Supplementary Material – Learn to Grow: A Continual Structure Learning Framework for Overcoming Catastrophic Forgetting

Xilai Li [*] [†] [1] **Yingbo Zhou** [*] [2] **Tianfu Wu** [1] **Richard Socher** [2] **Caiming Xiong** [2]

1. Additional Experimental Details for permuted MNIST

For all MNIST experiment, we use fully connected layer with three hidden layer, each with 300 hidden units, and one shared output layer for our method. For all other methods except pathnet and progressive net we used 3000 units in the first layer and 300 for the rest. For pathnet, each module in the first layer has 300 units, and the result layers has 30 units. We use 16 modules per layer, and 5 layers for pathnet, and restrict each mutation to only use 3 modules besides the output layer. For progressive net, the first layer has 300 units for each task, and the rest layers each has 30 units. Therefore, all competitive methods are having more or the same number of parameters as our methods.

For variational continual learning (VCL Nguyen et al., 2018), we used the official implementation at https://github.com/nvcuong/variational-continual-learning. For fair comparison with other methods, we set the coreset size to zero for VCL.

For (Shin et al., 2017) we used implementation from https://github.com/kuc2477/pytorch-deep-generative-replay. We tried various hyper-parameter settings, however, we are unable to get reasonable results on permutated MNIST. Performance was reasonable when the number of tasks is within five (average performance at around 96%). When number of tasks go beyond five, performance drops on previous tasks is quite significant. Reaching 60%

For DEN we use the official implementation at https://github.com/jaehong-yoon93/DEN, and we used Serrà et al. (2018) implementation of HAT, EWC, and IMM at https://github.com/joansj/hat. We used our own implemention for for Progressive Network and PathNet. All methods are trained using the same permutations and same subset of training data.

2. Additional Experimental Details for Split CIFAR-100

For all CIFAR-100 experiment, we use an Alexnet like structure. It contains three convolution and max pooling layers followed by two fully connected layers. The convolution layers are of size (4,4), (3,3) and (2,2) with 64, 128 and 256 filters, respectively. All convolution layers are followed by max pooling layer of size (2,2) and rectified linear activations. The two fully connected layers each have 2048 hidden units.

3. Additional Experiments on Visual Decathlon Dataset

In the multi-task continual learning experiments, the 10 tasks was trained in a random sequence except the first task was fixed to be ImageNet. This is just for fair comparison with other works such as Rebuffi et al. (2017) and Mallya & Lazebnik (2018), they are all using a light weight module to adapt ImageNet pretrained model to other of the 9 tasks. In real case, the tasks can come in any order, thus our framework would be much more flexible. As the tasks are trained in sequence, a super model is maintained that all the newly created weights and task-specific layers are stored. In this ResNet-26 model, all the Batch Normalization (BN) layers are treated as task-specific, which means each task has its own sets of BNs. Here, we fixed the weight during retraining when "reuse" is selected in the search phase. This means that the results of previous tasks would not be affected, i.e. no forgetting. We leave the evaluation of forgetting in the context of VDD dataset as future work.

In Table 1, we compare the results using our approach with other baselines. "Individual" means that each task is trained individually and weights are initialized randomly. "Classifier" means that only the last layer classifier could be tuned while the former 25 layers are transfer from ImageNet

[*]Equal contribution [1]Department of Electrical and Computer Engineering and the Visual Narrative Initiative, North Carolina State University, NC, USA. [2]Salesforce Research, Palo Alto, CA, USA.. Correspondence to: C. Xiong <cxiong@salesforce.com>, T. Wu <tianfu_wu@ncsu.edu>.

[†]Part of the work was done when X. Li was an intern at Salesforce Research in Summer 2018.

Proceedings of the 36th International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019. Copyright 2019 by the author(s).

		ImNet	C100	SVHN	UCF	OGlt	GTSR	DPed	Flwr	Airc.	DTD	Tot.
$\beta = 0.01$	acc	69.84	78.50	95.33	72.50	86.41	99.97	99.76	66.01	51.37	50.05	76.97
	#params	6.07	0.15	2.74	2.28	6.17	3.59	1.02	0.19	4.15	0.13	26.49
$\beta = 0.1$	acc	69.84	79.59	95.28	72.03	86.60	99.72	99.52	71.27	53.01	49.89	77.68
	#params	6.07	0.34	1.19	1.32	3.19	0.02	0.27	0.16	1.86	0.04	14.46
$\beta = 1.0$	acc	69.84	78.00	93.40	63.83	84.30	99.78	99.01	65.77	39.27	48.77	74.20
	# params	6.07	0.04	0.03	0.12	0.66	0.02	0.01	0.02	0.35	0.02	7.34

Table 1. Comparison of (top-1) validation classification accuracy (%) and total model size (in Million) on Visual Domain Decathlon dataset with parameter loss factor β of 0.01, 0.1, 1.0.

pretrained model and kept fixed during training. In this case, each task only adds a task-specific classifier and BNs, thus the overall model size is small. ”Adapter” add a 1x1 conv layer besides each 3x3 conv layer, and the outputs will be added before proceed to the next layer. Due to the lightweight 1x1 conv layer, each task will add approximately 1/9 of the whole model size. As shown in table 1, the results achieved by our framework is better than other baselines and the total model size is similar to ”Adapter” case. We can see that our approach gives best results in five out of nine tasks. Especially in task with small data size, e.g. VGG-Flowers and Aircraft, our method outperforms other baselines by a large margin.

Due to each choice has different parameter cost, we add a parameter loss function to L_{val} to penalize the choices that cost additional parameters. And the value of the loss function is proportional to the product of the additional parameter size and its corresponding weight value α_c^l. In table 2, we test it with three different scaling factor β of the parameter loss. We found that the scaling factor β can control the additional parameter size for each task. And we find that $\beta = 0.1$ gives the best average accuracy and can control the total model size approximate $2.3\times$ compared with the original model size.

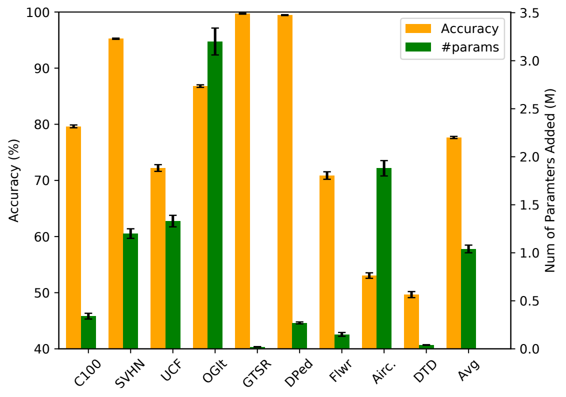

Figure 1. Statistics for performance and number of added parameters for each task of VDD dataset with 4 random task ordering. The first task is kept with ImageNet due to its large size and long training time. We observed that both accuracy and parameter growth are robust to different task ordering.

References

Mallya, A. and Lazebnik, S. Piggyback: Adding multiple tasks to a single, fixed network by learning to mask. *arXiv preprint arXiv:1801.06519*, 2018.

Nguyen, C. V., Li, Y., Bui, T. D., and Turner, R. E. Variational continual learning. In *International Conference on Learning Representations*, 2018.

Rebuffi, S.-A., Bilen, H., and Vedaldi, A. Learning multiple visual domains with residual adapters. In *Advances in Neural Information Processing Systems*, pp. 506–516, 2017.

Serrà, J., Surís, D., Miron, M., and Karatzoglou, A. Overcoming catastrophic forgetting with hard attention to the task. *arXiv preprint arXiv:1801.01423*, 2018.

Shin, H., Lee, J. K., Kim, J., and Kim, J. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems*, pp. 2990–2999, 2017.