IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 26, 2022, accepted March 14, 2022, date of publication March 23, 2022, date of current version March 30, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3161622

Resource-Efficient Multi-Task Deep Learning Using a Multi-Path Network

SOYEON PARK, JIHO LEE, AND EUNWOO KIM, (Member, IEEE)
School of Computer Science and Engineering, Chung-Ang University, Seoul 06974, South Korea

Corresponding author: Eunwoo Kim (eunwoo@cau.ac.kr)

This work was supported in part by Samsung Research Funding Incubation Center of Samsung Electronics under Project Number SRFC-IT2002-05, in part by the Chung-Ang University Graduate Research Scholarship in 2020, and in part by Korea Institute for Advancement of Technology (KIAT) grant funded by the Korea Government (MOTIE, Ministry of Trade, Industry and Energy) (P0012724, The Competency Development Program for Industry Specialist).

ABSTRACT Multi-task learning (MTL) improves learning efficiency compared to the single-task counterpart in that it performs multiple tasks at the same time. Due to the nature, it can achieve generalized performance as well as alleviate overfitting. However, it does not efficiently perform resource-aware inference from a single trained architecture. To address the issue, we aim to build a learning framework that minimizes the cost to infer tasks under different memory budgets. To this end, we propose a multi-path network with a self-auxiliary learning strategy. The multi-path structure contains task-specific paths in a backbone network, where a lower-level path predicts earlier with a smaller number of parameters. To alleviate the performance degradation from earlier predictions, a self-auxiliary learning strategy is presented. The self-auxiliary tasks convey task-specific knowledge to the main tasks to compensate for the performance leak. We evaluate the proposed method on an extensive set of multi-task learning scenarios, including multiple tasks learning, hierarchical learning, and curriculum learning. The proposed method outperforms existing multi-task learning competitors for most scenarios about by a margin of $1\% \sim 2\%$ accuracy on average while consuming $30\% \sim 60\%$ smaller computational cost.

INDEX TERMS Multi-task learning, resource-efficient learning, multi-path network.

I. INTRODUCTION

Deep learning has achieved great success in domains such as computer vision [1] and natural language processing [2]. A well-designed deep architecture is generally deployed for a single task [3], such as object detection [4], pose estimation [5] and image segmentation [6]. Despite the success in various fields, it is well-known that the approach requires heavy computational resources, especially when we address multiple tasks. This is generally from a common practice that a network is specialized for a single task. If the number of tasks grows, the number of parameters required increases accordingly. As a result, given a large number of tasks to be performed, it will become an intolerable burden for resource-constrained devices.

To overcome the limitation, one strategy is to learn tasks jointly, which is referred to as multi-task learning (MTL) [7]. MTL executes multiple tasks at the same time and is more efficient than its single-task counterpart. Due to the nature of exploiting knowledge of multiple different tasks, it can be employed in various fields, [8]–[10], and natural language processing [11], [12]. Despite its practical benefit, there might be unavoidable issues. First, we may encounter task interference during training when different tasks share the same network. This is because, unlike a single task, multiple knowledge from different tasks are learned using the same set of parameters, decreasing learning efficiency. Second, if a different computation budget is required, we usually need to define a new network, which introduces additional training effort. In other words, when we learn a network tailored for a specific budget, it is difficult to directly transfer the learned network to another one for a new budget. Therefore, it would be desirable to perform more flexible inference under different computation budgets in a learned network.

Many existing works try to solve one of the above two challenges. Some recent works [13]–[15] have made efforts to reduce the negative impact between tasks to prevent performance degradation. [13], [16] propose a selection module so

The associate editor coordinating the review of this manuscript and approving it for publication was Rongbo Zhu.

that relevant tasks are encouraged to share their features while irrelevant tasks are disentangled. [14] utilizes task-specific attention in a single shared backbone and [15] reparameterizes existing convolution modules into shared and task-specific modules. However, the above works do not perform budget-aware inference when different budgets are required, thus they need to produce multiple individual networks for different costs. Although [17] enables cost-aware inference by building a nested structure containing multiple networks of different sizes, it mainly focuses on a single task, not multiple tasks.

In this work, we develop a new network containing different inference paths, called a multi-path network, which learns multiple tasks simultaneously under different computation costs. The proposed architecture is structured hierarchically and consists of multiple paths of different hierarchy levels. The multiple paths correspond to internal networks of different sizes in the architecture, respectively (see Figure 1). The path under a lower-level (resp., higher-level) hierarchy enables earlier (resp., later) prediction with a smaller (resp., larger) number of parameters. The network possesses a nested structure such that an internal network of an earlier path is a subset of internal networks of later paths. Thus, the knowledge of a lower-level path is shared by a higher-level path. This is different from a common practice with multiple output branches at the end of a network [7]. We note here that some tasks are performed early with a partial set of parameters, and the other tasks can occupy the rest of the parameters. From this, tasks do not fully share the entire set of parameters, allowing different learning and inference flows and mitigating the strong influence (connection) between tasks. Besides, the multi-path network containing internal networks of diverse sizes can meet the requirement of different computation costs. Thus, it will reduce the effort to design and train additional networks from scratch. Note also that tasks of different natures can be efficiently addressed in the proposed approach. For instance, it can apply lower-level paths to easier tasks and higher-level paths to harder tasks. In addition, to avoid the performance degradation risk that occurred in the early prediction with a small number of parameters and to boost the performance of all target tasks, a self-auxiliary learning strategy is introduced through knowledge distillation [18]. The multi-path network receives rich task-specific knowledge from the self-auxiliary tasks that can improve the representation of the target tasks.

Additionally, to our knowledge, there is no comprehensive study of multi-task learning from various views. It is important to note that existing works in the multi-task learning literature have not extensively conducted a diverse set of scenarios, so their analyses might be limited. In this work, we study the proposed multi-task learning method under a wide range of scenarios, including multiple tasks learning, hierarchical learning, and curriculum learning (see Section IV). For those scenarios, we use three benchmark datasets: CIFAR-10 and CIFAR-100 [19], and Celeb-A [20]. We show from the experiments that the proposed method performs better than other approaches in most scenarios while effectively minimizing the harmful interference between tasks. We also analyze the computation cost of compared methods to verify that the proposed method is resource-efficient.

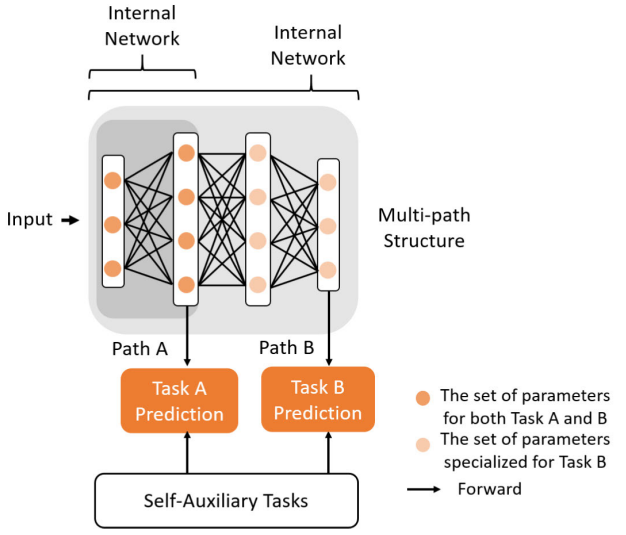


FIGURE 1. A graphical illustration of the proposed learning framework based on a multi-path network and self-auxiliary tasks. The network is composed of hierarchically constructed subnetworks and their inference paths, where the lower-level path (Path A) branches out at an earlier layer to perform Task A and the higher-level path (Path B) outputs at the end of the network to perform Task B. We also have auxiliary tasks corresponding to the main tasks, which are called self-auxiliary tasks. The self-auxiliary tasks (with their pre-trained networks) transfer task-specific knowledge to the main network to boost the performance of all tasks.

In summary, the main contributions of the proposed method are three folds:

- The proposed hierarchical structure can efficiently handle multiple tasks, including hierarchical and curriculum learning tasks.
- The proposed network contains multiple internal networks of different sizes and can address various computation budgets from a single training phase.
- The self-auxiliary learning strategy mitigates the performance leak by sharing the task-specific information with the multi-path structure.

The organization of this paper is as follows. We introduce related work in multi-task learning in Section II. In Section III, we explain the proposed multi-path structure and self-auxiliary learning strategy. In Section IV, we show the effectiveness of our work with other methods. Finally, the conclusion of this work is discussed in Section V.

II. RELATED WORK

A. MULTI-TASK LEARNING

The goal of multi-task learning (MTL) is to learn multiple tasks jointly [7]. MTL can be classified into two main categories. The first category introduces multiple individual networks proportional to the number of tasks with some

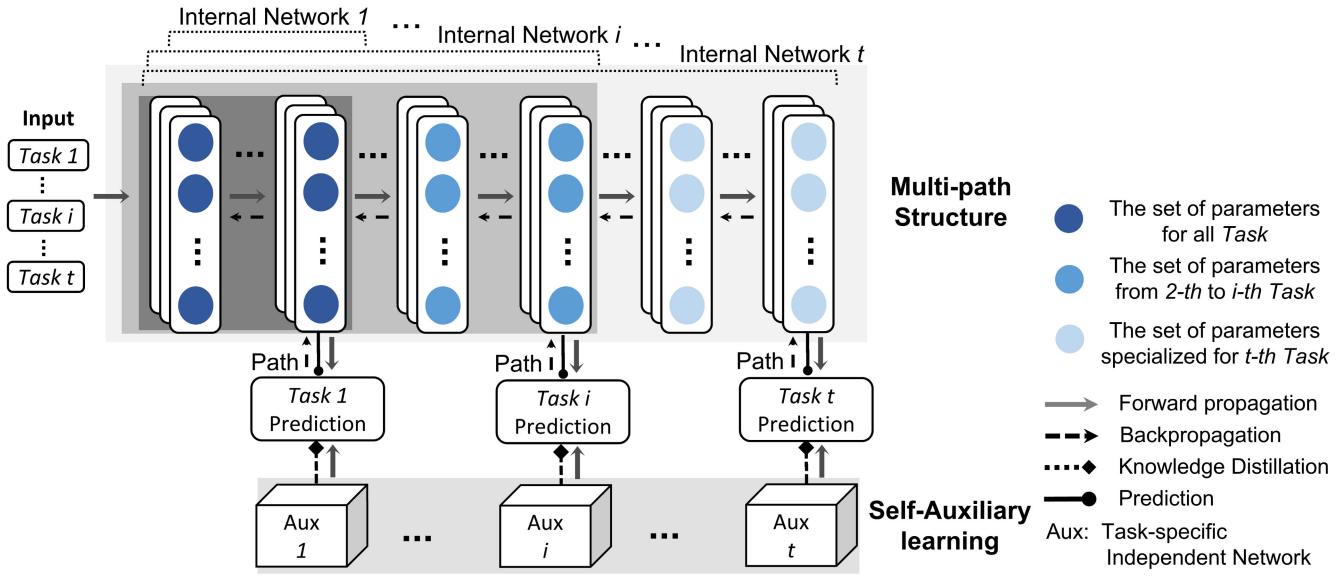


FIGURE 2. A graphical illustration of the proposed framework; the multi-path network with self-auxiliary tasks. The multi-path network contains hierarchical branches that are placed across the network, from shallow to deep layers. The lowest-level branch of the hierarchy (corresponding to Task 1) allows the task to be performed the earliest, and the highest-level branch (corresponding to Task *t*) performs the last. To boost the performance of the tasks, self-auxiliary learning is applied where self-auxiliary tasks are pre-trained and distilled to the main tasks.

constraints between the networks [21], [22]. [21] proposes a cross-stitch module that enables to share the task knowledge between individual networks. Likewise, [22] fuses the features from the different tasks to train them jointly. The other one learns multiple tasks using a single shared architecture [11], [17], [23]–[26]. [11], [23] utilize the auxiliary tasks to rich the feature representation. [24] suggests a soft-attention module for learning task-specific features. [25] attempts to hold important parameters for each task by iterative pruning. [17] constructs a resource-aware structure that compresses the network by splitting each channel into separated groups. Additionally, there are many studies to solve multi-task learning from an optimization point of view, such as Pareto optimal solution [27], multi-objective optimization [28], task priority [29], gradient surgery [30].

In this work, we are particularly interested in learning on a single shared structure since it is memory efficient. While most existing methods need to be trained from scratch if another computation budget is needed, the proposed method does not require extra training efforts due to the internal networks in the proposed architecture. Besides, task interference can be mitigated in the proposed method by allowing different learning and inference flows for tasks.

B. KNOWLEDGE DISTILLATION

Knowledge distillation (KD) is a method that transfers knowledge from a large-scale (teacher) network to a small-scale (student) network [31]. The student network is trained to mimic the teacher network by minimizing the difference of the predictions between student and teacher from the last layer [18], intermediate layers [32], [33], attention maps [34], and so on. Knowledge distillation has been applied to face recognition [35], style transfer [36], and natural language processing [37], [38] to compress the network, while maintaining the performance. KD is used not only in single task learning but also in multi-task learning [12], [39]. [12] suggests using KD when addressing multi-task learning in natural language processing. [39] employs KD to handle an imbalance problem while optimizing the multiple losses.

Apart from the previous works, we deploy KD to accelerate the performance in multi-task learning while achieving resource efficiency. To overcome the performance degradation that can arise with early predictions in the multi-path network, we distill knowledge from self-auxiliary tasks with pre-trained networks to learn the subnetworks in the proposed architecture.

III. METHOD

The proposed method aims to provide prediction at different computation costs from a single trained architecture while alleviating destructive interference that can arise when learning different tasks. To achieve this goal, we propose a learning framework based on a multi-path structure with a self-auxiliary learning strategy. The multi-path structure performs multiple tasks under different paths. Self-auxiliary learning assists the main tasks to make up for the performance loss with earlier paths. We introduce the multi-path network in Section III-B and the self-auxiliary learning strategy in Section III-C. Additionally, the notations used in this work are listed up in Table 1.

A. MULTI-TASK LEARNING

Multi-task learning (MTL) trains multiple tasks jointly, which can improve the learning efficiency and generalized

TABLE 1. The notations used in this work.

Notation	Meaning
i	Index of the task
$\mathcal{D}^{(i)}$	Dataset of the i^{th} task
$\mathcal{W}$	Set of parameters
$\mathcal{W}_{sh}$	Set of shared parameters
$\mathcal{W}_{task}^{(i)}$	Set of task-specific parameters of the i^{th} task
$\mathcal{W}_{main}^{(i)}$	Set of parameters of the i^{th} task assigned to the main task.
$\mathcal{W}_{aux}^{(i)}$	Set of parameters of the i^{th} task assigned to the auxiliary task
$\mathcal{L}_{multi}^{(i)}$	Loss of the i^{th} task for the main task
$\mathcal{L}_{aux}^{(i)}$	Loss of the i^{th} task for the auxiliary task

performance. We define a task as learning an attribute in a sample [40], [41], learning each hierarchy of hierarchically constructed dataset (in hierarchical classification) [17], [42], or learning a group of samples of different difficulty (in curriculum learning) [43], in this work, but not limited to.

In the conventional MTL method [7], the set of datasets is denoted as $\{\mathcal{D}^{(i)}\}_{1\le i\le t}$, where the number of tasks is t and the task index is $i \in [1, 2, \ldots, t]$. For the i^{th} task, the dataset $\mathcal{D}^{(i)}$ consists of the set of data samples $X^{(i)}$ and the corresponding set of labels $Y^{(i)}$, i.e., $\mathcal{D}^{(i)} = (X^{(i)}, Y^{(i)})$. A single shared network $h(\cdot)$ is a collection of the shared parameters $\mathcal{W}_{sh}$ and the task-specific parameters $\{\mathcal{W}_{task}^{(i)}\}_{1\le i\le t}$, and we define $\mathcal{W} = \{\mathcal{W}_{sh}, \mathcal{W}_{task}^{(1)}, \ldots, \mathcal{W}_{task}^{(t)}\}$. Given a collection of tasks, the loss function $\mathcal{L}$ is defined as follows:

$$\min_{\mathcal{W}} \sum_{i=1}^{t} \mathcal{L}(h(X^{(i)}, \mathcal{W}), Y^{(i)}). \tag{1}$$

Note that task-specific classifiers for multiple tasks occupy a small number of parameters in $\mathcal{W}$. There are studies [24], [25] considering how to distribute parameters of each task to $\mathcal{W}_{sh}$ to overcome the limitation of learning with a single fixed architecture. [24] introduces feature-level attention modules applied to the shared parameters for different tasks. In [25], the shared parameters are divided into task-specific disjoint groups. However, [24] requires an additional number of parameters, and it may be difficult for [25] to address many tasks because all tasks are learned independently.

B. MULTI-PATH NETWORK

Multi-task learning generally addresses different tasks within a single shared architecture, which reveals that the mixture of knowledge can negatively influence each other. Moreover, when different memory budgets or networks of different sizes are required, it is common to define individual networks corresponding to the requirements. To handle both problems, the proposed method is built on a multi-path structure (see Figure 2). The structure contains multiple different paths constructed hierarchically in a way that an earlier path entails a smaller number of parameters and a later path requires a larger number of parameters. Note that the hierarchical structure of the multi-path network indicates a nested network structure in such a way that an internal network corresponding to an earlier prediction path shares its parameters (and knowledge) with other internal networks corresponding to later prediction paths. Hence, different inference paths from the multi-path network may avoid negative task interference because tasks do not fully share the entire network (i.e., partial sharing can reduce the chance of interference due to different learning and inference flows). Besides, due to the different sizes of internal networks, the proposed method can handle diverse computation requirements.

The set of parameters for the i^{th} path (or the i^{th} internal network) is $\mathcal{W}_{main}^{(i)}$. The set of entire parameters of the multi-path network is $\mathcal{W}_{main}$ which is the same as the parameters in the last task $\mathcal{W}_{main}^{(t)}$. Note that the internal networks included in the proposed model have the following hierarchical (nested) property:

$$\mathcal{W}_{main}^{(m)} \subseteq \mathcal{W}_{main}^{(n)}, \quad m \le n, \forall m, n \in [1, 2, \ldots, t]. \tag{2}$$

Given the proposed network, the loss function for the i^{th} task is defined as:

$$\mathcal{L}_{multi}^{(i)} = \mathcal{H}(f(X^{(i)}, \mathcal{W}_{main}^{(i)}), Y^{(i)}), \tag{3}$$

where $\mathcal{H}(\cdot)$ denotes a loss function (e.g., cross entropy), and $f(\cdot)$ is the proposed multi-path network. By learning the multi-path network, we can perform tasks under different paths (and different computation costs). Moreover, if the overall structure of the network g and f are the same, the number of parameters for a task in the conventional MTL method is equal to the number of parameters for the last task in the proposed method. Thus, the proposed method can contain the existing multi-task learning approaches sharing a single network by adjusting the branches. It is a more general framework that can cover existing MTL approaches and can produce subnetworks of lower computation costs.

C. SELF-AUXILIARY LEARNING

We adopt knowledge distillation [18] to transfer the knowledge from one network to another one (see Figure 2). Since early prediction requiring fewer parameters may occur performance degradation, the presented self-auxiliary learning strategy overcomes the problem. Assume that we have pre-trained networks $g^{(i)}$'s for the self-auxiliary tasks (denoted as Aux i in Figure 2), respectively, before learning the multi-path network. In the proposed method, the multi-path network receives task-specific knowledge from the pre-trained networks using the distillation loss $\mathcal{L}_{aux}^{(i)}$ as follows:

$$\mathcal{L}_{aux}^{(i)} = KL\left(u\left(\frac{f\left(X^{(i)}, \mathcal{W}_{main}^{(i)}\right)}{T}\right) \middle\| u\left(\frac{g^{(i)}\left(X^{(i)}, \mathcal{W}_{aux}^{(i)}\right)}{T}\right)\right), \tag{4}$$

where $KL(\cdot)$ denotes the Kullback-Leibler divergence function and $u(\cdot)$ is the softmax function. $\mathcal{W}_{aux}^{(i)}$ denotes the set of

parameters for the i^{th} self-auxiliary task. The temperature T controls the amount of transferred knowledge. We adopt the self-auxiliary networks g with the same capacity as the main network f. When it comes to inference, the auxiliary networks are not used. This means the self-auxiliary learning tasks do not require additional parameters on inference.

D. TOTAL LOSS

The loss of each task is computed by combining the losses from the multi-path network (Eq. (3)) and self-auxiliary learning (Eq. (4)). The multi-path network learns the set of entire parameters $\mathcal{W}_{main}$ by minimizing the total loss function across all tasks, which is defined as

$$\min_{\mathcal{W}_{main}} \mathcal{L}_{total} = \sum_{i=1}^{t} \alpha \mathcal{L}_{multi}^{(i)} + (1-\alpha)\mathcal{L}_{aux}^{(i)}, \tag{5}$$

where α is a balancing factor between the two losses $\mathcal{L}_{multi}^{(i)}$ and $\mathcal{L}_{aux}^{(i)}$. Note that the self-auxiliary tasks are pre-trained before learning the multi-path network, and the pre-trained networks $g^{(i)}$'s (and $\mathcal{W}_{aux}^{(i)}$) are held fixed during training with Eq. (5). The overall learning procedure of the proposed method is described in Algorithm 1. Given a set of datasets $\{\mathcal{D}^{(i)}\}_{1\le i\le t}$, we train the parameters of the proposed multi-path network $\mathcal{W}_{main}$ with the pre-trained networks $\{\mathcal{W}_{aux}^{(i)}\}_{1\le i\le t}$. For N iterations, we calculate the loss for the multi-path structure as decribed in Eq. (3) and the loss value for the self-auxiliary tasks is obtained according to the Eq. (4). Then, the total loss is derived by adding the $\mathcal{L}_{multi}^{(i)}$ and $\mathcal{L}_{aux}^{(i)}$. We optimize the set of parameters $\mathcal{W}_{main}$ by the gradient update from $\mathcal{L}_{total}$.

Algorithm 1: Multi-Path Network

Input: the number of tasks t, dataset $\{\mathcal{D}^{(i)}\}_{1\le i\le t}$
Pretrained networks: $\{\mathcal{W}_{aux}^{(i)}\}_{1\le i\le t}$
Initialize: $\mathcal{W}_{main} = \{\mathcal{W}_{main}^{(i)}\}_{1\le i\le t}$
for *iterations 1,2,...,N*
  for $i \in \{1,\ldots,t\}$
    Compute $\mathcal{L}_{multi}^{(i)}$ using Eq. (3)
    Compute $\mathcal{L}_{aux}^{(i)}$ using Eq. (4)
  end for
  Compute $\mathcal{L}_{total}$ using Eq. (5)
  Update $\mathcal{W}_{main} \leftarrow \mathcal{W}_{main} - \eta \frac{\partial \mathcal{L}_{total}}{\partial \mathcal{W}_{main}}$
end for

IV. EXPERIMENTS

A. SETUP

1) DATASETS

We used five datasets: CIFAR-10, CIFAR-100 [19], split CIFAR-10, split CIFAR-100 and Celeb-A [20]. CIFAR-10 and split CIFAR-10 contain 50,000 training and 10,000 test images of the 32×32 size. Split CIFAR-10 is originated from CIFAR-10 but its total classes (and the corresponding number of samples) are partitioned into several groups (see below for more details). CIFAR-100 and split CIFAR-100 have 50,000 training and 10,000 test images of the same image resolution to CIFAR-10. We constructed split CIFAR-100 similar to split CIFAR-10. Celeb-A is a face image dataset composed of 40 attributes and has 162,770 training and 39,829 test images of the 218×178 resolution.

2) SCENARIOS

We constructed a range of scenarios to analyze the method from diverse perspectives, including multiple tasks learning, hierarchical learning, and curriculum learning. Multiple tasks learning handles different classes (tasks), and the corresponding datasets are split CIFAR-10 and split CIFAR-100. We split CIFAR-10 and CIFAR-100 into five disjoint tasks so that each task includes 2 and 20 consecutive classes, respectively. In addition, to show the robustness against task order on multi-task learning, we conducted experiments with respect to randomly produced task orders using the split CIFAR-100. In hierarchical learning, we learned both coarse- and fine-class tasks in CIFAR-100 that consists of two levels of hierarchy: 20 coarse-classes and 100 fine-classes. To introduce another hierarchical level of the dataset, in this work, we further grouped the 20 coarse classes into 3 super classes. Accordingly, we renamed the coarse class as the intermediate-class and the fine-class as the sub-class. Since the coarse-class task includes less information than the fine-class task, the internal network with smaller parameters for the coarse-class task is assigned. In curriculum learning, we presume that the number of samples in data indicates the difficulty of the task because it is well-known that learning becomes easier (resp., harder) when many (resp., small) samples are given. To conduct the scenario, we allocated easy, normal, and hard tasks (attributes) from Celeb-A (see Figure 5). We used a subset of the dataset containing eight attributes out of 40 to ease the learning procedure while almost preserving the distribution of the original dataset.

3) COMPARED METHODS

We compared six algorithms in the experiments: Hard parameter sharing [7] as a baseline method, NestedNet [17], PackNet [25] with batch normalization, MTAN [24] and AdaShare [44]. Most of the compared methods do not increase the size of the network regardless of the number of tasks similar to ours.

While NestedNet updates the set of parameters up to the i-th task, PackNet does not update the parameters allocated for the previous tasks. MTAN performs task-specific feature level attention, and AdaShare suggests a task-adaptive sharing approach that determines which layers to share between tasks, while minimizing resource efficiency. We used the same settings, such as network configuration and optimizer, for all the compared approaches and trained them from scratch.

TABLE 2. Results of the multiple tasks learning scenario for split CIFAR-10 and CIFAR-100 on ResNet-18. Classification accuracy, the ratio of the number of parameters, and FLOPs of the compared methods are provided. Bold font shows the best accuracy or the least computation cost for each dataset, and underline gives the second best accuracy or the second least computation cost.

Dataset		CIFAR-10			CIFAR-100		
Task	Method	Acc (%)	Params Ratio (↓)	FLOPs (G)	Acc (%)	Params Ratio (↓)	FLOPs (G)
Task 1	Baseline	98.18	1	0.035	78.13	1	0.035
	PackNet	98.85	0.4	**0.012**	78.52	0.4	**0.012**
	NestedNet	98.98	0.4	**0.012**	77.92	0.4	**0.012**
	MTAN	**99.74**	1.5	0.058	**78.91**	1.5	0.058
	AdaShare	98.30	1	0.035	78.12	1	0.035
	Ours	95.05	**0.2**	0.027	78.65	**0.2**	0.027
Task 2	Baseline	94.79	1	0.035	78.65	1	0.035
	PackNet	91.88	0.6	**0.020**	79.57	0.6	**0.020**
	NestedNet	92.42	0.6	**0.020**	**80.75**	0.6	**0.020**
	MTAN	**96.88**	1.5	0.058	75.78	1.5	0.058
	AdaShare	92.15	1	0.035	80.59	1	0.035
	Ours	95.05	**0.4**	0.028	73.96	**0.4**	0.028
Task 3	Baseline	95.05	1	0.035	**83.85**	1	0.035
	PackNet	94.08	0.7	**0.026**	76.65	0.7	**0.026**
	NestedNet	94.98	0.7	**0.026**	77.58	0.7	**0.026**
	MTAN	**98.18**	1.5	0.058	83.33	1.5	0.058
	AdaShare	95.30	1	0.035	78.20	1	0.035
	Ours	95.57	**0.6**	0.030	83.59	**0.6**	0.030
Task 4	Baseline	**98.44**	1	0.035	75.78	1	0.035
	PackNet	97.97	**0.8**	**0.029**	70.67	**0.8**	**0.029**
	NestedNet	98.10	**0.8**	**0.029**	71.95	**0.8**	**0.029**
	MTAN	**98.44**	1.5	0.058	**79.43**	1.5	0.058
	AdaShare	98.41	1	0.035	77.30	1	0.035
	Ours	97.92	**0.8**	0.033	76.30	**0.8**	0.033
Task 5	Baseline	**97.92**	1	0.035	**84.90**	1	0.035
	PackNet	96.07	1	0.035	76.63	1	0.035
	NestedNet	96.02	1	0.035	76.70	1	0.035
	MTAN	96.09	1.5	0.058	83.07	1.5	0.058
	AdaShare	97.66	1	0.035	82.50	1	0.035
	Ours	96.88	1	0.035	82.81	1	0.035

TABLE 3. Results of the hierarchical scenario using two backbones (a) ResNet-18 and (b) MobileNetV2. We also provide the computation information such as the ratio of the number of parameters. The bold is the best accuracy or least computation. The underline is the second best accuracy or second least computation.

Task		Super-class (3)		Intermediate-class (20)		Sub-class (100)	
BackBone	Method	Acc (%)	Params Ratio (↓)	Acc (%)	Params Ratio (↓)	Acc (%)	Params Ratio (↓)
ResNet-18	Baseline	**93.45**	1	76.49	1	64.27	1
	PackNet	92.25	0.5	72.18	0.8	58.41	1
	NestedNet	92.27	0.5	72.29	0.8	59.26	1
	MTAN	93.29	1.5	76.34	1.5	64.30	1.5
	AdaShare	92.28	1	74.28	1	62.42	1
	Ours	92.44	**0.4**	**77.72**	**0.6**	**66.82**	1
	Ours w/o distill	92.31		75.62		63.85	
MobileNetV2	Baseline	90.82	1	68.32	1	55.53	1
	PackNet	88.95	0.5	64.78	0.8	51.08	1
	NestedNet	89.60	0.5	66.82	0.8	53.17	1
	MTAN	90.10	1.3	65.79	1.3	53.56	1.3
	AdaShare	**91.56**	1	68.78	1	54.78	1
	Ours	90.76	**0.3**	**71.48**	**0.6**	**59.42**	1
	Ours w/o distill	**91.56**		69.91		57.25	

B. IMPLEMENTATION DETAILS

We built our method under ResNet and MobileNetV2 backbone architectures. While ResNet was used for the CIFAR datasets, MobileNetV2 (with multiplier 0.5) was deployed for ImageNet. Note that we did not downsize the CIFAR images at the input layer for MobileNetV2, since the size of

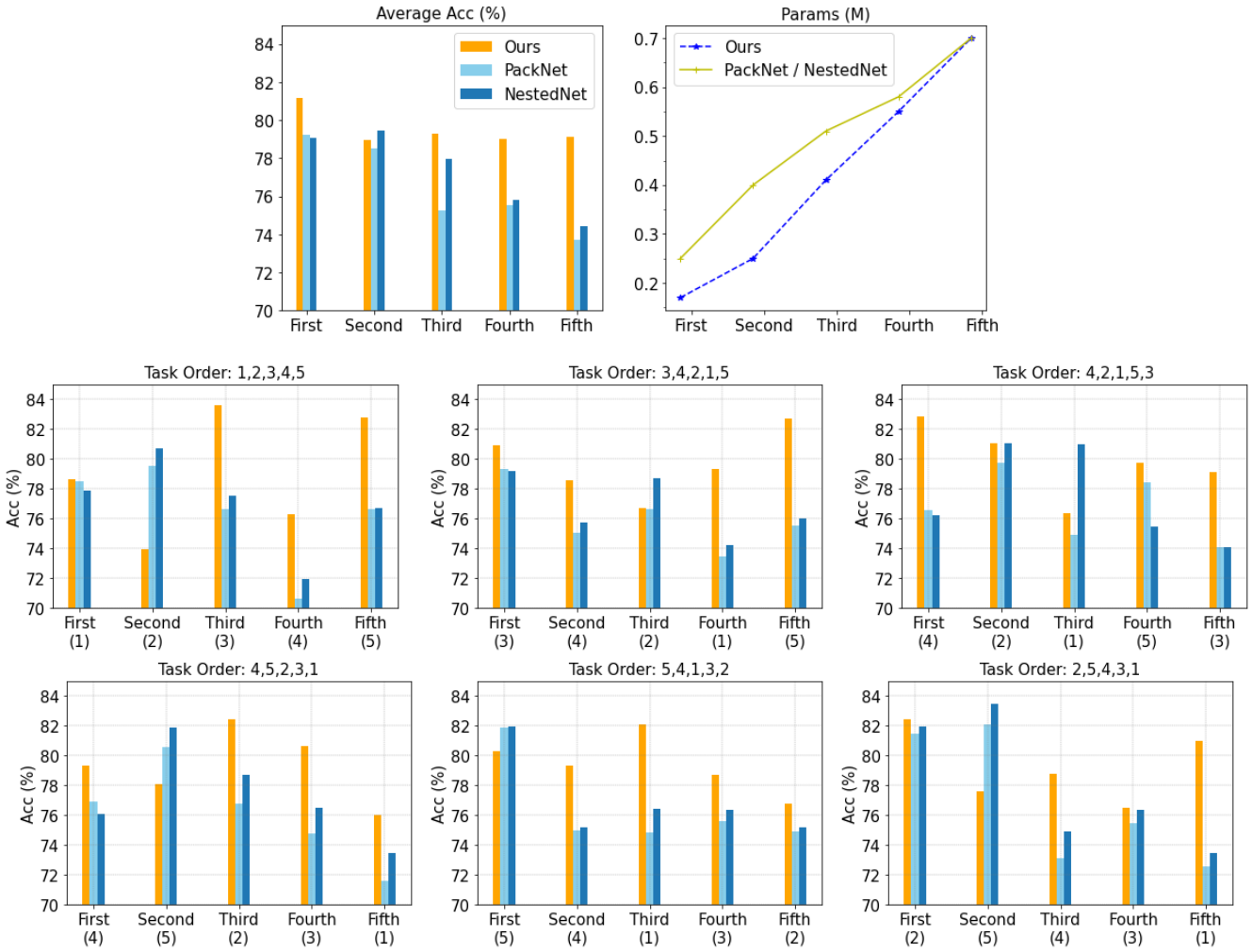


FIGURE 3. Results for split CIFAR-100 with random task orders on ResNet-18.

CIFAR is much smaller than that of ImageNet. We applied the SGD optimizer with the Nesterov momentum of 0.9 for both ResNet and MobileNetV2. The proposed network was trained with an initial learning rate of 0.1 and the learning rate was decayed by a factor of 10 when the training loss converges. The batch size of 128 for CIFAR-10 and CIFAR-100 and 64 for Celeb-A were used in all experiments. For the proposed method, we constructed five branches in the multiple tasks learning scenario and three branches in other scenarios. The branches take 25%, 36%, 58%, 79%, 100% of the number of parameters in the multiple tasks learning scenario and around 30%, 75%, 100% of the number of parameters in other scenarios, respectively. Implementation of the proposed method was conducted under the PyTorch library [45].

C. MULTIPLE TASKS LEARNING

For the first multiple tasks learning scenario, we used split CIFAR-10 and CIFAR-100. We partitioned the classes into five disjoint groups for both CIFAR-10 and CIFAR-100. Each group has 2 successive classes for CIFAR-10 and has 20 successive classes for CIFAR-100. Accordingly, we assigned the groups (tasks) to the branches by order of class labels, respectively. We compared with Baseline and major competitors sharing a similar idea to ours, PackNet, NestedNet, MTAN and AdaShare. For self-auxiliary learning, we set the temperature T to 2, and α was set to 1 to distill knowledge from self-auxiliary tasks to main tasks.

Table 2 shows the results of the scenario for three measures: accuracy, the number of parameters, and FLOPs. First, for split CIFAR-10, Baseline and MTAN show good performance but the computation cost (the number of parameters and FLOPs) is not efficient. Overall, the proposed method is comparable to other methods while consuming less number of parameters.

For split CIFAR-100, which is more challenging than split CIFAR-10 due to the large number of classes, the proposed method shows similar performance to Baseline, MTAN, and AdaShare. When it comes to the computation cost, the compared methods have twice as many parameters as the proposed method on average. Note that there is no significant

TABLE 4. Three-level class hierarchy of the CIFAR-100 dataset. The hierarchy contains 3 super-classes, 20 intermediate-classes, and 100 sub-classes.

Super-class (3)	Intermediate-class (20)	Sub-class (100)
Animals	Fish	Aquarium fish, Flatfish, Ray, Shark, Trout
	Insects	Bee, Beetle, Butterfly, Caterpillar, Cockroach
	Reptiles	Crocodile, Dinosaur, Lizard, Snake, Small mammals, Turtle
	Non-insect invertebrates	Crab, Lobster, Snail, Spider, Worm
	Aquatic mammals	Beaver, Dolphin, Otter, Seal, Whale
	Large carnivores	Bear, Leopard, Lion, Tiger, Wolf
	Large omnivores and Herbivores	Camel, Cattle, Chimpanzee, Elephant, Kangaroo
	Medium-sized mammals	Fox, Porcupine, Possum, Raccoon, Skunk
	People	Baby, Boy, Girl, Man, Woman
	Small mammals	Hamster, Mouse, Rabbit, Shrew, Squirrel
Things	Food containers	Bottle, Bowl, Can, Cup, Plate
	Household electrical device	Clock, Computer keyboard, Lamp, Telephone, Television
	Household furniture	Bed, Chair, Couch, Table, Wardrobe
	Large man-made outdoor things	Bridge, Castle, House, Road, Skyscraper
	Large natural outdoor scenes	Cloud, Forest, Mountain, Plain, Sea
	Vehicles 1	Bicycle, Bus, Motorcycle, Pickup truck, Train
	Vehicles 2	Lawn mower, Rocket, Streetcar, Tank, Tractor
Plants	Flowers	Orchid, Poppy, Rose, Sunflower, Tulip
	Fruit and Vegetables	Apple, Mushroom, Orange, Pear, Sweet pepper
	Trees	Maple tree, Oak tree, Palm tree, Pine tree, Willow tree

performance drop in our method compared to them (1% $\sim$ 2% drop on average). The self-auxiliary learning strategy enriches the task-specific knowledge of the multi-path architecture. However, the performance leak can be gained without the self-auxiliary learning strategy since the tasks predicted early consume a subset of the parameters. Consequently, it can maintain the performance while requiring fewer computations by applying the proposed method.

Furthermore, we demonstrated the proposed method under different randomly shuffled task orders as shown in Figure 3. Note that the number of parameters used to perform tasks in our multi-path network does not change from the experiment for the CIFAR-100 dataset. The results from the figure indicate that even if the task order is different, the proposed method performed better than other competitors, PackNet and NestedNet, with the smaller number of parameters. In particular, the proposal achieved the highest accuracy for four out of five tasks from the experiments, showing its excellence and robustness against task order.

D. HIERARCHICAL LEARNING

In this scenario, we learn class hierarchy from the coarsest to the finest in the proposed multi-path network. We used CIFAR-100 and constructed three levels of hierarchy (super-, intermediate-, sub-classes) as mentioned earlier (See more detail in Table 4). We learn super-classes with the lowest-level path, intermediate-classes to the intermediate-level path, and sub-classes to the highest-level path, respectively, to make the network learn the hierarchical structure of the dataset. The proposed approach was compared with Baseline, NestedNet, PackNet, MTAN, and AdaShare that can address class hierarchy. In this scenario, we used the same backbone architectures as the previous scenario. We set the temperature T to 2 and α to 1.

Table 3 shows the classification accuracy and computation cost among the compared methods for super-, intermediate-, and sub-classes of the CIFAR-100 dataset. As we can see in Table 3, the proposed method outperforms the other methods on ResNet-18 and MobileNetV2 with multiplier 0.5, respectively. Interestingly, although the competitors require a larger number of parameters than the proposed method, the performance of ours is higher than the other approaches.

We also performed another experiment to demonstrate the effectiveness of the self-auxiliary task, denoted as ours w/o distillation, in Table 3 denotes the accuracy with respect to different numbers of parameters of the proposed method without applying distillation (described in Section III-C). The proposed method without distillation does not suffer from a significant performance drop compared to ours with distillation. The proposal without distillation still gives competitive performance when compared to the other methods on both architectures while consuming a smaller amount of parameters. In addition, Table 3 reports the best performance of the proposed method without self-auxiliary learning on MobileNetV2 compared to other methods. The results indicate that the proposed hierarchical structure enables learning of the hierarchical knowledge better than the compared methods. At a lower-level path, the coarser knowledge is learned, and at a higher-level path, the richer knowledge is attained using the coarser knowledge, resulting in a performance gain.

E. CURRICULUM LEARNING

In the curriculum learning scenario, we experimented on Celeb-A that has attributes (tasks) of different distributions.

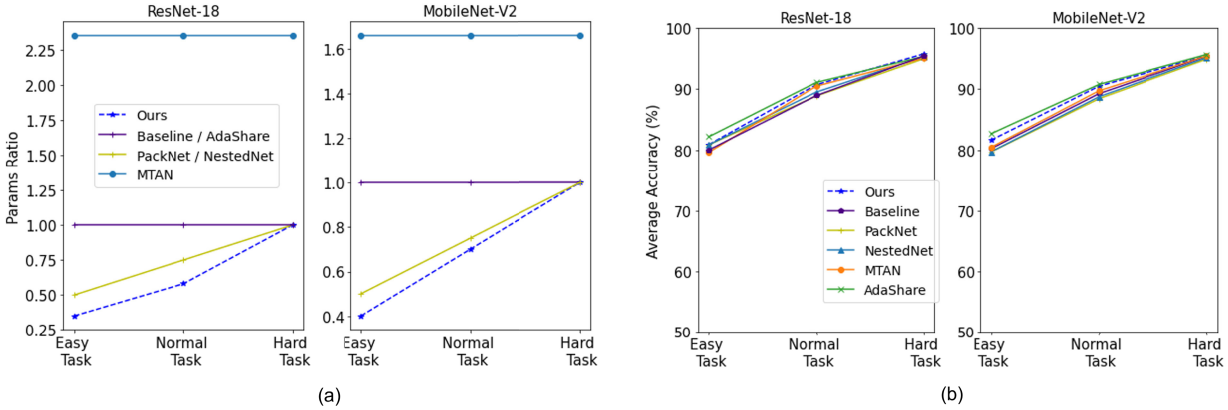


FIGURE 4. (a) The ratio of parameters used to perform each task. (b) The average accuracy of the each task.

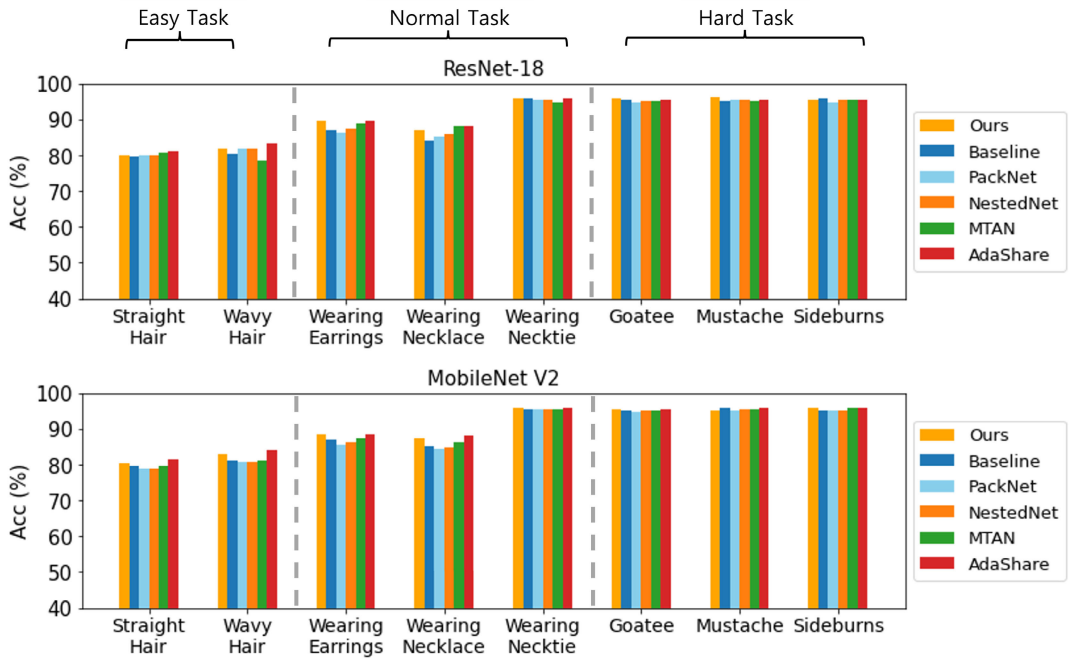


FIGURE 5. Results for the curriculum learning scenario on the Celeb-A using ResNet-18 and MobileNetV2.

For the scenario, we made three tasks according to difficulty; each, normal and hard tasks that contain small to large number of samples, respectively. Accordingly, the proposed multi-path network contains three branches such that we performed easy tasks in the early branch, normal tasks in the middle branch, hard tasks in the last branch, respectively. The temperature and α are set to 2 and 0.9 for both backbones.

The results of the curriculum learning scenario are summarized in Figure 4. For ResNet-18, the proposed method achieves the best or the second-best performance in the six tasks out of eight and for MobileNetV2, the proposed method maintains the best accuracy or the second best accuracy among all tasks. Multiple inference paths at different difficulties of tasks make the proposed method distribute the importance of parameters for each task, which can reduce the negative interference between tasks and thus yields better results than the compared methods. When it comes to parameter usage, our approach is superior to Baseline, PackNet, NestedNet, MTAN and AdaShare on average. Ours requires less than half the number of parameters when compared to Baseline, MTAN and AdaShare in the easy task on both architectures.

Figure 4 reports the ratio of parameters and average accuracy on the curriculum learning scenario. As shown in Figure 4 (a), the proposed method requires the smallest number of parameters, while MTAN consumes the largest number of parameters among the compared methods. PackNet and NestedNet also use a smaller amount of parameters compared

to Baseline. However, they require a larger number of parameters compared to ours while showing poorer performance shown in Figure 4. Figure 5 shows the accuracy of individual tasks. It can be seen that the performance of ours is similar or superior to the most methods on both architectures. Note that the required resource is reported in Figure 4 (a).

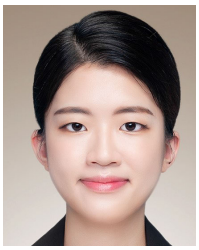
V. CONCLUSION

In this work, we have proposed a multi-task learning framework to resolve the combined problem of performing tasks under different computation costs and avoiding task interference. The proposed method is realized by a multi-path network with self-auxiliary learning. The multi-path network is constructed hierarchically to perform tasks under different levels of hierarchy (or prediction paths), resulting in the mitigation of negative interference. Furthermore, since the multi-path structure is composed of internal networks with different sizes, diverse memory budgets can be handled without extra training efforts. To boost the performance of the network, self-auxiliary learning has been presented, which supplements the task-specific knowledge to the main tasks. The proposed method has been demonstrated under an extensive set of experimental scenarios and has shown its efficiency with respect to both task performance and computational cost compared to other multi-task learning approaches. In future work, we investigate the scalability of the method for more challenging computer vision tasks other than classification tasks.

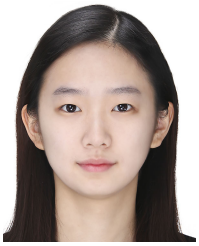
REFERENCES

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017, *arXiv:1706.03762*.
[3] S. Zagoruyko and N. Komodakis, "Wide residual networks," 2016, *arXiv:1605.07146*.
[4] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 580–587.
[5] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 483–499.
[6] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 3431–3440.
[7] R. Caruana, "Multitask learning," *Mach. Learn.*, vol. 28, no. 1, pp. 41–75, 1998.
[8] Z. Cao, C.-T. Lin, Y. Deng, and G.-W. Weber, "Guest editorial: Fuzzy systems toward human-explainable artificial intelligence and their applications," *IEEE Trans. Fuzzy Syst.*, vol. 29, no. 12, pp. 3577–3578, Dec. 2021.
[9] L. Hu, J. Zhang, X. Pan, H. Yan, and Z.-H. You, "HiSCF: Leveraging higher-order structures for clustering analysis in biological networks," *Bioinformatics*, vol. 37, no. 4, pp. 542–550, May 2021.
[10] L. Hu, K. C. C. Chan, X. Yuan, and S. Xiong, "A variational Bayesian framework for cluster analysis in a complex network," *IEEE Trans. Knowl. Data Eng.*, vol. 32, no. 11, pp. 2115–2128, Nov. 2020.
[11] X. Liu, P. He, W. Chen, and J. Gao, "Multi-task deep neural networks for natural language understanding," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, 2019, pp. 4487–4496.
[12] K. Clark, M.-T. Luong, U. Khandelwal, C. D. Manning, and Q. V. Le, "BAM! Born-again multi-task networks for natural language understanding," 2019, *arXiv:1907.04829*.
[13] X. Zhao, H. Li, X. Shen, X. Liang, and Y. Wu, "A modulation module for multi-task learning with applications in image retrieval," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 401–416.
[14] K.-K. Maninis, I. Radosavovic, and I. Kokkinos, "Attentive single-tasking of multiple tasks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 1851–1860.
[15] M. Kanakis, D. Bruggemann, S. Saha, S. Georgoulis, A. Obukhov, and L. V. Gool, "Reparameterizing convolutions for incremental multi-task learning without task interference," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2020, pp. 689–707.
[16] C. Ahn, E. Kim, and S. Oh, "Deep elastic networks with model selection for multi-task learning," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 6529–6538.
[17] E. Kim, C. Ahn, and S. Oh, "NestedNet: Learning nested sparse structures in deep neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8669–8678.
[18] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, *arXiv:1503.02531*.
[19] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, ON, Canada, Tech. Rep., 2009.
[20] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 3730–3738.
[21] I. Misra, A. Shrivastava, A. Gupta, and M. Hebert, "Cross-stitch networks for multi-task learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 3994–4003.
[22] 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 *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 3205–3214.
[23] D. Xu, W. Ouyang, X. Wang, and N. Sebe, "PAD-Net: Multi-tasks guided prediction-and-distillation network for simultaneous depth estimation and scene parsing," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 675–684.
[24] S. Liu, E. Johns, and A. J. Davison, "End-to-end multi-task learning with attention," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 1871–1880.
[25] A. Mallya and S. Lazebnik, "PackNet: Adding multiple tasks to a single network by iterative pruning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 7765–7773.
[26] E. Kim, C. Ahn, P. H. S. Torr, and S. Oh, "Deep virtual networks for memory efficient inference of multiple tasks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 2710–2719.
[27] X. Lin, H.-L. Zhen, Z. Li, Q.-F. Zhang, and S. Kwong, "Pareto multi-task learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 12060–12070.
[28] O. Sener and V. Koltun, "Multi-task learning as multi-objective optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 525–536.
[29] M. Guo, A. Haque, D.-A. Huang, S. Yeung, and L. Fei-Fei, "Dynamic task prioritization for multitask learning," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 270–287.
[30] 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*, vol. 33, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds. Red Hook, NY, USA: Curran Associates, 2020, pp. 5824–5836.
[31] L. J. Ba and R. Caruana, "Do deep nets really need to be deep?" 2013, *arXiv:1312.6184*.
[32] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "FitNets: Hints for thin deep nets," 2014, *arXiv:1412.6550*.
[33] Y. Guan, P. Zhao, B. Wang, Y. Zhang, C. Yao, K. Bian, and J. Tang, "Differentiable feature aggregation search for knowledge distillation," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2020, pp. 469–484.
[34] S. Zagoruyko and N. Komodakis, "Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer," in *Proc. ICLR*, 2017, pp. 1–13.
[35] P. Luo, Z. Zhu, Z. Liu, X. Wang, and X. Tang, "Face model compression by distilling knowledge from neurons," in *Proc. AAAI Conf. Artif. Intell.*, 2016, pp. 3560–3566.
[36] X. Chen, Y. Zhang, Y. Wang, H. Shu, C. Xu, and C. Xu, "Optical flow distillation: Towards efficient and stable video style transfer," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2020, pp. 614–630.

[37] Z. Sun, H. Yu, X. Song, R. Liu, Y. Yang, and D. Zhou, "MobileBERT: A compact task-agnostic BERT for resource-limited devices," in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics*, 2020, pp. 2158–2170.
[38] S. Hahn and H. Choi, "Self-knowledge distillation in natural language processing," 2019, *arXiv:1908.01851*.
[39] W.-H. Li and H. Bilen, "Knowledge distillation for multi-task learning," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2020, pp. 163–176.
[40] J. Rajasegaran, S. Khan, M. Hayat, F. S. Khan, and M. Shah, "ITAML: An incremental task-agnostic meta-learning approach," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 13588–13597.
[41] D. Abati, J. Tomczak, T. Blankevoort, S. Calderara, R. Cucchiara, and B. E. Bejnordi, "Conditional channel gated networks for task-aware continual learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 3931–3940.
[42] Z. Yan, H. Zhang, R. Piramuthu, V. Jagadeesh, D. DeCoste, W. Di, and Y. Yu, "HD-CNN: Hierarchical deep convolutional neural networks for large scale visual recognition," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 2740–2748.
[43] Y. Wang, W. Gan, J. Yang, W. Wu, and J. Yan, "Dynamic curriculum learning for imbalanced data classification," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 5017–5026.
[44] X. Sun, R. Panda, R. Feris, and K. Saenko, "AdaShare: Learning what to share for efficient deep multi-task learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 8728–8740.
[45] A. Paszke *et al.*, "PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, vol. 32, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2019, pp. 8024–8035.



SOYEON PARK received the B.S. degree in information statistics from Dongduk Women's University, Seoul, South Korea, in 2020. She is currently pursuing the M.S. degree with the School of Computer Science and Engineering, Chung-Ang University, Seoul. Her research interests include machine learning, deep learning, and computer vision.



JIHO LEE received the B.S. degree in electrical and electronics engineering from Chung-Ang University, Seoul, South Korea, in 2021, where she is currently pursuing the M.S. degree with the School of Computer Science and Engineering. Her research interests include machine learning, automated deep learning, and its application to real world problems.

EUNWOO KIM (Member, IEEE) received the B.S. degree in electrical and electronics engineering from Chung-Ang University, Seoul, South Korea, in 2011, and the M.S. and Ph.D. degrees in electrical engineering and computer science from Seoul National University, Seoul, in 2013 and 2017, respectively. He is currently an Assistant Professor with the School of Computer Science and Engineering, Chung-Ang University. From 2017 to 2018, he was a Postdoctoral Researcher with the Department of Electrical Engineering and Computer Science, Seoul National University. From 2018 to 2019, he was a Postdoctoral Researcher with the Department of Engineering Science, University of Oxford, Oxford, U.K. His research interests include machine learning, deep learning, model optimization, robotics, and computer vision.