Deep Elastic Networks with Model Selection for Multi-Task Learning

Chanho Ahn*
Dept. of ECE and ASRI
Seoul National University
mychahn@snu.ac.kr

Eunwoo Kim*
Department of Engineering Science
University of Oxford
ekim@robots.ox.ac.uk

Songhwai Oh
Dept. of ECE and ASRI
Seoul National University
songhwai@snu.ac.kr

Abstract

In this work, we consider the problem of instance-wise dynamic network model selection for multi-task learning. To this end, we propose an efficient approach to exploit a compact but accurate model in a backbone architecture for each instance of all tasks. The proposed method consists of an estimator and a selector. The estimator is based on a backbone architecture and structured hierarchically. It can produce multiple different network models of different configurations in a hierarchical structure. The selector chooses a model dynamically from a pool of candidate models given an input instance. The selector is a relatively small-size network consisting of a few layers, which estimates a probability distribution over the candidate models when an input instance of a task is given. Both estimator and selector are jointly trained in a unified learning framework in conjunction with a sampling-based learning strategy, without additional computation steps. We demonstrate the proposed approach for several image classification tasks compared to existing approaches performing model selection or learning multiple tasks. Experimental results show that our approach gives not only outstanding performance compared to other competitors but also the versatility to perform instance-wise model selection for multiple tasks.

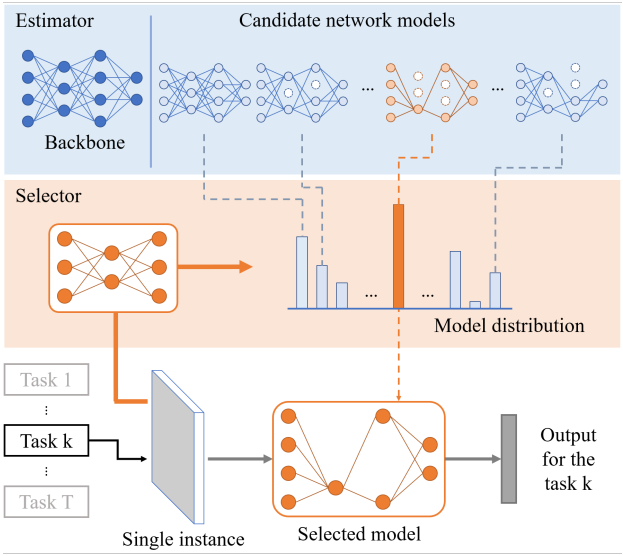

Figure 1. An overview of the proposed framework, which consists of an estimator and a selector. The estimator, whose structure is identical to the backbone network, includes multiple internal networks (models) of different configurations and scales. The selector outputs a probability distribution over the candidate models given an instance from a task. The model with the highest probability is chosen from the estimator to perform the assigned task.

1. Introduction

Multi-task learning (MTL) [5] simultaneously learns multiple tasks to improve generalization performance for the tasks. Most of recent MTL approaches [22–24, 29] are based on deep neural networks (DNNs) which have outstanding performance compared to traditional machine learning methods in computer vision and machine learning, such as image classification [10, 37], object detection [21], and pose estimation [26], to name a few.

Since it is believed that MTL methods using a DNN require a huge number of parameters and computing resources, a compact network with a small number of parameters and low computational complexity is highly desirable for many practical applications, such as mobile and embedded platforms [13]. To address this, there have been studies on designing compact DNNs, such as network pruning [9,34], knowledge distillation [12,28], network architecture search [39], and adaptive model compression [3, 20, 35]. However, these prior works have been applied to a single task problem and multiple tasks have been little considered in a single framework.

The MTL problem has a potential issue that the required number of parameters may increase depending on the number of tasks [5]. However, a single shared model for multiple tasks may cause performance degradation when associ-

*Indicates equal contribution

ated tasks are less relevant [29]. To avoid this issue, recent approaches [15, 16] proposed a network architecture which can contain several sub-models to assign the them to multiple tasks. Despite their attempts for MTL, they require human efforts to construct sub-models from the network architecture and assign the model to each task. For more flexible and adaptive model assignment for multiple tasks, it is desired to realize a model selection approach which automatically determines a proper sub-model depending on a given instance.

In this work, we aim to develop an instance-aware dynamic model selection approach for a single network to learn multiple tasks. To that end, we present an efficient learning framework that exploits a compact but high-performing model in a backbone network, depending on each instance of all tasks. The proposed framework consists of two main components of different roles, termed an *estimator* and a *selector* (see Figure 1). The estimator is based on a backbone (baseline) network, such as VGG [30] or ResNet [10]. It is structured hierarchically based on modularized blocks which consist of several convolution layers in the backbone network. It can produce multiple network models of different configurations and scales in a hierarchy. The selector is a relatively small network compared to the estimator and outputs a probability distribution over candidate network models for a given instance. The model with the highest probability is chosen by the selector from a pool of candidate models to perform the task. Note that the approach is learned to choose a model corresponding to each instance throughout all tasks. This makes it possible to share the common models or features across all tasks [7,15]. We design the objective function to achieve not only competitive performance but also resource efficiency (i.e., compactness) required for each instance. Inspired by [31], we introduce a sampling-based learning strategy to approximate the gradient for the selector which is hard to derive exactly. Both the estimator and the selector are trained in a unified learning framework to optimize the associated objective function, which does not require additional efforts (e.g., fine-tuning) performed in existing works [35, 39].

We perform a number of experiments to demonstrate the competitiveness of the proposed method, including model selection and model compression problems when a single or multiple tasks are given. For the experiments, we use an extensive set of benchmark datasets: CIFAR-10 and CIFAR-100 [18], Tiny-ImageNet[1], STL-10 [6], and ImageNet [19]. The experimental results on different learning scenarios show that the proposed method outperforms existing state-of-the-art approaches. Notably, our approach addresses both model selection and multi-task learning simultaneously in a single framework without introducing additional resources, making it highly efficient.

[1]https://tiny-imagenet.herokuapp.com/

2. Related Work

Model selection. In order to reduce the burden of an expert for designing a compact network, architecture search methods [39] were proposed to explore the space of potential models automatically. To shrink the daunting search space which usually requires a time-consuming exploration, methods based on a well-developed backbone structure find an efficient model architecture by compressing a given backbone network [2, 3]. Furthermore, the recent studies realizing such strategy [20, 33, 35] determine a different network model for each instance to reduce an additional redundancy. However, they usually achieve the lower performance compared to their backbone network [20, 33] or require additional fine-tuning process [35]. In contrast to them, we propose an efficient learning framework which can achieve better performance than the backbone network due to the dynamic model search and also does not includes an additional fine-tuning stage. Besides, our approach can be applied to learn multiple tasks simultaneously in a single framework, while aforementioned methods are limited to a single task.

Multi-task learning. The purpose of multi-task learning (MTL) is to develop a learning framework that jointly learns multiple tasks [5]. Note that we focus on a MTL method that learns a single DNN architecture for memory efficiency. There are several recent studies [11, 23, 24] that proposed a network structure in which parameters can be efficiently shared across tasks. Other approaches [15, 16, 22] suggest a single architecture which includes multiple internal networks (or models) so that they can assign different models to multiple tasks without increasing the parameters. However, they use a fixed model structure for each task and it requires expert efforts to assign the model to each task. In contrast, we propose a dynamic model selection for MTL which determines a proper model automatically for a given instance. Even if a recent MTL method [29] attempts model selection by a routing mechanism, it does not consider an optimized network structure associated with the number of parameters or FLOPs.

3. Approach

3.1. Overall framework

The goal of the proposed method is to develop a dynamic model selection framework when an input instance drawn from one of the target tasks is given. The proposed framework consists of two different components: an "estimator f" which is a network of the same size to the target backbone network and contains multiple different models of different network configurations, and a "selector g" which reveals a model with the highest probability in the estimator. Both estimator and selector are constructed based on

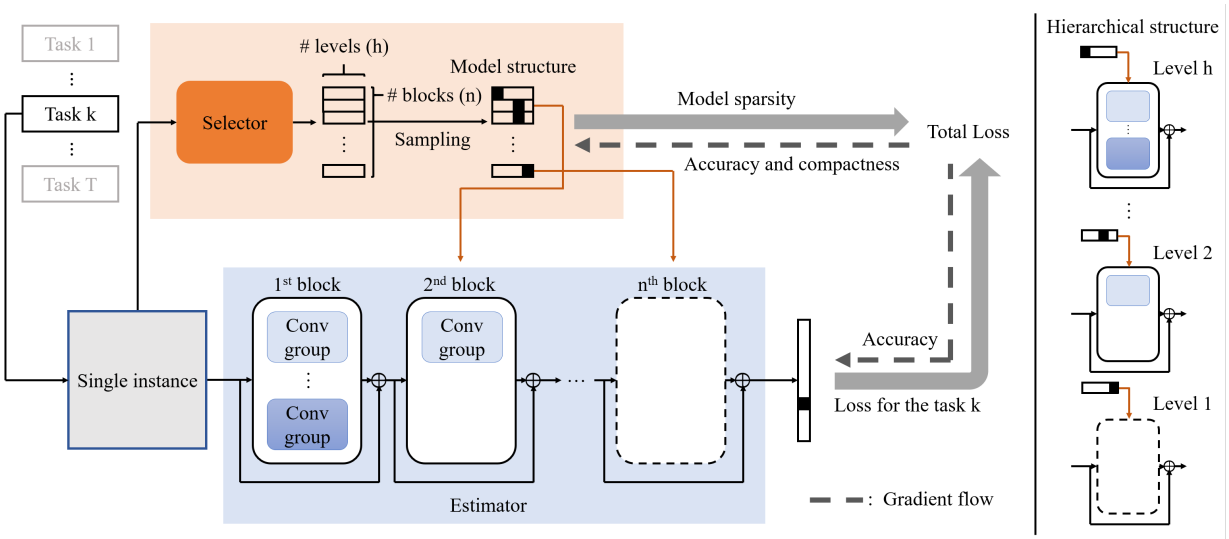

Figure 2. A graphical representation of the proposed framework which is based on a backbone network (a residual network [10]). The framework consists of an estimator and a selector. The estimator, whose structure is identical to the backbone network, contains n disjoint blocks. A block is defined as a collection of consecutive convolution layers (a block is the same as a residual block while keeping the number of channels). To simplify the hierarchical structure of each block, convolution layers in each block are divided into multiple groups. As shown on the right side of the figure, lower levels of hierarchy contain fewer convolution groups and higher levels contains more groups. The estimator can produce different network models by selecting convolution groups from zero to all groups in every block. The selector outputs a probability distribution over the convolution groups in every block, and a network model is determined from the distribution. The overall loss function consists of a prediction loss term (e.g., cross-entropy) from the determined network model and a sparse regularization term.

a CNN-based architecture, and the selector is designed to be much smaller than the estimator (see Section 4). The proposed approach explores a model search space and identifies an efficient network model to perform the given task in an instance-wise manner. The overall framework of the proposed approach is illustrated in Figure 1.

Note that there are a vast number of candidate models produced by the estimator, and this makes it difficult for the selector to explore the extensive search space. As a simplification strategy of the daunting task, we use a *block* notation to shrink the search space over the candidate models. A block is defined as a disjoint collection of multiple convolution (or fully connected) layers. The block is constructed as a hierarchical structure such that a lower level of hierarchy only refers fewer channels of hidden layers in the block and a higher level refers more channels, maintaining input and output dimensions of the block. Moreover, the lowest level of hierarchy can be constructed without any channels when the block is equivalent to a residual module [10]. This is similar to a layer skipping method in [35]. The hierarchical structure in a block is illustrated in Figure 2.

We determine a model structure by selecting a level of hierarchy in each block as follows: $z = (l_1, l_2, \cdots, l_n)$, where n is the number of the blocks in the estimator f and l_i denotes the selected level in the i-th block. Namely, a network model is collected in the estimator when the network model structure z is given. The inference of the determined network model is represented as follows:

$$f(\cdot; \theta_{est}, z, t) : \mathcal{X}_t \rightarrow \mathcal{Y}_t, \tag{1}$$

where θ_{est} is a set of parameters in the estimator, and $\mathcal{X}_t$ and $\mathcal{Y}_t$ denote input and output domains for task t, respectively. To address different input or output dimensions, we assume that the task ID is given beforehand.

The goal of the selector g is to find an appropriate network model for a given instance from a task by inferring the probability distribution over candidate models in the estimator. As mentioned earlier, we design the selector to produce a set of probability distributions over the modularized blocks (with their levels of hierarchy) as follows:

$$g(\cdot; \theta_{sel}) : \mathcal{X}_t \rightarrow [0, 1]^{h \times n}, \tag{2}$$

where θ_{sel} is a set of parameters of the selector and h is the number of levels of hierarchy in each block. We define the output of the selector as $\mathcal{C} \in [0, 1]^{h \times n}$ and each column of $\mathcal{C}$ reveals probabilities of selecting levels in the corresponding block (i.e., $\sum_i \mathcal{C}_{ij} = 1, \ \forall j$). Then, the probability of a candidate model for an instance x can be calculated as

$$P_{g(x;\theta_{sel})}(z; x) = \prod_{i=1}^{n} \mathcal{C}_i(l_i; x), \quad s.t. \ \ z = (l_1, \cdots, l_n), \tag{3}$$

where $\mathcal{C}_i(l_i; x) \in [0, 1]$ denotes the l_i-th element of the i-th column of $\mathcal{C}$, which means the probability that the l_i-th level is selected in the i-th block for an input x. Thus, we can represent up to h^n different candidate models, and one of them is selected to produce its corresponding model to perform the task. The overall framework is shown in Figure 2.

3.2. Optimization

The proposed approach is optimized to perform multi-task learning in an instance-wise manner within a single framework. We denote a set of datasets $\mathcal{D}$ as $\mathcal{D} = \{(x, y, t)|(x, y) \in \mathcal{D}_t, \ \forall t\}$, where x and y are an image and a label, respectively, and $\mathcal{D}_t$ is a dataset for task t. The proposed model selection problem is to minimize the loss functions for instances of all tasks while imposing the model size compact:

$$\begin{aligned} &J(\theta_{est}, \theta_{sel}) = \\ &\mathbb{E}_{(x,y,t)\in\mathcal{D}, z\sim g(x;\theta_{sel})} \left[\mathcal{L}(f(x; \theta_{est}, z, t), y) + \mathcal{S}(z)\right], \end{aligned} \tag{4}$$

where $\mathcal{L}(\cdot, \cdot)$ denotes a classification loss function (e.g., cross-entropy). $\mathcal{S}(z)$ is a sparse regularization term on the model structure z, which is defined as:

$$\mathcal{S}(z) = \rho \cdot \left(\frac{1}{n}\sum_{i=1}^{n} d_i(l_i)\right)^2, \quad s.t. \ z = (l_1, \cdots, l_n), \tag{5}$$

where $d_i(l_i)$ gives the ratio of the number of parameters determined by l_i from the total number of parameters in the i-th block, and ρ is a weighting factor. The square function in (5) can help enforce high sparsity ratio, and we have empirically found that it performs better than other regularization function, such as the l_1-norm.

The proposed approach involves alternating optimization steps for two sets of parameters, θ_{est} and θ_{sel}[2]. While θ_{est} can be updated by a stochastic gradient descent optimizer (SGD [4]), the gradient with respect to (4) for θ_{sel} is difficult to calculate without a exact expected value in (4). For this reason, we introduce a sampling-based approach to approximate the gradient. To describe the approximation, we introduce R which is equivalent to the loss function for θ_{sel} as follows:

$$\begin{aligned} &J_s(\theta_{sel}) = \mathbb{E}_{(x,y,t)\in\mathcal{D}, z\sim g(x;\theta_{sel})} \left[R(z; x, y, t)\right] \\ &s.t. \ R(z; x, y, t) \triangleq \mathcal{L}(f(x; \theta_{est}, z, t), y) + \mathcal{S}(z). \end{aligned} \tag{6}$$

Then, we can approximate the gradient value with sampled model structures, following the strategy in [31]:

$$\begin{aligned} &\nabla_{\theta_{sel}} J_s(\theta_{sel}) \\ &= \mathbb{E}_{(x,y,t)\in\mathcal{D}} \left[\sum_{\forall z} R(z; x, y, t) \nabla_{\theta_{sel}} P(z; x)\right] \\ &= \mathbb{E}_{(x,y,t)\in\mathcal{D}} \left[\sum_{\forall z} R(z; x, y, t) P(z; x) \frac{\nabla_{\theta_{sel}} P(z; x)}{P(z; x)}\right] \\ &= \mathbb{E}_{(x,y,t)\in\mathcal{D}, z\sim g(x;\theta_{sel})} \left[R(z; x, y, t) \nabla_{\theta_{sel}} \log P(z; x)\right] \\ &\approx \mathbb{E}_{(x,y,t)\in\mathcal{D}} \left[\sum_{z\in\mathcal{Z}} \frac{P(z; x)}{|\mathcal{Z}|} R(z; x, y, t) \nabla_{\theta_{sel}} \log P(z; x)\right], \end{aligned} \tag{7}$$

where $P(z; x) \triangleq P_{g(x;\theta_{sel})}(z; x)$. The last line approximates the expectation as the average for some randomly chosen samples z's which are collected from the same probability distribution when x is given. $\mathcal{Z}$ is a set of the collected z's and $|\mathcal{Z}|$ denotes the number of samples in $\mathcal{Z}$.

Note that the sampling scheme follows the common strategy in the reinforcement learning literature [25]. However, this can often lead to a worse network structure when the selected model is poor [36]. As a remedy, we apply the ϵ-greedy method [32] to allow more dynamic exploration at the earliest training time. In addition, we would like to note that the performance of the selected model may be sensitive to the initial distribution of the selector. For this reason, we use the following pre-determined distributions of the network model in the initial stage:

$$p(z_i) = \begin{cases} (1 - \tau)/h^n + \tau, & \text{if } z_i = z^*, \\ (1 - \tau)/h^n, & \text{otherwise,} \end{cases} \tag{8}$$

where τ is a weighting factor, $p(z_i)$ is a probability that the model structure z_i is selected, and z^* denotes the full model structure which includes all parameters in the estimator. In this work, we set τ to 0.75 in all conducted experiments. We increase the probability that the full model structure is selected more often in the initial stage, and it shows better performance compared to other initial distributions, such as a uniform distribution.

The overall training procedure of the proposed method, named deep elastic network (DEN), is summarized in Algorithm 1, where S denotes the number of stages. We optimize two sets of parameters, θ_{est} and θ_{sel}, during the several stages of the training process. At each stage, one of the above parameter sets is trained until it reaches the local optima.

4. Experiments

4.1. Experimental setup

Datasets. We evaluated the proposed framework on several classification datasets as listed in Table 1. For CIFAR-10,

[2] We call this alternating step as a stage.

Algorithm 1 Deep Elastic Network (DEN)

Input: $\mathcal{D}$, ρ
Initialize: θ_{est}, $\theta_{sel} \leftarrow$ Xavier-initializer [8], S
$p \leftarrow$ initial distribution in (8)
for s = 1 to S **do**
  repeat
    derive a model structure from p
    update θ_{est} w.r.t. (4)
  until convergence
  decay the learning rate for θ_{est}
  repeat
    update θ_{sel} using the gradient (7)
  until convergence
  decay the learning rate for θ_{sel}
  $p \leftarrow g(\cdot; \theta_{sel})$
end for

CIFAR-100, Tiny-ImageNet, and STL-10 datasets, we used the original image size. Mini-ImageNet is a subset of ImageNet [19] which has 50 class labels and each class has 800 training instances. We resized each image in the Mini-ImageNet dataset to 256×256 and center-cropped it to have the size of 224×224. As pre-processing techniques, we performed the random horizontal flip for all datasets and added zero padding of four pixels before cropping for CIFAR, Tiny-ImageNet, and STL-10 datasets. CIFAR-100 dataset includes two types of class categories for each image: 20 coarse and 100 fine classes. We used both of them for hierarchical classification; otherwise, we used the fine classes for the rest of the experiments.

Scenarios. We evaluated three scenarios for multi-task learning (MTL) and one scenario for network compression. For MTL, we organized two scenarios (*M1*, *M2*) using multiple datasets and one scenario (*M3*) using a single dataset with hierarchical class categories. For the first scenario, *M1*, we used three datasets of different image scales: CIFAR-100 (32×32), Tiny-ImageNet (64×64), and STL-10 (96×96). For *M2*, 50 labels are randomly chosen from the 1000 class labels in the ImageNet dataset and the chosen labels are separated into 10 disjoint subsets (tasks) each of which has 5 labels. *M3* is a special case of MTL (we call it hierarchical classification), which aims to predict two different labels (coarse and fine classes) simultaneously for each image. CIFAR-100 was used for the scenario *M3*. We also conducted the network compression scenario (*C1*) as a single task learning problem for CIFAR-10 and CIFAR-100, respectively.

Implementation details. We used ResNet-l [10] and WRN-l-r [38] as backbone networks in the MTL scenarios, where l is the number of layers and r is the scale factor on the number of convolutional channels. We borrowed a residual network architecture designed for ImageNet [19] to handle large-scale images and a WRN architecture designed for CIFAR [18] to handle small-scale images. We also used SimpleConvNet introduced in [27, 29] as a backbone network for Mini-ImageNet. SimpleConvNet consists of four 3x3 convolutional layers (32 filters) and three fully connected layers (128 dimensions for hidden units). In the network compression scenario, we used ResNeXt-l ($c \times s$d) [37] and VGG-l [30] to apply our methods in various backbones, where c and sd are the number of individual convolution blocks and unit depth of the convolution blocks in each layer, respectively [37]. The backbone networks are used as baseline methods performing an individual task in each scenario. To build the structure of the estimator, we defined a block as a residual module [10] and as two consecutive convolution layers for VGG networks. Then we split each block into multiple convolution groups along the channel dimension (2 or 3 groups in our experiments) to construct a hierarchical structure. Note that the lowest level of hierarchy does not have any convolution groups for ResNet, WRN, and ResNeXt, but has one group for VGG, and SimpleConvNet. The selector was designed with a network which is smaller than the estimator. The size of the selector is stated in each experiment.

Table 1. Summary of the datasets. The size represents the width and height of an input image for each dataset. # train and # test denote the number of images in the train and test sets, respectively.

Dataset	Size	# train	# test	# classes
CIFAR-10 [18]	32	50,000	10,000	10
CIFAR-100 [18]	32	50,000	10,000	100
Tiny-ImageNet	64	100,000	10,000	200
STL-10 [18]	96	5,000	8,000	10
Mini-ImageNet [29]	224	40,000	2,500	50

For the proposed method, named deep elastic network (DEN), the estimator was trained by the SGD optimizer with Nesterov momentum of 0.9, with the batch size of 256 for large-scale dataset (ImageNet) and 128 for other datasets. The ADAM optimizer [17] was used to learn the selector with the same batch size. The initial learning rates were 0.1 for the estimator and 0.00001 for the selector, and we decayed the learning rate with a factor of 10 when it converges (three or four decays happened in all experiments for both estimator and selector). All experiments are conducted in the TensorFlow environment [1].

Compared methods. We compared with four state-of-the-art algorithms considering resource efficiency for multi-task learning: PackNet*, NestedNet [15], Routing [29], and Cross-stitch [24]. PackNet* is a variant of PackNet [22], which considers group-wise compression along the channel dimension, in order to achieve practical speed-up like ours. Both PackNet* and NestedNet divide convolutional chan-

Table 2. Accuracy (%) on three tasks (datasets) of different input scales based on two different backbone networks: (a) ResNet-42 [10] and (b) WRN-32-4 [38]. We also provide FLOPs and the number of parameters for all compared methods. $[\cdot]$ denotes the number of required network models to perform the same tasks. Baseline requires three models to perform different tasks. ρ controls sparsity of the proposed method in (5). The bold and underline letters represent the best and the second best accuracy, respectively.

Dataset	Baseline [10]	NestedNet [15]	PackNet* [22]	DEN ($\rho = 1$)	DEN ($\rho = 0.1$)
CIFAR-100 (32×32)	<u>75.05</u>	74.53	72.22	74.30	**75.11**
Tiny-ImageNet (64×64)	<u>57.22</u>	56.71	56.49	56.74	**60.21**
STL-10 (96×96)	76.25	82.54	80.78	<u>83.90</u>	**87.58**
Average	69.51	71.26	69.83	<u>71.65</u>	**74.30**
FLOPs	2.91G	1.70G	1.70G	1.35G	1.61G
No. parameters	89.4M [3]	29.8M [1]	29.8M [1]	29.8M [1]	29.8M [1]

(a) ResNet-42

Dataset	Baseline [38]	NestedNet [15]	PackNet* [22]	DEN ($\rho = 1$)	DEN ($\rho = 0.1$)
CIFAR-100 (32×32)	75.01	74.09	73.56	<u>75.43</u>	**75.65**
Tiny-ImageNet (64×64)	**58.89**	57.87	57.17	58.17	<u>58.25</u>
STL-10 (96×96)	79.88	83.78	84.15	<u>87.54</u>	**87.56**
Average	71.26	71.91	71.63	<u>73.71</u>	**73.82**
FLOPs	2.13G	1.24G	1.24G	1.13G	1.14G
No. parameters	22.0M [3]	7.35M [1]	7.35M [1]	7.35M [1]	7.35M [1]

(b) WRN-32-4

nels into multiple disjoint groups and construct a hierarchical structure such that the i-th level of hierarchy includes i divided groups (the number of levels of hierarchy corresponds to the number of tasks). When updating the i-th level of hierarchy, NestedNet considers parameters in the i-th level but PackNet* considers parameters except those in the (i-1)-th level. For Routing and Cross-stitch, we used the results in [29] under the same circumstance. We also compared with BlockDrop [35], N2N [3], Pruning (which we termed) [14], and NestedNet [15] for the network compression problem. Note that we reported FLOPs and the number of parameters of the proposed method for the estimator in all experiments.

4.2. Multi-task learning

For the first scenario *M1* (on three tasks), we used both ResNet-42 and WRN-32-4 as backbone networks, respectively. The three tasks, Tiny-ImageNet, CIFAR-100, and STL-10, are assigned to the levels of hierarchy for PackNet* and NestedNet from the lowest to highest levels, respectively. The number of parameters and FLOPs of the selector are 1.49M and 0.15G for the ResNet-42 backbone and 0.37M and 0.11G for the WRN-32-4 backbone, respectively. The baseline method requires three separate networks, each trained independently. Table 2 shows the results with respect to accuracy, FLOPs and the number of parameters of the compared methods. Here, FLOPs denotes the average FLOPs for multiple tasks, and the number of parameters is measured from all networks required to perform the tasks. Overall, our approach outperforms other methods including the baseline method. In addition, we provide the results by varying the weighting factor ρ of our sparse regularizer in (5). As shown in the table, the performance is better when ρ is lower, and more compact model is selected when ρ is higher.

For the scenario *M2*, SimpleConvNet was used as a backbone network. Since the scenario contains a larger number of tasks than the previous scenario, PackNet* and NestedNet, which divide the model by human design, cannot be applied. We divided the convolution parts which takes most of the FLOPs in the network into two levels such that lowest level of hierarchy contains half the parameters of the highest level. The number of parameters of the selector is 0.4M, whereas the number of parameters of the estimator is 0.8M. In this scenario, the selector is not much smaller than the estimator because the estimator is constructed in sufficiently small size. However, the number of parameters of the selector for other scenarios are negligible compared to those of the estimator. The accuracy, FLOPs and the number of parameters of the compared methods are reported in Table 3. The result of the compared methods are reported in the work in [29]. Note that since the number of parameters and FLOPs are not precisely reported in the paper, we provide lower bounds. The proposed method shows a significant performance improvement compared to the other methods, even though ours uses lower average FLOPs than others for evaluations.

4.3. Hierarchical classification

For the scenario *M3*, we dealt with CIFAR-100 which has coarse and fine class categories for each image as described in Section 4.1. WRN-32-4 was used as a backbone network for this scenario. We compared with PackNet* and NestedNet, and the lowest and highest levels of hierarchy for them were allocated to perform the coarse and fine classifications, respectively. The structure of the selector in our method is equal to that in the scenario *M1*.

Table 4 summarizes the results of the compared meth-

Table 3. Accuracy (%) on the Mini-ImageNet dataset, FLOPs, and the number of parameters for all compared methods. Baseline uses the different last fully-connected layer for different tasks and shares other layers across the tasks. The bold and underline letters represent the best and the second best accuracy, respectively.

Method	Accuracy	FLOPs	No. params
Baseline	51.03	49.6M	0.8M
Cross-stitch [24]	56.03	$>$ 49.6M	$>$ 0.8M
Routing [29]	58.97	49.6M	$>$ 0.8M
DEN ($\rho = 1$)	<u>63.20</u>	33.3M	0.8M
DEN ($\rho = 0.1$)	**65.23**	39.1M	0.8M

ods for the coarse and fine classification problems. Our approach shows the best accuracy while giving the lowest FLOPs compared to the competitors except the baseline method for both problems. Furthermore, the proposed method has higher performance than the baseline method on average. Since each image has two different tasks (coarse and fine class categories), the selector exploits the same model structure and thus gives almost the same FLOPs.

4.4. Network compression

The goal of the network compression problem is to design a compact network model from a given backbone network while minimizing the performance degradation. We applied the proposed method to the network compression problem which is a single-task learning problem. We compared with BlockDrop [35] and NestedNet [15] on two backbone networks: ResNeXt [37] and VGG [30]. Since BlockDrop is developed for residual networks, we compared with it using ResNeXt. The CIFAR-10 and CIFAR-100 [18] datasets were used for the scenario, respectively. To verify the efficiency of the proposed method, we constructed our method with four levels of hierarchy for ResNeXt-29 (8×64d) and three levels for ResNeXt-29 (4×64d), respectively. The numbers of parameters of the selector are 3.9M and 3.6M for VGG and ResNeXt backbone networks, respectively.

Table 5 summarizes the classification accuracy of the compared approaches for the backbone networks. Overall, the proposed method shows the highest accuracy compared to other compression approaches. Our results with different ρ show that ρ can provide a trade-off between the network size and its corresponding accuracy. We also tested the proposed method (estimator) with a random selector, which reveals a model structure randomly among the candidate models in the estimator, to compare it with our model selection method. From the result, we can observe that the accuracy of the random selector is lower than the proposed selector, which reveals that the selector has potential to explore the desired model. Moreover, we compared with the state-of-the-art network compression methods, N2N [3], and Pruning [14], whose results were reported from their papers [3, 14]. Our approach has 94.47% classification accuracy using 5.8M parameters and the Pruning method has 94.15% accuracy using 6.4M parameters on the CIFAR-10 dataset. The proposed method also shows better performance than N2N and Pruning methods on the CIFAR-100 dataset.

Table 4. Hierarchical classification results on CIFAR-100. Baseline (WRN-32-4 [38]) requires two models to perform different tasks. The bold and underline letters represent the best and the second best accuracy, respectively.

Method	Accuracy	FLOPs	No. params
Baseline [38]	83.53	2.91G	14.7M
NestedNet [15]	<u>84.55</u>	1.46G	7.35M
PackNet* [22]	84.53	1.46G	7.35M
DEN ($\rho = 1$)	**84.87**	1.37G	7.35M

(a) Coarse classification (20)

Method	Accuracy	FLOPs	No. params
Baseline [38]	**76.32**	2.91G	14.7M
NestedNet [15]	75.84	2.91G	7.35M
PackNet* [22]	75.65	2.91G	7.35M
DEN ($\rho = 1$)	<u>75.93</u>	1.37G	7.35M

(b) Fine classification (100)

4.5. Quantitative results

The proposed instance-wise model selection for multi-task learning can associate similar features for similar images, which means that similar model structures can be selected for similar images. To verify this, we chose one input image (query) at each task and derived its output model distribution from the selector. Here, we measured the similarity between the distributions using l_2-distance. Then we collected four samples from each task, whose corresponding outputs have the similar model distribution to the query image. To do so, we constructed the proposed method based on the WRN-32-4 backbone architecture for the three tasks (datasets): CIFAR-100, Tiny-ImageNet, and STL-10. We set the size of input image to 32×32 for all the datasets to see the similarity under the same image scale. Figure 3 shows some selected images from all tasks for each query image. The results show that instance-wise model selection can be a promising strategy for multi-task learning as it can reveal the common knowledge across the tasks. We provide model distributions for instances from the test set in supplementary materials, along with the ablation study of using different numbers of levels.

5. Conclusion

In this work, we have proposed an efficient learning approach to perform resource-aware dynamic model selection for multi-task learning. The proposed method contains two main components of different roles, an estimator which produces multiple candidate models, and a selector which exploits a compact and competitive model among the candidate models to perform the designated task. We

Table 5. Network compression results on the CIFAR dataset. For FLOPs, we refer to the compression ratio from the baseline network for each model and dataset. The bold and underline letters represent the best and the second best accuracy, respectively. "rand sel" denotes that the random model structure is used without using the selector in the proposed method. The results of NestedNet are obtained from the lowest (L) to the highest (H) levels of hierarchy (including the intermediate level (M) for ResNeXt-29 (8×64d)).

Dataset		CIFAR-10			CIFAR-100		
Backbone	Method	Acc (%)	No. params	FLOPs	Acc (%)	No. params	FLOPs
VGG-16	Baseline [30]	**92.52**	38.9M	1.0×	**69.64**	38.9M	1.0×
	NestedNet [15], L	91.29	19.4M	2.0×	68.10	19.4M	2.0×
	NestedNet [15], H	<u>92.40</u>	38.9M	1.0×	<u>69.01</u>	38.9M	1.0×
	DEN ($\rho = 0.1$)	92.31	18.5M	2.4×	68.87	18.9M	1.7×
ResNet-18	N2N [3]	91.97	2.12M	–	69.64	4.76M	–
ResNet-34		93.54	3.87M	–	70.11	4.25M	–
ResNet-50	Pruning [14]	94.15	6.44M	–	74.10	9.24M	–
	DEN ($\rho = 1$)	94.50	4.25M	–	77.98	4.67M	–
ResNeXt-29 (8×64d)	Baseline [37]	**94.61**	22.4M	1.0×	**78.73**	22.4M	1.0×
	NestedNet [15], L	93.56	5.6M	4.0×	74.83	5.6M	4.0×
	NestedNet [15], M	93.64	11.2M	2.0×	74.98	11.2M	2.0×
	NestedNet [15], H	94.13	22.4M	1.0×	76.16	22.4M	1.0×
	BlockDrop [35]	93.56	16.9M	1.2×	78.35	15.5M	1.4×
	DEN + rand sel	90.55	9.8M	2.3×	69.67	9.8M	2.3×
	DEN ($\rho = 1$)	91.45	4.1M	5.5×	78.27	7.3M	3.0×
	DEN ($\rho = 0.1$)	**94.61**	8.7M	2.7×	<u>78.68</u>	13.5M	1.9×
ResNeXt-29 (4×64d)	Baseline [37]	<u>94.37</u>	11.2M	1.0×	**77.95**	11.2M	1.0×
	NestedNet [15], L	93.59	5.6M	2.0×	75.70	5.6M	2.0×
	NestedNet [15], H	94.11	11.2M	1.0×	76.36	11.2M	1.0×
	BlockDrop [35]	93.07	6.53M	1.7×	77.23	7.47M	1.5×
	DEN (rand sel)	87.33	5.6M	2.0×	65.44	5.6M	2.0×
	DEN ($\rho = 1$)	93.38	5.4M	2.1×	76.71	5.6M	2.0×
	DEN ($\rho = 0.1$)	**94.47**	5.8M	1.9×	<u>77.58</u>	6.3M	1.8×
ResNeXt-29 (2×64d)	Baseline [37]	93.60	5.6M	–	76.54	5.6M	–

Figure 3. Sampled images from each task (dataset) which have the similar model distribution to that of the query images (first column). The query images belong to CIFAR-100, Tiny-ImageNet, and STL-10 from top to bottom, respectively.

have also introduced a sampling-based optimization strategy to address the discrete action space of the potential candidate models. The proposed approach is learned in a single framework without introducing many additional parameters and much training efforts. The proposed approach has been evaluated on several problems including multi-task learning and network compression. The results have shown the outstanding performance of the proposed method compared to other competitors.

Acknowledgements: This research was supported in part by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) No. 2019-0-01190, [SW Star Lab] Robot Learning: Efficient, Safe, and Socially-Acceptable Machine Learning, and No. 2019-0-01371, Development of Brain-Inspired AI with Human-Like Intelligence, and by AIR Lab (AI Research Lab) of Hyundai Motor Company through HMC-SNU AI Consortium Fund.

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th Symposium on Operating Systems Design and Implementation (OSDI 16)*, 2016.

[2] Karim Ahmed and Lorenzo Torresani. MaskConnect: Connectivity learning by gradient descent. In *European Conference on Computer Vision (ECCV)*. Springer, 2018.

[3] Anubhav Ashok, Nicholas Rhinehart, Fares Beainy, and Kris M Kitani. N2N learning: network to network compression via policy gradient reinforcement learning. *arXiv preprint arXiv:1709.06030*, 2017.

[4] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[5] Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

[6] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of International Conference on Artificial Intelligence and Statistics*, 2011.

[7] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. DeCAF: A deep convolutional activation feature for generic visual recognition. In *International Conference on Machine Learning (ICML)*, pages 647–655, 2014.

[8] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of International Conference on Artificial Intelligence and Statistics*, 2010.

[9] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[11] Xiaoxi He, Zimu Zhou, and Lothar Thiele. Multi-task zipping via layer-wise neuron sharing. In *Advances in Neural Information Processing Systems (NIPS)*, 2018.

[12] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[13] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. MobileNets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[14] Yiming Hu, Siyang Sun, Jianquan Li, Xingang Wang, and Qingyi Gu. A novel channel pruning method for deep neural network compression. *arXiv preprint arXiv:1805.11394*, 2018.

[15] Eunwoo Kim, Chanho Ahn, and Songhwai Oh. NestedNet: Learning nested sparse structures in deep neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[16] Eunwoo Kim, Chanho Ahn, Philip HS Torr, and Songhwai Oh. Deep virtual networks for memory efficient inference of multiple tasks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

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

[18] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 and cifar-100 datasets. *URL: https://www.cs.toronto.edu/kriz/cifar.html (visited on Mar.1, 2016)*, 2009.

[19] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2012.

[20] Ji Lin, Yongming Rao, Jiwen Lu, and Jie Zhou. Runtime neural pruning. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

[21] Tsung-Yi Lin, Priyal Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2018.

[22] Arun Mallya and Svetlana Lazebnik. PackNet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

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

[24] 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 (CVPR)*, pages 3994–4003, 2016.

[25] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[26] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *European Conference on Computer Vision (ECCV)*. Springer, 2016.

[27] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. 2017.

[28] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. FitNets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014.

[29] Clemens Rosenbaum, Tim Klinger, and Matthew Riemer. Routing networks: Adaptive selection of non-linear functions for multi-task learning. *arXiv preprint arXiv:1711.01239*, 2017.

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

[31] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement

learning with function approximation. In *Advances in Neural Information Processing Systems (NIPS)*, 2000.

[32] Michel Tokic. Adaptive ε-greedy exploration in reinforcement learning based on value differences. In *Annual Conference on Artificial Intelligence*, pages 203–210. Springer, 2010.

[33] Andreas Veit and Serge Belongie. Convolutional networks with adaptive inference graphs. In *European Conference on Computer Vision (ECCV)*. Springer, 2018.

[34] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

[35] Zuxuan Wu, Tushar Nagarajan, Abhishek Kumar, Steven Rennie, Larry S Davis, Kristen Grauman, and Rogerio Feris. BlockDrop: Dynamic inference paths in residual networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[36] Jeremy Wyatt. Exploration and inference in learning from reinforcement. 1998.

[37] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[38] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

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