IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 5 November 2023, accepted 30 November 2023, date of publication 5 December 2023,
date of current version 13 December 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3339793

RESEARCH ARTICLE

Task-Aware Dynamic Model Optimization for Multi-Task Learning

SUJIN CHOI[1], HYUNDONG JIN[2], AND EUNWOO KIM[1,2], (Member, IEEE)
[1]Department of Artificial Intelligence, Chung-Ang University, Seoul 06974, Republic of Korea
[2]School of Computer Science and Engineering, Chung-Ang University, Seoul 06974, Republic of Korea

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

This work was supported in part by the Chung-Ang University Research Scholarship Grants, in 2022; and in part by the Institute of Information and Communications Technology Planning and Evaluation (IITP) Grant funded by the Korean Government through MSIT [Artificial Intelligence Graduate School Program (Chung-Ang University)] under Grant 2021-0-01341.

ABSTRACT Multi-task learning (MTL) is a field in which a deep neural network simultaneously learns knowledge from multiple tasks. However, achieving resource-efficient MTL remains challenging due to entangled network parameters across tasks and varying task-specific complexity. Existing methods employ network compression techniques while maintaining comparable performance, but they often compress uniformly across all tasks without considering individual complexity. This can lead to suboptimal solutions due to entangled network parameters and memory inefficiency, as the parameters for each task may be insufficient or excessive. To address these challenges, we propose a framework called Dynamic Model Optimization (DMO) that dynamically allocates network parameters to groups based on task-specific complexity. This framework consists of three key steps: measuring task similarity and task difficulty, grouping tasks, and allocating parameters. This process involves the calculation of both weight and loss similarities across tasks and employs sample-wise loss as a measure of task difficulty. Tasks are grouped based on their similarities, and parameters are allocated with dynamic pruning according to task difficulty within their respective groups. We apply the proposed framework to MTL with various classification datasets. Experimental results demonstrate that the proposed approach achieves high performance while taking fewer network parameters than other MTL methods.

INDEX TERMS Multi-task learning, resource-efficient learning, model optimization.

I. INTRODUCTION

Multi-task learning (MTL) [1], [2] has gained attention as a way to handle multiple tasks simultaneously and efficiently. The primary objective of MTL is to enhance generalization performance by sharing beneficial knowledge among tasks. MTL can be categorized into two main approaches: hard parameter sharing [1], [3], [4], [5], [6] and soft parameter sharing [7], [8], [9], [10], [11]. Hard parameter sharing [1], [3], [4], [5], [6] employs a single shared network, which offers the advantage of efficient resource utilization. Nevertheless, sharing knowledge between unrelated tasks in hard parameter sharing can result in task interference, leading to a suboptimal solution. Soft parameter sharing [7], [8], [10], [11] shares information across tasks through task-specific networks. Unlike hard parameter sharing, this approach shares information across tasks using task-specific networks, which reduces interference among tasks. Nevertheless, as the total number of tasks grows, memory consumption also increases, limiting the applicability of this approach in real-world situations with restricted resources.

The constraints of resource-limited applications have led to an increased need for resource-efficient MTL methods. However, achieving this efficiency is difficult due to several factors. Entangled parameters across tasks can negatively impact performance. Neglecting to address task-specific

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

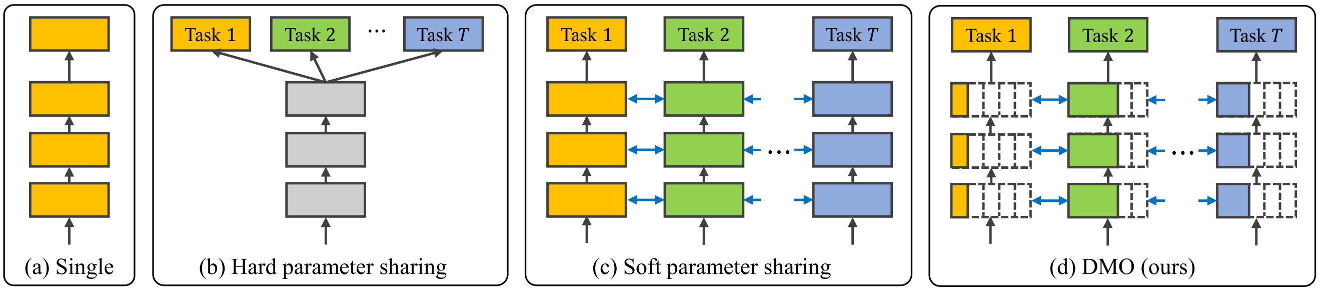

FIGURE 1. Overview of single-task and multi-task learning approaches. (a) represents a model (network) for a task. (b), (c) and (d) represent the models for performing multiple tasks, respectively. (b) represents a typical hard parameter-sharing approach with a single shared network learned for all tasks. The layers shared across tasks are represented by a gray box, while other color boxes represent task-specific layers. (c) represents a soft parameter-sharing approach, which contains task-specific backbones. (d) represents our method, dynamic model optimization (DMO), one of the soft-parameter sharing approaches. DMO leverages task-wise pruning to dynamically allocate network parameters based on the measured task-wise difficulty tailored to the specific requirements of each task. The colored box corresponds to the parameters that survive in the layer, while the white dotted box represents pruned parameters that contribute little to the network.

complexities may lead to resource inefficiency. These complexities involve different levels of detail across tasks, possibly requiring distinct network structures. Some tasks might need resource-intensive deep neural networks, while others can achieve satisfactory results with simpler, shallower models. To address these challenges, various MTL approaches [5], [6], [12] have been proposed. These studies focus on network compression through hard parameter sharing, addressing parameter entanglement between tasks. However, they learn a common representation for all tasks with a single shared network and are prone to task interference.

To address this limitation, we propose a task-aware Dynamic Model Optimization (DMO) framework based on a soft parameter-sharing model while adapting to the diverse complexities of individual tasks. The structure is illustrated in Figure 1. Our approach consists of 1) measuring task similarity and difficulty, 2) grouping tasks according to their similarity, and 3) allocating parameters based on their difficulty. We measure task-wise difficulty with sample-wise loss and calculate weight and loss-based similarity across tasks. We group tasks based on similarity and allocate parameters accordingly with dynamic pruning based on task difficulty. The proposed framework enables memory-efficient MTL that can handle varying complexity for different tasks by subnetworks that consume minimal parameters.

We evaluate the proposed framework to a wide range of MTL scenarios, including CIFAR-10 [13], STL-10 [14], MNIST [15], USPS [16], CIFAR-100 [13], and the Visual Decathlon Challenge datasets [17]. Experimental results show that the proposed method can achieve better memory efficiency compared to existing MTL methods [4], [6], [8], [10]. Specifically, our approach achieves an average reduction of over 80% in the total number of parameters compared to the soft parameter sharing approaches [7], [8], [10] while achieving competitive performance. The main contributions of our work are as follows:

- We propose the Dynamic Model Optimization (DMO) method that optimizes multiple tasks according to their complexity.
- DMO groups tasks based on inter-task similarity, determined by both weights and losses, enabling tasks within the same group to share knowledge.
- We produce memory-efficient subnetworks by dynamically allocating network parameters based on the relative difficulty of each task within a group.
- We demonstrate the effectiveness of DMO through extensive experiments on various benchmarks, showing that it outperforms existing MTL methods in terms of memory-accuracy trade-off.

II. RELATED WORK

A. MULTI-TASK LEARNING

In MTL, the overall loss is typically calculated as a weighted sum of task-specific loss functions, with the loss of each task being assigned a specific weight that denotes its relative importance. Previous works in the literatures [9], [12], [18], [19], [20], and [21] have aimed to optimize model parameters across all tasks simultaneously, considering the importance of each task in the overall learning process. Uncertainty [20] estimates task uncertainty using noise parameters for task weight assignment and demonstrates effective task balancing. Gradnorm [18] stabilizes MTL training by dynamically adjusting gradient magnitudes for task balance. Dynamic Priority [19] prioritizes tasks based on difficulty and improves learning efficiency compared to previous methods (e.g., Uncertainty and Gradnorm) through adaptive weight updates. BMTL [21] advances this by dynamically balancing tasks based on difficulty assessed via training loss values, optimizing tasks with high losses and potential for improvement. Motivated by Dynamic Priority [19] and BMTL [21], we aim to develop a memory-efficient regularization technique that allocates network parameters based on task complexity.

B. RESOURCE-EFFICIENT LEARNING

Resource-efficient learning aims to minimize model size, making it suitable for environments with limited resources. Model-level resource-efficient learning techniques, such

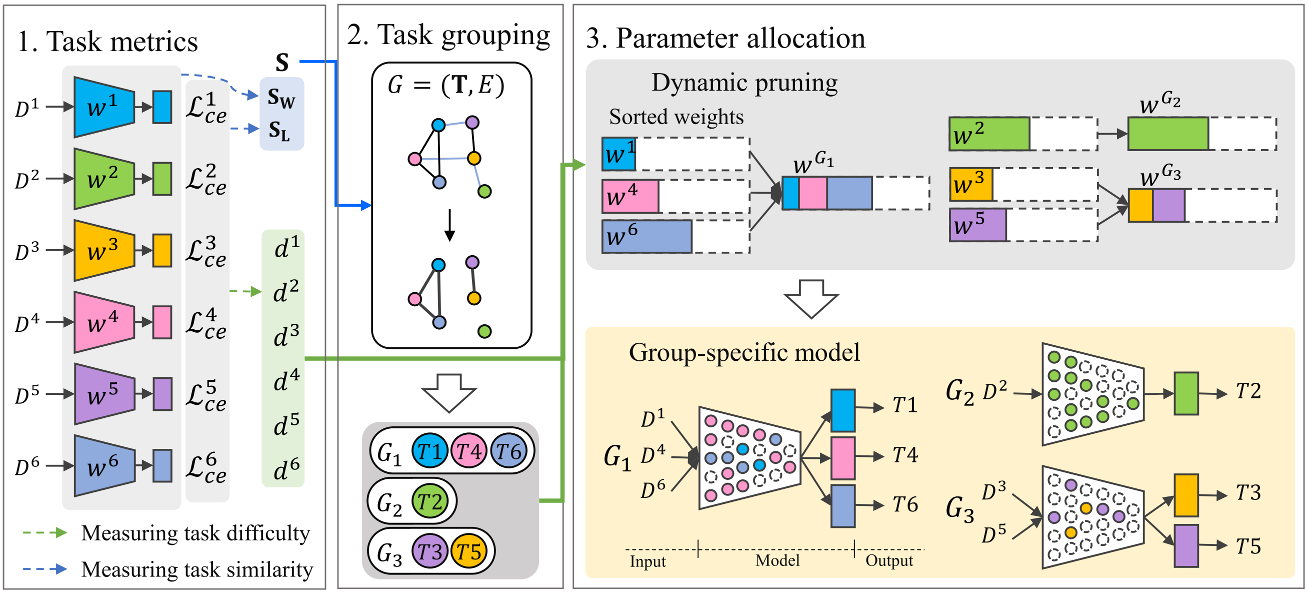
1. Task metrics
S
D^1 w^1 $\mathcal{L}_{ce}^1$
D^2 w^2 $\mathcal{L}_{ce}^2$
D^3 w^3 $\mathcal{L}_{ce}^3$
D^4 w^4 $\mathcal{L}_{ce}^4$
D^5 w^5 $\mathcal{L}_{ce}^5$
D^6 w^6 $\mathcal{L}_{ce}^6$
$\mathbf{S_W}$
$\mathbf{S_L}$
d^1 d^2 d^3 d^4 d^5 d^6
Measuring task difficulty
Measuring task similarity
2. Task grouping
$G = (\mathbf{T}, E)$
G_1 T1 T4 T6
G_2 T2
G_3 T3 T5
3. Parameter allocation
Dynamic pruning
Sorted weights
w^1 w^4 w^6
w^{G_1}
w^2 w^{G_2}
w^3 w^5 w^{G_3}
Group-specific model
G_1 D^1 D^4 D^6 T1 T4 T6
G_2 D^2 T2
G_3 D^3 D^5 T3 T5
Input
Model
Output

FIGURE 2. A visualization of the DMO framework. DMO groups tasks based on task similarity, and dynamically allocates network parameters to each group according to the measured task difficulty. In the first stage, task difficulty level d^t is measured using loss $\mathcal{L}_{ce}^t$, as shown in the green rounded rectangle. The weight similarity matrix $\mathbf{S_W}$, and the loss similarity matrix $\mathbf{S_L}$, are determined using weight w^t and loss $\mathcal{L}_{ce}^t$, as shown in the blue rounded rectangle. In the second stage, tasks of high similarity are grouped into a graph structure using maximal cliques for identification and grouping. During the parameter allocation stage, indicated by a gray rounded rectangle, colored rectangular boxes symbolize allocated task parameters. In contrast, white boxes represent pruned parameters that have made minimal contributions to improving model performance or accuracy. Finally, each yellow rounded box provides group-specific models, which are allocated with corresponding weights from previous stages.

as pruning, quantization, and ensembling of local and global features [2], [22], [23], effectively reduce memory usage while maintaining performance comparable to that of the original dense networks. Recent studies propose resource-efficient MTL methods targeting three aspects: sharing mechanisms [5], [24], interference reduction [6], and task grouping [25]. Sharing mechanisms enable tasks to share model components, reducing model size and resource usage. Interference reduction mitigates the negative effects of shared resources between tasks. Task grouping investigates the effective clustering of tasks to reduce interference and enhance performance. Sparse Sharing [24] improves performance with fewer parameters using task-specific masks. DiSparse [5] evaluates task importance for pruning but struggles to prevent task interference. TAPS [6] minimizes task interference with a small subset of layers but may lead to resource inefficiency. TAG [25] identifies task groups, but its heuristic approach may cause performance decline.

To address these challenges, we propose a technique that shares parameters based on task difficulty, accounting for the diverse complexities of each task. Additionally, we mitigate task interference by utilizing a soft parameter-sharing model and grouping similar tasks based on measuring their similarity.

III. METHOD

A. OVERALL FRAMEWORK

The DMO framework, shown in Figure 2, groups tasks based on similarity and dynamically allocates network parameters according to the difficulty of each task. Additionally, the proposed method is performed without requiring human intervention. It addresses parameter entanglement and memory inefficiencies by considering task similarity and complexity during grouping and allocation. The framework comprises three stages: measuring task difficulty and similarity, grouping tasks, and allocating parameters based on difficulty. Initially, task difficulty and similarity are measured using weight and loss metrics obtained by training individual task-specific networks. Subsequently, tasks are grouped based on these calculated similarities, where parameters within each group-specific network are shared among the tasks. Lastly, the measured task difficulty is mapped into a sparsity ratio to discard redundant parameters within each group-wise network. In conclusion, we derive memory-efficient sparse subnetworks.

B. MEASURING THE TASK SIMILARITY AND DIFFICULTY

To group similar tasks, we measure task similarity using two approaches: one based on task weights and the other on training losses. We employ loss as an indirect metric to evaluate the behavior of the network on various tasks. We utilize the Pearson correlation coefficient to identify linear relationships in task weight distributions, which helps us group tasks that could benefit from shared information. Specifically, we calculate the similarity between the weights w^i of the task i and w^j of the task j using the Pearson correlation coefficient, where $i \neq j, i, j \in \{1, \ldots, t, \ldots, T\}$, and T denotes the total number of tasks. This computation produces a symmetric matrix of dimensions $T \times T$, denoted as $\mathbf{S_W}$, representing the similarity scores for all task weight pairs. Note that we can calculate the similarity between weights of different dimensions using [26], [27]. Similarly, we define the loss for the task t as L^t. By calculating the Pearson correlation coefficients for all task losses, we derive a symmetric $T \times T$ correlation matrix, denoted as $\mathbf{S_L}$. Finally, we introduce a combined task similarity measure, $\mathbf{S}$, which integrates the two types of similarities, weight and loss. The formulation is as follows:

$$\mathbf{S} = \alpha \mathbf{S_W} + (1 - \alpha)\mathbf{S_L}, \tag{1}$$

where α is a parameter to balance two similarities.

To measure the relative difficulty of each task, we measure the difficulty of each sample. To differentiate relatively difficult samples among samples of the tasks, we use the average training loss across all task samples as a criterion. Specifically, we denote $\mathcal{L}_n^t$ as the training loss for the n-th sample of task t. We calculate the average training loss across tasks as

$$\mu = \frac{1}{T}\frac{1}{N^t}\sum_{t=1}^{T}\sum_{n=1}^{N^t}\mathcal{L}_n^t, \tag{2}$$

where N^t represents the number of samples for task t. We define the difficult sample if the loss of the sample is greater than a predefined value μ. The difficulty of task t is calculated by the ratio of difficult samples in task t as

$$d^t = \frac{1}{N^t}\sum_{n=1}^{N^t}\mathbb{1}_{[\mathcal{L}_n^t \geq \mu]}. \tag{3}$$

C. TASK GROUPING

To group tasks that share similar features, we use a graph-based approach to capture relationships between tasks. Each of these groups is associated with its distinct network. By doing so, we reduce the parameter redundancy of the model and promote knowledge sharing between tasks within the same group. We use a graph-based approach [28] to capture relationships between tasks. To create task groups based on similarity, we construct graph $G = (\mathbf{T}, E)$, where $\mathbf{T}$ denotes a set of tasks represented by nodes and E comprises edges connecting tasks with similarity scores $\mathbf{S}$ exceeding a threshold τ_{sim}, i.e.,

$$E = \{(i, j) \mid \mathbf{S}_{i,j} > \tau_{sim}, \forall i \neq j, i, j \in \mathbf{T}\}. \tag{4}$$

We define nodes as tasks in $\mathbf{T}$ and edges using similarity scores $\mathbf{S}$, allowing for efficient grouping that captures task relationships. In a graph, a clique is defined as a subset of nodes in which every node is directly interconnected with all other nodes within that subset. During the task grouping process, we form groups by identifying the largest cliques within the graph, representing strongly connected tasks with high similarity. We utilize the Bron-Kerbosch algorithm [29] to identify these largest cliques, denoted as $\mathcal{C}$, based on the similarity scores and a threshold value τ_{BK}. We set τ_{BK} as 0.8 in all experiments. From the discovered cliques in $\mathcal{C}$, we specify distinct task groups. For each clique from $\mathcal{C}$, we establish a new group consisting of tasks that have not yet been assigned to any existing group. Specifically, tasks from a particular clique, denoted as $C_k \in \mathcal{C}$, are assigned to a group G_k. A task t is included in the group if it has not already been assigned to other groups. The task assignment process continues until all cliques in $\mathcal{C}$ have been examined, thereby ensuring that all tasks with shared knowledge are assigned to their appropriate groups.

D. GROUP-SPECIFIC MODEL

We allocate parameters considering task-specific complexity by performing dynamic pruning to improve memory efficiency. We adopt unstructured magnitude-based pruning [30], [31], which facilitates the removal of redundant parameters without adhering to a specific structural pattern. Due to its flexibility in removing weights from any location, unstructured pruning offers higher compression rates. Subsequently, we merge the pruned parameters of tasks within a group into a group-specific model.

Specifically, for the tasks in the k-th group G_k, we prune parameters in their respective networks and subsequently unionize the survived task-specific parameter sets into the parameters of the group-specific network. For pruning the t-th task, we prune parameters in element-wise [31], with a difficulty-aware ratio of $1/\exp(d^t)$. This shows that the higher the difficulty, the lower the pruning ratio. Finally, the parameters for the network of the k-th group, w^{G_k}, are formed by merging the survived task-specific parameter sets belonging to the k-th group. Note that when the positions of these surviving parameters overlap, we prioritize parameters from tasks with a higher difficulty.

E. TOTAL LOSS

The proposed method trains multiple tasks simultaneously by summing the losses for all tasks. The total loss $\mathcal{L}_{total}$ comprises the sum of cross-entropy losses for group-specific models and a regularization term for the weights. We define $\mathcal{L}_{total}$ as

$$\mathcal{L}_{total} = \sum_{k=1}^{g}\mathcal{L}_{ce}^{G_k} + \|w^{G_k}\|_2, \tag{5}$$

where G_k is the k-th group, g is the total number of groups, and w^{G_k} is the set of weights for group G_k. DMO aims to improve memory efficiency by obtaining group-wise subnetworks that measure task difficulty and reduce unnecessary network parameters accordingly.

IV. EXPERIMENTS

A. SETUP

To demonstrate the effectiveness of the proposed method, we compared our method and other competitors to various

TABLE 1. Performance of various methods using ResNet-18 on four different tasks. Avg, #P, and Ratio denote the average accuracy, the number of model parameters expressed in millions, and the relative ratio of parameters compared with single-task learning.

	CIFAR-10	STL-10	USPS	MNIST	Avg	#P (Ratio)
Single-task learning	92.10	71.87	96.51	99.32	89.95	44.8 (1.00)
Hard parameter sharing [1]	79.72	70.82	51.66	98.03	75.06	11.2 (0.25)
MTAN [4]	81.92	72.21	95.06	94.53	85.93	19.1 (0.42)
Soft parameter sharing [7]	92.30	71.83	96.86	99.34	90.08	44.8 (1.00)
Soft RSA [32]	94.72	60.84	32.2	92.68	70.11	16.7 (0.37)
Cross-stitch [8]	89.79	68.60	96.71	99.22	88.58	56.0 (1.25)
NDDR [10]	88.53	69.11	96.36	99.27	88.32	54.5 (1.21)
TAPS [6]	79.49	60.30	94.72	98.99	83.37	25.0 (0.55)
DMO (ours)	91.50	71.56	96.56	99.31	89.73	6.8 (0.15)

MTL scenarios. For the first scenario of multi-task learning, we experimented with four classification datasets. We define each task as a dataset. We uniformly resized all images from the tasks CIFAR-10, STL-10, MNIST, and USPS into 72 × 72 pixels following [17]. CIFAR-10 and STL-10 have 10 classes with colored images, while MNIST and USPS have grayscale handwritten digit images. For the MTL scenario where we need to train with both color and grayscale images simultaneously, we converted the grayscale images into a 3-channel format. In the second scenario, we used the CIFAR-100 dataset [13], which consists of 50,000 training and 10,000 test color images of 32 × 32 pixels with 100 classes. To evaluate the effectiveness of the method in an MTL scenario with more tasks, we divided CIFAR-100 into 20 tasks based on the superclasses provided in the dataset. Here, a task corresponds to a five-class (superclass) classification task.

We also conducted another experiment using 10 datasets for the large-scale visual recognition challenge known as Visual Decathlon Challenge [17]. This challenge aims to address 10 image classification tasks simultaneously; each task belongs to a different visual domain. The datasets used in this challenge are FGVC-Aircraft Benchmark (Aircraft) [33], CIFAR-100 [13], Daimler Pedestrian classification (DPed) [34], Describable Textures Dataset (DTD) [35], German Traffic Sign Recognition Benchmark (GTSRB) [36], ILSVRC12 (ImageNet) [37], Omniglot [38], Street View House Numbers (SVHN) [39], UCF101 Dynamic Images (UCF101) [40], [41] and VGG-Flowers (Flowers102) [42].

We evaluated the proposed method and various MTL approaches, including single-task learning, hard parameter sharing [1], soft parameter sharing [7], and soft-parameter sharing method that groups tasks based on RSA [32] (we denote it as Soft RSA). We report on the accuracy of individual tasks and the average accuracy (Avg) across multiple tasks. Additionally, we present the relative ratio of parameter usage in comparison to single-task learning. Single-task learning involves separate learning for each task. We also compared our method to other recent MTL models, including cross-stitch [8], MTAN [4], NDDR [10], and TAPS [6]. We utilized the ResNet-18 network [43] as the backbone for the first two scenarios and an ImageNet pre-trained ResNet-18 for the last scenario. We used the SGD optimizer and set the momentum of 0.9. In the first scenario, we trained for 25 epochs with a batch size of 64 and an initial learning rate of 0.1. We trained for 200 epochs with the same parameters in the second scenario. In the third scenario, we trained for 25 epochs with a batch size of 128 and an initial learning rate of 0.01. We implemented using PyTorch [44].

TABLE 2. Performance of DMO and the other compared methods on 20 tasks from CIFAR-100. The table reports average accuracy (Avg), the number of parameters expressed in millions (#P), and the parameter ratio compared to the usage in single-task learning (Ratio).

Methods	Avg	#P (Ratio)
Single-task learning	80.47	223.5 (1.00)
Hard parameter sharing [1]	75.48	11.1 (0.04)
MTAN [4]	82.88	35.9 (0.16)
Soft parameter sharing [7]	82.16	223.5 (1.00)
Soft RSA [32]	77.19	111.8 (0.50)
Cross-stitch [8]	78.00	447.1 (2.00)
NDDR [10]	68.23	691.3 (3.09)
TAPS [6]	69.84	130.5 (0.58)
DMO (ours)	83.10	33.5 (0.14)

B. FOUR TASKS WITH DIFFERENT DIFFICULTY LEVELS

We compared the proposed method with existing approaches using four classification tasks, namely CIFAR-10, STL-10, USPS, and MNIST, whose results are shown in Table 1. Hard parameter sharing results in unsatisfactory performance compared to single-task learning due to task interference. Soft parameter sharing shows negligible improvement in the trade-off between average accuracy and parameter consumption compared to single-task learning. The proposed method, DMO, not only significantly reduces the number of parameters for performing each task compared to soft parameter sharing but also achieves performance similar to

TABLE 3. Performance on Visual Decathlon Challenge. The table provides details on the average accuracy (Avg) across all tasks, the number of parameters (#P) expressed in millions, and the relative ratio of parameters (Ratio) compared to the parameters of single-task learning.

Methods	Airc.	C100	Dped	DTD	GTSR	Flwr	Oglt	SVHN	UCF	Avg	#P (Ratio)
Single-task learning	33.69	68.57	99.76	42.23	99.80	70.19	86.21	94.28	77.46	74.69	101.68 (1.00)
Hard parameter sharing [1]	23.30	63.10	80.30	45.40	68.20	73.70	58.50	43.50	26.80	54.31	11.29 (0.11)
MTAN [4]	61.81	81.59	91.63	56.44	98.80	81.04	89.83	96.88	50.63	77.25	19.64 (0.19)
Soft parameter sharing [7]	28.47	65.90	99.57	38.83	99.40	61.47	80.83	92.90	67.29	65.24	101.68 (1.00)
Soft RSA [32]	70.65	42.30	87.40	79.13	64.09	81.60	17.83	67.78	62.50	63.70	45.78 (0.45)
Cross-stitch [8]	36.18	75.36	99.83	40.37	99.89	75.39	85.93	94.76	79.35	76.33	203.36 (2.00)
NDDR [10]	30.96	27.74	94.62	24.41	99.29	50.58	56.96	92.71	60.53	59.76	252.83 (2.48)
TAPS [6]	16.35	47.53	97.82	17.77	99.40	34.50	70.18	90.76	39.65	59.11	65.32 (0.64)
DMO (ours)	38.73	62.73	92.72	45.34	96.01	86.12	84.37	90.53	76.17	74.40	11.48 (0.11)

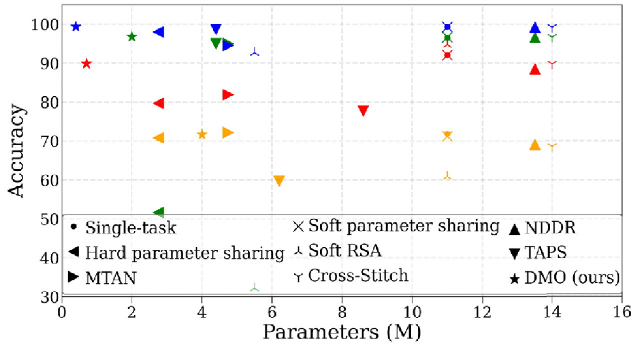

FIGURE 3. Performance of DMO in comparison to various MTL methods. We use multiple colors to represent different tasks: red for CIFAR-10, yellow for STL-10, green for MNIST, and blue for USPS. The x-axis indicates the number of parameters.

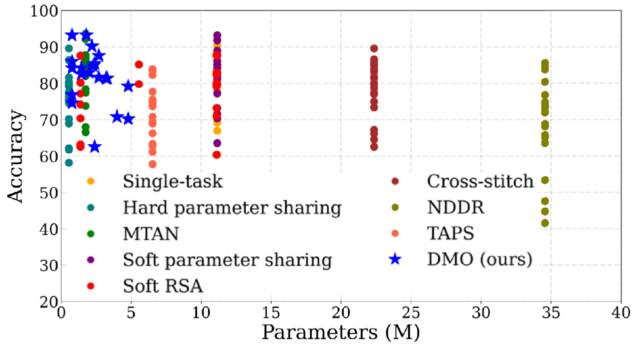

FIGURE 4. Performance of DMO and the other compared methods on 20 tasks from CIFAR-100. Identical symbols represent results with respect to accuracy and parameters for a task.

the hard parameter sharing approach. Specifically, DMO outperforms NDDR and simultaneously shows a substantial decrease of approximately 95% in the total number of parameters. This is achieved by grouping similar tasks, such as USPS and MNIST, and allocating parameters based on the task difficulty.

Figure 3 shows the accuracy and parameter consumption for each task. The proposed method consumes 94% fewer parameters than single-task learning, with only a marginal performance drop. For STL-10, MNIST, and USPS, DMO significantly reduces parameters while maintaining similar performance. On average, DMO results in an 84% parameter reduction across various tasks. Notably, it achieves a significant 97% decrease in parameters on the USPS dataset.

C. CIFAR-100: 20 TASKS

In Table 2, we evaluate the performance of various multi-task learning approaches on 20 tasks derived from the CIFAR-100 dataset. Single-task learning achieves an average accuracy of 80.47% by using 223.5M parameters. While hard parameter sharing [1] effectively reduces the parameter count to 11.1M, it comes at the cost of a noticeable decline in average accuracy, dropping it to 75.48%. MTAN [4] enhances an average accuracy of 82.88% by using 35.9M parameters. Conversely, soft parameter sharing [7] and Cross-stitch [8] give an average accuracy of 82.16% and 78.00%, respectively, employing 223.5 and 447.1M parameters. NDDR [10], with its substantial parameter count of 691.3M, only achieves an accuracy of 68.23%. This has approximately 20 times more parameters than the proposed method. TAPS [6] demonstrates a higher average accuracy than NDDR while using 130.5M parameters. However, it consumes about 3.9 times greater parameters than the proposed DMO and shows unsatisfactory performance compared to ours. Remarkably, DMO outperforms other existing methods concerning the trade-off between performance and parameter. It achieves an average accuracy of 83.10% while utilizing a significantly lower total number of parameters at 33.5M.

Figure 4 shows the number of parameters and the accuracy for each task. We demonstrate our method's accuracy and number of parameters compared to other MTL approaches. The figure shows that DMO performs better generalization across tasks than hard parameter sharing while maintaining parameter counts. DMO also achieves comparable performance to other methods that utilize significantly larger parameters.

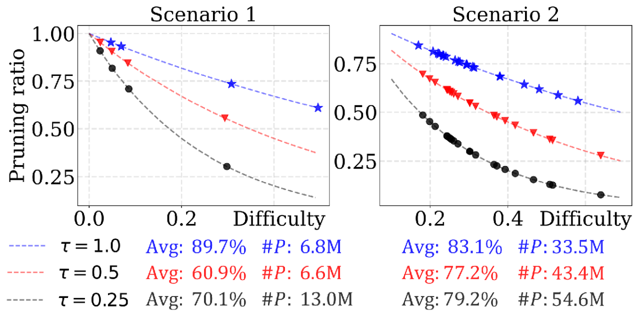

FIGURE 5. Correlation between task difficulty and pruning ratio for the scenarios 1 and 2 with respect to different values of τ. We report the average accuracy (Avg) and the parameters (#P). We denote different line colors for different values of τ. The symbols on each line represent the pruning ratio corresponding to the difficulty of a task.

TABLE 4. Performance according to the number of filters of DMO and the other compared methods on the first and second scenarios. The table reports the number of filters used in the first layer (#filters), the number of layers (#layers), and average accuracy (Avg), the number of parameters expressed in millions (#P).

Methods	#filters	Scenario 1		Scenario 2	
		Avg	#P	Avg	#P
Single-task learning	16	87.42	2.8	77.72	14.0
	32	87.58	11.2	80.05	56.0
	64	89.95	44.8	80.47	223.5
Soft parameter sharing [7]	16	89.46	2.8	78.55	14.0
	32	90.29	11.2	80.02	56.0
	64	90.08	44.8	82.16	223.5
DMO (ours)	16	86.58	0.2	74.37	0.8
	32	88.87	1.0	78.16	3.4
	64	89.73	6.8	83.10	33.5

Methods	#layers	Scenario 1		Scenario 2	
		Avg	#P	Avg	#P
Single-task learning	10	87.07	19.6	80.15	98.1
	14	87.65	21.1	80.17	105.5
	18	89.95	44.8	80.47	223.5
Soft parameter sharing [7]	10	90.23	19.6	80.13	98.1
	14	90.44	21.1	80.19	105.5
	18	90.08	44.8	82.16	223.5
DMO (ours)	10	85.67	1.3	79.48	7.7
	14	86.76	4.4	80.19	8.1
	18	89.73	6.8	83.10	33.5

D. VISUAL DECATHLON CHALLENGE

In Table 3, we compare the performance of various MTL methods on Visual Decathlon Challenge. Note that in single-task learning, the average accuracy is 74.69%, with a baseline parameter count of 101.68M. This method achieves high accuracy because it focuses on individual tasks. Soft parameter sharing [7] attains a similar accuracy of 65.24% with a similar parameter count to single-task learning. This method shares parameters among tasks, enabling better generalization. Cross-stitch [8] surpasses the soft parameter sharing approach with an average accuracy of 76.33%, but it increases the parameter count to 203.36M. It creates separate models for tasks and learns task-specific information but requires more parameters due to added complexity. NDDR [10] shows a slightly lower average accuracy of 59.76% and an increased parameter count of 252.86M compared to the cross-stitch method. This can be attributed to its design focus on tasks related to semantic segmentation and surface normal prediction. TAPS [6] is memory-efficient but shows an unsatisfying average accuracy of 59.11% compared to other MTL approaches. Compared to existing approaches, DMO demonstrates an average accuracy of 74.40% while using a significantly lower parameter count of 11.48 M. This memory-efficient method offers the benefits of parameter sharing by task grouping and parameter allocation within each group. The performance of the proposed method is slightly lower compared to the hard-parameter sharing method, MTAN, due to our focus on memory efficiency, which led us to use fewer parameters. However, when considering the trade-off between parameter usage and accuracy, our proposed method employs 58.45% fewer parameters than MTAN, with a marginal decrease in average accuracy.

E. ANALYSIS

1) CORRELATION BETWEEN TASK DIFFICULTY AND PRUNING RATIO

We analyzed the correlation to explore the relationship between task difficulty and the pruning ratio. To compare the performance and parameter usage associated with our proposed difficulty-aware pruning, we introduced a temperature hyperparameter τ. We set the difficulty-aware pruning ratio as $1/\exp(\frac{d^t}{\tau})$ and conducted experiments with varying values of τ. Figure 5 presents the difficulty and pruning ratio for each task in experiments conducted in the scenarios 1 and 2. This demonstrates that easier tasks are assigned higher pruning ratios, while more difficult ones have lower ratios, aligning the pruning ratio with task difficulty. Even at the same difficulty level, our default setting with $\tau = 1.0$ achieves a higher average accuracy despite a higher pruning ratio compared to other values of τ. This suggests that our method effectively eliminates unnecessary parameters and facilitates information sharing among grouped tasks.

2) IMPACT OF NETWORK HYPERPARAMETERS

To analyze the memory-accuracy trade-off of our proposed method for various network sizes, we experimented with the scenarios 1 and 2 by varying the sizes of the network's filters and the number of layers. Table 4 shows the average accuracy (Avg) and parameter usage (#P) for different network configurations. The proposed method demonstrates a higher memory-accuracy trade-off compared to other methods, even when adjusting the network's filters and layers.

V. CONCLUSION

In this paper, we have proposed Dynamic Model Optimization (DMO), a novel approach to multi-task learning that optimizes tasks according to their complexity. DMO produces memory-efficient subnetworks by grouping tasks based on similarity and allocating network parameters based on task difficulty. The experiments demonstrated that DMO effectively diminishes the number of model parameters by more than 80% while exhibiting similar performance compared to the soft parameter-sharing approach. Furthermore, DMO uses significantly fewer parameters than recent MTL methods while maintaining similar accuracy or improving upon it. Thus, DMO can be a strong candidate as a practical approach for designing efficient MTL networks. In this study, we acknowledge that the graph-based task grouping method has limitations, as the multitude of potential combinations may not always yield optimal results. Future work will focus on enhancing this approach and the efficiency of resource allocation by formulating more sophisticated algorithms and incorporating a broader range of influential factors beyond task difficulty.

ACKNOWLEDGMENT

(Sujin Choi and Hyundong Jin contributed equally to this work.)

REFERENCES

[1] R. Caruana, "Multitask learning," *Mach. Learn.*, vol. 28, no. 1, pp. 41–75, 1997.

[2] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "A survey of model compression and acceleration for deep neural networks," 2017, *arXiv:1710.09282*.

[3] I. Kokkinos, "UberNet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5454–5463.

[4] 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.

[5] X. Sun, A. Hassani, Z. Wang, G. Huang, and H. Shi, "DiSparse: Disentangled sparsification for multitask model compression," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 12372–12382.

[6] M. Wallingford, H. Li, A. Achille, A. Ravichandran, C. Fowlkes, R. Bhotika, and S. Soatto, "Task adaptive parameter sharing for multi-task learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 7551–7560.

[7] L. Duong, T. Cohn, S. Bird, and P. Cook, "Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser," in *Proc. Annu. Meeting Assoc. Comput. Linguistics*, 2015, pp. 845–850.

[8] 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.

[9] 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.

[10] Y. Gao, J. Ma, M. Zhao, W. Liu, and A. L. Yuille, "NDDR-CNN: Layerwise feature fusing in multi-task CNNs by neural discriminative dimensionality reduction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 3200–3209.

[11] S. Ruder, J. Bingel, I. Augenstein, and A. Søgaard, "Latent multi-task architecture learning," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 4822–4829.

[12] 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. 6528–6537.

[13] A. Krizhevsky, "Learning multiple layers of features from tiny images," Univ. Toronto, Canada, Tech. Rep., 2009.

[14] D. Wang and X. Tan, "Unsupervised feature learning with C-SVDDNet," *Pattern Recognit.*, vol. 60, pp. 473–485, Dec. 2016.

[15] L. Deng, "The MNIST database of handwritten digit images for machine learning research [best of the web]," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 141–142, Nov. 2012.

[16] J. J. Hull, "A database for handwritten text recognition research," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, no. 5, pp. 550–554, May 1994.

[17] S.-A. Rebuffi, H. Bilen, and A. Vedaldi, "Learning multiple visual domains with residual adapters," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017.

[18] Z. Chen, V. Badrinarayanan, C.-Y. Lee, and A. Rabinovich, "GradNorm: Gradient normalization for adaptive loss balancing in deep multitask networks," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 794–803.

[19] 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. (ECCV)*, 2018, pp. 270–287.

[20] R. Cipolla, Y. Gal, and A. Kendall, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 7482–7491.

[21] S. Liang, C. Deng, and Y. Zhang, "A simple approach to balance task loss in multi-task learning," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2021, pp. 812–823.

[22] J. Lee, L. Mukhanov, A. S. Molahosseini, U. Minhas, Y. Hua, J. M. del Rincon, K. Dichev, C.-H. Hong, and H. Vandierendonck, "Resource-efficient deep learning: A survey on model-, arithmetic-, and implementation-level techniques," 2021, *arXiv:2112.15131*.

[23] Z. Xing, S. Zhao, W. Guo, F. Meng, X. Guo, S. Wang, and H. He, "Coal resources under carbon peak: Segmentation of massive laser point clouds for coal mining in underground dusty environments using integrated graph deep learning model," *Energy*, vol. 285, Dec. 2023, Art. no. 128771.

[24] T. Sun, Y. Shao, X. Li, P. Liu, H. Yan, X. Qiu, and X. Huang, "Learning sparse sharing architectures for multiple tasks," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, 2020, pp. 8936–8943.

[25] C. Fifty, E. Amid, Z. Zhao, T. Yu, R. Anil, and C. Finn, "Efficiently identifying task groupings for multi-task learning," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 34, 2021, pp. 27503–27516.

[26] M. Raghu, J. Gilmer, J. Yosinski, and J. Sohl-Dickstein, "SVCCA: Singular vector canonical correlation analysis for deep learning dynamics and interpretability," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017.

[27] S. Kornblith, M. Norouzi, H. Lee, and G. Hinton, "Similarity of neural network representations revisited," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 3519–3529.

[28] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, *arXiv:1609.02907*.

[29] H. C. Johnston, "Cliques of a graph-variations on the Bron–Kerbosch algorithm," *Int. J. Comput. Inf. Sci.*, vol. 5, no. 3, pp. 209–238, Sep. 1976.

[30] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 28, 2015.

[31] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," 2018, *arXiv:1803.03635*.

[32] K. Dwivedi and G. Roig, "Representation similarity analysis for efficient task taxonomy & transfer learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 12379–12388.

[33] S. Maji, E. Rahtu, J. Kannala, M. Blaschko, and A. Vedaldi, "Fine-grained visual classification of aircraft," 2013, *arXiv:1306.5151*.

[34] S. Munder and D. M. Gavrila, "An experimental study on pedestrian classification," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 11, pp. 1863–1868, Nov. 2006.

[35] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi, "Describing textures in the wild," in *Proc. Comput. Vis. Pattern Recognit. (CVPR)*, 2014.

[36] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition," *Neural Netw.*, vol. 32, pp. 323–332, Aug. 2012.

[37] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.

[38] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum, "Human-level concept learning through probabilistic program induction," *Science*, vol. 350, no. 6266, pp. 1332–1338, Dec. 2015.

[39] Y. Netzer, "Reading digits in natural images with unsupervised feature learning," in *Proc. NIPS Workshop Deep Learn. Unsupervised Feature Learn.*, 2011.

[40] K. Soomro, A. R. Zamir, and M. Shah, "UCF101: A dataset of 101 human actions classes from videos in the wild," 2012, *arXiv:1212.0402*.

[41] H. Bilen, B. Fernando, E. Gavves, A. Vedaldi, and S. Gould, "Dynamic image networks for action recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 3034–3042.

[42] M.-E. Nilsback and A. Zisserman, "Automated flower classification over a large number of classes," in *Proc. 6th Indian Conf. Comput. Vis., Graph. Image Process.*, Dec. 2008, pp. 722–729.

[43] 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.

[44] A. Paszke et al., "Pytorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019.

SUJIN CHOI received the B.S. degree in mathematics and computer software from Kwangwoon University, Seoul, South Korea, in 2021, and the M.S. degree from the Department of AI, Chung-Ang University, Seoul, in 2023. Her research interests include deep learning, machine learning, multi-task learning, and computer vision.

HYUNDONG JIN received the B.S. degree in electrical and electronics engineering from Chung-Ang University, Seoul, South Korea, in 2020, and the M.S. degree from the School of Computer Science and Engineering, Chung-Ang University, in 2022, where he is currently pursuing the Ph.D. degree. His research interests include deep learning, machine learning, continual learning, and computer vision.

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. 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. He is currently an Associate Professor with the School of Computer Science and Engineering, Chung-Ang University. His research interests include machine learning, deep learning, model optimization, robotics, and computer vision.