KSM: Fast Multiple Task Adaption via Kernel-wise Soft Mask Learning

Li Yang, Zhezhi He, Junshan Zhang, Deliang Fan
School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ
{lyang166, zhezhihe, Junshan.Zhang, dfan}@asu.edu

Abstract

Deep Neural Networks (DNN) could forget the knowledge about earlier tasks when learning new tasks, which is known as catastrophic forgetting. To learn new task without forgetting, recently, the mask-based learning method (e.g. piggyback [10]) is proposed to address this issue by learning only a binary element-wise mask, while keeping the backbone model fixed. However, the binary mask has limited modeling capacity for new tasks. A more recent work [5] proposes a compress-grow-based method (CPG) to achieve better accuracy for new tasks by partially training backbone model, but with order-higher training cost, which makes it infeasible to be deployed into popular state-of-the-art edge-/mobile-learning. The primary goal of this work is to simultaneously achieve fast and high-accuracy multi task adaption in continual learning setting. Thus motivated, we propose a new training method called Kernel-wise Soft Mask (KSM), which learns a kernel-wise hybrid binary and real-value soft mask for each task. Such a hybrid mask can be viewed as a superposition of a binary mask and a properly scaled real-value tensor, which offers a richer representation capability without low-level kernel support to meet the objective of low hardware overhead. We validate KSM on multiple benchmark datasets against recent state-of-the-art methods (e.g. Piggyback, Packnet, CPG, etc.), which shows good improvement in both accuracy and training cost.

1. Introduction

It is well-known that human and animals can learn new tasks without forgetting old ones. However, a practical limitation of Deep Neural Network (DNN) is their high degree of specialization to a single task and domain. For example, given a backbone DNN model, conventional fine-tuning of the model for new tasks could easily result in the forgetting of old knowledge upon earlier tasks, thus degrading the performance. Such phenomenon is known as *catastrophic forgetting*, which widely exists in continual learning [6]. The continual learning refers that a model is incrementally updated over a sequence of tasks, performing knowledge transfer from old tasks to the new one.

The typical way to alleviate the catastrophic forgetting issue is to fine-tune the backbone model w.r.t the new task with regularization [8, 6, 14, 1], thus preventing drastic weight update. Nevertheless, such method has limited success when many new tasks need to be learned. Different from that, Piggyback [10], a mask-based continual learning method, is proposed to address this issue by learning only binary (i.e., 0,1) element-wise masks w.r.t the weights, while keeping the backbone model fixed. Such mask is then multiplied by the fixed network weights, determining relevant or irrelevant for the current task. Since it only updates binary masks for each new task during training, it can be trained in fast manner, but with limited modeling capacity. To further improve the adaption capacity without forgetting, the compress-grow-based approach (e.g., CPG [5]) compresses (via pruning) and selectively expands the model iteratively. After punning, it utilizes the Piggyback method to learn a mask for the preserved weights as shown in Fig.1, and also retrains the released weights for current task. If the accuracy goal is not attained, it will expand the model by adding new filters. Such method outperforms Piggyback [10], as it involves additional task-specific parameters, but with order-higher training cost.

Motivation: Although Piggyback could learn new tasks in a fast manner, the binary mask has limited representation capacity, which gets non-ideal accuracy gain. As the countermeasure, CPG improves the representation capacity via combining the mask learning and additional task-specific parameters retraining. However, such complex training procedure suffers from extremely high training cost (i.e., training time and computing resources) that makes it impossible to deploy into state-of-the-art popular edge based or mobile computing based continual learning domain. These limitations motivate us to explore a new mask-based learning method that can rich the representation capacity, and more importantly, without involving additional training cost.

Contribution: In this work, we propose a new learning method called *Kernel-wise Soft Mask* (KSM), which learns a kernel-wise hybrid binary and real-value soft mask

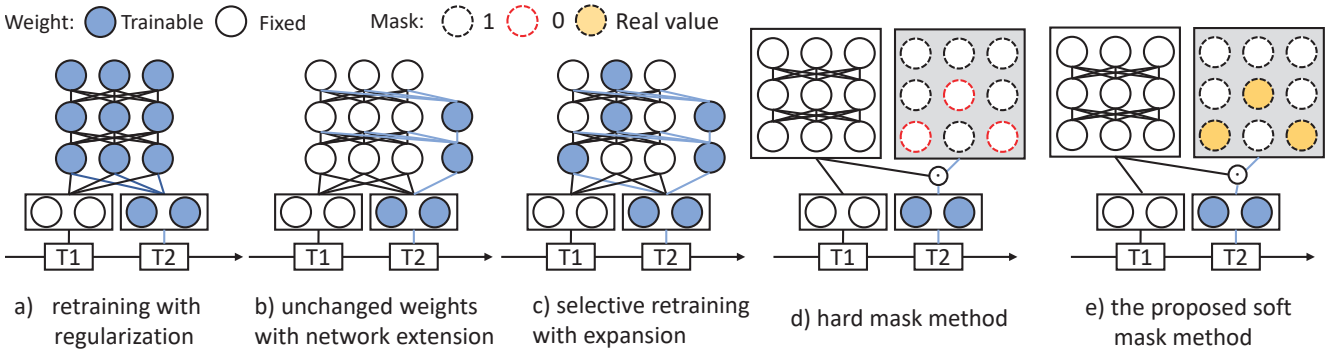

Figure 1: Overview of neural network approaches to overcome catastrophic forgetting, we consider the setting where each task retrains a new classifier. Except that, for the backbone model: a) retraining while regularizing to prevent catastrophic forgetting with previously learned tasks; b) unchanged weights with network extension for representing new tasks; c) selective retraining with possible expansion[5]; d) the hard mask method[10]; e) the proposed soft mask method.

for each new task, while keeping the backbone model fixed. The KSM method has the capability to mitigate the well-known catastrophic forgetting issue, to better transfer knowledge from old tasks, and more importantly, to improve the training efficiency. Our method is distinguished from prior works in the following aspects:

1. **Kernel-wise mask sharing.** To reduce the mask size and improve the computation efficiency in hardware, we design the mask in kernel-wise, instead of element-wise. For instance, only a single mask value is utilized to represent a 3 by 3 kernel, thus the mask size would reduce by 9 times.

2. **Soft mask.** To boost the knowledge representation ability without involving additional training cost, we decompose the mask into a binary mask and partial real-value scaling coefficient tensor.

3. **Softmax trick.** To obtain a better binary mask, we propose to leverage the softmax trick to relax the gradient approximation for mask during training.

Benefiting from the techniques above, the proposed KSM method could achieve similar to CPG (or even better) accuracy, while keeping similar to Piggyback (or even better) training speed.

2. Related Work

2.1. Dynamic architecture for continual learning

Dynamic architecture method addresses the catastrophic forgetting issue by selectively retraining and expanding the network architecture. [17] proposes to expand the network by generating new sub-network with fixed size for each task, while fixing the backbone model. [22] first selectively retrains the backbone network while expanding with limited

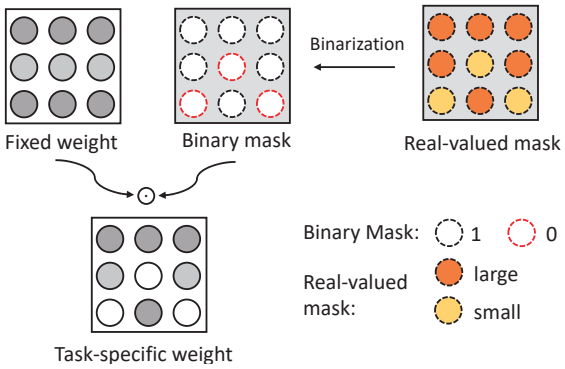

Figure 2: Illustration of Piggyback to learn a binary mask given a backbone model [10].

neurons by group-sparsity regularization, and then splits and duplicates the neurons to avoid catastrophic forgetting. Beyond that, PackNet [11] avoids this issue by identifying weights important for prior tasks through network pruning, while keeping the important weights fixed after training for a particular task. In contrast to directly expanding model architecture, [21] adds additional task-specific parameters for each task and selectively learns the task-shared parameters together. CPG [5] combines the model pruning, weight selection and model expansion methods, which gradually prunes the task-shared weights and then learns additional task-specific weights. Moreover, it could uniformly expand the model channels in each layer if the current accuracy can not meet the accuracy requirement.

2.2. Multi-domain learning

Multi-domain learning [13, 15] aims to build a model, which can adapt a task into multiple visual domains without forgetting previous knowledge, and meanwhile using as fewer parameters as possible. [15] proposes to recom-

bine the weights of the backbone model via controller modules in channel-wise. [9] proposes domain-specific attention modules for the backbone model. One of the most related method is Piggyback [10], which solves the issue by learning task-specific binary masks for each task, as illustrated in 1(d). They achieve this by generating the real-value masks which own the same size with weights, passing through a binarization function to obtain binary masks, which are then applied to existing weights. We denote the real-value mask and binary mask as $\mathbf{M}^r$ and $\mathbf{M}^b$ respectively, then, the binarization function is given by:

$$\text{Forward}: \quad \mathbf{M}^{\text{b}} = \begin{cases} 1 & \text{if } \mathbf{M}^{\text{r}} \geq \tau \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

$$\text{Backward}: \quad \nabla \mathbf{M}^{\text{b}} = \nabla \mathbf{M}^{\text{r}} \tag{2}$$

Where τ is a constant threshold value. However, the gradient of binarization is non-differential during back-propagation. They use the Straight-Through Estimator (STE) [4] to solve this problem, which estimates the gradient of real-value mask by the gradient of binary mask as shown in Fig 2.

3. Kernel-wise Soft Mask Method

Different from the conventional multi-task learning where the data of all tasks is available at training time, we consider a continual learning setting in which new tasks ($\{\mathcal{T}_1, \mathcal{T}_2, ..., \mathcal{T}_N\}$) arrive sequentially and past mask cannot be used for training future tasks. Given a convolution layer, we denote the weights $\mathbf{W}^{(l)} \in \mathbb{R}^{c_{\text{in}} \times c_{\text{out}} \times kh \times kw}$, where $c_{\text{in}}, c_{\text{out}}, kh, kw$ refers the weight dimension of l-th layer, including #output channel, #input channel, kernel height and width, respectively. We also denote the dataset of the t-th task ($\mathcal{T}_t$) as $\mathbf{D}_t = \{\boldsymbol{x}_t, \boldsymbol{y}_t\}$, where $\boldsymbol{x}_t$ and $\boldsymbol{y}_t$ are vectorized input data and label pair. To adapt the pre-trained backbone model with the parameter $\{\mathbf{W}_1\}$ from the initial task $\mathcal{T}_1$ to a new task $\mathcal{T}_t$, we intend to learn a task-specific kernel-wise soft mask $\mathbf{M}_t \in \mathbb{R}^{c_{\text{in}} \times c_{\text{out}} \times 1 \times 1}$ that is applied to the fixed parameter $\mathbf{W}_1$ to provide good performance. To reduce the mask size, each make element is shared by a $kh \times kw$ kernel. Based on this idea, the optimization objective can be mathematically formalized as:

$$\min_{\mathbf{M}_t} \mathcal{L}\Big(f(\boldsymbol{x}_t; \{\mathbf{M}_t \times \mathbf{W}_1\}), \boldsymbol{y}_t\Big) \tag{3}$$

Such soft mask based method differs from prior mask-based counterparts [10] in following aspects:

1. **Kernel-wise Mask Sharing**. Since the task-specific weights are refactorized from the backbone model via the task-specific mask, the size of mask directly determines the computation and model overhead for domain adaption objective. Instead of utilizing the mask in the identical size with weights as in [10], we introduce the compact mask where each mask element is shared by the kernel $kh \times kw$. Such kernel-wise mask method properly alleviates the computation and memory burden, and importantly, without accuracy degradation as demonstrated in our experiments.

2. **Soft Mask.** In contrast to prior works leveraging binary mask ($\mathbf{M}_t^b \in \{0, 1\}$), the proposed soft mask ($\mathbf{M}_t \in \mathbb{R}$) replaces the zero values inside binary mask with real-valued scaling coefficients. Such simple modification empowers the mask with a richer representation capability. However, although kernel-wise mask can greatly reduce the mask size, the additional memory usage caused by the real-valued $\mathbf{M}_t$ (32 bits) is still too large. For example, given a model with 3 by 3 kernel size in all layers, comparing with binary mask (1 bit), the soft mask will cause $\frac{32}{3\times3} \approx 3.5\times$ additional memory consumption. To address this issue, we divided the soft mask into two parts: one is binary

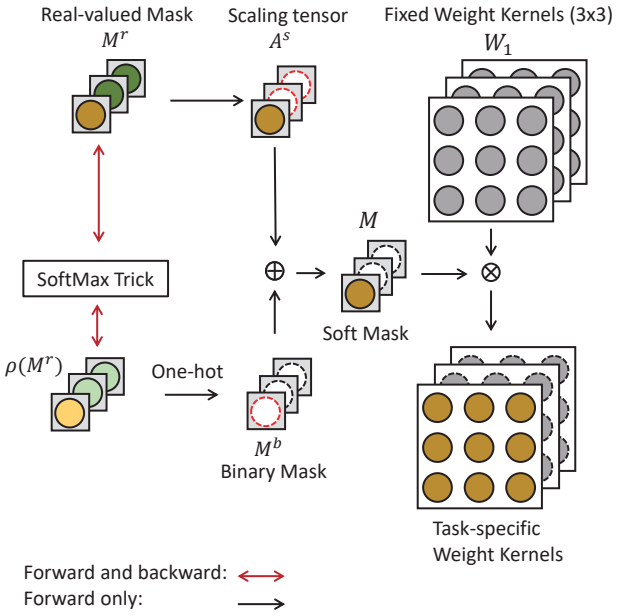

Figure 3: Overview of the proposed soft mask (KSM) learning method. Give a task t, we aim to learn a task-specific soft mask $\mathbf{M}_t$, by refactoring the fixed backbone weight to favor the current task. $\mathbf{M}_t$ is decomposed into a binary mask $\mathbf{M}_t^{\text{b}}$ and a scaling coefficient tensor $\mathbf{A}_t^{\text{s}}$ (4). To obtain $\mathbf{M}_t^{\text{b}}$, the learnable real-value mask $\mathbf{M}_t$ pass through a logistic function (6) and a softmax function (8) successively. In addition, scaling tensor $\mathbf{M}_t^{\text{b}}$ is generated by $\mathbf{M}_t^{\text{r}}$ (5). During training backward, the real-value mask can be updated without gradient estimation. After training, only the soft mask is saved for testing.

mask $\mathbf{M}_t^b$; the other is a scaling coefficient tensor $\mathbf{A}_t^s$ in sparse pattern. It can be expressed as:

$$\mathbf{M}_t = \mathbf{M}_t^b + \mathbf{A}_t^s. \tag{4}$$

3. **Softmax trick for better gradient calculation.** Since the soft mask above includes binary portion, there still exists the non-differential issue. Instead of utilizing Straight-Through Estimator (STE) to approximate gradient in the binary mask counterpart, we propose to leverage the softmax trick to relax the categorical objective. Compared to the STE method, the softmax trick could provide better gradient calculation, to achieve higher accuracy on new tasks.

Fig. 1 depicts the evolution from prior implementation to our method. More details of our soft mask-based method are presented in the following subsections.

3.1. Soft mask

As in Piggyback method [10], the adopted binary mask is generated by binarizing trainable real-valued masks. We conjecture that the magnitude of these masks have the in-nature property to represent the importance levels w.r.t the corresponding weights of the backbone model. Inspired by that, we aim to utilize the real-valued mask to represent scaling coefficient tensor.

However, there are two issues that impede us directly use the trainable real-valued mask: 1) Constructing the real-valued scaling coefficient tensor in dense pattern will largely increase the model size; 2) The magnitude of values in the real-value mask is typically very small (i.e. 0.01), even with negative values. To solve the fist issue, we introduce an additional trainable real-value scale coefficient tensor $\mathbf{A}^s$ as a replacement of the zero elements in the binary mask counterpart, so as to create a soft mask and avoid mask size increasing significantly due to those real values. In this way, it can improve the learning capacity without time-consuming re-training of zeroed-out weights in CPG [5]. To solve the second issue, we normalize those values to treat them as the scaling factor of each kernel when learning new tasks. In practice, we normalize the real-valued masks between 0 and 0.5 in all the experiments. As shown in Fig. 3, the above two steps can be achieved by inverting '0' and '1' in the generated binary mask $\mathbf{M}^b$, followed by multiplying with the real-value mask $\mathbf{M}^r$. Fig. 4 gives a toy example to show how to generate the scaling coefficient tensor by real-valued masks, which can be formulated as:

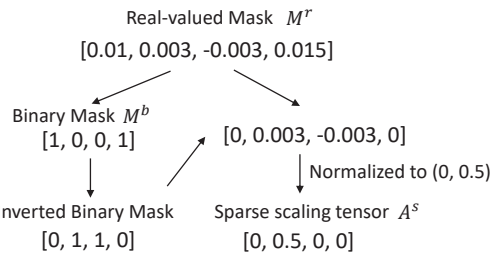

Figure 4: An example to generate the scaling coefficient tensor with four values.

$$\mathbf{A}^s = \text{normal}(\mathbf{M}_t^r) \tag{5}$$

Where $\widetilde{\mathbf{M}}^b$ inverts 0 and 1 in the $\mathbf{M}^b$. The 'detach' is used to only grasp the values of $\mathbf{M}^r$ without influence the back-propagation. Note that, since all the masks are set in kernel-wise, each mask value will be applied on a kernel weight as shown in Fig. 3.

In short, we generate the soft mask $\mathbf{M}$ by combining the binary mask $\mathbf{M}^b$ and the scaling factor $\mathbf{A}^s$ as shown in Eq. (4). It can be understood as we fix the important kernels ('1' in binary mask) and scale the unimportant kernels ('0' in binary mask) to be different trainable levels for the new task. The soft mask is generated in this way, mainly for the following two reasons:

1. Directly utilizing the already existing real-value mask does not involve additional trainable parameters or changing the backbone model architecture, indicating that can be trained with no extra cost.

2. These scaling factors increase the model capacity for the new task, with very small mask size increase due to the facts that 1) real-values occupy a small portion in the mask and 2) our kernel-wise mask dimension is already much smaller than traditional element-wise mask. We will quantify the overhead and the sparsity level in the analysis later.

3.2. Soft mask learning via softmax trick

[10] proposes a masking method, where they train a real-value mask followed by a hard threshold function to binarize the mask as depicted in Eq. (1). However, the binarization function is not differentiable, the general solution is to skip the threshold function during backpropagation and update the real mask by directly using gradients computed from binary mask, which is known as Straight Through Estimator (STE) as shown in Eq. (2). Different from that, we propose a method to eliminate the gradient estimation step and make whole mask learning compatible with existing gradient based backpropagation training process.

First, we relax the binarization function in Eq. (1) to a continuous logistic function:

$$\sigma(\mathbf{M}^r) = \frac{1}{1 + \exp(-k(\mathbf{M}^r - \tau))} \tag{6}$$

where k is a constant. Note that the logistic function becomes closer to hard thresholding function when k is increasing. The partial derivative of the logistic function is:

$$\frac{\partial \sigma(\mathbf{M}^r)}{\partial \mathbf{M}^r} = k \cdot \sigma(\mathbf{M}^r) \cdot (1 - \sigma(\mathbf{M}^r)) \tag{7}$$

In this work, we treat $\sigma(\mathbf{M}^r)$ as a probability mask to estimate the importance level of the corresponding weight kernels to save training time without involving extra parameters.

When considering it as a probability mask, sampling from a Bernoulli distribution is a reasonable and popular way to generate, but such sampling procedure is not differentiable. To overcome this issue, we leverage the softmax trick, which performs a differential sampling to approximate a categorical random variable. Summarizing, we define $p(\cdot)$ using the softmax trick as

$$p(\mathbf{M}^r) = \frac{\exp((\log \pi_0)/T)}{\exp((\log \pi_0)/T) + \exp((\log \pi_1)/T)} \tag{8}$$

Where π_0 and π_1 represent $1 - \sigma(\mathbf{M}^r)$ and $\sigma(\mathbf{M}^r)$ respectively. The temperature T is a hyper-parameter to adjust the range of input values, meanwhile choosing larger one could avoid gradient vanishing during back-propagation. Note that the output of $p(\mathbf{M}^r)$ closer to a Bernoulli sample as T towards to 0.

Benefiting from the differentiable property of Eq. (6) and Eq. (8), the real-value mask $\mathbf{M}^r$ can be embedded with existing gradient based backpropagation training without gradient approximation. During training, most values in the distribution of $p(\mathbf{M}^r)$ will move towards either 0 and 1. To represent $p(\mathbf{M}^r)$ as binary format, we use the one-hot code of $p(\mathbf{M}^r)$ during training forward, which has no influence on the real-value mask to be updated for back-propagation.

In the end, the soft mask is generated as described in Eq. (4). During forward, the input-output relationship of one layer is given by $\boldsymbol{y} = \mathbf{W}_1 \cdot (\mathbf{M}^{\mathrm{b}} + \mathbf{A}^{\mathrm{s}})\boldsymbol{x}$. According to the chain rule in the back-propagation, the gradient of such soft mask is given by:

$$\nabla \mathbf{M}^s = (\frac{\partial E}{\partial \boldsymbol{y}}) \cdot (\frac{\partial \boldsymbol{y}}{\partial p(\mathbf{M}^r)}) \cdot (\frac{\partial p(\mathbf{M}^r)}{\partial \sigma(\mathbf{M}^r)}) \cdot (\frac{\partial \sigma(\mathbf{M}^r)}{\partial \mathbf{M}^r}) \tag{9}$$

Where the partial derivative of each term is given by:

$$\begin{aligned} \frac{\partial E}{\partial \boldsymbol{y}} &= \nabla \boldsymbol{y} \\ \frac{\partial \boldsymbol{y}}{\partial p(\mathbf{M}^r)} &= \boldsymbol{x}^T \cdot \mathbf{W}_1 \\ \frac{\partial p(\mathbf{M}^r)}{\partial \sigma(\mathbf{M}^r)} &= -\frac{p(\mathbf{M}^r)(1 - p(\mathbf{M}^r))}{T\sigma(\mathbf{M}^r)(1 - \sigma(\mathbf{M}^r))} \end{aligned} \tag{10}$$

By doing so, the proposed method can optimize the soft mask in an end-to-end manner, where every step is differentiable. We illustrate the complete algorithm in Algorithm 1. During training, we save the optimized $\mathbf{M}^*$, and then directly apply it to the corresponding weight for testing.

Algorithm 1 The proposed soft mask learning

Require: Give the initial task τ_1 and the backbone model with the parameter $\mathbf{W}_1$, the threshold τ and temperature T

for Task t = 2, ..., N **do**
  Get data $\boldsymbol{x}_t$ and label $\boldsymbol{y}_t$
  $\mathbf{M}_t^b \leftarrow$ one-hot$(p(\mathbf{M}_t^r))$
  $\widetilde{\mathbf{M}}_t^b \leftarrow$ invert$(\mathbf{M}_t^b)$
  $\mathbf{M}_t \leftarrow \mathbf{M}_t^b + \widetilde{\mathbf{M}}_t^b \cdot$ normal$(\mathbf{M}_t^r$.detach$())$
  $\mathbf{M}_t^* \leftarrow \min_{\mathbf{M}_t^s} \mathcal{L}\Big(f(\boldsymbol{x}_t; \mathbf{W}_1 \cdot \mathbf{M}_t), \boldsymbol{y}_t\Big)$
  During testing, execute $f(\boldsymbol{x}_t; \mathbf{W}_1 \cdot \mathbf{M}_t^*)$
end for

4. Experiments

4.1. Datasets and backbone architectures

To make a fair comparison with prior works, similarly, we use VGG16-BN [19] and ResNet50 [3] as the backbone architectures for the following datasets:

ImageNet-to-Sketch In this experiments, six image classification datasets are used: CUBS [20], Stanford Cars [7], Flowers [12], WikiArt [18] and Sketch [2]. We use the ResNet50 as the backbone model which are trained on ImageNet dataset [16], then fine-tunes the fine-grained datasets sequentially.

Twenty Tasks of CIFAR-100 We divide the CIFAR-100 dataset into 20 tasks. Each task has 5 classes, 2500 training images, and 500 testing images. In the experiment, VGG16-BN model (VGG16 with batch normalization layers) is employed to train the 20 tasks sequentially.

4.2. Competing methods to compare

To test the efficacy of our KSM method, we compare it with recent several representative methods in three categories:

- **Whole model fine-tuning:** Fine-tuning the whole model for each task individually
- **PiggyBack [10]** It fixes the backbone weights and then learns a binary mask to select partial weights for new tasks.
- **PackNet [11]**: It first prunes unimportant weights, and then fine-tunes them for learning new tasks.
- **CPG [5]**: It combines PackNet and PiggyBack to gradually prune, pick and grow the backbone model for learning new tasks sequentially.

Table 1: The accuracy (%) and training cost (s) on Twenty Tasks of CIFAR-100. Considering those accuracy and training time comparison, we could achieve best average accuracy and around $\sim 10\times$ faster than CPG.

Methods		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	**Avg**
PackNet	Acc	66.4	80.0	76.2	78.4	80.0	79.8	67.8	61.4	68.8	77.2	79.0	59.4	66.4	57.2	36.0	54.2	51.6	58.8	67.8	83.2	67.5
	Time	334	360	370	379	382	385	385	389	234	358	370	378	384	385	384	337	359	371	377	382	365
Piggyback	Acc	65.8	78.2	76.4	79.8	86.0	81.0	79.4	82.4	81.8	86.4	87.8	76.0	82.8	80.6	48.2	70.4	65.0	71.80	87.80	90.6	77.1
	Time	100	150	102	113	154	102	121	119	97	130	84	110	96	120	106	97	97	106	110	119	111
CPG	Acc	66.6	76.2	78.2	80.6	86.4	83.0	81.4	82.4	82.0	86.8	86.8	81.4	82.8	82.0	50.4	72.4	66.2	71.2	85.6	91.6	78.7
	Time	629	2101	2123	2120	2121	2127	2116	2120	2122	2121	2122	2115	2127	2125	2126	2114	2124	2126	2123	2125	2046
Ours	Acc	67.2	78.0	78.8	78.4	85.6	82.6	80.2	83.4	82.6	89.4	88.4	80.6	83.2	80.8	52.8	73.2	67.8	72.6	88.0	92.0	**79.2**
	Time	130	81	111	123	123	127	62	106	88	78	95	85	73	88	90	90	80	95	96	65	**94.3**

4.3. Results on ImageNet-to-Sketch dataset

In this experiment, following the same settings in the works of CPG [5] and Piggyback [10], we train each task for 30 epochs using the Adam optimizer. The initial learning rate is 1e-4, which is decayed by a factor of 10 after 15 epochs.

4.3.1 Accuracy comparison

The accuracy of the five classification tasks is tabulated in Table 2. For the first ImageNet task, CPG and PackNet perform slightly worse than the others, since both methods have to compress the model via pruning. Then, for the following five fine-grained tasks, the proposed method could achieve all better accuracy comparing with Piggyback and PackNet. Even comparing with the individually fine-tuning whole model for each task, we could still achieve better performance except WikiArt dataset. In comparison to CPG that requires one order more training time (Fig.5), our method achieves better accuracy in tasks of CUBS, Flowers and Sketch. However, we admit that, owing to a small portion of real-values in the hybrid mask, our method needs slightly more model size than other methods. Note that, the model size reported in 2 includes both the backbone model and introduced mask size.

Table 2: Accuracy on ImageNet-to-Sketch dataset

Dataset	Finetune	PackNet	Piggyback	CPG	Ours
ImageNet	76.26	75.71	76.16	75.81	76.16
CUBS	82.83	80.41	81.59	83.59	**83.81**
Cars	91.83	86.11	89.62	**92.80**	92.14
Flowers	96.56	93.04	94.77	96.6	**96.94**
WikiArt	75.60	69.40	71.33	**77.15**	75.25
Sketch	80.78	76.17	79.91	80.33	**81.12**
Model Size (MB)	554	115	121	121	146

4.3.2 Training time comparison

To do fair comparisons, all the methods are trained on the single Quadro RTX-5000 GPU with the same batch size and epochs. Fig. 5 summarizes the whole training time for each method. First, our method slightly outperforms Piggyback, since the proposed soft mask learning method (as illustrated in Eq. (6) and Eq. (4)) is faster than the binarization function in real hardware implementation. Then, ours and Piggyback could both achieve better speed than PackNet, since PackNet needs to retrain weights which is slower than training a mask. Last, it is obvious that CPG requires $\sim 10\times$ more training time than all rest methods, while 3 out of 5 tasks have lower accuracy than ours as shown in Table 2.

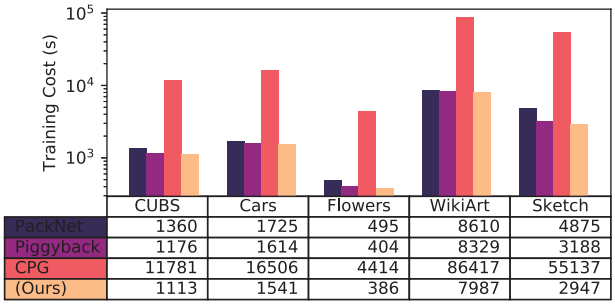

Figure 5: Training cost on ImageNet-to-Sketch datasets with various continual learning methods.

4.4. Results on twenty tasks of CIFAR-100

Different from the ImageNet-to-Sketch setting that relies on a pre-trained model on ImageNet dataset, in this experiment, we first train a task from scratch as the backbone model. Afterward, we fix the backbone model weights and learn the task-specific mask for the rest tasks sequentially. To conduct a fair comparison, we follow the same configuration as CPG, and select the same task as the initial task-1. Note that, since this work only focuses on continual learning without model expansion, all the CPG results are without expansion based on their open source code.

4.4.1 Accuracy and training time comparison

Similar phenomenon can be observed with the ImageNet-to-Sketch setting. Table 1 shows the accuracy and training

Table 3: The accuracy on Twenty Tasks of CIFAR-100 with different initial tasks. The accuracy of individual task on these five settings is slightly different. Nevertheless, the average accuracy is better than PackNet and Piggyback. Comparing with CPG, better accuracy could be achieved on three different initial types.

Initial	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	**Avg**
1	67.2	78.0	78.8	78.4	85.6	82.6	80.2	83.4	82.6	89.4	88.4	80.6	83.2	80.8	52.8	73.2	67.8	72.6	88.0	92.0	79.2
5	67.0	77.2	77.6	79.2	84.8	82.6	78.0	85.2	82.8	88.8	88.4	80.8	84.2	81.4	50.2	71.8	67.0	71.2	86.0	91.8	78.8
10	67.8	77.2	76.6	79.4	82.8	81.6	80.8	83.4	82.0	88.6	88.2	81.2	85.0	80.2	53.4	73.8	68.6	74.4	87.2	91.2	**79.3**
15	67.6	78.2	77.0	77.0	81.8	82.6	78.4	83.4	83.2	86.6	88.4	80.0	83.0	78.0	51.2	70.8	67.8	67.8	86.4	91.0	78.0
20	66.8	75.6	77.2	76.6	85.4	81.0	79.0	84.0	82.2	87.4	86.4	79.0	83.8	80.4	49.0	70.8	66.4	72.0	88.2	93.6	78.2

cost for these methods. Our method could achieve completely better results than Piggyback and PackNet. In addition, comparing with CPG, we also could achieve better results in most tasks. In terms of training time, our method is around $\sim 10\times$ faster than CPG.

Considering those accuracy and training time comparison, it shows our method could perform outstanding knowledge transfer based on a weak backbone model, which only trains on 2 classes. It is worthy to note that the initial task indeed influences the performance of rest tasks, since we fix the backbone weights all the time. In the next section, we will show that, even with different initial tasks, in all cases, our method could learn a mask that achieves good knowledge representation for new task.

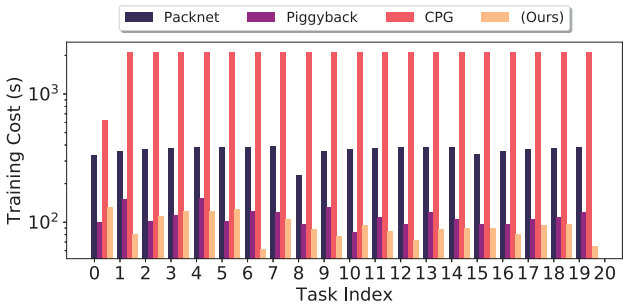

Figure 6: Training cost on twenty tasks of CIFAR-100 with various continual learning methods.

4.5. Ablation Study and Analysis

4.5.1 Kernel-wise, soft mask and softmax trick

We study the individual effect of the three main techniques of our proposed method on ImageNet-to-Sketch dataset setting. As shown in Table. 4, we denote the 'Piggyback-Soft' as replacing the 0 values in piggyback's binary mask with scaling factors, and denote the 'Ours-softmax' as we only use the proposed softmax trick to generate binary mask. Also, we name the 'Ker-wise' and 'Ele-wise' as kernel-wise and element-wise mask respectively. The 'Ours-Softmax' achieves better results than Piggyback, which means the proposed differentiable mask learning process with softmax trick could generate better optimization, since we don't have gradient estimation. In addition, 'Piggyback-Soft' achieves better results than 'Piggyback' proving that adding scaling factors to zeroed-out weights indeed improves the task-specific representation ability. Also, changing the mask to kernel-wise has very minor or neglect-able influence for performance. In the end, the 'Ours-Full' combines all three techniques, showing best overall performance in all datasets.

Table 4: The ablation study on the proposed method.

Method	CUBS	Cars	Flowers	WikiArt	Sketch
Piggyback	81.59	89.62	94.77	71.33	79.91
Piggyback - Ker-wise	81.76	89.57	94.88	70.30	79.95
Piggyback - Soft	82.26	91.17	95.85	73.12	80.22
Ours - Softmax	82.86	91.71	96.67	74.06	80.70
Ours - Ele-wise	83.79	92.18	96.90	75.0	81.10
Ours - Full	83.81	92.14	96.94	75.25	81.12

4.5.2 The Effect of Different Initial Tasks

Different from the ImageNet-to-Sketch dataset setting that heavily relies on a strong pre-trained model, we randomly select a task and then train it from scratch as the initial model in Twenty Tasks of CIFAR-100 setting. To explore how does the initial task affects the performance of rest tasks, we randomly select five different tasks as the initial task as shown in Fig. 6. Thus, the accuracy of these five settings on each individual task is slightly different, since they own different domain shift levels. Nevertheless, the average accuracy is better than PackNet and Piggyback. Comparing with CPG, better accuracy could be achieved on three different initial types, which indicates that the proposed method could balance the domain shift with different initial tasks.

4.5.3 Scaling Tensor Distribution on ImageNet-to-Sketch Setting

Fig. 7 shows the distribution of the scaling tensors on five different tasks respectively. We select two representative layers of the ResNet50 model: first layer and last layer. In practice, we normalize the real-valued mask between 0 and 0.5. Three observations could be drawn across all tasks: 1)

end layers have more scaling tensors than front layers. Especially, the last layer has the highest scaling tensor ratio in the soft mask; 2) the number of scaling tensors almost gradually increases from 0 to 0.5, which depends on the learning property of the real-valued mask; 3) different from Flowers, cars and CUBS dataset, WikiArt and Sketches have more number of scaling tensors between 0 and 0.2, which reflect these two tasks are more difficult in terms of domain shift.

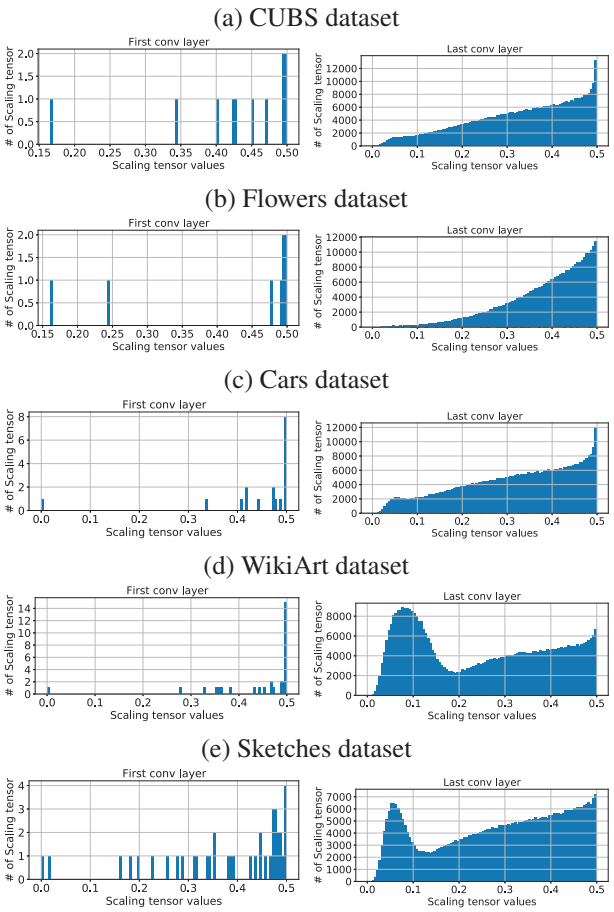

Figure 7: The scaling tensor distribution on ImageNet-to-Sketch setting

4.5.4 Architecture and Soft Mask Visualization

Fig. 8 visualizes the ratio of '1' values in binary mask and the scaling factor. Two observations could be drawn from all the tasks: 1) Within a task, end layers need more changes than front layers, especially the last convolutional layer. 2) The scaling factor ratio may link to the domain shift difficulty. For example, the largest dataset WikiArt has a higher ratio the that of smallest dataset Flowers.

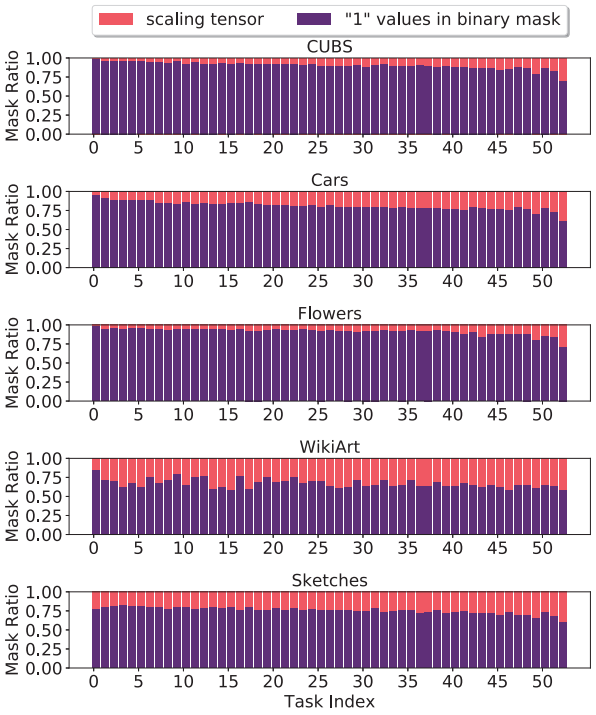

Figure 8: The ratio of two mask types visualization on ResNet50 for ImageNet-to-Sketches dataset.

5. Conclusion

In this work, we propose a novel kernel-wise soft mask method for multiple task adaption in the continual learning setting, which learns a hybrid binary and real-value soft mask of a given backbone model for new tasks. Comprehensive experiments on the ImageNet-to-Sketch dataset and twenty tasks of CIFAR-100 indicate that, with no need of using weight regularization and model expansion, the proposed method can run $\sim 10\times$ faster than the state-of-the-art CPG based learning method with similar accuracy performance. In addition, we analyze the effect of different backbone models. Even with a weak backbone model, the proposed method also could learn reasonable information for new tasks. We show that we can achieve better results compared with the related prior mask-based methods.

Acknowledgement. This work is supported in part by the National Science Foundation under Grant No.2005209, No.1931871, No. 2019548

References

[1] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420*, 2018. 1

[2] Mathias Eitz, James Hays, and Marc Alexa. How do humans sketch objects? *ACM Transactions on graphics (TOG)*, 31(4):1–10, 2012. 5

[3] 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*, pages 770–778, 2016. 5

[4] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. In *Advances in neural information processing systems*, pages 4107–4115, 2016. 3

[5] Ching-Yi Hung, Cheng-Hao Tu, Cheng-En Wu, Chien-Hung Chen, Yi-Ming Chan, and Chu-Song Chen. Compacting, picking and growing for unforgetting continual learning. In *Advances in Neural Information Processing Systems*, pages 13669–13679, 2019. 1, 2, 4, 5, 6

[6] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017. 1

[7] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 554–561, 2013. 5

[8] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017. 1

[9] Shikun Liu, Edward Johns, and Andrew J Davison. End-to-end multi-task learning with attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1871–1880, 2019. 3

[10] Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 67–82, 2018. 1, 2, 3, 4, 5, 6

[11] 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*, pages 7765–7773, 2018. 2, 5

[12] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pages 722–729. IEEE, 2008. 5

[13] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with residual adapters. In *Advances in Neural Information Processing Systems*, pages 506–516, 2017. 2

[14] Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. *arXiv preprint arXiv:1810.11910*, 2018. 1

[15] Amir Rosenfeld and John K Tsotsos. Incremental learning through deep adaptation. *IEEE transactions on pattern analysis and machine intelligence*, 2018. 2

[16] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015. 5

[17] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016. 2

[18] Babak Saleh and Ahmed Elgammal. Large-scale classification of fine-art paintings: Learning the right metric on the right feature. *arXiv preprint arXiv:1505.00855*, 2015. 5

[19] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 5

[20] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011. 5

[21] Jaehong Yoon, Saehoon Kim, Eunho Yang, and Sung Ju Hwang. Scalable and order-robust continual learning with additive parameter decomposition. *arXiv preprint arXiv:1902.09432*, 2019. 2

[22] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547*, 2017. 2