Dynamic Neural Network for Multi-Task Learning Searching across Diverse Network Topologies

Wonhyeok Choi, Sunghoon Im*
Department of Electrical Engineering & Computer Science, DGIST, Daegu, Korea
{smu06117, sunghoonim}@dgist.ac.kr

Abstract

In this paper, we present a new MTL framework that searches for structures optimized for multiple tasks with diverse graph topologies and shares features among tasks. We design a restricted DAG-based central network with read-in/read-out layers to build topologically diverse task-adaptive structures while limiting search space and time. We search for a single optimized network that serves as multiple task adaptive sub-networks using our three-stage training process. To make the network compact and discretized, we propose a flow-based reduction algorithm and a squeeze loss used in the training process. We evaluate our optimized network on various public MTL datasets and show ours achieves state-of-the-art performance. An extensive ablation study experimentally validates the effectiveness of the sub-module and schemes in our framework.

1. Introduction

Multi-task learning (MTL), which learns multiple tasks simultaneously with a single model has gained increasing attention [3, 13, 14]. MTL improves the generalization performance of tasks while limiting the total number of network parameters to a lower level by sharing representations across tasks. However, as the number of tasks increases, it becomes more difficult for the model to learn the shared representations, and improper sharing between less related tasks causes negative transfers that sacrifice the performance of multiple tasks [15,36]. To mitigate the negative transfer in MTL, some works [6, 25, 32] separate the shared and task-specific parameters on the network.

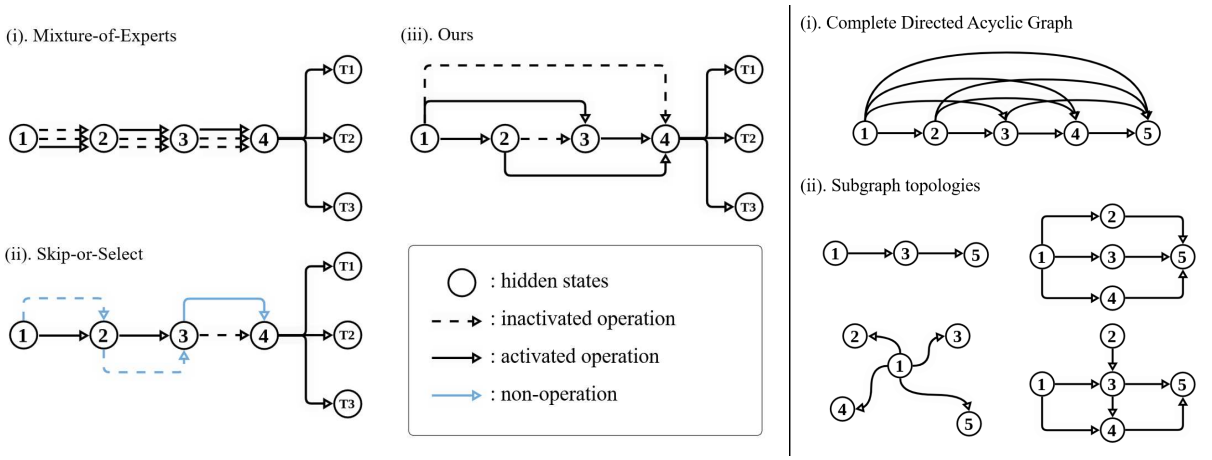
More recent works [21,29,38] have been proposed to dynamically control the ratio of shared parameters across tasks using a Dynamic Neural Network (DNN) to construct a task adaptive network. These works mainly apply the cell-based architecture search [19, 27, 41] for fast search times, so that the optimized sub-networks of each task consist of fixed or simple structures whose layers are simply branched, as shown in Fig. 1a. They primarily focus on finding branching patterns in specific aspects of the architecture, and feature-sharing ratios across tasks. However, exploring optimized structures in restricted network topologies has the potential to cause performance degradation in heterogeneous MTL scenarios due to unbalanced task complexity.

We present a new MTL framework searching for sub-network structures, optimized for each task across diverse network topologies in a single network. To search the graph topologies from richer search space, we apply Directed Acyclic Graph (DAG) for the homo/heterogeneous MTL frameworks, inspired by the work in NAS [19, 27, 40]. The MTL in the DAG search space causes a scalability issue, where the number of parameters and search time increase quadratically as the number of hidden states increases.

To solve this problem, we design a restricted DAG-based central network with read-in/read-out layers that allow our MTL framework to search across diverse graph topologies while limiting the search space and search time. Our flow-restriction eliminates the low-importance long skip connection among network structures for each task, and creates the required number of parameters from $O(N^2)$ to $O(N)$. The read-in layer is the layer that directly connects all the hidden states from the input state, and the read-out layer is the layer that connects all the hidden states to the last feature layer. These are key to having various network topological representations, such as polytree structures, with early-exiting and multi-embedding.

Then, we optimize the central network to have compact task-adaptive sub-networks using a three-stage training procedure. To accomplish this, we propose a squeeze loss and a flow-based reduction algorithm. The squeeze loss limits the upper bound on the number of parameters. The reduction algorithm prunes the network based on the weighted adjacency matrix measured by the amount of information flow in each layer. In the end, our MTL framework constructs a compact single network that serves as multiple task-specific networks with unique structures, such as chain, polytree, and parallel diverse topologies, as presented in Fig. 1b. It also dynamically controls the amount of sharing representation among tasks.

*Corresponding author



(a) Graph representation of existing DNN-based methods (i).-(ii). for MTL and ours (iii). (b) Topologies of DAG (i). and its sub-graph (ii).

Figure 1. **Graph representation of various neural networks.** (a) Graph representation of existing dynamic neural network for multitask learning and ours. (b) Topologies of a completed Directed Acyclic Graph (DAG) and the output sub-graph of DAG structure.

The experiments demonstrate that our framework successfully searches the task-adaptive network topologies of each task and leverages the knowledge among tasks to make a generalized feature. The proposed method outperforms state-of-the-art methods on all common benchmark datasets for MTL. Our contributions can be summarized as follows:

- We present for the first time an MTL framework that searches both task-adaptive structures and sharing patterns among tasks. It achieves state-of-the-art performance on all public MTL datasets.
- We propose a new DAG-based central network composed of a flow restriction scheme and read-in/out layers, that has diverse graph topologies in a reasonably restricted search space.
- We introduce a new training procedure that optimizes the MTL framework for compactly constructing various task-specific sub-networks in a single network.

2. Related Works

Neural Architecture Search (NAS) Neural Architecture Search is a method that automates neural architecture engineering [8]. Early works [2, 40, 41] use reinforcement learning based on rewarding the model accuracy of the generated architecture. Alternative approaches [24, 30, 37] employ evolutionary algorithms to optimize both the neural architecture and its weights. These methods search for an adequate neural architecture in a large discrete space. Gradient-based NAS methods [4, 19, 39] of formulating operations in a differentiable search space are proposed to alleviate the scalability issues. They generally use the convex combination from a set of operations instead of determining a single operation. Most NAS approaches [19, 27, 40] adopt the complete DAG as a search space, to find the architecture in the various network topologies. However, DAG-based MTL frameworks have not been proposed, because of their considerably high computational demands.

Multi-Task Learning (MTL) Multi-task learning in deep neural networks can be categorized into hard and soft parameter sharing types [31]. Hard parameter sharing [3, 13, 14], also known as shared-bottom, is the most commonly used approach to MTL. This scheme improves generalization performance while reducing the computational cost of the network, by using shared hidden layers between all tasks. However, it typically struggles with the negative transfer problem [15, 36] which degrades performance due to improper sharing between less relevant tasks.

On the other hand, soft-parameter sharing [25, 32] alleviate the negative transfer problem by changing the shared parameter ratio. These approaches mitigate the negative transfer by flexibly modifying shared information, but they cannot maintain the computational advantage on the classic shared-bottom model. Recently, advanced approaches have been proposed to adjust shared parameters using a dynamic neural network [21, 22, 29, 38] and NAS [10].

NAS-style MTL MTL frameworks using a dynamic neural network (DNN) can be divided into two categories. One employs the Mixture-of-Experts (MoE) [33], which is designed for conditional computation of per-sample, to MTL by determining the experts of each task [9, 21, 22]. They have a fixed depth finalized task-specific sub-network, because they choose experts from a fixed number of modular layers. This causes a critical issue with task-balancing in the heterogeneous MTL. The other adopts the skip-or-select policy to select task-specific blocks from the set of residual blocks [38] or a shared block per layer [12, 29]. These methods only create a simple serial path in the finalized sub-network of each task, and a parallel link cannot be reproduced. Moreover, they heuristically address the unbalanced

task-wise complexity issues in the heterogenous MTL (*e.g.* manually changing the balancing parameters based on the task complexity [29,38]). Thus, none of the aforementioned works focus on finding the optimal task-specific structure in the MTL scenario.

3. Method

We describe our MTL framework, which searches for optimized network structures tailored to each task across diverse graph topologies, while limiting search time. Sec. 3.1 describes the composition of the searchable space of our central network and our flow-restriction method for efficiently balancing the topological diversity of task-specific sub-networks and searching space. Sec. 3.2 introduces our mechanism to determine the task-adaptive sub-network in the central network and describes the overall training process and loss function. The overall pipeline of our method is illustrated in Fig. 2.

3.1. The Central Network with Diverse Topologies

Our central network composes a graph $G = (V, E)$ with layers E in which the N hidden states $V = \{v_1, ..., v_N\}$ are topologically sorted:

$$E = \{e_{ij}\}_{i,j\in\{1,...,N\}}, \text{ where } i < j, \tag{1}$$

$$e_{ij}(\,\cdot\,;\theta_{ij}) : \mathbb{R}^{N^{v_i}} \rightarrow \mathbb{R}^{N^{v_j}}, \tag{2}$$

where e_{ij} is the operation that transfer the state v_i to v_j with the weight parameters $\theta_{ij} \in \Theta$, and N^{v_k} is the number of the elements of hidden state v_k, respectively. We adopt the DAG structure [19, 27, 40] for the network. However, the optimized structure from DAG is searched from $2^{N(N-1)/2}$ network topologies, which are too large to be optimized in time. To address the issue while maintaining diversity, we propose a flow-restriction and read-in/read-out layers.

Flow-restriction The flow-restriction eliminates the low-importance long skip connection among network structures for each task by restricting $j - i \leq M$ where M is the flow constant. Regulating the searchable edges in the graph reduces the required number of parameters and searching time from $O(N^2)$ to $O(N)$, but it sacrifices the diversity and capacity of the network topologies.

To explain the topological diversity and task capacity of sub-networks, we define the three components of network topology, as follows:

1. $\mathcal{D}(G) = \max(\{\text{Distance}(v_i, v_j)\}_{v_i,v_j\in V})$,
2. $\mathcal{W}(G) = \max(\{\text{Out}_{v_i}\}_{v_i\in V})$,
3. $\mathcal{S}(G_s, G) = |E_s|/|E|$,

where Out_{v_i} is the out-degree of the vertex v_i and $\text{Distance}(\cdot)$ is the operation that counts the number of layers (or edges) between two connected vertices. The network depth $\mathcal{D}(G)$ is equal to the longest distance between two vertices in the graph G. The network width $\mathcal{W}(G)$ is equal to the maximum value of the out-degrees of hidden states in the graph G. The sparsity $\mathcal{S}(G_s, G)$ of the sub-graph G_s is the ratio of finalized edges $|E_s|$ over entire edges $|E|$. The first two components are measurements of the topological diversity of the finalized sub-network, while the last one is for the sub-network capacity. While a complete DAG has the full range of depth and width components, the flow-restricted DAG has the properties of depth and width components as follows:

Property 1. $\min(\{\mathcal{D}(G_s)\}_{G_s\subseteq G}) = \lceil(|V|/M)\rceil$,

Property 2. $\max(\{\mathcal{W}(G_s)\}_{G_s\subseteq G}) = M$,

where $\{G_s\}$ is the entire sub-graph of G. The min-depth property (Prop. 1) can cause the over-parameterized problem when the capacity of the task is extremely low. The max-width property (Prop. 2) directly sacrifices the diversity of network topologies in the search space.

Read-in/Read-out layers We design read-in/read-out layers to mitigate these problems. The read-in layer embeds the input state v_0 into all hidden states $v_i \in V$ with task-specific weights $\alpha_i^k \in \mathcal{A}$ for all K tasks $\mathcal{T} = \{T_k\}_{k\in\{1,...,K\}}$ as follows:

$$v_i^k = \sigma(\alpha_i^k) \cdot v_0, \tag{3}$$

where $\sigma(\cdot)$ is the sigmoid function. Then, the central network sequentially updates the hidden state v_1^k to v_N^k with the task-specific weights $\gamma_{ij}^k \in \Gamma$ that correspond to e_{ij}^k:

$$v_j^k = \frac{1}{\text{In}_{v_j^k}} \sum_{e_{ij}\in E} (\sigma(\gamma_{ij}^k) \cdot e_{ij}(v_i^k; \theta_{ij})), \tag{4}$$

where $\text{In}_{v_j^k}$ is the in-degree of v_j^k. Note that Γ is the adjacency matrix of graph G. Finally, the read-out layer aggregates all hidden state features $\{v_i^k\}_{i\in\{1,...,N\}}$ with the task-specific weights $\beta_i^k \in \mathcal{B}$ and produces the last layer feature v_L^k for each task k as follows:

$$v_L^k = \sum_{i\in\{1,...,N\}} (\sigma(\beta_i^k) \cdot v_i^k). \tag{5}$$

The final prediction $\hat{\mathbf{y}}_k$ for each task T_k is computed by passing the aggregated features v_L^k through the task-specific head $H^k(\cdot)$ as follows:

$$\hat{\mathbf{y}}^k = H^k(v_L^k). \tag{6}$$

All upper-level parameters $\mathcal{A}$, $\mathcal{B}$, and Γ are learnable parameters, and their learning process is described in Sec. 3.2. The read-in/read-out layers enable the optimized network to have a multi-input/output sub-network. The read-out layer

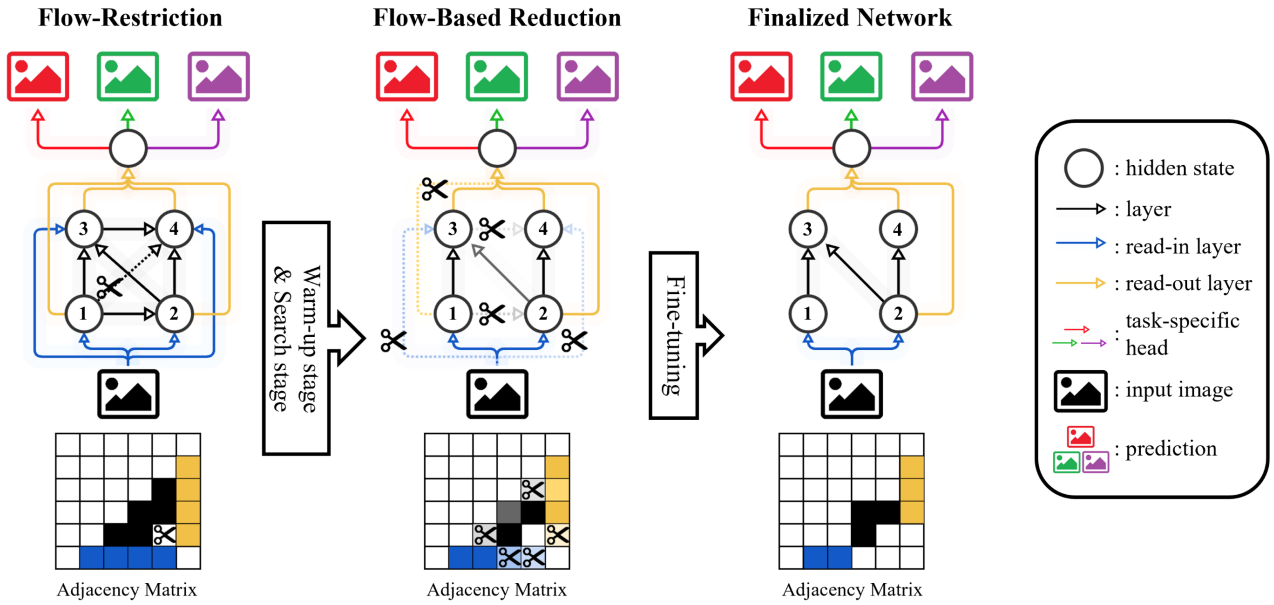


Figure 2. **Overall pipeline.** Our central network follows a DAG-based structure with read-in/out layers, and task-specific heads. The long skip connection is cut by our flow-restriction. Our framework with a 3-task MTL learning scenario consists of three stages including warm-up, search, and fine-tuning stages. The warm-up stage only learns the parameters of the main network Θ and task-specific weights. The search stage learns the upper-level parameters $\mathcal{A}, \mathcal{B}, \Gamma$, and task-specific weights. Then, flow-based reduction eliminates the low-importance edges from the network. The fine-tuning stage re-trains the network with the remaining important parameters.

aggregates all hidden states of the central network during the search stage, allowing a specific task to use the early hidden states to output predictions while ignoring the last few layers. These early-exit structures help alleviate the over-parameterized problem in simple tasks.

3.2. Network Optimization and Training Procedure

We describe the entire training process for our MTL framework, which consists of three stages, including warm-up, search, and fine-tuning stages.

Warm-up stage As with other gradient-based NAS, our framework has upper-level parameters that determine the network structure and parameters. This bilevel optimization with a complex objective function in an MTL setup makes the training process unstable. For better convergence, we initially train all network parameters across tasks for a few iterations. We train the weight parameters of the central network Θ that shares all operations E across tasks. We fix all values of the upper-level parameters $\mathcal{A}, \mathcal{B}$, and Γ as 0, which becomes 0.5 after the sigmoid function $\sigma(\cdot)$, and freeze them. We train the network parameters Θ in Eq. 4 with a task loss as follows:

$$\mathcal{L}_{task} = \sum_{k=0}^{K} \mathcal{L}_{T_k}(\hat{\mathbf{y}}_{T_k}, \mathbf{y}_{T_k}), \tag{7}$$

where $\mathcal{L}_{T_k}$ is the task-specific loss, which is the unique loss function for each task.

Search stage After the warm-up stage, we unfreeze the upper-level parameters $\mathcal{A}, \mathcal{B}$, and Γ and search the network topologies appropriate to each task. We train all these parameters and network parameters Θ simultaneously by minimizing the task loss and the proposed squeeze loss $\mathcal{L}_{sq}$ as follows:

$$\mathcal{L}_{train} = \mathcal{L}_{task} + \lambda_{sq}\mathcal{L}_{sq}, \tag{8}$$

$$\mathcal{L}_{sq} = \sum_{k=0}^{K}(\max((\sum_{\gamma_{ij}\in\Gamma}(\sigma(\gamma_{ij})) - \kappa), 0)), \tag{9}$$

where λ_{sq} is the balancing hyperparameter, and κ is a constant number called the budget, that directly reduces the sparsity of the central network. This auxiliary loss is designed to encourage the model to save computational resources.

Fine-tuning stage Lastly, we perform a fine-tuning stage to construct a compact and discretized network structure using the trained upper-level parameters $\mathcal{A}, \mathcal{B}$, and Γ. To do so, we design a flow-based reduction algorithm that allows the network to obtain high computational speed by omitting low-importance operations, as described in Alg. 1. It measures the amount of information flow of each layer e_{ij} in the central network by calculating the ratio of edge weight with respect to other related edges weight. Then, it sequentially removes the edge which has the lowest information flow. Alg. 1 stops when the edge selected to be deleted is

Algorithm 1: Flow-based Reduction

Input: $\Gamma \in \mathbb{R}^{N\times N}, \mathcal{A} \in \mathbb{R}^{N}, \mathcal{B} \in \mathbb{R}^{N}$
Output: $\hat{\Gamma}, \hat{\mathcal{A}}, \hat{\mathcal{B}}$ // Discretized params.

initialize zero matrix $\Psi, \hat{\Psi} \in \mathbb{R}^{(N+2)\times(N+2)}$
$N_\alpha = \text{argmax}(\mathcal{A})$
$N_\beta = \max(N_\alpha + 1, \text{argmax}(\mathcal{B}))$
$\Gamma[: N_\alpha, :] \leftarrow 0$ // remove edges $< N_\alpha$
$\Gamma[:, N_\beta :] \leftarrow 0$ // remove edges $> N_\beta$
$\Psi[1 : N, 1 : N] \leftarrow \Gamma$ // merge $\Gamma, \mathcal{A}, \mathcal{B}$ into Ψ
$\Psi[0, N_\alpha + 1 : N_\beta + 1] \leftarrow \mathcal{A}[N_\alpha : N_\beta]$
$\Psi[N_\alpha + 1 : N_\beta + 1, N + 1] \leftarrow \mathcal{B}[N_\alpha : N_\beta]$
while *True* **do**
  initialize zero matrix $S \in \mathbb{R}^{(N+2)\times(N+2)}$
  for $i \leftarrow 0$ *to* $\{N + 1\}$ **do**
    for $j \leftarrow 0$ *to* $\{N + 1\}$ **do**
      $S[i, j] \leftarrow$
      $\psi_{ij}\big(\frac{1}{\text{In}_{v_i}} \sum_{\psi_{ki}\in\Psi}(\psi_{ki}) / \sum_{\psi_{ik}\in\Psi}(\psi_{ik}) +$
      $\frac{1}{\text{Out}_{v_j}} \sum_{\psi_{jk}\in\Psi}(\psi_{jk}) / \sum_{\psi_{kj}\in\Psi}(\psi_{kj})\big)$
  $\psi_{ij} \leftarrow 0$, where $S[i, j]$ is nonzero min value
  if *graph builded from* Ψ *is reachable* **then**
    $\hat{\Psi} \leftarrow \Psi$
  else
    $\hat{\Psi}[\hat{\Psi} > 0] \leftarrow 1$ // discretization
    $\hat{\Gamma} \leftarrow \hat{\Psi}[1 : N, 1 : N]$ // split into $\Gamma, \mathcal{A}, \mathcal{B}$
    $\hat{\mathcal{A}} \leftarrow \hat{\Psi}[0, 1 : N]$
    $\hat{\mathcal{B}} \leftarrow \hat{\Psi}[1 : N, N + 1]$
    return $\hat{\Gamma}, \hat{\mathcal{A}}, \hat{\mathcal{B}}$

the only edge that can reach the graph. We use the simple Depth-first search algorithm to check the reachability of $\hat{\Gamma}$ between hidden state v_{N_α} to v_{N_β}. All the output $\hat{\mathcal{A}}, \hat{\mathcal{B}}, \hat{\Gamma}$ in Alg. 1, which is the discretized binary adjacency matrix, represent the truncated task-adaptive sub-network. After the reduction, we fix the upper-level parameters and only re-train the network parameters Θ, and we do not use the sigmoid function in Eq. 3-5

4. Experiments

We first describe the experimental setup in Sec. 4.1. We compare our method to state-of-the-art MTL frameworks on various benchmark datasets for MTL in Sec. 4.2. We also conduct extensive experiments and ablation studies to validate our proposed method in Sec. 4.3-4.5.

4.1. Experimental Settings

Dataset We use four public datasets for multi-task scenarios including Omniglot [17], NYU-v2 [34], Cityscapes [7], and PASCAL-Context [26]. We use these datasets, configured by previous MTL works [29, 38], not their original sources.

- **Omniglot** Omniglot is a classification dataset consisting of 50 different alphabets, and each of them consists of a number of characters with 20 handwritten images per character.
- **NYU-v2** NYU-v2 comprises images of indoor scenes, fully labeled for joint semantic segmentation, depth estimation, and surface normal estimation.
- **Cityscapes** Cityscapes dataset collected from urban driving scenes in European cities consists of two tasks: joint semantic segmentation and depth estimation.
- **PASCAL-Context** PASCAL-Context datasets contain PASCAL VOC 2010 [34] with semantic segmentation, human parts segmentation, and saliency maps, as well as additional annotations for surface normals and edge maps.

Competitive methods We compare the proposed framework with state-of-the-art methods [1, 11, 12, 18, 20, 22, 23, 25, 28, 29, 32, 38] and various baselines including a single task and a shared-bottom. The single-task baseline trains each task independently using a task-specific encoder and task-specific head for each task. The shared-bottom baseline trains multiple tasks simultaneously with a shared encoder and separated task-specific heads.

We compare our method with MoE-based approaches, including Soft Ordering [23], Routing [28], and Gumbel-Matrix [22], as well as a NAS approach [18] on Omniglot datasets. CMTR [18] can modify parameter count, similar to our method. We compare our method with other soft-parameter sharing methods including Cross-Stitch [25], Sluice network [32], and NDDR-CNN [11] and the dynamic neural network (DNN)-based methods including MTAN [20], DEN [1], and Adashare [38] for the other three datasets. We provide the evaluation results of two recent works, LTB [12] and PHN [29] for PASCAL-Context datasets because only the results are reported in their papers, but no source codes are provided.

Multi-task scenarios We set up multi-task scenarios with the combination of several tasks out of a total of seven tasks, including classification $\mathcal{T}_{cls}$, semantic segmentation $\mathcal{T}_{sem}$, depth estimation $\mathcal{T}_{dep}$, surface normal prediction $\mathcal{T}_{norm}$, human-part segmentation $\mathcal{T}_{part}$, saliency detection $\mathcal{T}_{sal}$, and edge detection $\mathcal{T}_{edge}$. We follow the MTL setup in [38] for three datasets including Omniglot, NYU-v2, and cityscapes, and [29] for PASCAL-Context. We simulate a homogeneous MTL scenario of a 20-way classification task in a multi-task setup using Omniglot datasets by following [23]. Each task predicts a class of characters in a single alphabet set. We use the other three datasets for heterogeneous MTL. We set three tasks including segmentation, depth estimation, and normal estimation for NYU-v2 and two with segmentation, depth estimation for Cityscapes.

Method	Test Acc. (%)	# of Param ↓
Soft Ordering [23]	66.59	**0.27**
CMTR [18]	87.19	-
MoE [28]	92.19	9.08
Gumbel-Matrix [22]	93.52	9.08
Single Task	93.48	20.00
Shared Bottom	93.25	1.00
Ours ($M = 3$)	94.99	0.91
Ours ($M = 5$)	**95.71**	1.37
Ours ($M = 7$)	95.68	1.46

Table 1. Evaluation on **Omniglot datasets**.

Method	$\Delta_{\mathcal{T}_{sem}} \uparrow$	$\Delta_{\mathcal{T}_{norm}} \uparrow$	$\Delta_{\mathcal{T}_{dep}} \uparrow$	$\Delta_{\mathcal{T}} \uparrow$	# of Param ↓
Single-Task	0.0	0.0	0.0	0.0	3.00
Shared Bottom	-7.6	+7.5	+5.2	+1.7	**1.00**
Cross-Stitch [25]	-4.9	+4.2	+4.7	+1.3	3.00
Sluice [32]	-8.4	+2.9	+4.1	-0.5	3.00
NDDR-CNN [11]	-15.0	+2.9	-3.5	-5.2	3.15
MTAN [20]	-4.2	+8.7	+3.8	+2.7	3.11
DEN [1]	-9.9	+1.7	-35.2	-14.5	1.12
AdaShare [38]	+8.8	+7.9	+10.1	+8.9	**1.00**
Ours ($M = 5$)	+11.9	+7.9	+8.8	+9.5	1.04
Ours ($M = 7$)	**+13.4**	**+9.2**	+10.7	**+11.1**	1.31
Ours ($M = 9$)	+13.2	+9.0	**+10.9**	+11.0	1.63

Table 2. Evaluation on **NYU-v2 datasets**.

We set five tasks $\mathcal{T}_{sem}$, $\mathcal{T}_{part}$, $\mathcal{T}_{norm}$, $\mathcal{T}_{sal}$, and $\mathcal{T}_{edge}$ as used in [29] for PASCAL-Context datasets.

Evaluation metrics We follow the common evaluation metrics utilized in the competitive methods. We use an accuracy metric for the classification task. The semantic segmentation task is measured by mean Intersection over Union (mIoU) and pixel accuracy. We use the mean absolute and mean relative errors, and relative difference as the percentage of $\delta = \max(\hat{\mathbf{d}}/\mathbf{d}, \mathbf{d}/\hat{\mathbf{d}})$ within thresholds $1.25^{\{1,2,3\}}$ for the depth estimation task. For the evaluation of the PASCAL-Context datasets, we follow the same metrics used in [29] for all tasks. As reported in [38], we report a single relative performance $\Delta_{\mathcal{T}_i}$ in Tab. 1-4 for each task $\mathcal{T}_i$ with respect to the single-task baseline, which defined as:

$$\Delta_{\mathcal{T}_i} = \frac{100}{|\mathcal{M}|} \sum_{j=0}^{|\mathcal{M}|} (-1)^{l_j} \frac{(\mathcal{M}_{\mathcal{T}_i,j} - \mathcal{M}^{single}_{\mathcal{T}_i,j})}{\mathcal{M}^{single}_{\mathcal{T}_i,j}}, \quad (10)$$

where $\mathcal{M}_{\mathcal{T}_i,j}$ and $\mathcal{M}^{single}_{\mathcal{T}_i,j}$ are the j-th metric of i-th task $\mathcal{T}_i$ from each method and the single task baseline, respectively. The constant l_j is 1 if a lower value represents better for the metric $\mathcal{M}_{\mathcal{T}_i,j}$ and 0 otherwise. The averaged relative performance for all tasks $\mathcal{T}$ is defined as:

$$\Delta_{\mathcal{T}} = \frac{1}{|\mathcal{T}|} \sum_{i=1}^{|\mathcal{T}|} \Delta_{\mathcal{T}_i}. \quad (11)$$

Method	$\Delta_{\mathcal{T}_{sem}} \uparrow$	$\Delta_{\mathcal{T}_{dep}} \uparrow$	$\Delta_{\mathcal{T}} \uparrow$	# of Param ↓
Single-Task	0.0	0.0	0.0	2.00
Shared Bottom	-3.7	-0.5	-2.1	1.00
Cross-Stitch [25]	-0.1	**+5.8**	+2.8	2.00
Sluice [32]	-0.8	+4.0	+1.6	2.00
NDDR-CNN [11]	+1.3	+3.3	+2.3	2.07
MTAN [20]	+0.5	+4.8	+2.7	2.41
DEN [1]	-3.1	-1.6	-2.4	1.12
AdaShare [38]	+1.8	+3.8	+2.8	1.00
Ours ($M = 5$)	+3.5	+3.9	+3.7	**0.96**
Ours ($M = 7$)	+7.5	+3.1	+5.3	1.16
Ours ($M = 9$)	**+8.3**	+4.8	**+6.6**	1.31

Table 3. Evaluation on **Cityscapes datasets**.

Method	$\Delta_{\mathcal{T}_{sem}} \uparrow$	$\Delta_{\mathcal{T}_{part}} \uparrow$	$\Delta_{\mathcal{T}_{sal}} \uparrow$	$\Delta_{\mathcal{T}_{norm}} \downarrow$	$\Delta_{\mathcal{T}_{edge}} \uparrow$	$\Delta_{\mathcal{T}} \uparrow$	# of Param ↓
Single-Task	0.0	0.0	0.0	**0.0**	0.0	0.0	5.00
Shared Bottom	-6.6	-0.7	-3.4	-14.3	0.0	-5.0	**1.00**
Cross-Stitch [25]	-1.3	+3.6	-0.2	-1.4	0.0	+0.1	5.00
Sluice [32]	-1.6	-1.2	-0.5	-2.9	-6.0	-2.4	5.00
NDDR-CNN [11]	-1.1	-2.6	0.0	-5.0	0.0	-1.7	5.61
MTAN [20]	-3.6	-0.7	-0.3	-5.0	-6.0	-3.1	5.21
AdaShare [38]	-1.4	**+4.0**	-0.5	-0.7	0.0	+0.3	**1.00**
LTB [12]	-6.9	-1.9	+0.2	-1.4	0.0	-2.0	3.19
PHN [29]	-6.6	-1.6	-1.0	**0.0**	0.0	-1.8	2.51
Ours ($M = 5$)	-0.3	+3.4	+0.9	0.0	**0.0**	+0.8	1.93
Ours ($M = 7$)	**0.0**	-0.2	+1.7	**+1.4**	**0.0**	+0.6	1.91
Ours ($M = 9$)	**0.0**	+3.6	**+1.8**	**+1.4**	**0.0**	**+1.4**	2.31

Table 4. Evaluation on **PASCAL-Context datasets**.

The absolute task performance for all metrics is reported in the supplementary material.

Network and training details For our central network, we set 8 hidden states, the same as the existing MoE-based works [22, 28] and use the same classification head for Omniglot datasets. We set 12 hidden states, the same as the VGG-16 [35], except for the max-pooled state, and use the Deeplab-v2 [5] decoder structure as all task heads for all the other datasets, respectively. We use the Adam [16] optimizer to update both upper-level parameters and network parameters. We use cross-entropy loss for semantic segmentation and L2 loss for the other tasks. For a fair comparison, we train our central network from scratch without pre-training for all experiments. We describe more details on the network structure and hyperparameter settings in the supplementary material.

4.2. Comparison to State-of-the-art Methods

We report the performance of the proposed method with different flow constants M and compare it with state-of-the-art methods in Tab. 1-4 with four different MTL scenarios. Tab. 1 shows that our framework with any flow constant M outperforms all the competitive methods for the homogeneous MTL scenario with Omniglot datasets. Ours has a similar number of parameters to the shared-bottom baseline. All the other experiments for heterogeneous MTL scenarios in Tab. 2-4 show that our frameworks achieve the best performance among all state-of-the-art works. Even with the flow constant $M = 5$, our

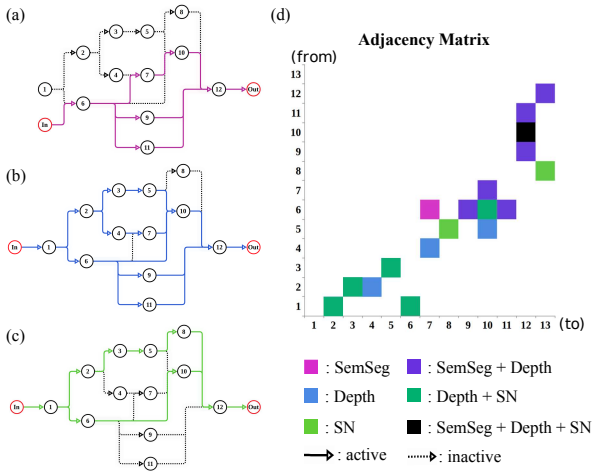


Figure 3. **Graph Representation of Task-adaptive Sub-network** The finalized sub-network topologies ($M = 7$) trained with NYU-v2 datasets is illustrated as graph. (a-c) The task-adaptive sub-network of semantic segmentation, depth estimation, and surface normal, respectively. (d) The adjacency matrix where color represents the discretized value for the activated edge of each task.

Task	$\mathcal{D}$	$\mathcal{W}$	$\mathcal{S}$
Semantic Seg.	5	3	0.103
Depth	7	3	0.192
Surface normal	7	2	0.128

Table 5. Topologies analysis on **NYU-v2 datasets**.

model outperforms AdaShare [38] for both the NYU-v2 and Cityscapes datasets, while keeping almost the same number of parameters (NYU-v2: 1.00 vs. 1.04 and Cityscapes 1.00 vs. 0.96). With the flow constant $M = 7, 9$, our method outperforms all the baselines by a large margin. The results from the PASCAL-Context datasets in Tab. 4 show that all baselines suffer from negative transfer in several tasks, as the number of tasks increases. Only Adashare and Cross Stitch slightly outperform the single-task baseline (see the performance $\Delta_{\mathcal{T}}$). On the other hand, ours with $M = 9$ achieves the best performance without any negative transfers for all tasks.

Interestingly, the required parameters of the search space increase almost in proportion to the increase of the flow constant, but there is no significant difference in the number of parameters of the finalized networks. For example, the required number of parameters for the network with the flow constant $M = 3, 5, 7$ is 2.77, 4.23, and 5.38, respectively. This demonstrates that the proposed flow-based reduction algorithm is effective in removing low-relative parameters while maintaining performance. Specifically, we observe that the total performance of our framework with $M = 7$ is slightly better than the $M = 9$ setup in Tab. 2 despite its smaller architecture search space. To investigate this, we further analyze the tendency in performance and computational complexity with respect to the flow constant in Sec. 4.4.

4.3. Analysis of Topologies and Task Correlation

To demonstrate the effectiveness of the proposed learning mechanism, we visualize our finalized sub-network topologies in Fig. 3-(a-c) and the adjacency matrix for NYU-v2 3-task learning in Fig. 3-(d). We also analyze the diversity and capacity of task-adaptive network topologies in Tab. 5 with network depth $\mathcal{D}$, width $\mathcal{W}$, and sparsity $\mathcal{S}$ described in Sec. 3.1. These analyses provide three key observations on our task-adaptive network and the correlation among tasks.

First, *the tasks of segmentation and surface normal hardly share network parameters*. Various task-sharing patterns are configured at the edge, but there is only one sharing layer between the two tasks. This experiment shows a low relationship between tasks, as it is widely known that the proportion of shared parameters between tasks indicates task correlation [22, 29, 38].

Second, *long skip connections are mostly lost*. The length of the longest skip connection in the finalized network is 5, and the number of these connections is 2 out of 18 layers, even with the flow constant of 7. This phenomenon is observed not only in NYU-v2 datasets but also in the other MTL datasets. This can be evidence that the proposed flow-restriction reduces search time while maintaining high performance even by eliminating the long skip connection in the DAG-based central network.

Lastly, *Depth estimation task requires more network resources than segmentation and surface normal estimation tasks*. We analyze the network topologies of the finalized sub-network of each task in NYU-v2 datasets using three components defined in Sec. 3. The depth $\mathcal{D}$ and width $\mathcal{W}$ of the sub-network increase in the order of semantic segmentation, surface normal prediction, and depth estimation tasks. Likewise, the sparsity $\mathcal{S}$ of the depth network is the highest. This experiment shows that the depth network is the task that requires the most network resources.

4.4. Performance w.r.t. Flow-resctriction

We analyze performance and computational complexity with respect to the flow constant M for the NYU-v2 and Cityscapes datasets. We report the rate of performance degradation with respect to the complete DAG search space in Fig. 4. We observe that the reduction rate of the final performance does not exceed 3% even with considerably lower flow constants. The performance is saturated at a flow constant of about 7 or more. This means that the proposed method optimizes the task adaptive sub-network regardless of the size of the network search space, if it satisfies the minimum required size.

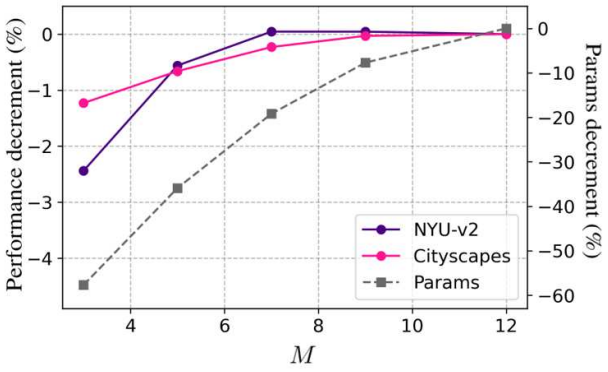


Figure 4. **Model performance with respect to the proposed flow-restriction.** We plot the degradation ratio of the performance (left y-axis) and parameter (right y-axis) by changing the flow constant M. We measure the final averaged task performance with NYU-v2 and Cityscapes datasets marked by purple and pink circle markers, respectively. We also measure the number of parameters marked by gray square markers.

To demonstrate the effectiveness of our flow-based reduction (FBR) algorithm, we compare it to two other reduction algorithms (random and threshold) for Cityscapes datasets in Fig. 5. The random reduction literally removes the edges randomly, and the thresholding method sequentially removes the edge which has the lowest value in the adjacency matrix Γ. We measure the rate of performance degradation of the pruned network of each reduction algorithm with respect to the non-reduced network while changing the sparsity $\mathcal{S}$. Note that our method automatically determines the sparsity, so for this experiment only, we add a termination condition that stops the network search when a certain sparsity is met. The results show that the proposed flow-reduction method retains performance even with a low sparsity rate. This means that our method efficiently prunes the low-related edge of the network compared to the other methods.

4.5. Ablation Study on Proposed Modules

We conduct ablation studies on the four key components of our framework; the flow-restriction, read-in/out layers, flow-based reduction, and squeeze loss. We report the relative task performance and the number of parameters of the finalized network with/without the components in Tab. 6. The results show that our framework, including all components, achieves the lowest number of parameters and the second-best performance. Our method without flow-based reduction achieves the best performance. However, the finalized network from this setup has about a five-times larger number of parameters than ours because the network has never been pruned in a training process. This demonstrates that our restricted DAG-based central network is optimized

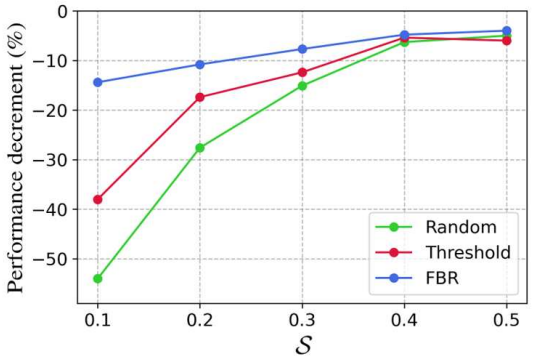


Figure 5. **Model Performance with respect to the network sparsity.** We plot the performance degradation rate by changing network sparsity. We compare our flow-based reduction algorithm to two other schemes; random selection and thresholding.

Method	$\Delta_{\mathcal{T}_{sem}} \uparrow$	$\Delta_{\mathcal{T}_{dep}} \uparrow$	$\Delta_{\mathcal{T}_{norm}} \uparrow$	$\Delta_{\mathcal{T}} \uparrow$	# of Param $\downarrow$
Ours (M=7)	+13.4	**+9.2**	+10.7	+11.1	**1.31**
w/o flow-restriction	+13.2	**+9.2**	+10.4	+11.0	1.80
w/o read-in/out	+11.7	+8.3	+10.4	+10.1	1.43
w/o flow-based reduction	**+14.2**	**+9.2**	**+11.1**	**+11.5**	6.50
w/o $\mathcal{L}_{sq}$	+13.2	+8.8	+10.7	+10.9	1.38

Table 6. **Ablation study on the proposed modules (NYU-v2).**

to build compact task-adaptive sub-networks with performance close to the optimized sub-network from a complete DAG-based network.

5. Conclusions

In this paper, we present a new MTL framework to search for task-adaptive network structures across diverse network topologies in a single network. We propose flow restriction to solve the scalability issue in a complete DAG search space while maintaining the diverse network topological representation of the DAG search space by adopting read-in/out layers. We also introduce a flow-based reduction algorithm that prunes the network efficiently while maintaining overall task performance and squeeze loss, limiting the upper bound on the number of network parameters. The extensive experiments demonstrate that the submodule and schemes of our framework efficiently improve both the performance and compactness of the network. Our method compactly constructs various task-specific subnetworks in a single network and achieves the best performance among all the competitive methods on four MTL benchmark datasets.

Acknowledgement

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. RS-2023-00210908).

References

[1] Chanho Ahn, Eunwoo Kim, and Songhwai Oh. Deep elastic networks with model selection for multi-task learning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6529–6538, 2019. 5, 6

[2] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. *arXiv preprint arXiv:1611.02167*, 2016. 2

[3] Hakan Bilen and Andrea Vedaldi. Integrated perception with recurrent multi-task neural networks. *Advances in neural information processing systems*, 29, 2016. 1, 2

[4] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332*, 2018. 2

[5] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017. 6

[6] Ying Chen, Jiong Yu, Yutong Zhao, Jiaying Chen, and Xusheng Du. Task's choice: Pruning-based feature sharing (pbfs) for multi-task learning. *Entropy*, 24(3):432, 2022. 1

[7] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 5

[8] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20(1):1997–2017, 2019. 2

[9] Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A Rusu, Alexander Pritzel, and Daan Wierstra. Pathnet: Evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734*, 2017. 2

[10] Yuan Gao, Haoping Bai, Zequn Jie, Jiayi Ma, Kui Jia, and Wei Liu. Mtl-nas: Task-agnostic neural architecture search towards general-purpose multi-task learning. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, pages 11543–11552, 2020. 2

[11] Yuan Gao, Jiayi Ma, Mingbo Zhao, Wei Liu, and Alan L Yuille. Nddr-cnn: Layerwise feature fusing in multi-task cnns by neural discriminative dimensionality reduction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3205–3214, 2019. 5, 6

[12] Pengsheng Guo, Chen-Yu Lee, and Daniel Ulbricht. Learning to branch for multi-task learning. In *International Conference on Machine Learning*, pages 3854–3863. PMLR, 2020. 2, 5, 6

[13] Junshi Huang, Rogerio S Feris, Qiang Chen, and Shuicheng Yan. Cross-domain image retrieval with a dual attribute-aware ranking network. In *Proceedings of the IEEE international conference on computer vision*, pages 1062–1070, 2015. 1, 2

[14] Brendan Jou and Shih-Fu Chang. Deep cross residual learning for multitask visual recognition. In *Proceedings of the 24th ACM international conference on Multimedia*, pages 998–1007, 2016. 1, 2

[15] Zhuoliang Kang, Kristen Grauman, and Fei Sha. Learning with whom to share in multi-task feature learning. In *ICML*, 2011. 1, 2

[16] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[17] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015. 5

[18] Jason Liang, Elliot Meyerson, and Risto Miikkulainen. Evolutionary architecture search for deep multitask networks. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 466–473, 2018. 5, 6

[19] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018. 1, 2, 3

[20] Shikun Liu, Edward Johns, and Andrew J Davison. End-to-end multi-task learning with attention. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1871–1880, 2019. 5, 6

[21] Jiaqi Ma, Zhe Zhao, Jilin Chen, Ang Li, Lichan Hong, and Ed H Chi. Snr: Sub-network routing for flexible parameter sharing in multi-task learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 216–223, 2019. 1, 2

[22] Krzysztof Maziarz, Efi Kokiopoulou, Andrea Gesmundo, Luciano Sbaiz, Gabor Bartok, and Jesse Berent. Flexible multi-task networks by learning parameter allocation. *arXiv preprint arXiv:1910.04915*, 2019. 2, 5, 6, 7

[23] Elliot Meyerson and Risto Miikkulainen. Beyond shared hierarchies: Deep multitask learning through soft layer ordering. *arXiv preprint arXiv:1711.00108*, 2017. 5, 6

[24] Risto Miikkulainen, Jason Liang, Elliot Meyerson, Aditya Rawal, Daniel Fink, Olivier Francon, Bala Raju, Hormoz Shahrzad, Arshak Navruzyan, Nigel Duffy, et al. Evolving deep neural networks. In *Artificial intelligence in the age of neural networks and brain computing*, pages 293–312. Elsevier, 2019. 2

[25] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3994–4003, 2016. 1, 2, 5, 6

[26] Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun, and Alan Yuille. The role of context for object detection and semantic segmentation in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 891–898, 2014. 5

[27] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameters sharing. In *International conference on machine learning*, pages 4095–4104. PMLR, 2018. 1, 2, 3

[28] Prajit Ramachandran and Quoc V Le. Diversity and depth in per-example routing models. In *International Conference on Learning Representations*, 2018. 5, 6

[29] Dripta S Raychaudhuri, Yumin Suh, Samuel Schulter, Xiang Yu, Masoud Faraki, Amit K Roy-Chowdhury, and Manmohan Chandraker. Controllable dynamic multi-task architectures. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10955–10964, 2022. 1, 2, 3, 5, 6, 7

[30] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V Le, and Alexey Kurakin. Large-scale evolution of image classifiers. In *International Conference on Machine Learning*, pages 2902–2911. PMLR, 2017. 2

[31] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017. 2

[32] Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. Sluice networks: Learning what to share between loosely related tasks. *arXiv preprint arXiv:1705.08142*, 2, 2017. 1, 2, 5, 6

[33] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017. 2

[34] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *European conference on computer vision*, pages 746–760. Springer, 2012. 5

[35] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 6

[36] Trevor Standley, Amir Zamir, Dawn Chen, Leonidas Guibas, Jitendra Malik, and Silvio Savarese. Which tasks should be learned together in multi-task learning? In *International Conference on Machine Learning*, pages 9120–9132. PMLR, 2020. 1, 2

[37] Masanori Suganuma, Shinichi Shirakawa, and Tomoharu Nagao. A genetic programming approach to designing convolutional neural network architectures. In *Proceedings of the genetic and evolutionary computation conference*, pages 497–504, 2017. 2

[38] Ximeng Sun, Rameswar Panda, Rogerio Feris, and Kate Saenko. Adashare: Learning what to share for efficient deep multi-task learning. *Advances in Neural Information Processing Systems*, 33:8728–8740, 2020. 1, 2, 3, 5, 6, 7

[39] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. Snas: stochastic neural architecture search. *arXiv preprint arXiv:1812.09926*, 2018. 2

[40] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016. 1, 2, 3

[41] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018. 1, 2