Orchestrate Latent Expertise: Advancing Online Continual Learning with Multi-Level Supervision and Reverse Self-Distillation

Supplementary Material

8. Implementation Details

Here's the overall algorithm of our proposed method MOSE in a standard OCL training scenario.

Algorithm 1 Training Algorithm of MOSE

Require:
$\mathcal{D} = \{\mathcal{D}_t\}_{t \leq T}$: training dataset
Network F with its experts $\{E_i\}_{i \leq n}$
$\text{Aug}(\cdot)$: augmentation transformation
Ensure: Multi-level supervision of sequential experts and reverse self-distillation in the OCL network
for all task $t \leq T$ **do**
  for all incoming batch: $\mathcal{B}^t \sim \mathcal{D}_t$ **do**
    Memory retrieval: $\mathcal{B}^{\mathcal{M}} \sim \mathcal{M}$, $\mathcal{B} = \mathcal{B}^t \cup \mathcal{B}^{\mathcal{M}}$
    Augmentation: $\tilde{\mathcal{B}} = \mathcal{B} \cup \text{Aug}(\mathcal{B})$
    Calculate $\mathcal{L}_{\text{MLS}}$ and $\mathcal{L}_{\text{RSD}}$ over $\tilde{\mathcal{B}}$
    Update F, $\{E_i\}_{i \leq n}$ with $\mathcal{L}_{\text{MOSE}} = \mathcal{L}_{\text{MLS}} + \mathcal{L}_{\text{RSD}}$
    Memory update: $\mathcal{M} \leftarrow \mathcal{M}, \mathcal{B}^t$
  end for
end for

Buffer	200	500	1000	2000
SD	11.9±0.8	22.0±0.8	30.6±0.7	41.9±0.4
RSD (ours)	**20.2±0.5**	**28.3±0.7**	**35.1±0.4**	**45.1±0.3**

Table 5. **SD *v.s.* RSD.** ACC over Split CIFAR-100 for four different memory buffer sizes.

Model	Time	# Params	#FLOPs	ACC ↑	AF ↓
SCR+$\mathcal{L}_{ce}$	8.8min	11.29M	558.55M	22.3±0.4	45.8±0.2
w/ MLS	17.3min	12.81M	586.78M	31.6±0.8	39.1±1.2
w/ MLS,RSD	17.5min	12.81M	590.64M	35.1±0.4	36.9±0.3

Table 6. **Training Cost of Components.** Recorded for Split CIFAR-100 with $M = 5k$.

8.1. Network Architecture

Our model is based on the architecture of a full-width ResNet18 [25] as mentioned in Sec. 5.1, where the original embedding dimension of feature vector output by the feature extractor f_θ is $d = 512$. The feature maps produced by each block are of different sizes in shape: (channel, height, width). Therefore the feature alignment modules $\{a_{\omega_i}\}_{i=1}^4$ are designed to further encode those feature maps into the same dimension space with as little computational cost as possible. Here we list the details of the necessary components of $\{a_{\omega_i}\}_{i=1}^4$ in Tab. 7.

As for projection heads $\{p_{\psi_i}\}_{i=1}^4$ and $\{g_{\phi_i}\}_{i=1}^4$, we use fully connected layers $\text{Linear}(512, l)$ and $\text{Linear}(512, |\mathcal{C}|)$ to project the feature vector into a new embedding space of dimension $l = 128$ (as SCR [40] and OCM [21]) and the logit space with dimension equal to the number of image classes $|\mathcal{C}|$. Alignment modules and projection heads will all be dropped once the network's continual learning procedure is accomplished. At the testing phase, we only calculate the feature means for each class in the feature space with dimension $d = 512$ and use the NCM classifier [40, 44]. There is no other additional storage cost.

8.2. Evaluation Metrics

We first train all continual learners task-wise with corresponding training samples, then test them with test samples for all classes when training is complete. We use two commonly used metrics Average Accuracy (**ACC**) and Average Forgetting (**AF**) to evaluate their performance on benchmarks following [20, 21, 41, 51, 52]:

$$\begin{aligned} \text{ACC} &= \frac{1}{T}\sum_{t=1}^{T} \text{acc}_{t,T}, \\ \text{AF} &= \frac{1}{T-1}\sum_{t=1}^{T-1}\left(\max_{i \leq T-1} \text{acc}_{t,i} - \text{acc}_{t,T}\right), \end{aligned} \tag{11}$$

where $a_{i,j}$ is the test accuracy of task i when the model has been trained on task from 1 to j, and T is the total number of tasks. ACC represents the final remaining skills for all incrementally learned tasks, which is ultimately the optimization goal of a continual learner, and AF shows how the CL algorithm resists the catastrophic forgetting issue.

9. Additional Ablation Studies

In this section, we present the additional results of the ablation study to discuss the effect of our choices of each algorithm component on the final OCL performance (average task accuracy). We take Split CIFAR-100 with 10 tasks as the benchmark, and all tests are conducted over different setups with memory buffer sizes $M = 1000$, 2000, and 5000 for 15 different random runs.

9.1. RSD Variations

In Sec. 4.2, we use the task-wise final test accuracy ($a_{i,T}$) of each expert with or without the RSD loss within Tab.1. It shows that RSD successfully transfers useful knowledge from different teacher experts $E_{i\leq 3}$ to the final predictor $E_4 = F$. Compared to traditional self-distillation (SD) [72], RSD enhances the final predictor on purpose. We show the difference of ACC between SD and RSD in Tab. 5, indicating the effectiveness of our artificial modification to distillation direction.

We further test RSD with different experts being the student expert who learns from others. Average accuracy and $\mathbb{E}_{i\leq t}a_{i,t}$ and final task-wise test accuracy $a_{t,T}$ of different task t are depicted in Fig. 5 and Fig. 6. They suggest that: (1) choosing a different expert as the student does not significantly affect the MOE performance; (2) with a larger memory buffer size M, variation grows and experts at deeper stages exhibit greater potential of learning from other experts. Detailed results data of $a_{t,T}$ in Tab. 8 make these points more clear.

9.2. Data Augmentation

As described in Sec.5.1, we use a transformation operation combining random horizontal flip, random grayscale, and random resized crop, following SimCLR [11] and OCM [21] (we denote this data augmentation combination as *SimCLR*). Our proposed method MOSE also utilizes the contrastive loss $\mathcal{L}_{scl}$ of SCR [40], where a different data augmentation combination is used: random resized crop, random horizontal flip, color jitter, and random grayscale (we denote it as *SCR*). For situations with both types of augmentation and whether inner flip operation (proposed in OCM [21]) is used to double the training samples, we conducted a contrast experiment whose results are recorded in Fig. 7 (average accuracy and final task-wise accuracy) and Tab. 9 (final task-wise accuracy).

We conclude from the above tests that, the choice of augmentation indeed affects the performance of our method MOSE. The usage of a more complex and diverse augmentation combination does not guarantee better accuracy. For example, the insertion of color jitter or inner flip results in quite different task-wise performance. However, increasing the number of training samples through augmentation is validated to be an effective approach. This type of improvement remains consistent across different memory buffer sizes.

The inner flip operation rotates half of the image vertically. Thus there are in total $2 \times 2 = 4$ different possible flipped versions of one image. On top of that, OCM [21] and OnPro [67] also utilize global rotation and create $4 \times 4 = 16$ augmented versions of each single image. In order not to further increase the number of computations and GPU memory usage of training so that the comparison with other methods is fair, we choose to use inner flip only to double the training samples. Even so, the final performance improvement brought by MOSE is already evident.

9.3. NCM Classifiers

For each expert $\{E_i\}_{i\leq 4}$ we choose to use the NCM classifier to output the final prediction. Compared to the traditional linear output head, the NCM classifier is based on feature representation and is constrained by the memory buffer size. Here we show the results of using these two types of classifiers (along with the MOE version) under our MOSE framework in Fig. 8 (average accuracy and final task-wise accuracy) and Tab. 10 (final task-wise accuracy).

From the above tests, we see that these two types of classifiers present similar inter-task behavior: the last trained task has a significantly larger test accuracy than the previous ones. The representation-based NCM classifier generally outperforms the linear classifier, indicating its superiority within our proposed framework which relies on multi-level feature representation learning.

10. Baseline Codes and Efficiency

For all baselines tested in Sec. 5, the common hyperparameters are fixed to give a fair comparison, including batch size $B = 10$, memory buffer size $B^{\mathcal{M}} = 64$ and random seed 0. For other algorithm-specific setups, we keep the default implementation based on their source codes in Tab. 11 and refer to the training details of OCM [21].

10.1. Running Time Comparison.

We run all of our experiments on an NVIDIA RTX3090-Turbo GPU. Here to display the training efficiency, we record the training time over the Split CIFAR-100 dataset for all OCL baselines including MOSE in Fig. 9. Notice that the introduction of data augmentation also increases the training time. Tab. 6 demonstrates the training cost and computational complexity for components of MOSE.

Our proposed MOSE achieves a relatively efficient result compared to other baselines with a training time of around 15 minutes over the Split CIFAR-100 dataset (50000 images in total). The calculation of expert-wise loss doubles the training time of each image compared to ER [10] and SCR [40] which have the same supervision loss as MOSE.

11. Overfitting-Underfitting Dilemma

To further investigate the multi-level expert design in addressing the proposed overfitting-underfitting dilemma, we record the new task accuracy $a_{t,t}$ for each expert with or without using RSD (take the final expert E_4 as the student), as well as average BOF value of old tasks (see Eq. 5).

Accuracy $a_{t,t}$ represents the performance of learning each new incoming task, and BOF indicates how well the

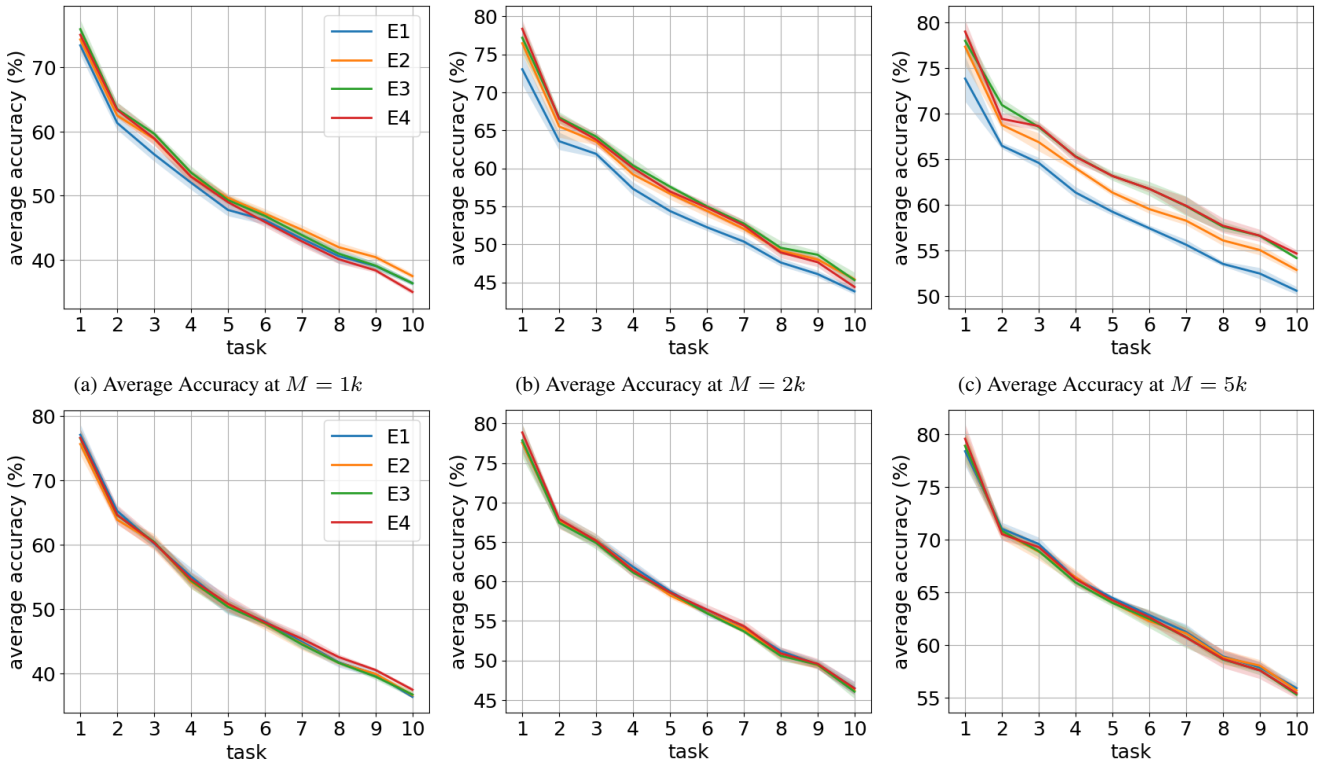
buffer overfitting issue is dealt with (lower is better). All tests are conducted over the dataset Split CIFAR-100 with three different memory sizes for 15 random runs. Based on results in Fig. 10 and Fig. 11, we explain the effectiveness of MOSE in solving the overfitting-underfitting dilemma as follows:

1. From Fig. 10, it is clear that experts have different performances in learning new tasks: deeper ones (E_3 and E_4) generally learn better than experts (E_1 and E_2) from shallower layers. The RSD loss has no noticeable effect on E_4 the last expert's learning of the new task, which means its ability to avoid the underfitting problem of the new task is preserved.
2. From Fig. 11, experts show varying degrees of buffer overfitting, with shallower experts doing slightly better. The average BOF value of the last expert E_4 is significantly improved with the help of RSD. This is benefited from the various feature representations learned by all experts. This is consistent with the discussion of the ablation study in Sec. 5.2). In other words, knowledge of feature representation in shallower experts $E_{i\leq 3}$ is transferred to the final predictor $E_4 = F$ and it helps alleviate the buffer overfitting problem of old tasks.

The above statement holds for OCL with different memory sizes, demonstrating the excellent compatibility of multi-level feature learning and self-distillation and showcasing a synergy within MOSE where the sum is greater than its parts.

Expert	a_{ω_1}	a_{ω_2}	a_{ω_3}	a_{ω_4}
input shape	(16, 32, 32)	(128, 16, 16)	(256, 8, 8)	(512, 4, 4)
composition	[Conv3x3(C_{in}, C_{in}, s=2, g=C_{in}) Conv1x1(C_{in}, C_{in}, s=1, g=1) BatchNorm(C_{in}) ReLU() Conv3x3(C_{in}, C_{in}, s=1, g=C_{in}) Conv1x1(C_{in}, $C_{\text{in}} \times 2$, s=1, g=1) BatchNorm($C_{\text{in}} \times 2$) ReLU()] $\times 3$ +AdaptiveAvgPool2d(1,1)	[Conv3x3(C_{in}, C_{in}, s=2, g=C_{in}) Conv1x1(C_{in}, C_{in}, s=1, g=1) BatchNorm(C_{in}) ReLU() Conv3x3(C_{in}, C_{in}, s=1, g=C_{in}) Conv1x1(C_{in}, $C_{\text{in}} \times 2$, s=1, g=1) BatchNorm($C_{\text{in}} \times 2$) ReLU()] $\times 2$ +AdaptiveAvgPool2d(1,1)	[Conv3x3(C_{in}, C_{in}, s=2, g=C_{in}) Conv1x1(C_{in}, C_{in}, s=1, g=1) BatchNorm(C_{in}) ReLU() Conv3x3(C_{in}, C_{in}, s=1, g=C_{in}) Conv1x1(C_{in}, $C_{\text{in}} \times 2$, s=1, g=1) BatchNorm($C_{\text{in}} \times 2$) ReLU()] $\times 1$ +AdaptiveAvgPool2d(1,1)	Identity() +AdaptiveAvgPool2d(1,1)
output dimension	512	512	512	512
# of parameters	268800	254976	202752	0

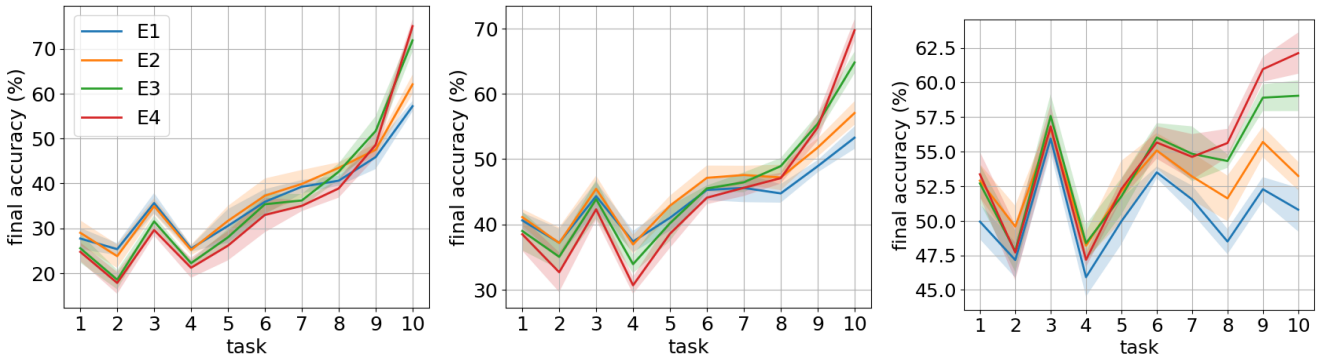
Table 7. **Composition of Alignment Modules.** Each alignment module contains consecutive similar blocks to down-sample the maps' sizes and increase their channel dimensions. *Conv3x3* and *Conv1x1* are convolutional layers with kernel size $(3, 3)$ and $(1, 1)$, respectively. s stands for stride and g means number of groups. *BatchNorm*, *ReLU*, and *AdaptiveAvgPool2d* are common network layers: batch normalization, ReLU activation, and adaptive average 2D pooling layers. *Identity()* function outputs the input feature map directly.



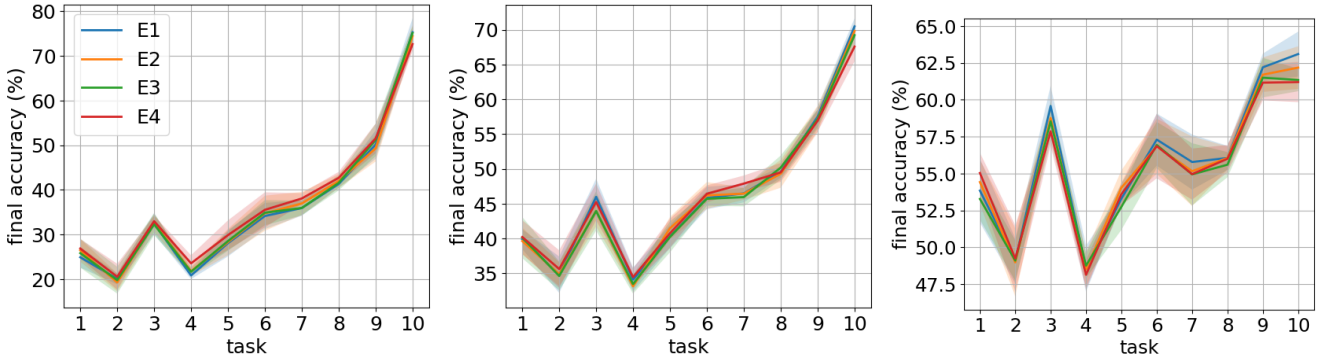
(a) Average Accuracy at $M = 1k$ (b) Average Accuracy at $M = 2k$ (c) Average Accuracy at $M = 5k$

(d) MOE Average Accuracy at $M = 1k$ (e) MOE Average Accuracy at $M = 2k$ (f) MOE Average Accuracy at $M = 5k$

Figure 5. **Average Accuracy with Different Student Expert.** Here presents the average test accuracy $\mathbb{E}_{i \leq t} a_{i,t}$ at different task t during training. E1, E2, E3, and E4 denote the student experts used in our proposed RSD. **(a)**, **(b)** and **(c)** are accuracy results of corresponding student expert; **(d)**, **(e)** and **(f)** are accuracy results of their MOE version, which is the accuracy of averaged output logits across all experts.



(a) Final Accuracy at $M = 1k$ (b) Final Accuracy at $M = 2k$ (c) Final Accuracy at $M = 5k$

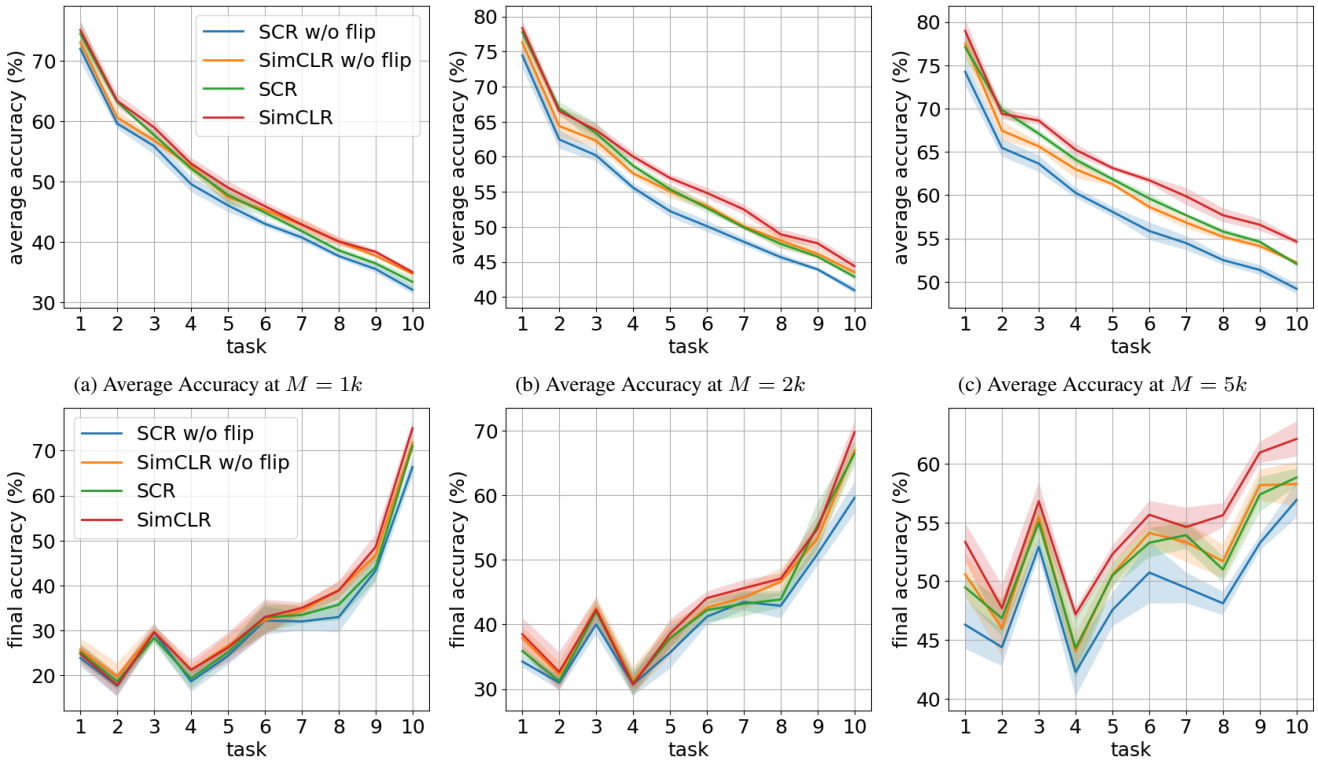


(d) MOE Final Accuracy at $M = 1k$ (e) MOE Final Accuracy at $M = 2k$ (f) MOE Final Accuracy at $M = 5k$

Figure 6. **Final Task-wise Accuracy with Different Student Expert.** Here presents the final task-wise test accuracy $a_{t,T}$ of different task t after OCL training is complete. E1, E2, E3, and E4 denote the student experts used in our proposed RSD. **(a)**, **(b)** and **(c)** are accuracy results of corresponding student expert; **(d)**, **(e)** and **(f)** are accuracy results of their MOE version.

Experts		task 1	task 2	task 3	task 4	task 5	task 6	task 7	task 8	task 9	task 10	Average
$M = 1k$	E_1	27.72	**25.36**	**35.62**	**25.50**	30.54	35.90	39.26	40.60	45.86	57.18	36.35
	E_2	**29.00**	23.80	34.94	25.18	**31.66**	**37.26**	**39.88**	**43.46**	47.48	62.04	**37.47**
	E_3	25.56	18.56	31.54	22.24	28.00	35.36	36.20	42.60	**51.66**	71.84	36.36
	E_4	24.76	17.84	29.62	21.22	26.14	32.98	35.00	38.88	48.58	**75.00**	35.00
$M = 2k$	E_1	40.60	**37.18**	44.32	**37.36**	41.00	45.28	45.58	44.74	48.90	53.28	43.82
	E_2	**41.10**	37.14	**45.44**	36.92	**42.86**	**47.12**	**47.56**	47.18	51.76	57.06	**45.41**
	E_3	38.98	35.02	43.86	33.90	40.16	45.50	46.44	**48.92**	**55.40**	64.80	45.30
	E_4	38.48	32.66	42.30	30.66	38.56	44.08	45.60	47.08	54.84	**69.74**	44.40
$M = 5k$	E_1	49.94	47.16	55.94	45.92	49.96	53.50	51.54	48.50	52.28	50.80	50.55
	E_2	52.90	**49.58**	56.66	48.20	52.14	55.08	53.20	51.62	55.70	53.24	52.83
	E_3	52.70	47.80	**57.58**	**48.38**	51.76	**56.02**	**54.84**	54.32	58.90	59.04	54.13
	E_4	**53.36**	47.70	56.82	47.18	**52.30**	55.66	54.62	**55.62**	**60.96**	**62.12**	**54.63**

Table 8. **Final Task-wise Accuracy with Different Student Expert**. Here presents the final task-wise test accuracy $a_{t,T}$ of different task t after OCL training is complete. E_1, E_2, E_3, and E_4 denote the student experts used in our proposed RSD.



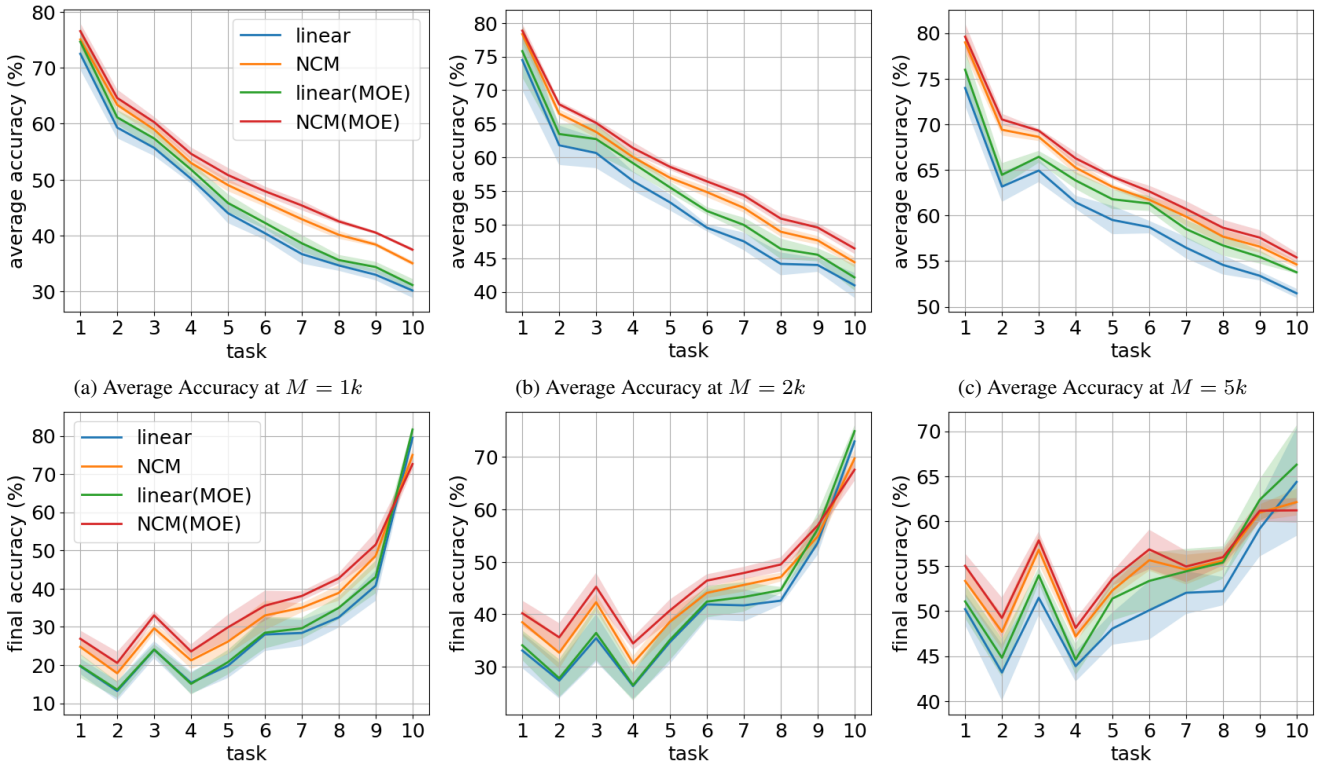
(a) Average Accuracy at $M = 1k$ (b) Average Accuracy at $M = 2k$ (c) Average Accuracy at $M = 5k$

(d) Final Accuracy at $M = 1k$ (e) Final Accuracy at $M = 2k$ (f) Final Accuracy at $M = 5k$

Figure 7. **Data augmentation.** Here presents the average test accuracy $\mathbb{E}_{i \leq t} a_{i,t}$ and final task-wise accuracy $a_{t,T}$ at different task t during training. Four different training setups, where we use *SCR* or *SimCLR* augmentation combination and whether inner flip is introduced, are tested with three memory buffer sizes. **(a)**, **(b)** and **(c)** are average test accuracy results; **(d)**, **(e)** and **(f)** are final task-wise accuracy results.

	Experts	task 1	task 2	task 3	task 4	task 5	task 6	task 7	task 8	task 9	task 10	Average
$M = 1k$	SCR w/o flip	23.88	17.66	29.56	18.64	24.34	32.18	32.00	32.98	43.30	66.36	32.09
	SimCLR w/o flip	**25.74**	**19.74**	**29.78**	21.18	**26.64**	32.26	34.36	**38.98**	46.68	71.84	34.72
	SCR	25.18	18.62	28.32	19.32	25.16	32.92	33.44	35.72	44.12	71.08	33.39
	SimCLR	24.76	17.84	29.62	**21.22**	26.14	**32.98**	**35.00**	38.88	**48.58**	**75.00**	**35.00**
$M = 2k$	SCR w/o flip	34.26	30.98	40.02	30.70	35.64	41.20	43.48	42.88	50.94	59.58	40.97
	SimCLR w/o flip	37.92	32.20	**42.54**	**31.30**	37.66	42.62	44.16	46.64	53.34	66.94	43.53
	SCR	35.88	31.22	41.98	30.90	37.86	42.22	43.18	43.86	**55.38**	66.46	42.89
	SimCLR	**38.48**	**32.66**	42.30	30.66	**38.56**	**44.08**	**45.60**	**47.08**	54.84	**69.74**	**44.40**
$M = 5k$	SCR w/o flip	46.30	44.38	52.94	42.26	47.56	50.74	49.44	48.12	53.24	56.92	49.19
	SimCLR w/o flip	50.60	46.00	55.48	44.02	50.58	54.12	53.32	51.70	58.18	58.28	52.23
	SCR	49.48	46.84	55.00	44.30	50.50	53.28	53.92	51.00	57.38	58.84	52.05
	SimCLR	**53.36**	**47.70**	**56.82**	**47.18**	**52.30**	**55.66**	**54.62**	**55.62**	**60.96**	**62.12**	**54.63**

Table 9. **Data augmentation.** Here presents the final task-wise accuracy $a_{t,T}$ at different task t during training. Four different training setups, where we use *SCR* or *SimCLR* augmentation combination and whether inner flip is introduced, are tested with three memory sizes.



(a) Average Accuracy at $M = 1k$ (b) Average Accuracy at $M = 2k$ (c) Average Accuracy at $M = 5k$

(d) Final Accuracy at $M = 1k$ (e) Final Accuracy at $M = 2k$ (f) Final Accuracy at $M = 5k$

Figure 8. **NCM or Linear Classifier.** Here presents the average test accuracy $\mathbb{E}_{i \leq t} a_{i,t}$ and final task-wise accuracy $a_{t,T}$ at different task t during training. Four different training setups, where we use *NCM* or *linear* classifier along with its MOE versions, are tested with three memory buffer sizes. **(a)**, **(b)** and **(c)** are average test accuracy results; **(d)**, **(e)** and **(f)** are final task-wise accuracy results.

Experts		task 1	task 2	task 3	task 4	task 5	task 6	task 7	task 8	task 9	task 10	Average
$M = 1k$	linear	19.68	13.22	24.00	15.36	19.82	28.00	28.40	32.48	40.80	79.50	30.13
	NCM	24.76	17.84	29.62	21.22	26.14	32.98	35.00	38.88	48.58	75.00	35.00
	linear(MOE)	19.80	13.54	24.08	15.08	20.76	28.46	29.62	34.98	43.04	**81.64**	31.10
	NCM(MOE)	**26.88**	**20.56**	**33.00**	**23.56**	**29.88**	**35.54**	**38.08**	**42.70**	**51.52**	72.62	**37.43**
$M = 2k$	linear	33.08	27.34	35.44	26.30	34.70	41.88	41.70	42.60	53.62	72.98	40.96
	NCM	38.48	32.66	42.30	30.66	38.56	44.08	45.60	47.08	54.84	69.74	44.40
	linear(MOE)	34.08	27.80	36.44	26.46	35.12	42.42	43.30	44.60	56.24	**74.94**	42.14
	NCM(MOE)	**40.20**	**35.60**	**45.24**	**34.44**	**40.74**	**46.42**	**47.88**	**49.50**	**56.92**	67.58	**46.45**
$M = 5k$	linear	50.22	43.18	51.48	43.88	48.06	50.10	52.04	52.22	59.18	64.36	51.47
	NCM	53.36	**47.70**	56.82	47.18	52.30	55.66	54.62	55.62	60.96	62.12	54.63
	linear(MOE)	51.08	44.82	53.98	44.62	51.40	53.36	54.42	55.40	**62.38**	**66.28**	53.77
	NCM(MOE)	**55.04**	49.24	**57.86**	**48.14**	**53.60**	**56.86**	**54.96**	**56.00**	61.16	61.20	**55.41**

Table 10. **Data augmentation.** Here presents the final task-wise accuracy $a_{t,T}$ at different task t during training. Four different training setups, where we use *NCM* or *linear* classifier along with its MOE versions, are tested with three memory buffer sizes.

Baseline	Source Code Links
AGEM	https://github.com/facebookresearch/agem
ER and MIR	https://github.com/optimass/Maximally_Interfered_Retrieval
GSS	https://github.com/rahafaljundi/Gradient-based-Sample-Selection
ASER and SCR	https://github.com/RaptorMai/online-continual-learning
ER-AML	https://github.com/pclucas14/AML
GDumb	https://github.com/drimpossible/GDumb
OCM	https://github.com/gydpku/OCM
OnPro	https://github.com/weilllllls/OnPro
GSA	https://github.com/gydpku/GSA
DER++	https://github.com/aimagelab/mammoth
IL2A	https://github.com/Impression2805/IL2A
Co^{2}L	https://github.com/chaht01/Co2L
LUCIR	https://github.com/hshustc/CVPR19_Incremental_Learning
CCIL	https://github.com/sud0301/essentials_for_CIL
BiC	https://github.com/sairin1202/BIC
SSIL	https://github.com/hongjoon0805/SS-IL-Official

Table 11. **Baseline Source Code Links.** We list the official source links of all tested baseline algorithms.

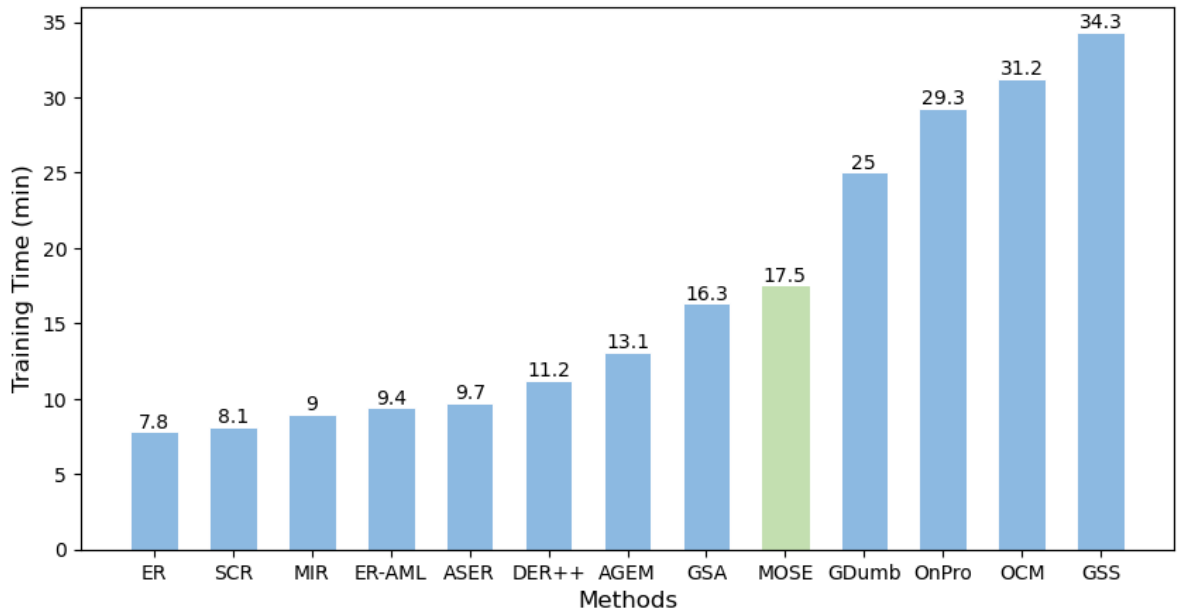
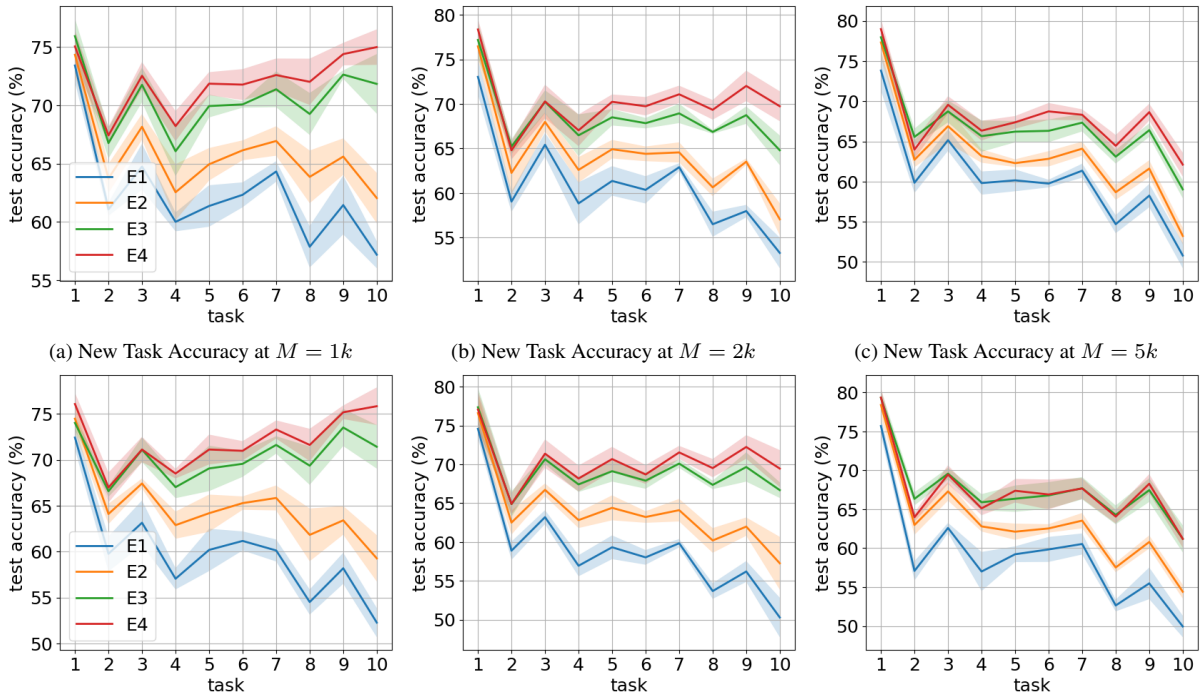


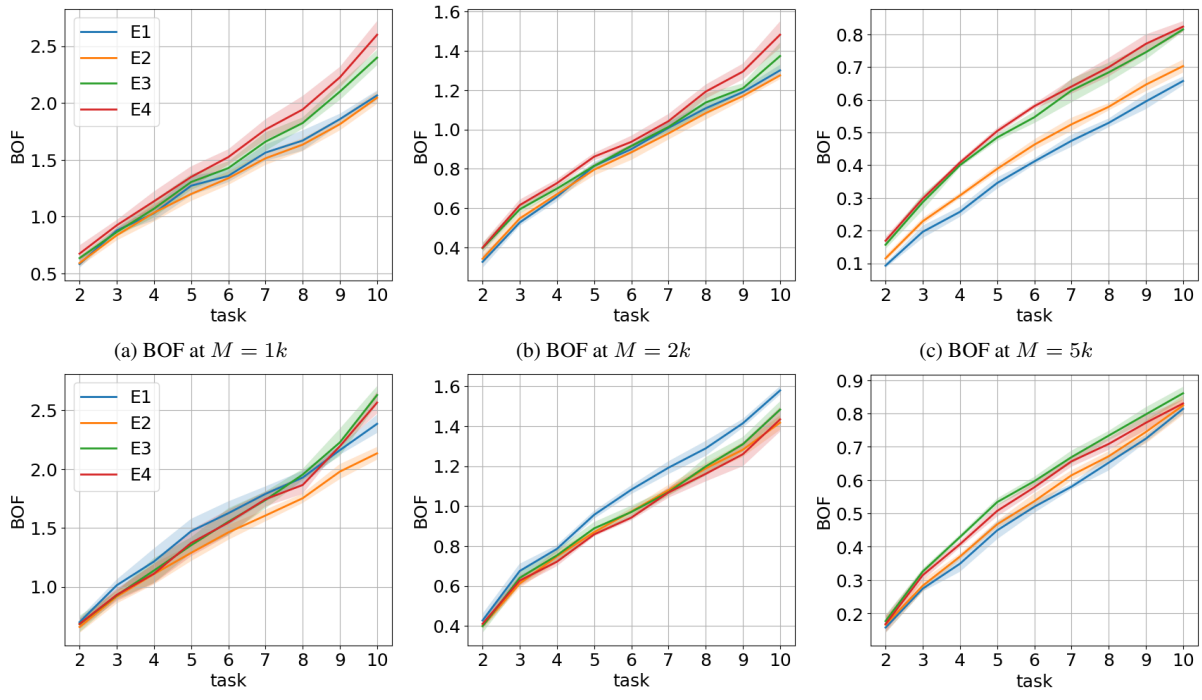
Figure 9. **Training Time (min).** Here we show the training time of all tested OCL methods over the dataset Split CIFAR-100 with memory size $M = 5000$. Time for our proposed method MOSE is colored in green for better visual presentation.



(a) New Task Accuracy at $M = 1k$ (b) New Task Accuracy at $M = 2k$ (c) New Task Accuracy at $M = 5k$

(d) New Task Accuracy with RSD at $M = 1k$ (e) New Task Accuracy with RSD at $M = 2k$ (f) New Task Accuracy with RSD at $M = 5k$

Figure 10. **New Task Accuracy with or without RSD.** Here presents the new task accuracy $a_{t,t}$ at different task t during training. The results of all experts with three memory buffer sizes are depicted. **(a)**, **(b)** and **(c)**: no RSD loss $\mathcal{L}_{\text{RSD}}$; **(d)**, **(e)** and **(f)**: with RSD loss $\mathcal{L}_{\text{RSD}}$.



(a) BOF at $M = 1k$ (b) BOF at $M = 2k$ (c) BOF at $M = 5k$

(d) BOF with RSD at $M = 1k$ (e) BOF with RSD at $M = 2k$ (f) BOF with RSD at $M = 5k$

Figure 11. **Average BOF with or without RSD.** Here presents the average BOF value (Eq. 5) at different task t during training. The results of all experts with three memory buffer sizes are depicted. **(a)**, **(b)** and **(c)**: no RSD loss $\mathcal{L}_{\text{RSD}}$; **(d)**, **(e)** and **(f)**: with RSD loss $\mathcal{L}_{\text{RSD}}$.