EXTENDING CONDITIONAL CONVOLUTION STRUCTURES FOR ENHANCING MULTITASKING CONTINUAL LEARNING

Cheng-Hao Tu*, Cheng-En Wu* and Chu-Song Chen*†‡
* Institute of Information Science, Academia Sinica, Taipei, Taiwan
E-mail: {andytu28, chengen, song}@iis.sinica.edu.tw
† Research Center for Information Technology Innovation, Academia Sinica, Taipei, Taiwan
‡ MOST Joint Research Center for AI Technology and All Vista Healthcare, Taipei, Taiwan

***Abstract*—Conditional operations have received much attention in recent deep learning studies to facilitate the prediction accuracy of a model. A recent advance toward this direction is the conditional parametric convolutions (CondConv), which is proposed to exploit additional capacities provided by the deep model weights to enhance the performance, whereas the computational complexity of the model is much less influenced. CondConv employs input-dependent fusion parameters that can combine multiple columns of convolution kernels adaptively for performance improvement. At runtime, the columns of kernels are on-line combined into a single one, and thus the time complexity is much less than that of employing multiple columns in a convolution layer under the same capacity. Although CondConv is effective for the performance enhancement of a deep model, it is currently applied to individual tasks only. As it has the nice property of adding model weights with computational efficiency, we extend it for multi-task learning, where the tasks are presented incrementally. In this work, we introduce a sequential multi-task (or continual) learning approach based on the CondConv structures, referred to as CondConv-Continual. Experimental results show that the proposed approach is effective for unforgetting continual learning. Compared to current approaches, CondConv is advantageous to offer a regular and easy-to-implement way to enlarge the neural networks for acquiring additional capacity and provides a cross-referencing mechanism for different task models to achieve comparative results.**

***Index Terms*—Convolutional Neural Network (CNN), Multi-task Learning, Continual Learning, Conditional Computation, Deep Learning**

I. Introduction

Continual learning of multiple tasks is essential for many applications in computer vision. In continual learning, the tasks are presented in a sequential manner, and each task is learned only based on its own training data without accessing to the data of previous or future tasks. Although fine-tuning the model learned from previous tasks usually provides satisfiable performance on the current task, it also drastically degrades the model's original performance on previous tasks. Such phenomenon is called *catastrophic forgetting* [1], [2] and is the major challenge in continual learning.

To resolve this issue, many approaches have been proposed in recent years [3], [4], [5], [6], [7], [8] to mitigate or avoid forgetting on previous tasks while learning the current task. In this paper, we introduce an architecture-expansion approach for unforgetting continual learning, which is based on the conditional parametric convolution (CondConv) structure proposed in [9]. As the CondConv operation can acquire capacity of the model without adding much computational complexity, it is potentially highly suitable for multitask learning. In this paper, we utilize this technique for sequential multitasking and propose a new method for continual learning.

The proposed approach has several characteristics. First, it can avoid forgetting based on the additional capacity acquired for new tasks. The function mappings learned for predicting the labels of previous tasks remain exactly unchanged for the sequential tasks, without much influencing the time complexity for the model inference. Second, it can exploit the model weights learned for previous tasks to improve the current task by taking advantage of the CondConv structure for input adaptive model combination. In the experiments, we compare our approach with several state-of-the-art methods (such as [10], [11], [12], [13], [5], [14]) and show that our approach obtains more favorable results, in the cases with or without task boundaries. Our source code is available at https://github.com/ivclab/CondConvContinual.

The rest of this paper is organized as follows. In Section II, we give a survey of the related works. In Section III, we review the principle of conditionally parametric convolution (CondConv). In Section IV, we introduce the proposed approach of this work. Experimental results are presented in Section V. Finally, conclusions are given in Section VI.

II. Related Work

In this section, we review the related work including continual lifelong learning and conditional structures.

A. Continual Lifelong Learning

Existing continual learning literature can be divided into three categories, regularization, memory replay and adaptive architectures, according to the way on how they overcome the problem of catastrophic forgetting.

Regularization-based methods [3], [4], [15], [16] limit the network from updating too much on the current task to avoid deviating too much from previous tasks. Although the methods lessen the catastrophic forgetting, they usually fail to solve the forgetting problem well and perform non-favorably when the sequence of tasks is long.

In contrast, memory-replay methods [17], [18], [19], [5] store images of previous tasks through a generative model, so that the training data can be re-generated or re-synthesized to train together with the data of future tasks. Although the above methods alleviate the forgetting to a certain extent, they cannot guarantee unforgetting of previous tasks because when re-training all of the tasks, the capacity of the original model could be insufficient, and a model growing or adaptation would be required. Besides, retraining all tasks need to take more resources, which could not be affordable in practice for a long sequence of tasks.

To ensure unforgetting, architecture-adaptation methods [6], [7], [20], [21], [8] assign each sub-network of a large network to a task and memorize the previous tasks by keeping their parameters unchanged. In this research direction, ProgressiveNet [10] and CPG [8] expand the feature channels of layers to acquire additional capacity for learning new tasks. Learning to Grow [21] utilizes NAS (Network Architecture Search) to decide suitable structures to expand. However, network expansion in feature channels usually accompanies with increasing the model size and inference time. In particular, computational resources are extremely valuable when we would like to deploy our models to embedding devices. Thus, these methods usually introduce network compression [11], [8] or smaller structures [21] to limit their model capacity and make trade-off between accuracy and inference speed. In this paper, we utilize conditional computation [9] in our convolutional layers to gain the model capacity equal to a new full model but the inference time can also be controlled at the same time.

B. Conditional Structures

Conditional computation aims at expanding the model capacity while keeping the inference speed under control. The major idea is to activate a sub-net of the full network via adaptive routing functions for different input samples.

However, routing functions are usually discrete functions that is extremely challenging to learn. SkipNet [22] and BlockDrop [23] use reinforcement learning to skip blocks within a network for various input examples. To enforce the idea that similar examples should be processed by similar sub-nets, Gross et al. [24] introduce an two-stage learning approach by first clustering images into groups and then feeding them into different branches of the network. Unlike them, CondConv [9] utilize soft routing functions by applying fully connected layers on the global feature maps followed by sigmoid activations. Therefore, CondConv is much easier to optimize via the standard gradient descent algorithm.

In continual learning, data from different tasks can essentially have different distributions, which makes them suitable to be processed by different sub-nets like conditional computation; thus maintain the inference speed at the same time. In this paper, we utilize CondConv as our conditional structure due to its simplicity in optimization and implementation.

III. Conditional Convolution

Given an input image $\mathbf{x}$, a conditional convolution (CondConv) layer [9] consists of n kernels and computes its output feature maps as follows:

$$\text{CondConv}_n(\mathbf{x}) = \sigma((\alpha_1\mathbf{W}_1 + \alpha_2\mathbf{W}_2 + \cdots + \alpha_n\mathbf{W}_n) * \mathbf{x}), \tag{1}$$

where $\mathbf{W}_i$ are the kernels conditionally combined with parameters α_i, $*$ denotes the convolution operation, and $\sigma(\cdot)$ denotes the activation function.

In Eq. 1, if the combination coefficients $\{\alpha_{1:n}\}$ are fixed, the linear combination yields an operation equivalent to only a single kernel, which provides the weight capacity no larger than the application of a single kernel. In CondConv, the coefficients $\{\alpha_{1:n}\}$ are non-fixed but adaptive to the input $\mathbf{x}$. Each α_i $(i = 1 \cdots n)$ is an input-dependent scalar computed via a global average polling followed by a fully-connected layer as follows:

$$\alpha_i = \text{Sigmoid}(\text{GlobalAveragePool}(\mathbf{x})\mathbf{R}_i + \mathbf{b}_i), \tag{2}$$

where $\mathbf{R}_i$ and $\mathbf{b}_i$ are learnable routing weight and bias that adaptively assign conditional parameters based on the global feature maps. Because $\{\alpha_{1:n}\}$ are input-dependent, CondConv is thus a non-linear operation via the conditional combination.

CondConv introduces n convolutional kernels to increase model weight capacity. As multiple kernels are used, the model capacity of a CondConv layer is larger than that of the ordinary convolution layer. Therefore, CondConv offers a more abundant representation power than the standard convolution. Although the representation space is large, CondConv is still more efficient to compute than a convolution layer containing multiple parallel kernels. Since the nonlinear activation function is executed after the linear kernels in Eq. 1, the kernels can be combined into a single kernel at first in the forward mapping. Thus, the convolution is only computed once (instead of n times). The output size of a CondConv layer is maintained to be the same as that of a single-kernel convolution. Compared to the convolution operation, the global average pooling followed by a routing weight combination in Eq. 2 consumes much less computational complexity.

Although the model size is n-times larger than a single-column model, running the model in existing frameworks (such as PyTorch [25] or TensorFlow [26]) remains efficient and manageable. It has been shown in [9] that the computation is still effectively feasible even when a large number of kernels are combined, such as $n = 32$.

IV. Sequential Multitasking Continual Learning

In this section, we first formulate the problems in Section IV-A, and then present our approach in Section IV-B.

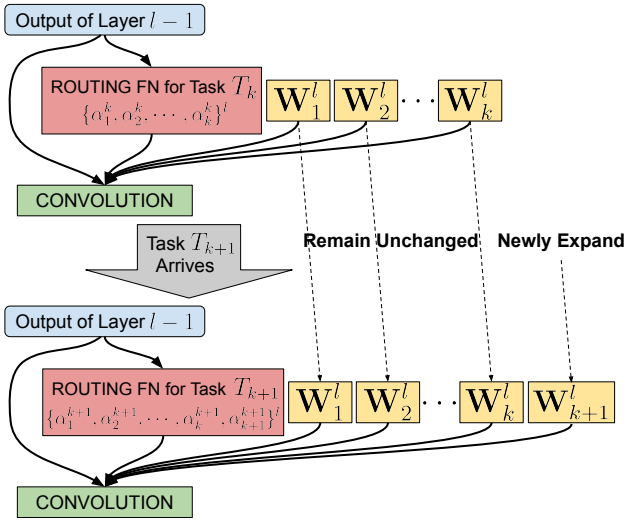

Fig. 1. An illustration of using CondConv structures for sequential multitasking. When the next task T_{k+1} arrives, for each CondConv layer, we expand a new kernel $\mathbf{W}_{k+1}^l$ and initialize a new set of task-specific routing parameters $\{\alpha_{1:(k+1)}^{k+1}\}^l$. During the learning of task T_{k+1}, the task-specific routing parameters $\{\alpha_{1:(k+1)}^{k+1}\}^l$ and the newly added kernel $\mathbf{W}_{k+1}^l$ are updated while other kernels $\{\mathbf{W}_1^l, \mathbf{W}_2^l, \cdots, \mathbf{W}_k^l\}$ remain unchanged. Therefore, our method completely avoids forgetting because the function mappings are exactly preserved.

A. Problem Statement

Assume that there are N sequentially coming tasks $\mathcal{T} = \{T_1, T_2, \cdots, T_N\}$. Each task T_k has its corresponding dataset $\mathcal{D}_k = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{n_k}$. For the task T_k, we can only access its dataset $\mathcal{D}_k$, and the dataset becomes unavailable once we start to learn the next task T_{k+1}. Under this setting, our objective is to build a model M capable of performing all the N tasks.

Specifically, given an input image $\mathbf{x}_i$ along with its belonging task k, $M(\mathbf{x}_i, k)$ outputs the predictions for $\mathbf{x}_i$ under task k. Such a sequential multitasking setting is referred to as continual learning *with task boundary*. Another more challenging setting only inputs the image $\mathbf{x}_i$ and $M(\mathbf{x}_i)$ outputs its prediction in the absence of the task-index information (i.e. *without task boundary*). In this paper, we focus on solving the case with task boundary. However, our approach can also be extend to the case without task boundary via a simple modification (as demonstrated in our experiments).

B. Conditional Convolution on Sequential Multitasking

To incorporate conditional convolution structures in sequential multitasking learning, we employ a deep neural network model containing a backbone part and a predictor (classifier) part. We use CondConv layers in the backbone part to extract features using conditionally combined kernels learned from different tasks. Specifically, consider a deep model M_k for the task T_k. It contains a backbone network $\mathcal{B}$ and a task-specific classifier C_k to perform predictions. In our approach, each CondConv layer consists of k kernels after learning the task T_k. Let L be the number of CondConv layers in $\mathcal{B}$. We denote $\mathbf{W}_i^l$ as the i-th kernel of the l-th layer, $\mathcal{W}_i = \{\mathbf{W}_i^1, \cdots, \mathbf{W}_i^L\}$, and use $\mathcal{W}_{1:k}$ to represent $\bigcup_{i=1}^k \mathcal{W}_i$.

As different tasks may have images from different distributions, we employ task-specific routing parameters $\{\alpha_{1:k}^k\}^l$ ($l = 1 \cdots L$) for the task T_k so that they can combine the kernels based on the feature maps more accurately. Let $\mathcal{A}_k = \bigcup_{l=1}^L \{\alpha_{1:k}^k\}^l$ be the set of all task-specific routing parameters for task T_k in all CondConv layers. Once the next task T_{k+1} is coming, we allocate a new classifier C_{k+1} and routing parameters $\{\alpha_{1:(k+1)}^{k+1}\}^l$ and expand a new kernel $\mathbf{W}_{k+1}^l$ for each layer, $l = 1 \cdots L$. The prediction for T_{k+1} given an image $\mathbf{x}$ is then computed as

$$M_{k+1}(\mathbf{x}) = C_{k+1}(\mathcal{B}(\mathbf{x}; \mathcal{W}_{1:(k+1)}, \mathcal{A}_{k+1})), \tag{3}$$

where all the previous kernels $\mathcal{W}_{1:k}$ are co-used and conditionally combined with the newly added kernel, $\mathcal{W}_{k+1}$, via $\{\alpha_i^{k+1} | i = 1 \cdots (k+1)\}^l$ in layers $l = 1 \cdots L$. More specifically, during the training of task T_{k+1}, we aim at solving the optimization problem defined as follows:

$$\underset{\mathcal{W}_{k+1}, \mathcal{A}_{k+1}, C_{k+1}}{\arg\min} \sum_{(\mathbf{x}, \mathbf{y}) \in \mathcal{D}_{k+1}} l_{k+1}(M_{k+1}(\mathbf{x}), \mathbf{y}), \tag{4}$$

where l_{k+1} is the loss function of task T_{k+1}.

Note that only the newly expanded kernels $\mathcal{W}_{k+1}$ is updated in Eq. 4, whereas $\mathcal{W}_{1:k}$ remain fixed, as shown in Fig 1. Hence, the models $M_{1:k}$ (and the associated deep function mappings) learned for the old tasks keep unchanged. Thus, our method exactly maintains the performance of previously learned tasks and achieves unforgetting continual learning when the task is specified in the inference stage (i.e. with task boundary). In addition, as the previous kernels are co-used, our method can exploit the learned experience of old tasks to facilitate the learning of new tasks under the full expansion of model capacity. Compared to the approaches increasing the columns parallelly for continual learning (such as ProgressiveNet [10]), our method leverages the conditional structure in an incremental manner and exploits additional capacities with computational efficiency.

We refer to our method as the *CondConv-Continual* approach. A series of experiments are conducted to verify its performance in the next section.

V. Experiments

To demonstrate the effectiveness of the proposed continual learning with CondConv, we perform the experiments under three different settings as follows.

Twenty Tasks of CIFAR-100: In the first setting, we use CIFAR-100 dataset [27] and separate it into 20 independent tasks. We show that our approach is competitive against state-of-the-art methods, such as CPG [8], when scaling up to twenty tasks. CPG is an unforgetting continual learning approach assuming the existence of task boundary. We use the CPG code released by the authors for the implementation[1]

[1]https://github.com/ivclab/CPG

TABLE I
THE RESULTS OF CIFAR-100 TWENTY TASKS

	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}	T_{12}	T_{13}	T_{14}	T_{15}	T_{16}	T_{17}	T_{18}	T_{19}	T_{20}	Avg.
Scratch	65.4	76.0	75.0	78.0	83.0	77.8	79.2	81.8	82.2	86.8	83.4	79.4	84.2	78.4	48.0	68.2	63.8	70.2	85.8	88.6	76.8
Finetuning	65.4	75.4	74.5	74.7	81.2	77.2	73.2	80.4	81.0	84.8	86.0	76.6	81.6	77.5	46.6	67.2	63.2	69.7	84.4	88.6	75.5
CPG[8]	63.6	76.8	76.2	74.4	83.0	79.6	79.2	82.2	80.6	87.0	85.2	77.6	82.4	81.6	51.0	67.8	68.4	67.2	85.8	90.2	77.0
Ours	65.4	77.4	75.2	78.4	81.4	77.6	77.6	82.2	82.2	86.8	85.4	77.8	83.8	80.2	50.6	71.0	67.8	69.8	86.8	91.2	77.4

TABLE II
STATISTICS OF THE FINE-GRAINED DATASETS

Dataset	#Train	#Eval	#Classes
ImageNet	1,281,167	50,000	1,000
CUBS	5,994	5,794	200
Stanford Cars	8,144	8,041	196
Flowers	2,040	6,149	102
WikiArt	42,129	10,628	195
Sketch	16,000	4,000	250

in the experiment and compare it with our approach in terms of both the prediction accuracy and inference speed. In this setting, the sizes of training datasets among tasks are similar.

Fine-grained Image Classification Tasks: In the second experiment, we follow the setting in PackNet [11] and PiggyBack [12] on six fine-grained datasets, including the entire ImageNet [28], CUBS [29], Stanford Cars [30], Flowers [31], Wikiart [32] and Sketch [33]. ImageNet serves as the first task, followed by five relatively smaller-scale datasets. This experiment aims to show the performance of our approach based on a large-scale first task. We compare our method with PackNet [11], PiggyBack [12], and CPG [8].

Five Tasks of ImageNet-50: The last experiment follows the setting in the recent studies of continual learning [5] and [14]. The ImageNet-50 dataset is split into five 10-way classification tasks, and the experiments focus on continual learning *without task boundary*. In this experiment, we extend our approach to a task-boundary-free version simply via a random-weight strategy. As the code of [14] are not available yet, we compare our approach with them based on the results shown in [14], and demonstrate that our approach is highly competitive and can achieve better performance.

A. CIFAR-100 Twenty Tasks Results

We divide CIFAR-100 into 20 tasks based on their super classes. Each task is a 5-way classification problem. In each task, we randomly split the dataset so that each category contains 400 images for training, 100 images for validation and 100 images for testing. A 4-layer convolutional neural network (CNN) is adopted as our backbone. Each layer outputs 64-channel feature maps followed by batch normalization, ReLU and max pooling layers.

The results are shown in Table I. In this table, 'Scratch' means that each task is trained independently from scratch using the 4-layer CNN. 'Finetuning' is also referred to as the case where each task in the sequence is learned independently, but the results are obtained by fine-tuning from the CNN models of all previous tasks; the average fine-tuning performance is reported. 'CPG' means that we run the codes released from the authors of CPG [8] with the 4-layer CNN model to obtain the results. As CPG repeats a loop of weight-picking, weight-pruning and model-growing for unforgetting continual learning, the model will be expanded in the learning process. In CPG, the model expansion is performed by increasing the number of both the input and output nodes of a layer, instead of adding a parallel column. Additional binary masks have to be recorded in CPG (one mask per task) to choose the fragmental weights associated with each task in the runtime for inference. In the following, we report the comparison results on both the classification accuracy and inference time.

As shown in Table I, the performance of 'Scratch' is better than that of 'Finetuning' on the classification accuracy (%), indicating that there exists negative transfer among tasks. Nevertheless, our method that conditionally combines kernels trained on different tasks can still extract suitable features based on input samples, achieving more favorable results. In addition, our approach also performs better than the state-of-the-art approach, CPG, on unforgetting continual learning.

As for the model size (in terms of the number of weights), because the 4-layer CNN model contains less redundancy to be exploited than a large model, CPG grows about 16.34 times larger. Our model grows 20 times larger; however, as depicted in Section III, the output size of each layer is the same as that of a single-column convolution layer and thus our model remains efficient. We further measure the execution time by running 10,000 forward passes using batch size 64 on a NVIDIA Tesla V100 GPU, and found that CPG takes 78.99 seconds while our model takes only 59.10 seconds, which is 33% faster. This result demonstrates that our model is more computationally efficient and thus can acquire better performance by exploiting larger model capacity.

B. Fine-grained Six Tasks Results

In this experiment, we follow the standard setting used in Piggyback [12] and CPG [8], where six fine-grained classification datasets are used in a continual learning pipeline. The statistics of these datasets are listed in Table II. In this experiment, the base model used is ResNet-50 [34] for all of the approaches compared.

The results are shown in Table III. In this table, 'Finetuning' means that the tasks are fine-tuned from the model of the first task, ImageNet. As ImageNet is a large-scale dataset, it gives a strong pre-trained model for the other tasks. We use the 'Finetuning' result as the baseline for comparison in the table.

TABLE III
THE RESULTS OF FINE-GRAINED SIX TASKS

Dataset	ImageNet	CUBS	Stanford Cars	Flowers	WikiArt	Sketch	Total Gain
Finetuning	-	83.41	92.85	97.12	74.19	79.7	-
Scratch	76.16	42.03	62.94	46.24	55.12	69.48	−151.46
ProgressiveNet[10]	76.16	78.94	89.21	93.41	74.94	76.35	−14.42
PackNet[11]	76.16	81.59	89.62	94.77	71.33	79.91	−10.05
Piggyback[12]	76.16	81.59	89.62	94.77	71.33	79.91	−10.05
CPG[8]	75.81	83.59	92.80	96.62	77.15	80.33	+2.87
Ours	76.16	84.26	92.61	97.16	78.32	80.77	+5.85

TABLE IV
THE RESULTS OF IMAGENET-50 FIVE TASKS (WITHOUT TASK BOUNDARY)

Method	Accuracy
DGMw[5]	17.82
DGMa[5]	15.16
CCGN[14]	35.24
Ours	61.32

As can be seen, unlike the previous case of 20 even tasks, the 'Finetuning' results considerably outperform the 'Scratch' results in this experiment.

In our approach, the six tasks are continually learned and thus resulting in six kernels in each CondConv layer in our model. Similarly, for each of the second to last tasks, instead of conditionally combining with all previous tasks, we combine them with the first task (i.e. ImageNet) only. Since ImageNet is a large-scale dataset, it enables the learned kernels to extract powerful features that can be easily transferred to other tasks. As shown in Table III, our model surpasses ProgressiveNet [10], PackNet [11] and Piggyback [12] by a large margin. Our model even manages to outperform CPG [8] by gaining an additional 3% accuracy. As each task only combines with ImageNet, we only need to load the corresponding kernels in CondConv layers instead of all kernels and thus greatly decrease the on-line memory usages.

This also provides another direction of improvement of our method that we can consider only conditional combination with the most related tasks to further boost the inference speed. In this setting, we have the knowledge that ImageNet provides easy-to-transfer features by observing the performance differences between 'Scratch' and 'Finetuning', and also from the other approaches such as PiggyBack and PackNet. However, automatically and efficiently selecting related tasks still remains challenging, and we leave this as our future work.

C. ImageNet-50 Five Tasks Results (Without Task Boundary)

In this experiment, we follow the setting in [14] and [5], which breaks the task boundary (i.e., assumes that no task information is given) in the inference stage. A subset of ImageNet [28] is constructed by randomly sampling 50 classes, and we split them into five tasks so that each task is a 10-way classification problem. In each category, the training and validation images of ILSVRC-2012 dataset are used for training and evaluation, respectively.

Note that our approach is an unforgetting continual learning that assumes the existence of task boundary, and we aim to break the task boundary without retraining the networks. Specifically, once a new task T_k is coming and after learning its model M_k using our approach, we simply use this model to predict not only the labels of the task T_k, but the labels of all tasks $T_{1:N}$, so as to avoid requiring the task information in the inference time. Because the labels of tasks $T_{\{1:N\}\setminus\{k\}}$ are newly added in the final classification layer of our model M_k, we merely set the corresponding new weights associated with the newly added labels as random numbers. As an input image can be predicted by N models, the final prediction is made by selecting the class with the largest score among the N predictions. Such a easy-to-implement strategy is a post processing of our models, and is tested on the case without task boundary.

Through the simple modification as depicted above, our method outperforms other methods significantly on breaking the task boundary, as shown in Table IV. The idea behind this simple strategy is that images of the k-th tasks have different distributions than images of the others. The model of k-th task predicts peak scores if input images come from a distribution similar to the training distribution; otherwise it tends to predict uniformly distributed scores. Therefore, selecting the class with the largest score still helps, even when there exists un-trained weights of new classes in each task. Leveraging on a strong unforgetting model, the results reveal that a simple modification can yield favorable performance for the case without task boundary. This technique is essentially a post processing of our predictions and can be regarded as a strong baseline for continual learning without task boundary.

VI. CONCLUSION AND FUTURE WORK

In this paper, we introduce a new approach, CondConv-Continual, which can achieve unforgetting continual learning for multiple tasks. Our approach leverages the celebrated structure, CondConv, which can provide additional capacities for a deep CNN model to enhance the performance with much less influence on the computational complexity. Experimental results show that our approach is highly competitive and performs better than existing ones on unforgetting continual learning with task boundary. Besides, when a simple random-weight strategy is applied to our model, the performance is more favorable than the recent approaches without task boundary. The results reveal that CondConv-Continual has a high potential to be used for continual learning.

In the future, we plan to study the relationships among the sequential tasks so that we can pick the relevant kernels from previous tasks to simplify the model.

ACKNOWLEDGMENT

The authors gratefully acknowledge the financial support from the Ministry of Science and Technology, Taiwan (MOST 109-2634-F-001-009) and National Center for

High-performance Computing (NCHC) of National Applied Research Laboratories (NARLabs) in Taiwan for providing computational and storage resources.

REFERENCES

[1] J. L. McClelland, B. L. McNaughton, and R. C. O'reilly, "Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory." *Psychological review*, 1995.

[2] B. Pfulb and A. Gepperth, "A comprehensive, application-oriented study of catastrophic forgetting in dnns," in *Proceedings of the International Conference on Learning Representations*, 2019.

[3] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the national academy of sciences*, 2017.

[4] A. Chaudhry, P. K. Dokania, T. Ajanthan, and P. H. Torr, "Riemannian walk for incremental learning: Understanding forgetting and intransigence," in *Proceedings of the European Conference on Computer Vision*, 2018.

[5] O. Ostapenko, M. Puscas, T. Klein, P. Jahnichen, and M. Nabi, "Learning to remember: A synaptic plasticity driven framework for continual learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[6] A. Rosenfeld and J. K. Tsotsos, "Incremental learning through deep adaptation," *IEEE transactions on pattern analysis and machine intelligence, Early Access*, 2018.

[7] G. I. Parisi, X. Ji, and S. Wermter, "On the role of neurogenesis in overcoming catastrophic forgetting," in *Proceedings of NeurIPS Workshop on Continual Learning*, 2018.

[8] S. C. Y. Hung, C.-H. Tu, C.-E. Wu, C.-H. Chen, Y.-M. Chan, and C.-S. Chen, "Compacting, picking and growing for unforgetting continual learning," in *Proceedings of Advances in Neural Information Processing Systems*, 2019.

[9] B. Yang, G. Bender, Q. V. Le, and J. Ngiam, "Condconv: Conditionally parameterized convolutions for efficient inference," in *Proceedings of Advances in Neural Information Processing Systems*, 2019.

[10] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks," *arXiv*, 2016.

[11] A. Mallya and S. Lazebnik, "Packnet: Adding multiple tasks to a single network by iterative pruning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[12] A. Mallya, D. Davis, and S. Lazebnik, "Piggyback: Adapting a single network to multiple tasks by learning to mask weights," in *Proceedings of the European Conference on Computer Vision*, 2018.

[13] C.-Y. Hung, C.-H. Tu, C.-E. Wu, C.-H. Chen, Y.-M. Chan, and C.-S. Chen, "Compacting, picking and growing for unforgetting continual learning," in *Proceedings of Advances in Neural Information Processing Systems*, 2019.

[14] D. Abati, J. Tomczak, T. Blankevoort, S. Calderara, R. Cucchiara, and B. E. Bejnordi, "Conditional channel gated networks for task-aware continual learning," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[15] P. Dhar, R. V. Singh, K.-C. Peng, Z. Wu, and R. Chellappa, "Learning without memorizing," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[16] J. Schwarz, W. Czarnecki, J. Luketina, A. Grabska-Barwinska, Y. W. Teh, R. Pascanu, and R. Hadsell, "Progress & compress: A scalable framework for continual learning," in *Proceedings of the International Conference on Machine Learning*, 2018.

[17] H. Shin, J. K. Lee, J. Kim, and J. Kim, "Continual learning with deep generative replay," in *Proceedings of Advances in Neural Information Processing System*, 2017.

[18] M. Riemer, T. Klinger, D. Bouneffouf, and M. Franceschini, "Scalable recollections for continual lifelong learning," in *Proceedings of AAAI Conference on Artificial Intelligence*, 2019.

[19] M. Riemer, I. Cases, R. Ajemian, M. Liu, I. Rish, Y. Tu, and G. Tesauro, "Learning to learn without forgetting by maximizing transfer and minimizing interference," in *Proceedings of the International Conference on Learning Representations*, 2019.

[20] S. C. Hung, J.-H. Lee, T. S. Wan, C.-H. Chen, Y.-M. Chan, and C.-S. Chen, "Increasingly packing multiple facial-informatics modules in a unified deep-learning model via lifelong learning," in *Proceedings of International Conference on Multimedia Retrieval*, 2019.

[21] X. Li, Y. Zhou, T. Wu, R. Socher, and C. Xiong, "Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting," in *Proceedings of the International Conference on Machine Learning*, 2019.

[22] X. Wang, F. Yu, Z.-Y. Dou, T. Darrell, and J. E. Gonzalez, "Skipnet: Learning dynamic routing in convolutional networks," in *Proceedings of the European Conference on Computer Vision*, 2018.

[23] Z. Wu, T. Nagarajan, A. Kumar, S. Rennie, L. S. Davis, K. Grauman, and R. Feris, "Blockdrop: Dynamic inference paths in residual networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[24] S. Gross, M. Ranzato, and A. Szlam, "Hard mixtures of experts for large scale weakly supervised vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[25] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Proceedings of Advances in Neural Information Processing Systems*, 2019.

[26] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, 2016.

[27] A. Krizhevsky, "Learning multiple layers of features from tiny images," Tech. Rep., 2009.

[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[29] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, "The Caltech-UCSD Birds-200-2011 Dataset," California Institute of Technology, Tech. Rep. CNS-TR-2011-001, 2011.

[30] J. Krause, M. Stark, J. Deng, and L. Fei-Fei, "3d object representations for fine-grained categorization," in *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013.

[31] M.-E. Nilsback and A. Zisserman, "Automated flower classification over a large number of classes," in *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008.

[32] B. Saleh and A. Elgammal, "Large-scale classification of fine-art paintings: Learning the right metric on the right feature," in *ICDMW*, 2015.

[33] M. Eitz, J. Hays, and M. Alexa, "How do humans sketch objects?" *ACM Trans. Graph.*, vol. 31, no. 4, 2012.

[34] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.