More Classifiers, Less Forgetting: A Generic Multi-classifier Paradigm for Incremental Learning

Yu Liu[1], Sarah Parisot[2,3], Gregory Slabaugh[2], Xu Jia[2], Ales Leonardis[2], and Tinne Tuytelaars[1]

[1]KU Leuven [2]Huawei Noah's Ark Lab [3]Mila
{firstname.lastname}@kuleuven.be {firstname.lastname}@huawei.com

Content of this supplementary material:

1. Incremental Learning Pipeline of MUC-MAS and MUC-LwF

In the main paper, we introduce how to train MUC with parameter regularization (PR) and activation regularization (AR), respectively. In addition, we present the objective functions for MUC-MAS and MUC-LwF. Due to limited space in the paper, we move the algorithmic pipeline in this supplementary material. Algorithm 1 and Algorithm 2 summarize the incremental learning pipeline for MUC-MAS and MUC-LwF, respectively. MUC-MAS and MUC-LwF employ two different regularization strategies, and one major difference is that MUC-MAS performs a post-training step to estimate the importance weights and stability factors for the current task. However, MUC-LwF has no need to do this post-training step. It is worthy noting that both MUC-MAS and MUC-LwF are generic methods. Here, we use the regularization terms from MAS [1] and LwF [4]. Following these pipelines, other PR and AR methods can be applied to MUC as well.

Algorithm 1: Incremental Learning Pipeline of MUC-MAS

Input: In-distribution data $\{X^t, Y^t\}_{t=1}^T$; Out-of-distribution data X^{out}; Feature extractor F; Main classifiers $\{M^t\}_{t=1}^T$; Side classifiers $\{S_{1:K}^t\}_{t=1}^T$.

for $t = 1; t \leq T$ **do**
- **if** $t == 1$ **then**
  - // Learn the first task
  - Train F and M^1: optimize $\mathcal{L}_{CE}(F, M^1)$, in Eq.1.
  - Freeze F and train $S_{1:K}^1$: optimize $\mathcal{L}_{CE}(F, S_{1:K}^1) + \mathcal{L}_{CD}(F, S_{1:K}^1)$, in Eq.5.
  - Estimate importance weights and stability factors: $\boldsymbol{\alpha}^1, \boldsymbol{\gamma}^1, \boldsymbol{\delta}^1$, in Eq.6,10,7.
- **else**
  - // Learn subsequent tasks
  - Update F and train M^t: optimize $\mathcal{L}_{CE}(F, M^t) + \lambda \mathcal{L}_{REG}^{PR}$, in Eq.1,11.
  - Freeze F and train $S_{1:K}^t$: optimize $\mathcal{L}_{CE}(F, S_{1:K}^t) + \mathcal{L}_{CD}(F, S_{1:K}^t)$, in Eq.5.
  - Estimate importance weights and stability factors: $\boldsymbol{\alpha}^t, \boldsymbol{\gamma}^t, \boldsymbol{\delta}^t$, in Eq.6,10,7.
- **end**

end

Algorithm 2: Incremental Learning Pipeline of MUC-LwF

Input: In-distribution data $\{X^t, Y^t\}_{t=1}^T$; Out-of-distribution data X^{out}; Feature extractor F; Main classifiers $\{M^t\}_{t=1}^T$; Side classifiers $\{S_{1:K}^t\}_{t=1}^T$.

for $t = 1; t \leq T$ **do**
- **if** $t == 1$ **then**
  - // Learn the first task
  - Train F and M^1: optimize $\mathcal{L}_{CE}(F, M^1)$, in Eq.1.
  - Freeze F and train $S_{1:K}^1$: optimize $\mathcal{L}_{CE}(F, S_{1:K}^1) + \mathcal{L}_{CD}(F, S_{1:K}^1)$, in Eq.5.
- **else**
  - // Learn subsequent tasks
  - Update F and train M^t: optimize $\mathcal{L}_{CE}(F, M^t) + \lambda \mathcal{L}_{REG}^{AR}$, in Eq.1,14.
  - Freeze F and train $S_{1:K}^t$: optimize $\mathcal{L}_{CE}(F, S_{1:K}^t) + \mathcal{L}_{CD}(F, S_{1:K}^t)$, in Eq.5.
- **end**

end

2. Experimental Details and Evaluation Metrics

We employ a "single-head" evaluation that does not require task identities. The decision space contains all seen classes learned from a sequence of tasks. This "single-head" evaluation is widely-used for class-incremental learning, despite that it is more challenging than the "multi-head" evaluation [3]. In addition to the top-1 accuracy, we evaluate the methods with the forgetting ratio introduced in [6] as follows:

$$\rho^{\leq t} = \frac{1}{t}\sum_{i=1}^{t}\rho^{i\leq t}, \tag{1}$$

$$\rho^{i\leq t} = \frac{A^{i\leq t} - A_R^t}{A_J^{i\leq t} - A_R^t} - 1, \tag{2}$$

where $A^{i\leq t}$ is the accuracy of task i after learning task t, A_R^t is the accuracy through random guess, and $A_J^{i\leq t}$ is the upper-bound accuracy with joint training. Note that, the forgetting ratio belongs to $[-1, 0]$ and less negative ratio means less forgetting.

3. Analysis of Hyper-parameter λ in MUC-MAS

As discussed in Table 4 of the main paper, we study the effect of the hyper-parameter λ. Despite the fact that the accuracy performance when λ=0.02 is better, we find the accuracy of new tasks becomes even lower than that of old tasks in the end. To be specific, we show the accuracy on new and old tasks in Fig. 1. By increasing λ from 0.01 to 0.02, MUC-MAS trades new tasks accuracy for higher accuracy of old tasks. The accuracy of new tasks will be lower than that of old tasks when $\lambda = 0.02$, This behavior is opposite to some practical scenarios. For incremental learning, it is important to retain new tasks accuracy instead of purely pushing old tasks accuracy with larger λ. Hence, we choose to use $\lambda = 0.01$ in the experiments.

4. Histogram Statistics of Stability Factors in MUC-MAS

Although our MUC-MAS is built on the pre-existing importance weight, we expand it with a new stability factor, which is able to quantify how stable the importance weight per parameter is. In Fig. 2, we further provide the statistics of stability factors of all parameters in the feature extractor. Here, we focus on the five tasks from `CIFAR-100` and depict the number of parameters according to the stability factors. We can see that each task has a different histogram statistics with respect to the stability factors. It suggests that each parameter adaptively learns different stability factors for the five tasks. Intuitively, it is consistent with the fact that one certain parameter is stable for one task while it may become unstable for other tasks.

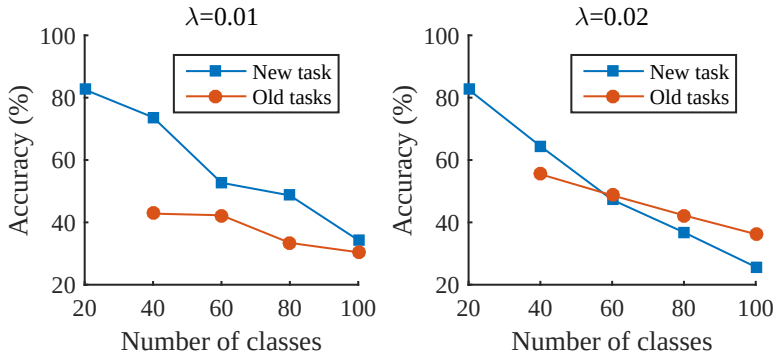

Fig. 1: The accuracy of old and new tasks on `CIFAR-100` when λ=0.01 or 0.02. When $\lambda = 0.02$, the new tasks accuracy begins to be lower than that of old tasks quickly. It is not a desirable behavior in practice. Instead, we choose to set λ=0.01.

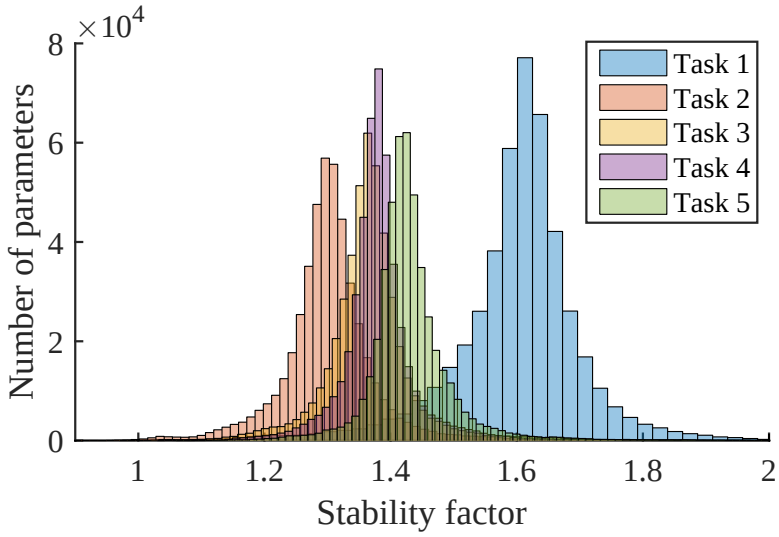

Fig. 2: Histogram statistics of stability factors of network parameters when incrementally learning five tasks in `CIFAR-100`. The tasks demonstrate different statistics distributions. Most parameters learn stability factors ranging between 1.2 and 1.6.

5. Quantitative Analysis for Soft Labels in MUC-LwF

In Fig.7 of the main paper, we has shown the qualitative analysis for the soft labels in MUC-LwF when incrementally learning a few tasks. This test aims to further analyze the soft labels in a quantitative way. Specifically, when learning task t, we feed its data into the old model and capture the soft labels from the main classifier and three side classifiers. Then, we use the L1-norm distance to measure the discrepancy among the soft-labels vectors. Notably, the dimension of the vectors keeps on increasing together with learning more tasks. As reported in Table 1, the side classifiers have a remarkable discrepancy with the main classifier. In particular, the discrepancy increases largely after $t = 2$. This verifies the effectiveness of the classifier discrepancy loss used in MUC.

Table 1: L1-norm distances for the soft labels captured from the main classifier (Main) and three side classifiers (Side-1, Side-2 and Side-3). #Dim indicates the dimension of soft labels. Based on the large distances, we can know that the side classifiers learn different soft labels, compared with the main classifier. This experiment is conducted for the 5-tasks scenario in `CIFAR-100`.

Tasks	#Dim	Side-1 v.s. Main	Side-2 v.s. Main	Side-3 v.s. Main
t=2	20	0.238	0.248	0.238
t=3	40	1.568	0.820	1.160
t=4	60	1.662	1.590	1.398
t=5	80	1.640	1.560	1.408

6. Analysis of Memory Cost and Training Time

In this section, we analyze the complexity for the methods including MUC and the baselines (Table 2). (1) Compared with MAS, MUC-MAS has a slight increase in the number of network parameters while it needs more memory to store the importance weights and stability factors. In addition, MUC-MAS spends more training time to compute those auxiliary information. (2) Like LwF, MUC-LwF does not need extra memory to store auxiliary information. These two methods are comparable in terms of memory and time cost. In one word, MUC spends some extra cost, however, being able to leverage multiple classifiers is its main strength.

7. Training Procedure of Learning with Exemplars

Here, we demonstrate the details about how MUC is trained with access to a fixed budget of exemplars (Sec.4.5 in the main paper). Following iCaRL [5], the budget memory stores $B = 2000$ exemplars for previous tasks. Suppose that the model has learned t tasks and each task contains g classes, we randomly

Table 2: Detailed analysis of memory cost and training time for T=5 tasks on `CIFAR-100`. Both MUC-MAS and MUC-LwF use K=3 side classifiers.

Method	Network parameters	Importance weights	Stability factors	Total memory cost	Total training time
MAS	472.8K	472.8K	-	945.6K	0.92h
MUC-MAS	492.3K	984.6K	492.3K	1969.2K	1.83h
LwF	472.8K	-	-	472.8K	0.67h
MUC-LwF	492.3K	-	-	492.3K	0.85h

select $\lfloor \frac{B}{t \cdot g} \rfloor$ exemplars per old class and store them into the budget memory. Similar with the observation in [2,7], using the herding algorithm [5] to select the exemplars does not show remarkable improvements against random selection. Next, we perform an additional fine-tuning stage after learning the new task $t+1$. We need to select $\lfloor \frac{B}{t \cdot g} \rfloor$ exemplars for each new class, to make a balanced exemplar set between old and new classes. Then we fine-tune the model with the mixed exemplars from old and new classes. After the fine-tuning stage, task $t+1$ will be added into the set of old tasks, and we thereby need to update the number of exemplars per old class to be $\lfloor \frac{B}{(t+1) \cdot g} \rfloor$. Likewise, the fine-tune step will be performed after learning the task $t+2$.

References

1. Aljundi, R., Babiloni, F., Elhoseiny, M., Rohrbach, M., Tuytelaars, T.: Memory aware synapses: Learning what (not) to forget. In: ECCV. pp. 144–161 (2018)
2. Castro, F.M., Marín-Jiménez, M.J., Guil, N., Schmid, C., Alahari, K.: End-to-end incremental learning. In: ECCV. pp. 241–257 (2018)
3. Farquhar, S., Gal, Y.: Towards robust evaluations of continual learning. CoRR **abs/1805.09733** (2018)
4. Li, Z., Hoiem, D.: Learning without forgetting. In: ECCV. pp. 614–629 (2016)
5. Rebuffi, S., Kolesnikov, A., Sperl, G., Lampert, C.H.: icarl: Incremental classifier and representation learning. In: CVPR. pp. 5533–5542 (2017)
6. Serrà, J., Suris, D., Miron, M., Karatzoglou, A.: Overcoming catastrophic forgetting with hard attention to the task. In: ICML. pp. 4555–4564 (2018)
7. Zhang, J., Zhang, J., Ghosh, S., Li, D., Tasci, S., Heck, L.P., Zhang, H., Kuo, C.J.: Class-incremental learning via deep model consolidation. CoRR **abs/1903.07864** (2019)