast to these methods, we produce flexible and sparse sharing MTL networks by learning hybrid binary and real-value masks to achieve channel-level and layer-level architecture selection in a differentiable manner.

2.2. Sparse networks

Pruning [24] and Neural Architecture Search (NAS) [25] are the dominant sparsity methods to find compact neural networks. Classical pruning methods adopted a pre-defined strategy to remove neurons, channels, or even layers according to varying removal criteria, such as magnitude [24]. For example, *Karnin et al.* [26] removed weights based on the sensitivity. *Han et al.* [24] pruned weights according to their L_2 or L_1 norms. *Dong et al.* [27] explored weight pruning based on second derivative information. Such methods typically first train the dense model fully and then select parameters for removal. Another method pruned a network by approximating l_0-regularization with reparameterization [28].

NAS methods aim to seek architectures with optimal performance potential. Using reinforcement learning, previous NAS methods have already obtained the appropriate architectures whose performance outperformed the manually designed ones [29]. For example, MnasNet [30] introduces a novel factorized hierarchical search space and used a reinforcement-learning based search algorithm to discover resource-efficient mobile CNN models. However, such methods required high computational costs during searching. To reduce the computational cost, recent work utilized more efficient search techniques, such as gradient-based methods [6]. They rely on gradient estimation methods (e.g., STE [31]) for estimating the gradient of discrete variables, which can be biased and have high variance. Other recent methods use reparameterization tricks to make continuous relaxations of discrete variables [32]. Our work follows this direction, guiding the learning process of task-specific sub-networks by considering the importance of channels and layers to highlight the task-relevant features while suppressing the interference or noise parts.

3. Methodology

In this section, we introduce *dual-mask*. We begin by introducing the channel selection and layer selection modules. Following that, we show how to integrate the two modules with existing MTL methods based on task-conditional networks. Finally, we present the progressive training method with continuation, and loss function.

Consider a set of K tasks $\mathcal{T} = \{\mathcal{T}_1, \dots, \mathcal{T}_K\}$ to be learned, associated with a dataset $\mathcal{D} = \{(\mathbf{x}_n, y_{n,k})\}_{n\in[N],k\in[K]}$, where N refers to the number of samples and K denotes the number of tasks. A typical MTL model contains a standard shared convolutional neural network of depth D and a different prediction layer tailored for each task k, namely Backbone Network $\mathcal{E}$, to learn all tasks simultaneously. In this configuration, the convolutional filters of the network $\mathcal{E}$ are regarded as model parameters. Let $\mathbf{h}_{k,d}, \mathbf{h}_{k,d+1}, \mathbf{w}_d^{\mathcal{E}}$ be the input feature maps, output feature maps, and model parameters of shared convolutional kernel for task k in layer d, respectively, where $\mathbf{h}_{k,d} \in \mathbb{R}^{p_d\times w_d\times h_d}$, $\mathbf{h}_{k,d+1} \in \mathbb{R}^{p_{d+1}\times w_{d+1}\times h_{d+1}}$, $\mathbf{w}_d^{\mathcal{E}} \in \mathbb{R}^{p_{d+1}\times p_d\times s_d\times s_d}$, and p_d, w_d, h_d, s_d correspond to the dimensions of the channel, height, width and kernel size in the dth representation, respectively. Formally, the output of the dth convolutional layer for task k can be defined as $\mathbf{h}_{k,d} = f(\mathbf{h}_{k,d-1} * \mathbf{w}_d^{\mathcal{E}})$, where $f(\cdot)$ is an activation function (e.g. ReLU) and $*$ is the convolution operation function.

In this paper, we explore a new progressively sparse multi-task architecture learning method, namely *dual-mask*. The backbone architecture of our method can be the same as the typical MTL method. To achieve flexible architecture selection, two different kinds of architectures within $\mathcal{E}$ from coarse to fine-grained are explored, including channels and layers. This method decides which channels and layers are selected to be shared among tasks and which are selected as task-specific.

3.1. Channel selection module

Recent works show that, in the typical MTL model, the salient channels for different tasks are distinct [8]. We assume that dynamically activating the sub-networks of important channels for different tasks from $\mathcal{E}$ independently emerges as a promising method to mitigate the task interference. Therefore, the channel selection module is proposed to generate sparse channel selection mask which serves to identify the important channels for tasks. Since the selected channels among tasks usually have partial overlap, this module also achieves channel-level feature sharing while mitigating the task interference. This module is illustrated in Fig. 2(a).

Specifically, given the trainable variable $\mathbf{s}_{k,d}^c \in \mathbb{R}^{p_d}$, the channel selection module aims to learn the parameters of $\mathcal{E}$ and the binary channel selection mask $\mathbf{m}_{k,d}^c$ for each task and layer jointly in a differentiable manner, where p_d is the number of channels in dth layer. Since each element of $\mathbf{M}^c$ can be discrete, it usually utilizes the Heaviside step function $H : \mathbb{R}_{\neq 0} \to \{0, 1\}$ to perform the binarization process. However, during training, this process cannot directly back-propagate since the discrete channel selection mask is non-differentiable. To prevent this, one solution is to use logistic function to relax the variable $\mathbf{s}_{k,d}^c$ for each task and layer:

$$\mathbf{z}_{k,d}^c = \sigma(\beta \cdot \mathbf{s}_{k,d}^c) = \frac{1}{1 + \exp(-\beta \cdot \mathbf{s}_{k,d}^c)}, \tag{1}$$

where β is a hyperparameter. Note that the logistic function $\sigma(\cdot)$ converges to discrete endpoints when β approaches infinity. $\mathbf{z}_{k,d}^c$ as a

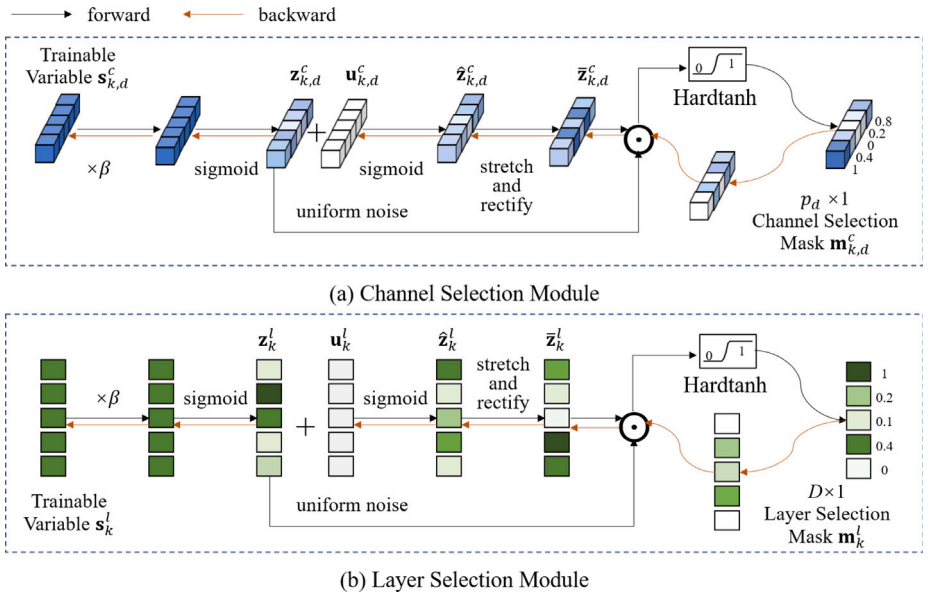

Fig. 2. The details of the Channel Selection Module and Layer Selection Module for training are illustrated in (a) and (b), respectively.

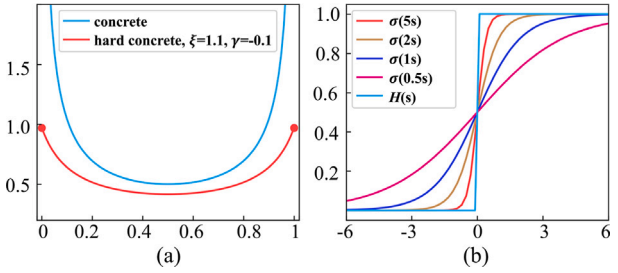

Fig. 3. (a) The binary concrete and the hard concrete distribution with location $\log z = 0$ and temperature $\beta = 0.5$; (b) Optimization with different functions in Continuation.

probability vector can be used to estimate the importance level of the corresponding channels to tasks, which saves training time without involving extra parameters.

However, with Eq. (1), the train-time architecture is dense, which means all channels always have non-zero contribution to tasks, and thus tend to suffer from task interference. To maintain a sparse architecture, we introduce hard-concrete distribution [28] (see Fig. 3 (a)) conditioned on probability value $\mathbf{z}^c_{k,d}$ to relax the discrete masks to continuous variable, which is a stretched version of binary concrete distribution [32]:

$$\hat{\mathbf{z}}^c_{k,d} = \sigma((\log \mathbf{u}^c_{k,d} - \log(1-\mathbf{u}^c_{k,d}) + \log \mathbf{z}^c_{k,d})/\beta), \tag{2}$$

$$\bar{\mathbf{z}}^c_{k,d} = \hat{\mathbf{z}}^c_{k,d}(\zeta-\gamma)+\gamma, \tag{3}$$

$$\hat{\mathbf{m}}^c_{k,d} = \min(1, \max(0, \bar{\mathbf{z}}^c_{k,d})), \tag{4}$$

where $u^c_{k,d,i}$ samples from the U(0, 1) uniform distribution, and hyper-parameters ζ and γ reflect stretching level. Under this setup, β is the temperature of the sigmoid that controls how closely the hard-concrete distribution approximates the binary distribution. Eq. (4) employs hard-tanh function to sample hard values $\hat{\mathbf{m}}^c_{k,d}$ during the forward pass and gradients are obtained from soft values $\bar{\mathbf{z}}^c_{k,d}$ during the backward pass. By using the straight-through estimator [33], the gradient of hardtanh(·) is estimated as:

$$\text{hardtanh}'(z) = \begin{cases} 1, & \text{if} \quad 0 \le z \le 1 \\ 0, & \text{otherwise.} \end{cases}$$

Due to the hard-concrete approximation, $\hat{\mathbf{m}}^c_{k,d}$ is now a continuous value in the range of [0, 1]. In inference time, we do not add uniform noise and therefore $\min(1, \max(0, \sigma(\log \mathbf{z}^c_{k,d}/\beta)(\zeta-\gamma)+\gamma))$ is employed to generate $\hat{\mathbf{m}}^c_{k,d}$.

The above-mentioned method ignores the importance of channels. To further improve it, we provide a new relaxation method that additionally considers channel importance, allowing it to pay more attention to task-relevant essential channels $\mathbf{z}_k$, making the sampling much more effective, and Eq. (4) can be rewritten as:

$$\bar{\mathbf{m}}^c_{k,d} = \mathbf{z}^c_{k,d} \odot \bar{\mathbf{z}}^c_{k,d}, \qquad \mathbf{m}^c_{k,d} = \min(1, \max(0, \bar{\mathbf{m}}^c_{k,d})). \tag{5}$$

The advantages of our method are: (1) the channel selection masks are learned in an end-to-end manner without heuristic-based predefining or updating adopted in MR [8]. (2) The additional channel importance branch could better guide the sampling toward finding an advantaged architecture. (3) In addition, the proposed relaxation method can model the hybrid binary and real-value masks, which enhances the representation capacity of channel-wise sharing architectures.

3.2. Layer selection module

Existing deep CNNs based MTL contain layer sparsity, as demonstrated in [7]. This suggests that not all layers are of equal importance for each task. Consequently, we aim to select a sub-network that contains part of layers in $\mathcal{E}$ for each task, thereby the task-specific execution path is distinct. To achieve this, we introduce the layer as another structural unit to support coarse-grained feature sharing, similar to the role of channels in channel-wise selection. Our layer selection module aims to generate sparse layer selection mask, which serves to select the most relevant layers for each task. The structure of the layer selection module is illustrated in Fig. 2(b).

Specifically, given the trainable variable $\mathbf{s}^l_k \in \mathbb{R}^D$ for task k, as the same with channel selection module, we first compute $\mathbf{z}^l_k$ as follows to estimate the importance level of the representations in different layers to various tasks separately:

$$\mathbf{z}^l_k = \sigma(\beta \cdot \mathbf{s}^l_k). \tag{6}$$

Then, we employ the proposed relaxation method in Section 3.1 on $\mathbf{z}^l_k$ to generate the discrete and sparse layer selection mask, which is computed by:

$$\hat{\mathbf{z}}^l_k = \sigma((\log \mathbf{u}^l_k - \log(1-\mathbf{u}^l_k) + \log \mathbf{z}^l_k)/\beta), \tag{7}$$

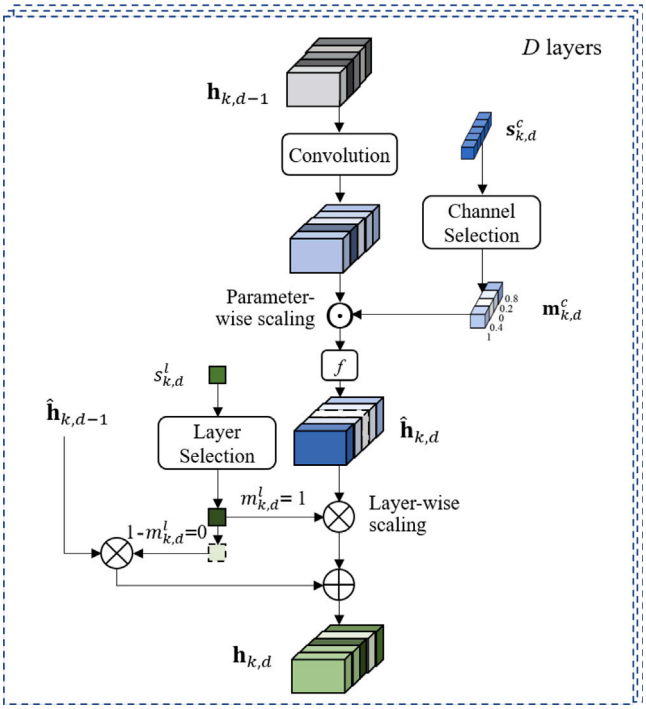

Fig. 4. Overview of proposed *dual-mask*.

$$\bar{\mathbf{z}}_k^l = \hat{\mathbf{z}}_k^l(\zeta - \gamma) + \gamma, \tag{8}$$

$$\bar{\mathbf{m}}_k^l = \mathbf{z}_k^l \odot \bar{\mathbf{z}}_k^l, \tag{9}$$

$$\mathbf{m}_k^l = \min(1, \max(0, \bar{\mathbf{m}}_k^l)), \tag{10}$$

where $\mathbf{m}_k^l \in \mathbb{R}^D$ determines the layer subset selected to execute for task k. If $m_{k,d}^l = 0$, the dth layer is skipped; otherwise, it is selected. During training, the layer selection module uses the proposed relaxation method based on hard-concrete distribution [28] to get the continuous mask $\mathbf{m}_k^l$ for backpropagation. Therefore, both mask $\mathbf{m}_k^l$ and the parameters of MTL network are jointly learned in an end-to-end manner.

3.3. Dual-mask network architecture

Fig. 4 illustrates an overview of *dual-mask*. Here, we use the fully shared convolutional neural network as the backbone to integrate channel and layer selection modules, which is similar to task-conditional based MTL networks. The design principle of the proposed *dual-mask* is to apply our designed two modules within the MTL model's each convolutional layer in both training and testing, so that the channels and layers are learned selectively for all tasks, contributing to reducing the task interference.

As a running example, when the network performs feedforward operations for a single input $\mathbf{h}_{k,d-1}$ of task k, it will output feature maps of the dth convolutional layer first. Then, the channel selection and layer selection modules generate masks $\mathbf{m}_{k,d}^c, m_{k,d}^l$ for task k, respectively. Next, $\mathbf{m}_{k,d}^c$ and $m_{k,d}^l$ are multiplied to convolution output to adjust the final output $\mathbf{h}_{k,d}$ that determines the subset of channels and layers used to perform actual calculation. Specifically, with $\mathbf{m}_{k,d}^c$, the output of dth layer for task k is computed as:

$$\hat{\mathbf{h}}_{k,d} = f((\mathbf{h}_{k,d-1} * \mathbf{w}_d^{\mathcal{E}}) \odot \mathbf{m}_{k,d}^c), \tag{11}$$

where $\odot$ denotes the channel-wise multiplication. Given the ith data instance $\mathbf{x}_i$, since the input of the first layer in our method is $\mathbf{x}_i$ for tasks, thus $\mathbf{h}_{k,0} = \mathbf{x}_i$. By introducing $\mathbf{m}^c$ and task-specific forward pass, the outputs are now task-dependent. Moreover, the less informative features are pruned and only the task-relevant features are passed to the next layer. With $m_{k,d}^l$, the output of dth layer for task k is computed as:

$$\begin{aligned}\mathbf{h}_{k,d} &= g(\hat{\mathbf{h}}_{k,d-1}; \hat{\mathbf{h}}_{k,d}; m_{k,d}^l), \\ &= \hat{\mathbf{h}}_{k,d-1} \cdot (1 - m_{k,d}^l) + \hat{\mathbf{h}}_{k,d} \cdot m_{k,d}^l,\end{aligned} \tag{12}$$

where $g(\cdot)$ is a gate function to ensemble the results from the current layer and the previous layer. In this way, the unimportant layers are directly skipped, and the salient layers are passed. In other cases, the dth layer can be instantiated as convolutional layers with weighted shortcut connections, similar to ResNets.

3.4. Progressive training with continuation

To further obtain approximately discrete solutions for channel selection and layer selection masks in a continuous optimization scheme, we draw inspiration from recent studies on continuation methods [34], which simplify the complex optimization problem by turning it into a series of sub-problems that are easier to optimize. Specifically, we construct a sequence of functions $\mathbb{F} = \{f_\beta : \mathbf{s} \mapsto \sigma(\beta\mathbf{s}) \mid \beta \in [1, \infty)\}$ indexed by β. In training process, $\mathbb{F}$ starts at $\beta = 1$ with an easier objective function, which is standard sigmoid function $\sigma(\beta\mathbf{s}) = \sigma(\mathbf{s})$. By periodically increasing β, $\sigma(\beta\mathbf{s})$ becomes harder to optimize, while the objectives converge to discrete endpoints $\lim_{\beta\to\infty} \sigma(\beta\mathbf{s}) = H(\mathbf{s})$, as shown in Fig. 3(b).

3.4.1. Sparsity loss functions

Since each task has its own forward and backward pass, different tasks are optimized individually and sequentially. Therefore, the loss function of our problem for each task is shown as follows, which optimizes the trade-off between model performance and sparsification.

$$\mathcal{L}_k = \mathcal{L}_k^p(\mathbf{W}^{\mathcal{E}}, \mathbf{Z}^{c,l}, \mathbf{M}^{c,l}) + \lambda_s(\sum_d^D \| \mathbf{m}_{k,d}^c \|_1 + \| \mathbf{m}_k^l \|_1), \tag{13}$$

where λ_s is the hyperparameter. $\mathcal{L}_k^p$ represents the prediction goals of task k. The L_1-norm is used to penalize the number of non-zero parameters to encourage $\mathbf{m}^{c,l}$ to be sparse. Note that since the whole network is differentiable as for the concrete form of $\mathbf{m}^{c,l}$, the optimization of the designed loss function is straightforward.

3.4.2. Training strategy

Rather than use a fixed temperature schedule across training steps, we propose an exponential temperature schedule in which tasks have the same η to control the speed at which the temperature increases progressively during training. The temperature schedule is defined as:

$$\beta_r = \min(\beta_{max}, \beta_0 \cdot (\eta)^r), \tag{14}$$

where $\beta_0 = 1$, β_{max} is the maximum value of β. r refers to the number of rounds required to update β. In each round, T_r is the number of training iterations under the hyperparameter β_r. Therefore, the total number of training epochs is $T = \sum_r^R T_r$. Overall, the proposed progressive training strategy effectively mitigates task interference by controlling the sparsity of selected layers and channels in the task-specific sub-networks, while also helping to capture inductive bias across tasks by preventing premature sparse of the learned task-specific sub-networks. We illustrate the proposed progressive training algorithm in Algorithm 1.

Algorithm 1 Progressive Training with Continuation

Input: Base Network $\mathcal{E}$; Data for K tasks $D = \{\mathbf{x}_n, \mathbf{y}_n\}_{n=1}^{N}$
Output: Sparse masks $\mathbf{M}^c, \mathbf{M}^l$

Randomly initialize $\mathbf{W}^{\mathcal{E}}$; Set each $s \in \{\mathbf{S}^c, \mathbf{S}^l\} = 1$
for $r = 1$ **to** R **do**
  for $t = 1$ **to** T_r **do**
    for $k = 1$ **to** K **do**
      Compute $\mathbf{m}_k^c$ using Equations (1)–(3) and (5)
      Compute $\mathbf{m}_k^l$ using Equations (6)–(10)
      Train $\mathcal{E}_k(\cdot; \mathbf{W}^{\mathcal{E}}, \mathbf{m}_k^c, \mathbf{m}_k^l)$ using Equation (13)
    end for
  end for
  Update β_r using Equation (14)
end for

4. Experiments

In this section, we evaluate the effectiveness of our proposed method using standard MTL benchmarks. First, our method is compared against classical MTL methods as well as SOTA multi-task architecture learning methods. Then, a hyperparameter analysis is conducted to illustrate how to choose the best hyperparameters for optimal model performance. Next, we conduct ablation studies to evaluate different components within our proposed method. Finally, we provide illustrations of the learned task-specific sub-networks.

4.1. Experimental settings

Brief descriptions of three datasets, baselines, implementation details and evaluation metrics are provided as follows. The codes are openly accessible and can be found at https://github.com/Jiebuaa/dual-mask-mtl.

4.1.1. Datasets

We evaluate the effectiveness of our method on three MTL benchmarks: indoor scene understanding dataset NYUv2 [35], street scene understanding dataset CityScapes [36], and facial attribute detection dataset Celeb-A [8]. The NYUv2 dataset contains 1449 indoor images with 288×384 resolution, and we use the official train/val splits, utilizing 795 images for training and 654 images for validation. It provides 3 tasks, including 13-class semantic segmentation, depth estimation, and surface normal estimation. CityScapes [36] dataset offers 3475 high resolution street-view images, resized to 128×256, focusing on 7-class semantic segmentation and depth estimation. It includes 2975 training images, and 500 validation images. Celeb-A dataset contains over 200K celebrity images, each annotated with 40 binary facial attribute labels. To reduce the computational cost, we divide the 40 attributes into eight groups according to the pre-processing procedure provided by [8]. Each group is regarded as a prediction task, with a dataset allocation of 16K images for training and 20K images for validation, all resized to 64×64.

4.1.2. Baselines

We compare *dual-mask* with nine baselines. First, we consider a single task baseline, namely STL, which trains each task separately. Second, we employ three popular MTL baselines, including Shared-Bottom (where tasks share the bottom layers but have distinct last layers), Cross-Stitch [5] networks and MTL with Covariance Alignment module (MTL-CA, for short) [37]. Third, three task-conditional based multi-task solutions were considered, including Task Routing (TR) [14], SE-MTL [22] and MR [8]. The dissimilarities among these methods are adopting different parameter partitioning strategies. Specifically, TR [14] uses fixed binary masks which are not trainable and fixed during training. SE-MTL [22] uses trainable soft masks. MR [8] adopts non-differentiable binary masks and applies a discrete updating method to them. In addition, we compare our method with a multi-task optimization method, namely PCGrad [38], which was combined with a SOTA multi-task learning algorithm, MTAN [39]. Finally, we consider a SOTA multi-task architecture learning method, namely AdaShare [7]. To ensure a fair comparison, we employ an identical backbone and task-specific layers across all baselines and our method.

Table 1
Hyperparameter settings on different datasets.

Datasets	NYUv2	Celeb-A	CityScapes
η	4	8	4
β_{max}	1000	300	800
ζ	3	2	3.5
γ	0	0.1	−0.2

4.1.3. Implementation details

On the NVIDIA GeForce RTX 3080 GPU, PyTorch is utilized to implement all baselines. We use SegNet [40], a popular architecture for CityScapes and NYUv2 datasets, as backbone and split the last convolution layer for tasks. For Celeb-A dataset, we use ResNet-18 as the backbone and split the last fully connected layer for tasks. All methods are optimized by Adam with a fixed learning rate of $1e^{-4}$. The batch sizes are tailored to the dataset: 2 for NYUv2 dataset, 8 for CityScapes dataset, and 256 for Celeb-A dataset. For a fair comparison, these hyperparameter settings are identical for both our method and baselines. The reported results are averaged over five seeds. In our method, training iteration T_r is set to 400 for both CityScapes and NYUv2 datasets, and to 40 for Celeb-A. Correspondingly, the number of iteration rounds R is consistently 40 across three datasets. We use 10% of total epochs for model warm-up without sampling for each. The early stop strategy is used to capture the best performance. We use grid search to find the best hyperparameters for our method and baselines, such as η. Table 1 provides the key hyperparameter settings for our method, tailored for each dataset.

4.1.4. Evaluation metric

We use standard MTL evaluation metrics as [8,39]. Specifically, for scene understanding tasks, we use mean Intersection over Union (mIoU) and Pixel Accuracy (Pix. Acc.) for semantic segmentation task, Mean Error (Mean Err.) and Medium Error (Med. Err.) for surface normal prediction task, and Averaged absolute Error (Abs. Err.) and Relative Error (Rel. Err.) for depth estimation task. For facial attribute detection, we utilize Precision, Recall and F-Score as metrics. In addition, we also report the ratio #P of a model's trainable model parameters w.r.t. the Shared-Bottom method, which is defined as $\frac{\text{The number of a model's trainable model parameters}}{\text{The number of Shared-Bottom's trainable parameters}}$.

4.2. Results and analysis

We compare *dual-mask* with nine baselines across the Celeb-A, CityScapes and NYUv2 datasets, as shown in Tables 2–4. On NYUv2 dataset, as shown in Table 2, *dual-mask* outperforms all baselines on five of six metrics, and ranking second in the remaining one metric. Notably, the performance of all MTL baselines on the Surface Normal Prediction task lags behind STL, indicating that the channels or layers used by Semantic Segmentation and Depth Estimation should be carefully selected in order to improve the performance of Surface Normal Prediction. Our method provides improved performance with respect to STL on this task, whose behavior suggests that our method can reduce task interference even more than other task-conditional based MTL baselines, including SE-MTL [22], TR [14] and MR [8]. For Depth Estimation task, STL achieves the worst performance on 2 metrics out of 2, showing that it needs more inductive bias from other tasks. In contrast, our method achieves the second-best performance on 1

Table 2
NYUv2 results. #P denotes the ratio of a model's trainable model parameters compared to those of the Shared-Bottom model. The best scores are highlighted in bold.

Model	# P	Semantic segmentation (Higher better)		Depth estimation (Lower better)		Surface normal prediction (Lower better)	
		mIoU	Pix. Acc.	Abs. Err.	Rel. Err.	Mean Err.	Med. Err.
STL	≈3	17.48	55.57	68.37	29.47	29.93	23.32
Shared-Bottom	1	17.47	54.98	59.28	25.44	32.32	27.02
Cross-Stitch [5]	≈3	17.03	55.24	59.42	25.47	31.72	26.41
MTL-CA [37]	1	17.97	55.79	58.93	25.37	31.92	26.65
SE-MTL [22]	1	16.27	53.46	59.33	26.65	32.19	25.73
TR [14]	1	16.85	54.61	60.65	27.40	30.78	24.08
MR [8]	1	17.28	55.25	60.91	27.83	30.80	24.19
PCGrad [38]	≈4	19.84	57.13	59.80	**25.22**	31.08	27.03
AdaShare [7]	1	19.07	57.26	64.60	29.21	31.58	26.39
Our	1	**22.84**	**59.28**	**57.38**	25.35	**27.03**	**20.52**

Table 3
Celeb-A results. MR* indicates the results reported from the original paper [Pascal et al. 2021].

Model	#P	Multi-attribute classification (Higher better)		
		Precision	Recall	F-Score
STL	≈8	67.39	59.80	62.82
Shared-Bottom	1	70.66	58.40	62.79
Cross-Stitch	≈8	70.67	59.18	63.22
MTL-CA	1	70.02	59.36	63.19
SE-MTL	1	70.60	62.54	65.71
TR	1	71.07	61.77	65.35
MR	1	71.50	62.19	65.76
MR*	1	71.24	63.04	66.23
PCGrad	≈9	71.22	59.35	63.20
AdaShare	1	71.78	61.30	65.21
Our	1	**74.09**	**65.42**	**68.61**

Table 4
CityScapes results. The best scores are highlighted in bold.

Model	#P	Semantic segmentation (Higher better)		Depth estimation (Lower better)	
		mIoU	Pix. Acc.	Abs. Err.	Rel. Err.
STL	≈2	51.77	91.02	0.0141	**23.39**
Shared-Bottom	1	50.49	90.95	0.0148	32.80
Cross-Stitch	≈2	51.17	91.09	0.0143	26.84
MTL-CA	1	51.50	91.03	0.0148	30.15
SE-MTL	1	44.98	87.07	0.0186	33.67
TR	1	50.88	90.62	0.0137	25.54
MR	1	50.93	90.62	0.0136	24.73
PCGrad	≈3	52.20	91.65	0.0149	30.73
AdaShare	1	54.02	92.17	**0.0131**	28.97
Our	1	**54.77**	**92.19**	0.0133	25.52

metric out of 2 for Depth Estimation task. The above-mentioned results demonstrate that the task-specific sub-networks learned by our method capture more inductive bias across tasks while effectively mitigating task interference.

Similarly, in Celeb-A 8-Task Learning, it can be observed that *dual-mask* significantly outperforms baselines on all metrics, see Table 3. Compared to STL and Cross-Stitch, which use separated backbones for tasks, our method obtains superior performance with fewer parameters. For other task-conditional based MTL baselines, including SE-MTL [22], TR [14] and MR [8], we observe that all of them significantly outperform STL, showing that the learned dense masks or heuristically selected sparse masks used by these methods are able to efficiently reduce task interference. Our method provides improved performance with respect to SE-MTL [22], TR [14] and MR [8], showing that it is able to capture optimal masks. Moreover, our method achieves higher performance than the SOTA multi-task architecture learning method (AdaShare), which shows the importance of fine-grained channel partitioning.

On CityScapes dataset, as in Table 4, *dual-mask* outperforms baselines on two of four metrics, achieves second on one metric, and achieves third on another metric. On Semantic Segmentation task, it can be observed that *dual-mask* and AdaShare obtain similar performance, and both of them outperform other baselines significantly. The experimental results indicate that layer-level sharing architecture is more important than channel-level one under this dataset. Shared-Bottom obtains worse performance than STL on both two tasks, suggesting that fully sharing of features among these tasks increases task interference. For Depth Estimation task, STL obtains the best performance in 1 metric, which shows that this task needs more task-specific inductive bias. MR [8] methods improve more performance than STL on one metric in Depth Estimation task while obtaining worse performance on the Semantic Segmentation task. Our method optimizes MTL model in a more balanced way, yielding better overall results.

4.3. Hyperparameter analysis

In this section, we explore the influence of several critical hyperparameters on NYUv2 dataset, including β^{max}, η, ζ and γ. The influence of four hyperparameters is evaluated independently. Specifically, we vary the values of β^{max}, η, ζ and γ and plot the results in Fig. 5(a), (b), (c) and (d), respectively. The experimental results show that Depth Estimation task is greatly affected by the value of β^{max}, ζ and γ. For example, the performance of the Depth Estimation task decreases when β^{max} grows from 1. When ζ is more than 1, the performance of Depth Estimation task improves; however, when ζ is larger than 3, the performance declines. A moderate value for ζ (e.g., 3) may be preferable to achieve good performance, and γ (e.g., 0) is similar. For other tasks, their results are slightly affected by the value of four hyperparameters. For example, with varying η, the performance of the Normal Estimation problem oscillates around 27.0. In addition, for different tasks, the preferred values of β^{max} and η vary. A modest value of β^{max} (e.g., 1000) to balance all tasks may be ideal to attain overall better results, and η (e.g., 4) is similar. Thus, we choose $\beta^{max} = 1000, \eta = 4, \zeta = 3$ and $\gamma = 0$ to avoid too heavy effort on hyperparameter searching.

4.4. Ablation study

4.4.1. Effect of different masks in dual-mask

Here, we investigate the effect of the designed masks on the performance of our method under different sparsity ratios. Specifically, we compare two versions of our method with MR and AdaShare on NYUv2, one "Our ($\mathbf{m}^c$)" only making the channel selection masks sparse and the other "Our ($\mathbf{m}^{c,l}$)" making both of the channel and layer

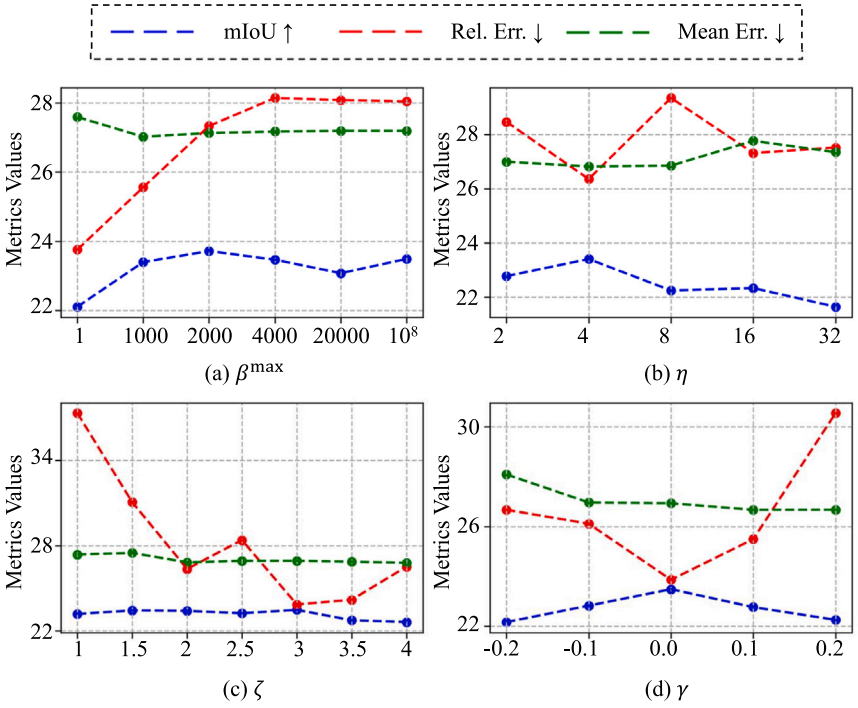

Fig. 5. Analysis of hyperparameters on NYUv2 dataset.

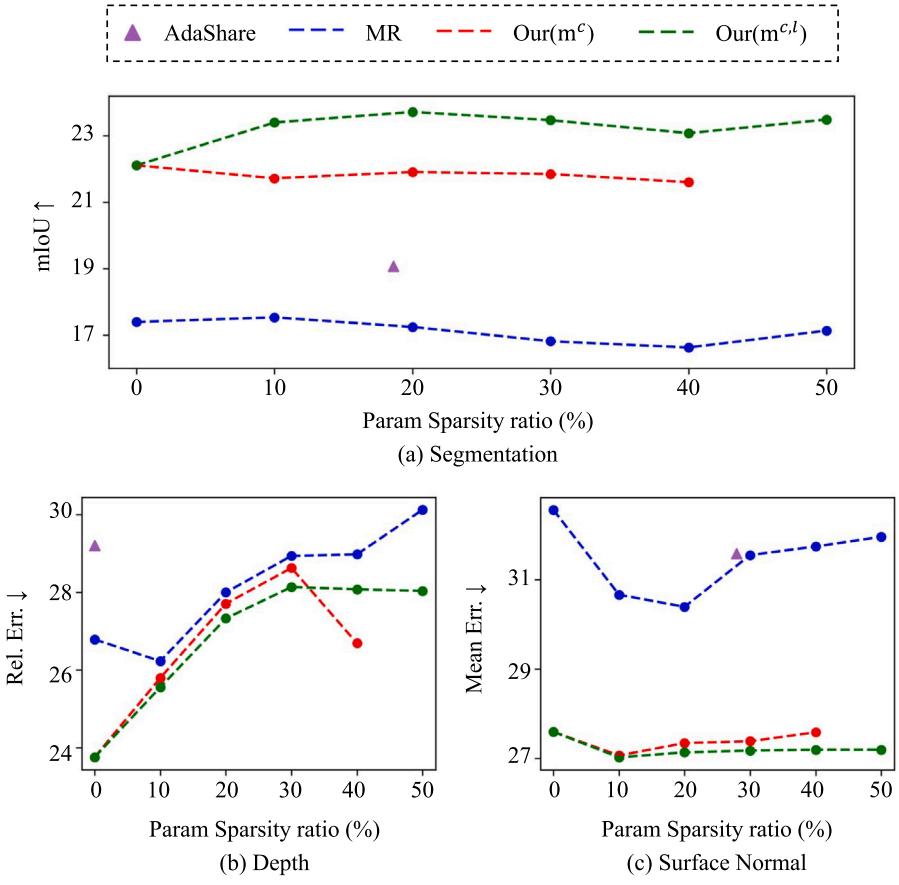

Fig. 6. Performance comparison with different masks on NYUv2 dataset.

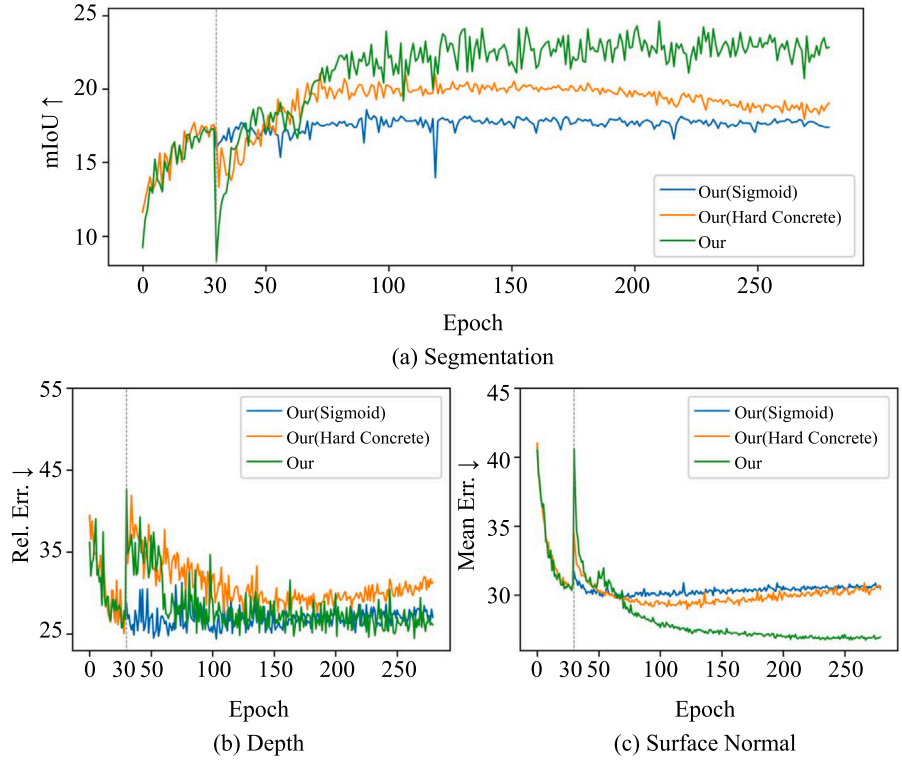

Fig. 7. Effects of different relaxation methods on NYUv2 dataset.

selection masks sparse. The sparsity ratio is defined as $1-\frac{\sum_d^D \|\mathbf{m}_{k,d}^c \cdot m_{k,d}^l\|_0}{\sum_d^D p_d}$ for each task, which shows the ratio of selected channels across all layers. The results are shown in Fig. 6, from it, we have the following observations: (1) We observe that our methods hold their performance when the sparsity ratio is less than 0.5. (2) Under the same sparsity ratio, "Our ($\mathbf{m}^c$)" provides improved performance on all tasks compared to MR. This observation suggests that "Our ($\mathbf{m}^c$)" with differentiable channel selection masks effectively enhances inductive bias across tasks and mitigates task conflicts, outperforming heuristic method MR. In addition, "Our ($\mathbf{m}^{c,l}$)" provides improved performance in most of the cases than "Our ($\mathbf{m}^c$)". These observations suggest that both channel selection and layer selection masks positively contribute to the performance of our method. (3) Compared with AdaShare, "Our ($\mathbf{m}^{c,l}$)" can achieve better performance, and generate more sparse task-specific sub-networks. (4) "Our ($\mathbf{m}^{c,l}$)" achieves the best performance when the sparsity ratio is 0.9, which indicates that tasks are closely related and need fewer parameters to handle task interference. Therefore, we choose 0.9 as the sparsity ratio and "Our ($\mathbf{m}^{c,l}$)" to deploy.

4.4.2. Effect of relaxation methods

To validate the effectiveness of the proposed relaxation method, we explore the effects of different relaxation methods. We denote "Our (Sigmoid)" as our method using solely the logistic trick to generate masks. Also, "Our (Hard-Concrete)" as our method only uses the Hard-Concrete sampling to generate masks. As shown in Fig. 7, we observe that the performance of "Our (Sigmoid)" method is worse than "Our (Hard-Concrete)" method, showing that the effectiveness of Hard-Concrete sampling method. Our method performs better than other variants on all tasks, demonstrating its superiority. The results also show that importance-guided auxiliary gradient indeed benefits for convergence to an optimal solution.

4.4.3. Effect of training strategy

We evaluate two training schedules: fixed and proposed exponential temperature schedule, and explore their effects on MTL model performance. The former forces the model to be sparse at the beginning; the latter aims to achieve a similar sparsity in a progressive manner. The experimental results are shown in Fig. 8. It is observed that the proposed exponential temperature schedule works better on all tasks after warm-up phase than the method based on fixed temperature schedule, which demonstrates that the proposed training strategy is effective in capturing more inductive bias across tasks. Furthermore, we can also observe that two methods spend similar iterations reaching convergence. Such behavior shows that the proposed progressive training strategy does not require an extra number of iterations during training.

4.5. Illustrations of the learned sparsity architecture

To better understand how the channels and layers are utilized by different tasks in *dual-mask*, we visualize the task-specific sub-networks captured by our method on NYUv2 dataset, as shown in Fig. 9. In Fig. 9, we visualize the number of channels finally selected in each layer in our method. This figure illustrates that our method discretely selects varying numbers of channels per layer for tasks, with the remaining channels being skipped. In addition, we visualize the task activation paths at the layer level on the Celeb-A dataset, as shown in Fig. 10. Specifically, we choose eight convolution layers as an example to show the layer-wise architectures learned by eight tasks. Each node refers to a convolution layer whose numbering is the layer index. And each blue node represents the layer with at least one task selected, whereas the white slashed node represents the layer that is not selected by any task. Different colored arrows between layers represent the execution paths taken by the different tasks. It can be seen that different tasks have different execution paths. The aforementioned results show that our method is capable of learning diverse task-specific sub-networks for each task in terms of two hierarchical levels: channel and layer.

5. Conclusion

In this paper, we proposed a progressively sparse multi-task architecture learning method, namely *dual-mask*, with the goal of disentangling the single network into different task-specific sparse sub-networks

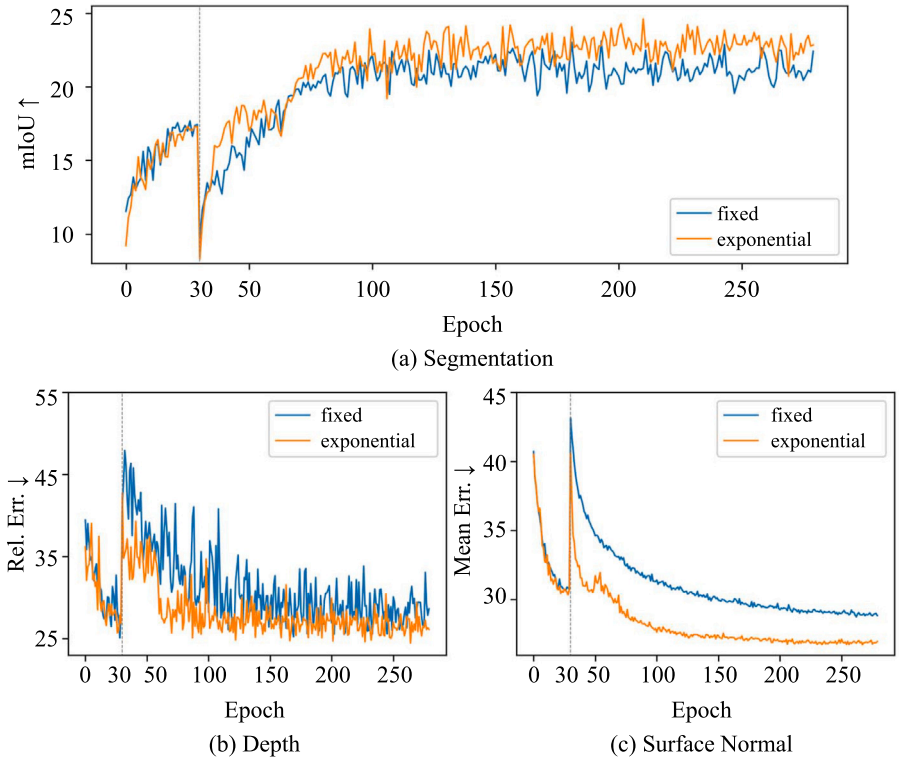

Fig. 8. Effects of different training strategies on NYUv2 dataset.

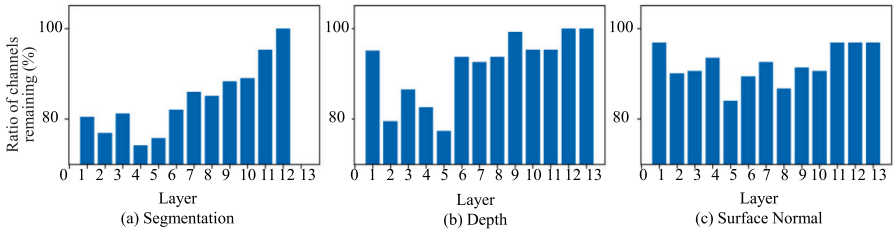

Fig. 9. Visualization task-specific sub-networks obtained by *dual-mask* on NYUv2 dataset.

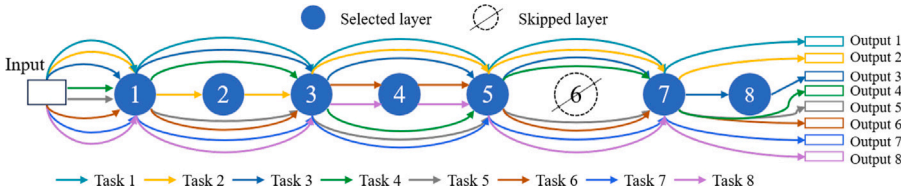

Fig. 10. Visualization layer-wise sub-networks obtained by *dual-mask* on Celeb-A dataset. Here, we choose eight convolution layers as an example to show the layer-wise architectures learned by eight tasks. Each blue node represents the layer with at least one task selected, whereas the white slashed node represents the layer that is not selected by any task. The arrows with different colors between layers represent the learned layer-wise architectures for different tasks. 'Input' indicates the input of layer 1, which is consistent across all tasks, whereas 'Output 1' and so on indicate the outputs of layer 8, which vary across tasks. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

with salient channels and layers in an end-to-end manner. In contrast to current MTL methods, which work with a fixed backbone network, our method enables the flexible and fine-grained feature sharing based on backbone network that is achieved by the designed channel selection and layer selection modules. In channel selection and layer selection modules, we propose an importance-guided relaxation method to jointly train the model parameters, channel and layer selection masks. By learning the sparse and high-quality masks, our method can effectively identify salient channels and layers for each task, activating distinct task-specific subnetworks, thereby contributing to reducing task interference. The proposed progressive training strategy increases inductive bias across tasks effectively, alleviating the model performance drop raised by sparsity at the early training stages. Extensive experimental results on the multiple standard MTL benchmarks show that our method achieves superior performance than the SOTA multi-task methods. We believe that our method can offer a new perspective on pruning and architecture optimization on MTL and has wide practical application scenarios. In addition, we will extend

our method to more complex MTL problem scenarios, including those where not all task sample labels are available.

CRediT authorship contribution statement

Jiejie Zhao: Writing – original draft, Methodology, Validation, Formal analysis. **Tongyu Zhu:** Writing – review & editing, Supervision, Conceptualization. **Leilei Sun:** Writing – review & editing, Methodology, Formal analysis, Conceptualization. **Bowen Du:** Project administration, Methodology, Supervision, Funding acquisition. **Haiquan Wang:** Writing – review & editing, Visualization, Investigation. **Lei Huang:** Writing – review & editing, Methodology, Formal analysis, Conceptualization.

Declaration of competing interest

All authors have participated in (a) conception and design, or analysis and interpretation of the data; (b) drafting the article or revising it critically for important intellectual content; and (c) approval of the final version.

This manuscript has not been submitted to, nor is under review at, another journal or other publishing venue.

The authors have no affiliation with any organization with a direct or indirect financial interest in the subject matter discussed in the manuscript.

Data availability

Data will be made available on request.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant 51991395, and in part by the S&T Program of Hebei Province, China, under Grant 225A0802D.

References

[1] H. Cuevas-Velasquez, A. Galán-Cuenca, R.B. Fisher, A.J. Gallego, Efficient multi-task progressive learning for semantic segmentation and disparity estimation, Pattern Recognit. 154 (2024) 110601.
[2] H. Yu, Q. Dai, Self-supervised multi-task learning for medical image analysis, Pattern Recognit. 150 (2024) 110327.
[3] L. Zhou, H. Zhao, J. Leng, MTCNet: Multi-task collaboration network for rotation-invariance face detection, Pattern Recognit. 124 (2022) 108425.
[4] B. Huang, Z. Wang, G. Wang, K. Jiang, Z. Han, T. Lu, C. Liang, PLFace: Progressive learning for face recognition with mask bias, Pattern Recognit. 135 (2023) 109142.
[5] I. Misra, A. Shrivastava, A. Gupta, M. Hebert, Cross-stitch networks for multi-task learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 3994–4003.
[6] M. Ding, X. Lian, L. Yang, P. Wang, X. Jin, Z. Lu, P. Luo, HR-NAS: searching efficient high-resolution neural architectures with lightweight transformers, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 2982–2992.
[7] X. Sun, R. Panda, R. Feris, K. Saenko, Adashare: Learning what to share for efficient deep multi-task learning, Adv. Neural Inf. Process. Syst. 33 (2020) 8728–8740.
[8] L. Pascal, P. Michiardi, X. Bost, B. Huet, M.A. Zuluaga, Maximum roaming multi-task learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 35, 2021, pp. 9331–9341.
[9] Z. Wang, C. Li, X. Wang, Convolutional neural network pruning with structural redundancy reduction, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 14913–14922.
[10] C. Ren, D. Dai, H. Yan, Robust classification using L2,1-norm based regression model, Pattern Recognit. 45 (7) (2012) 2708–2718.
[11] J. Zhan, Y. Luo, C. Guo, Y. Wu, J. Meng, J. Liu, YOLOPX: Anchor-free multi-task learning network for panoptic driving perception, Pattern Recognit. 148 (2024) 110152.
[12] H. Ye, D. Xu, Taskexpert: Dynamically assembling multi-task representations with memorial mixture-of-experts, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2023, pp. 21828–21837.
[13] T. Chen, X. Chen, X. Du, A. Rashwan, F. Yang, H. Chen, Z. Wang, Y. Li, Adamv-moe: Adaptive multi-task vision mixture-of-experts, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2023, pp. 17346–17357.
[14] G. Strezoski, N.v. Noord, M. Worring, Many task learning with task routing, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 1375–1384.
[15] M. Kanakis, D. Bruggemann, S. Saha, S. Georgoulis, A. Obukhov, L. Van Gool, Reparameterizing convolutions for incremental multi-task learning without task interference, in: European Conference on Computer Vision, 2020, pp. 689–707.
[16] Y. Gao, H. Bai, Z. Jie, J. Ma, K. Jia, W. Liu, Mtl-nas: Task-agnostic neural architecture search towards general-purpose multi-task learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 11543–11552.
[17] E. Yang, Z. Wang, L. Shen, S. Liu, G. Guo, X. Wang, D. Tao, Adamerging: Adaptive model merging for multi-task learning, in: 12th International Conference on Learning Representations, 2024.
[18] Z. Yue, Y. Zhang, J. Liang, Learning conflict-noticed architecture for multi-task learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 37, 2023, pp. 11078–11086.
[19] S. Vandenhende, S. Georgoulis, B. De Brabandere, L. Van Gool, Branched multi-task networks: Deciding what layers to share, in: Proceedings BMVC 2020, 2019.
[20] E. Meyerson, R. Miikkulainen, Beyond shared hierarchies: Deep multitask learning through soft layer ordering, in: 6th International Conference on Learning Representations, 2018.
[21] F.J. Bragman, R. Tanno, S. Ourselin, D.C. Alexander, J. Cardoso, Stochastic filter groups for multi-task cnns: Learning specialist and generalist convolution kernels, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 1385–1394.
[22] K.-K. Maninis, I. Radosavovic, I. Kokkinos, Attentive single-tasking of multiple tasks, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 1851–1860.
[23] H. Shi, S. Ren, T. Zhang, S.J. Pan, Deep multitask learning with progressive parameter sharing, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2023, pp. 19924–19935.
[24] S. Han, J. Pool, J. Tran, W.J. Dally, Learning both weights and connections for efficient neural networks, in: Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 1, 2015, pp. 1135–1143.
[25] Y. Hu, N. Belkhir, J. Angulo, A. Yao, G. Franchi, Learning deep morphological networks with neural architecture search, Pattern Recognit. 131 (2022) 108893.
[26] E.D. Karnin, A simple procedure for pruning back-propagation trained neural networks, IEEE Trans. Neural Netw. 1 (2) (1990) 239–242.
[27] X. Dong, S. Chen, S. Pan, Learning to prune deep neural networks via layer-wise optimal brain surgeon, Adv. Neural Inf. Process. Syst. 30 (2017).
[28] C. Louizos, M. Welling, D.P. Kingma, Learning sparse neural networks through l_0 regularization, in: 6th International Conference on Learning Representations, 2018.
[29] A. Vahdat, A. Mallya, M.-Y. Liu, J. Kautz, Unas: Differentiable architecture search meets reinforcement learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 11266–11275.
[30] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, Q.V. Le, Mnasnet: Platform-aware neural architecture search for mobile, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 2820–2828.
[31] Y. Bengio, N. Léonard, A.C. Courville, Estimating or propagating gradients through stochastic neurons for conditional computation, 2013, CoRR abs/1308.3432.
[32] C. Maddison, A. Mnih, Y. Teh, The concrete distribution: A continuous relaxation of discrete random variables, in: 5th International Conference on Learning Representations, 2017.
[33] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, Y. Bengio, Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1, 2016, arXiv preprint arXiv:1602.02830.
[34] E.L. Allgower, K. Georg, Numerical Continuation Methods: an Introduction, vol. 13, Springer Science & Business Media, 2012.
[35] N. Silberman, D. Hoiem, P. Kohli, R. Fergus, Indoor segmentation and support inference from rgbd images, in: European Conference on Computer Vision, Springer, 2012, pp. 746–760.
[36] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, B. Schiele, The cityscapes dataset for semantic urban scene understanding, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 3213–3223.
[37] S. Wu, H.R. Zhang, C. Ré, Understanding and improving information transfer in multi-task learning, in: 8th International Conference on Learning Representations, 2020.
[38] T. Yu, S. Kumar, A. Gupta, S. Levine, K. Hausman, C. Finn, Gradient surgery for multi-task learning, Adv. Neural Inf. Process. Syst. 33 (2020) 5824–5836.
[39] S. Liu, E. Johns, A.J. Davison, End-to-end multi-task learning with attention, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 1871–1880.

[40] V. Badrinarayanan, A. Kendall, R. Cipolla, Segnet: A deep convolutional encoder-decoder architecture for image segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 39 (12) (2017) 2481–2495.

Jiejie Zhao received the Ph.D. degree in computer science and engineering from Beihang University, Beijing, China, in 2022. She is currently a research assistant in Zhongguancun Laboratory, Beijing, China. Her research interest includes data mining, multi-task learning and deep learning.

Tongyu Zhu received the B.S. degree from Tsinghua University in 1992, and the M. S. degree from Beihang University in 1999. He is currently an associate professor with the State Key Laboratory of Complex and Critical Software Environment in School of Computer Science and Engineering, Beihang University, Beijing, China. His research interests include intelligent traffic information processing and network application.

Leilei Sun received the BS and MS degrees from the School of Control Theory and Control Engineering, Dalian University of Technology, in 2009 and 2012, respectively. He is currently an associate professor with the State Key Laboratory of Complex and Critical Software Environment in School of Computer Science and Engineering, Beihang University, Beijing, China. His research interests include data mining, machine learning, and artificial intelligence.

Bowen Du received the BS degree from Shijiazhuang Tiedao University, China, and the Ph.D. degree in computer science and engineering from Beihang University, China. He is currently an associate professor with the State Key Laboratory of Complex and Critical Software Environment, Beihang University. His research interests include smart city technology, multi-source data fusion, big data mining, and citizen travel pattern mining.

Haiquan Wang received the Ph.D. degree in computer science and engineering from Beihang University, Beijing, China, in 2013. He is currently an Associate Professor with the School of Software, Beihang University. His research interests include intelligent transport systems and software engineering. He has been studying intelligent transport systems in recent years, hosting or participating in many national projects, including the National Nature Science Foundation of China, National High Technology Research and Development Program of China.

Lei Huang is currently an associate professor with the State Key Laboratory of Complex and Critical Software Environment in Institute of Artificial Intelligence, Beihang University, Beijing, China. He received his B.Sc. and Ph.D. degrees in 2010 and 2018, respectively, from the School of Computer Science and Engineering, Beihang University, China. From 2015 to 2016, he visited the Vision and Learning Lab, University of Michigan, Ann Arbor, as a joint Ph.D. student. His research interests include deep learning, semi-supervised learning, and their application to computer vision tasks.