OPTIMIZING CLASS DISTRIBUTION IN MEMORY FOR MULTI-LABEL CONTINUAL LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Continual learning, which tries to learn from a data stream with non-stationary distribution, is an important yet challenging problem. One of the most effective ways to solve this problem is replay-based methods, in which a replay buffer called memory is maintained to keep a small part of past samples and the model rehearses these samples to keep its performance on old distribution when learning on new distribution. Most existing replay-based methods focus on single-label problems in which each sample in the data stream has only one label. But many real applications are multi-label problems in which each sample may have more than one label. To the best of our knowledge, there exists only one method, called partition reservoir sampling (PRS), for multi-label continual learning problems. PRS suffers from low speed due to its complicated process. In this paper, we propose a novel method, called optimizing class distribution in memory (OCDM), for multi-label continual learning. OCDM formulates the memory update mechanism as an optimization problem and updates the memory by solving this problem. Experiments on two widely used multi-label datasets show that OCDM outperforms other state-of-the-art methods including PRS in terms of accuracy, and its speed is also much faster than PRS.

1 INTRODUCTION

The ability to learn multiple tasks continuously, which is usually called continual learning, is essential for people and other animals to live in real environment (Parisi et al., 2019). However, deep neural networks typically lack this ability. Specifically, when learning on the data with new distribution, the neural network may forget the old knowledge it has learned so that its performance on the old data distribution will drop significantly. This problem is often referred to as catastrophic forgetting (Goodfellow et al., 2013).

Two different settings are often considered in continual learning. The first setting is offline continual learning. In this setting, the model learns multiple tasks in a specific order. At each time the whole dataset of one task or several classes is available for training the model (Hou et al., 2019; Rajasegaran et al., 2020). Thus, the model can perform multiple epochs on the current dataset to achieve the desired performance. The second setting is online continual learning. In this setting, the data of all the tasks or classes will be formulated into a data stream with a non-stationary distribution. At each time only a small batch of samples are available for training the model, so the model can only traverse each task once (Lopez-Paz & Ranzato, 2017; Aljundi et al., 2019a). Some works (Aljundi et al., 2019b;c; Chrysakis & Moens, 2020) considered a more challenging online continual learning setting where both task boundaries and task labels are not available to the model during training. Lacking the task boundaries and task labels means the model does not know when the old task ends and when the new task begins, which brings more difficulties for the model.

Three different types of methods are proposed to solve continual learning problems, including regularization-based methods (Kirkpatrick et al., 2017; Zenke et al., 2017; Ahn et al., 2019), expansion-based methods (Yoon et al., 2017; Li et al., 2019; Hung et al., 2019) and replay-based methods (De Lange et al., 2019; Lopez-Paz & Ranzato, 2017; Guo et al., 2019). Among them, replay-based methods have always shown superior performance. These replay-based methods maintain a buffer called memory to keep a small part of old samples and rehearse saved samples to alleviate catastrophic forgetting. For these methods, keeping a proper class distribution in memory

is critical for the model to perform well. In general, keeping an equal number of samples in the memory for each old class is a common choice for many existing replay-based methods (Rebuffi et al., 2017; Liu et al., 2020; Rajasegaran et al., 2020; Wu et al., 2019), because this could put equal attention on each old task or class. However, in online continual learning, especially when the task boundaries and task labels are not available to the model, controlling the class distribution in memory is difficult. Several works consider this challenging setting and propose different memory update mechanisms (Aljundi et al., 2019c; Kim et al., 2020). However, most of these methods either fail to control the class distribution in the memory or can only be used for the single-label setting in which each sample in the data stream has only one label. But many real applications are multi-label problems in which each sample may have more than one label. When the samples in the data stream are multi-label ones, the data distribution for different classes is always imbalanced, which brings more difficulties in controlling the class distribution in memory than single-label setting. As a result, the class distribution in the memory may also become imbalanced such that the model fails to keep its performance on the classes with too few saved samples. To the best of our knowledge, there exists only one method, called partition reservoir sampling (PRS) (Kim et al., 2020), for multi-label continual learning problems. As we will see from the experimental results in this paper, PRS suffers from low speed due to its complicated process.

In this paper, we propose a simple but effective method, called optimizing class distribution in memory (OCDM), to control the class distribution in memory under the online multi-label continual learning setting without task boundaries and task labels. The main contributions of OCDM are outlined as follows:

- OCDM formulates the memory update process for online continual learning into a selection problem and proposes an optimization problem to represent it.
- OCDM proposes a greedy algorithm to solve the above optimization problem with linear time complexity, which can control the class distribution in memory for online multi-label continual learning.
- Experiments on two widely used multi-label datasets MSCOCO and NUSWIDE show that OCDM outperforms other state-of-the-art methods including PRS in terms of accuracy, and its speed is also much faster than PRS.

2 Related Work

Replay-based methods maintain a memory to keep a small part of past samples. In the offline single-label continual learning setting where each sample has only one single label, controlling the class distribution in memory is trivial. Keeping an equal number of samples in the memory for each old task or old class is a common choice for many replay-based methods (Rebuffi et al., 2017; Liu et al., 2020; Rajasegaran et al., 2020; Wu et al., 2019). As for online continual learning without task boundaries and task labels, the model has little information about the data stream, so it is challenging to control class distribution in memory even for the single-label setting. Reservoir sampling (RS) (Vitter, 1985) has been widely adopted by many existing replay-based methods (Gupta et al., 2020; Buzzega et al., 2020) for memory update. Under the single-label setting, this memory update mechanism can guarantee that the expectation of the number of samples stored in memory for each class is proportional to its total number of samples in the data stream. Therefore, RS performs poorly in a more realistic environment where the class distribution is highly imbalanced. Gradient-based sample selection (GSS) (Aljundi et al., 2019c) selects (saves) samples to maximize the gradient direction in memory and shows better performance than RS in the imbalanced data stream setting. There also exist works which propose class balancing mechanisms to keep an equal number of samples for each class in order to keep the performance of those minor classes (Chrysakis & Moens, 2020). However, these methods only consider the single-label setting.

To the best of our knowledge, partition reservoir sampling (PRS) (Kim et al., 2020) is the only work to tackle the multi-label problem under the online continual learning setting. PRS has two processes called sample in and sample out when a new batch is coming and memory is full. In the sample in process, memory decides whether let the current new sample come into the memory or not. If the answer is true, the sample out mechanism will be performed to decide which samples will be moved out. However, the process of PRS is too complicated so that the speed of memory update is

too slow, while the speed is important because the model needs to react quickly to the changes in real environment.

3 METHODOLOGY

In this section, we first give the problem definition, and then propose our memory update method OCDM.

3.1 PROBLEM DEFINITION

Under the online continual learning setting, there is a data stream $\mathcal{S}$ containing a sequence of datasets for different tasks. At each time t, a batch of data $\mathcal{B}_t = \{(\mathbf{x}_i^t, \mathbf{y}_i^t)\}_{i=1}^{b_t}$ is available to the model, where b_t is the batch size at time t and it is usually small. $\mathbf{x}_i^t$ is an input sample and $\mathbf{y}_i^t$ is a multi-hot label vector. The model does not have the global information about the whole data stream. That is to say, the length of the data stream, the task boundaries, and the task labels are not available to the model during the whole training process. The only thing the model has access to is the current new batch of samples $\mathcal{B}_t$. Furthermore, there is a memory $\mathcal{M}$ with a fixed size M, which is used to keep a small part of samples from the data stream. At each step, the model needs to use the new samples $\mathcal{B}_t$ to update $\mathcal{M}$ according to the current class distribution in memory.

3.2 UPDATING MEMORY WITH AN OPTIMIZATION PROBLEM

Since the memory size is fixed and the batch size is often smaller than the memory size, the memory $\mathcal{M}$ is not full when the first several batches come from the data stream. Like the previous methods RS (Vitter, 1985), GSS (Aljundi et al., 2019c) and PRS (Kim et al., 2020), in this case, OCDM will add the new samples into the memory without any selection mechanism in order to make full use of memory space. When the memory is full, the model needs to decide which new samples should be added to the memory and which old samples should be removed from the memory, that is, to update the memory.

When the t-th batch of data $\mathcal{B}_t$ with size b_t is coming and the memory is full, the process of memory update can be regarded as selecting M samples from $M + b_t$ samples to be kept in $\mathcal{M}$, and then removing the remaining b_t samples. Different from PRS that treats the samples kept in $\mathcal{M}$ and new samples $\mathcal{B}_t$ differently, OCDM treats all these samples equally. Specifically, if the target distribution $\mathbf{p}$, which we want to control the class distribution in memory to be, is given,

$$\begin{aligned} &\mathbf{p} = [p_1, p_2, ..., p_{C-1}, p_C], \\ &p_i \geq 0, \ \sum_{i=1}^{C} p_i = 1, \end{aligned} \tag{1}$$

where C represents the total number of classes kept in $\mathcal{M} \cup \mathcal{B}_t$. Then selecting M samples from $M + b_t$ samples can be represented as the following optimization problem:

$$\begin{aligned} \min_{\Omega} \ & Dist(\mathbf{p}_\Omega, \mathbf{p}) \\ s.t. \ & \Omega \subseteq [M + b_t], \\ & |\Omega| = M, \end{aligned} \tag{2}$$

where $Dist(\cdot, \cdot)$ represents a function that measures the distance between two distributions, such as KL-difference, $[M + b_t] = \{1, 2, ..., M + b_t\}$ is the indices of $\mathcal{M} \cup \mathcal{B}_t$, Ω represents the indices of the selected dataset. $\mathbf{p}_\Omega = [p_{\Omega,1}, p_{\Omega,2}, ..., p_{\Omega,C}]$ represents the class distribution of the selected dataset and its j-th component can be represented as

$$p_{\Omega,j} = \frac{m_{\Omega,j}}{\sum_i m_{\Omega,i}}, \tag{3}$$

where $m_{\Omega,j}$ is the number of samples belonging to the j-th class in the selected dataset.

Solving the optimization problem in (2) means selecting a subset $\{(\mathbf{x}_i, \mathbf{y}_i) | i \in \Omega, (\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{M} \cup \mathcal{B}_t\}$ with M elements such that the class distribution $\mathbf{p}_\Omega$ of this selected dataset is closest to the target distribution $\mathbf{p}$. We show the whole process in Figure 1. In the following two subsections, we will introduce the target distribution $\mathbf{p}$ and the selection strategy based on problem in (2).

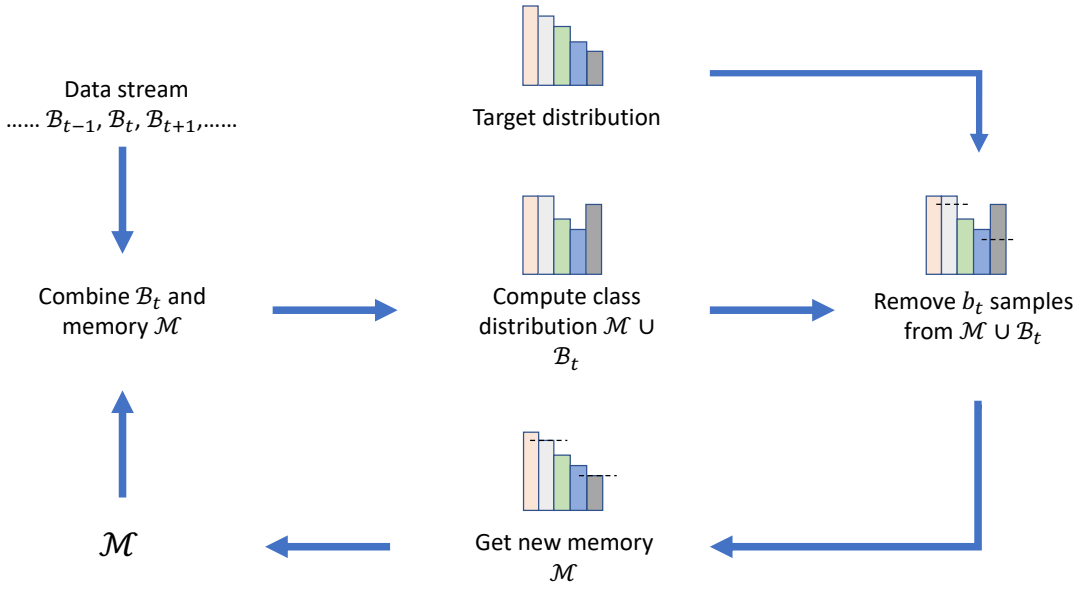


Figure 1: The process of OCDM. At each step the memory receives a batch of new samples and then removes some samples to minimize the distance between the current class distribution and the target class distribution.

3.2.1 TARGET DISTRIBUTION

We adopt similar techniques in PRS to formulate our target distribution, that is, the model uses the running class frequency to set the target class distribution in $\mathcal{M}$. Specifically, when the t-th batch of new samples comes, the model can compute the running frequency of each class and set the target distribution for the i-th class in memory as follows:

$$p_i = \frac{(n_i)^\rho}{\sum_{i=1}^{C}(n_j)^\rho}, \tag{4}$$

where ρ is a power of allocation, n_i is the running frequency for class i. Please note that when $\rho = 1$, the number of samples kept in memory for each class is proportional to the total number of samples in the data stream. Thus, the major classes will keep much more samples in memory than the minor classes. When $\rho = 0$, the target distribution puts an equal allocation for all the classes, so the memory is updated to have the same number of samples in memory for all the classes.

For a single-label setting without any prior knowledge, keeping an equal number of samples in memory for each class is an ideal choice and has been adopted by many existing relay-based methods. Under the imbalanced online continual learning setting, keeping class distribution in memory balanced becomes more critical to keep the model's performance on those classes with too few training samples. Previous work PRS has shown that balanced class distribution in memory is ideal for the model to get desired performance in multi-label setting in which the class distribution in the data stream is always imbalanced. Therefore, we will set ρ to 0 by default unless otherwise stated. In the experiment, we will change ρ between 0 and 1 to show the effect of ρ on the model's performance.

3.2.2 SELECTION STRATEGY

One challenge we have not tackled is how to solve the optimization problem in (2). This problem is a discrete optimization problem, and the number of possibilities is limited. Thus, the most straightforward way to solve (2) is to transverse all the possibilities and find the optimal value. However, the total number of possibilities in (2) is $\binom{M+b_t}{M}$. Taking our experimental setting as an example,

Algorithm 1 Optimizing Class Distribution in Memory (OCDM)

Input: data stream $\mathcal{S}$, memory $\mathcal{M}$, memory size M.
while data stream $\mathcal{S}$ is not over **do**
  Get a batch of new samples $\mathcal{B}_t = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{b_t}$ from $\mathcal{S}$;
  if $\mathcal{M}$ is not full **then**
    Get the remaining space r in $\mathcal{M}$;
    Select $\min(b_t, r)$ samples $\mathcal{V}_t$ from $\mathcal{B}_t$ randomly and save in $\mathcal{M}$;
    $\mathcal{B}_t \leftarrow \mathcal{B}_t \backslash \mathcal{V}_t$;
    $b_t \leftarrow b_t - |\mathcal{V}_t|$;
  end if
  if $b_t > 0$ **then**
    $\Omega_0 \leftarrow [M + b_t]$;
    for $k \in [1, 2, ..., b_t]$ **do**
      Get index i_k from Ω_{k-1} according to (6);
      $\Omega_k \leftarrow \Omega_{k-1} \backslash \{i_k\}$;
    end for
    Select M samples with indices Ω_{b_t} as a new memory and remove the remaining samples;
  end if
end while

the size of the memory M is set to 1000, and the batch size of the data stream b_t is set to 10. At this case, we can get:

$$\binom{M+b_t}{M} = \binom{M+b_t}{b_t} \geq \left(\frac{M+b_t}{b_t}\right)^{b_t} = 101^{10} \geq O(10^{20}). \tag{5}$$

This shows that obtaining the optimal solution through transversing methods is impossible.

In OCDM, we design a greedy algorithm to solve this problem. Specifically, the purpose of (2) is to select M samples from $M + b_t$ samples. Equivalently, the purpose of (2) can also be restated as deleting b_t samples from $M + b_t$ samples. Through the above analysis, we have known that it is too complicated to delete b_t samples from $M + b_t$ samples at one time. Hence, we divide the deletion process into b_t subprocesses, and each subprocess deletes one sample. Assume $\Omega_0 = [M + b_t]$ is the indices of the whole dataset $\mathcal{M} \cup \mathcal{B}_t$ available to the model, and in the k-th ($1 \leq k \leq b_t$) subprocess, Ω_k is selected from Ω_{k-1} by removing one sample $(\mathbf{x}_{i_k}, \mathbf{y}_{i_k})$, where $\Omega_k = \Omega_{k-1} \backslash \{i_k\}$. The deleted index i_k needs to satisfy

$$i_k = \underset{j \in \Omega_{k-1}}{\arg\min}\, Dist(\mathbf{p}_{\Omega_{k-1} \backslash \{j\}}, \mathbf{p}), \tag{6}$$

where $\mathbf{p}_{\Omega_{k-1} \backslash \{j\}}$ represents the class distribution of the subsets with indices $\Omega_{k-1} \backslash \{j\}$. We show the whole process of our method in Algorithm 1.

For a sequence of length n, the time complexity of finding the sample with the smallest objective value in (6) is $O(n)$. Hence, the total time complexity of updating memory $\mathcal{M}$ with the new batch $\mathcal{B}_t$ is

$$O(\sum_{i=0}^{b_t-1}(M + b_t - i)) = O(b_t(M + b_t) - \frac{b_t(b_t-1)}{2}) = O(b_t M). \tag{7}$$

The second equation in (7) is derived from $M \gg b_t$ in common conditions. This shows that the time complexity of using greedy algorithm to solve (2) is linear. Note that when all the samples have only one label and ρ is set to 0 in (4), our selection strategy can get the optimal solution because each deleted sample selected by (6) must belong to the class with the largest number of samples.

4 EXPERIMENTS

We compare our method OCDM with other continual learning methods under the online multi-label continual learning setting without task boundaries and task labels.

4.1 EXPERIMENTAL SETTINGS

Dataset Following the previous work PRS (Kim et al., 2020), we adopt two widely used multi-label datasets MSCOCO (Lin et al., 2014) and NUSWIDE (Chua et al., 2009) to construct multi-label continual learning data streams. We do not directly use the datasets COCOseq and NUSWIDEseq in PRS (Kim et al., 2020) as our datasets, because we find that the proportion of samples with only one label in the constructed datasets of PRS is too high. Too many samples with one label make the model only need to pay attention to single-label samples to control the class distribution in memory, and hence the problem can be transformed into a single-label problem. Therefore, we construct data streams for MSCOCO and NUSWIDE by ourselves in order to increase the proportion of samples with more than one label. Specifically, we construct four and five tasks for MSCOCO and NUSWIDE, respectively, and each task contains exclusive class labels. For MSCOCO, the total number of classes for all tasks is 42 and the total number of samples in the data stream is 11044. For NUSWIDE, the total number of classes for all tasks is 37 and the total number of samples in the data stream is 19704. We show the detailed information of our datasets and the difference between our datasets and the datasets of PRS (Kim et al., 2020) in appendix of the supplementary material. Furthermore, we also give the experimental results on the datasets MSCOCOseq and NUSWIDEseq of PRS (Kim et al., 2020) in appendix.

Since the multi-label datasets always have an imbalanced class distribution, we follow the previous PRS work and divide all classes into three parts: majority, moderate, and minority, according to the number of samples in each class. Specifically, classes with a sample size of more than 500 are classified as majority, those with less than 100 are classified as minority, and those with a sample size between 100 and 500 are classified as moderate.

Training Details and Architecture Following the previous online continual learning methods (Aljundi et al., 2019a;c), in all the experiments we set both the data batch size and replay size as 10 for data stream and memory. It is common for recent multi-label methods to finetune a model pre-trained on ImageNet to get a target model. We follow this operation and choose ResNet101 pre-trained on ImageNet as our training model. We train the model using Adam optimizer with learning rate $lr = 1e-4$, running averages of gradient and its square $(\beta_1 = 0.9, \beta_2 = 0.999)$, numerical stability term $\epsilon = 1e-4$ which is used by PRS (Kim et al., 2020). We set $\rho = 0$ unless otherwise stated. Furthermore, we perform data augmentation with random crops and horizontal flips to the samples in the data stream and memory for all the baselines and our methods. We perform all experiments on four NVIDIA TITAN Xp GPUs.

Baseline and Evaluation Metric We compare our methods with three different baselines including RS (Vitter, 1985), PRS (Kim et al., 2020), GSS-Greedy (Aljundi et al., 2019c). RS, PRS and GSS-Greedy are the state-of-the-art replay-based continual learning methods. Besides, we also add multitask and finetune to compare with our method. Multitask learns the model with all the previous data when a new task is coming, which could be regarded as the upper bound of all the continual learning methods. Finetune learns the model without any memory, which could be regarded as the lower bound of all the methods.

We report several commonly used multi-label classification metrics including the average overall F1 (OF1), per-class F1 (CF1), and mean average precision (mAP). Besides, the forgetting metric (denoted as forget) is also reported, as in the previous continual learning works (Kim et al., 2020; Aljundi et al., 2019a). However, for any method, a low forget value does not means this method has high performance, because if this method performs poorly during the whole training, its will get a low forget value.

4.2 EXPERIMENTAL RESULTS

We perform all the experiments on MSCOCO and NUSWIDE for three times. Table 1 and Table 2 show the average results for all the methods, where the memory size is set to 1000 for the replay-based methods. We can find that RS and GSS-Greedy get competitive results in the majority classes. However, the results for moderate classes and minority classes are not satisfactory for RS and GSS-Greedy. As for PRS and our method OCDM, the results are competitive for all classes, so the total performance for PRS and our method is much better than RS and GSS-Greedy. Furthermore, OCDM outperforms PRS in most cases in terms of accuracy.

Table 1: Results of the continual learning methods on MSCOCO with memory size being 1000.

	Majority			Moderate			Minority			Total			Time (s)
Methods	CF1	OF1	mAP	CF1	OF1	mAP	CF1	OF1	mAP	CF1	OF1	mAP	1000 steps
Multitask	60.2	46.3	61.1	59.9	58.7	64.8	53.1	48.7	53.2	58.8	52.8	61.2	-
Finetune	1.2	7.6	18.3	10.4	14.2	18.1	10.2	19.1	13.7	8.9	13.4	17.1	-
–forget	55.7	49.9	47.2	16.5	15.9	25.3	7.2	4.1	18.4	23.6	19.3	28.8	-
RS	**53.1**	40.1	50.2	43.6	43.6	45.0	22.6	24.7	29.0	41.9	38.9	42.4	**342.8**
–forget	12.2	20.4	21.2	10.0	7.6	8.9	35.3	32.9	28.5	14.1	14.9	15.6	-
GSS-Greedy	48.7	36.2	47.8	35.0	38.0	37.7	11.2	16.6	19.9	32.7	33.8	35.9	4715.6
–forget	18.9	29.9	29.7	27.8	20.5	26.3	50.2	44.7	31.6	30.7	25.5	29.0	-
PRS	48.8	46.5	47.4	45.3	42.9	45.2	42.4	41.8	41.0	45.5	43.4	44.7	718.4
–forget	12.8	14.5	19.1	10.3	13.4	11.1	15.7	18.1	14.2	11.3	12.1	12.1	-
OCDM (ours)	52.4	**51.0**	**52.6**	**48.6**	**46.1**	**48.1**	**46.3**	**43.9**	**44.6**	**48.9**	**46.5**	**48.2**	351.4
–forget	17.5	18.4	21.6	10.3	9.2	11.3	14.6	17.7	25.5	12.2	11.5	14.3	-

Table 2: Results of the continual learning methods on NUSWIDE with memory size being 1000.

	Majority			Moderate			Minority			Total			Time (s)
Methods	CF1	OF1	mAP	CF1	OF1	mAP	CF1	OF1	mAP	CF1	OF1	mAP	1000 steps
Multitask	35.1	24.4	25.5	29.7	29.9	27.0	17.6	17.7	16.1	30.6	25.4	24.4	-
Finetune	1.9	7.1	6.9	0.0	0.0	6.0	0.4	0.8	4.6	0.9	5.5	6.1	-
–forget	36.8	10.6	24.7	17.4	20.2	34.9	0.0	0.0	20.8	23.4	10.5	25.2	-
RS	**27.7**	19.8	18.3	18.1	17.9	16.9	0.2	0.2	7.8	18.9	17.5	15.7	**349.7**
–forget	16.3	5.8	15.0	21.2	21.8	17.9	16.8	20.8	20.8	17.3	9.0	16.4	-
GSS-Greedy	26.2	18.9	17.8	19.4	19.4	17.4	1.2	1.3	9.7	19.4	17.5	16.1	4782.9
–forget	12.9	5.3	15.8	14.3	14.1	16.3	8.0	8.9	22.0	12.9	5.9	16.7	-
PRS	24.0	22.2	19.1	24.2	23.7	20.3	**23.0**	**22.4**	18.8	24.2	22.8	19.5	1210.9
–forget	17.5	10.3	18.7	28.3	27.1	24.3	24.6	24.2	30.9	21.7	14.4	22.7	-
OCDM (ours)	25.3	**24.9**	**21.0**	**26.2**	**26.2**	**20.5**	21.0	22.0	**19.0**	**24.9**	**24.9**	**20.5**	370.7
–forget	18.1	8.2	17.4	20.0	19.9	21.7	30.3	30.0	32.9	22.4	13.8	22.8	-

Table 1 and Table 2 also report the time consumed by each replay-based method per 1000 steps in the training process. Among them, GSS-Greedy is the slowest one, which is also consistent with the experimental results in previous work (Chaudhry et al., 2020). Because the memory update operation of RS is straightforward, RS is the fastest one among all the replay-based methods. Our method OCDM and PRS have the mechanism to control the class distribution in memory, so more time is consumed than RS. The speed of OCDM is comparable with that of RS, but PRS is much slower than RS and OCDM.

Changing the Memory Size We also follow previous work Aljundi et al. (2019a); Kim et al. (2020) and vary the memory size in order to find whether the conclusion is changed with other memory sizes. Specifically, we set the memory size to 300 and 500, and show the results in Table 3. We can find that the performance of all the methods is lower than that with $M = 1000$, which is

Table 3: Results of the continual learning methods on NUSWIDE and MSCOCO with memory size varing.

	MSCOCO						NUSWIDE					
	$M = 300$			$M = 500$			$M = 300$			$M = 500$		
Methods	CF1	OF1	mAP	CF1	OF1	mAP	CF1	OF1	mAP	CF1	OF1	mAP
RS	31.2	29.0	33.4	37.2	33.7	38.2	16.0	14.4	13.3	16.2	14.5	13.1
GSS-Greedy	28.7	27.7	32.4	29.9	29.5	33.0	16.8	14.6	13.8	17.7	16.4	14.8
PRS	36.9	34.4	36.3	41.6	38.9	40.0	**18.0**	16.9	16.0	20.8	20.0	17.4
OCDM (ours)	**40.0**	**37.1**	**40.0**	**43.8**	**41.2**	**43.2**	17.8	**17.8**	**16.5**	**21.4**	**21.1**	**18.7**

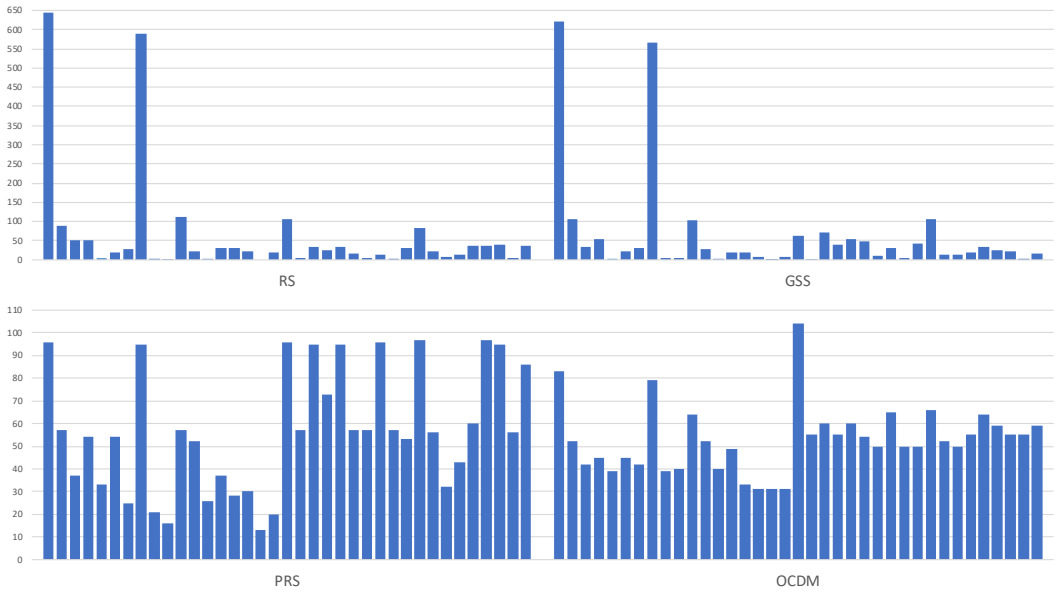


Figure 2: Class distribution in memory after training on NUSWIDE with memory size $M = 1000$.

consistent with intuition. However, our method OCDM still achieves the best performance in most cases.

Class Distribution in Memory after Training We show the class distribution in memory for RS, GSS-Greedy, PRS and OCDM in Figure 2 after training on NUSWIDE. The class distribution for RS and GSS-Greedy is highly imbalanced, where the amount of two classes is more than 500 and is much larger than those of other classes. For PRS and OCDM, the class distribution in memory is relatively balanced, while our method OCDM is more balanced than PRS. Specifically, in PRS there are ten classes with more than 70 samples and 3 classes with less than 30 samples in memory. While for our method OCDM, there are only 3 classes with more than 70 samples and no classes with less than 30 samples in memory. The class distribution in memory for MSCOCO after training is similar, which is given in appendix.

Performance Change during Training We show the changes in mAP after each task ends when training on MSCOCO in Figure 3a. It can be seen that the mAP of all methods gradually decreases as the number of tasks increases, but our method is consistently better than PRS and RS. The results on NUSWIDE is similar and we give it in appendix. This result is more important than the result at the end of training because the data stream may not be terminated in an actual environment. Hence, it is more important to maintain good performance in the middle of the data stream.

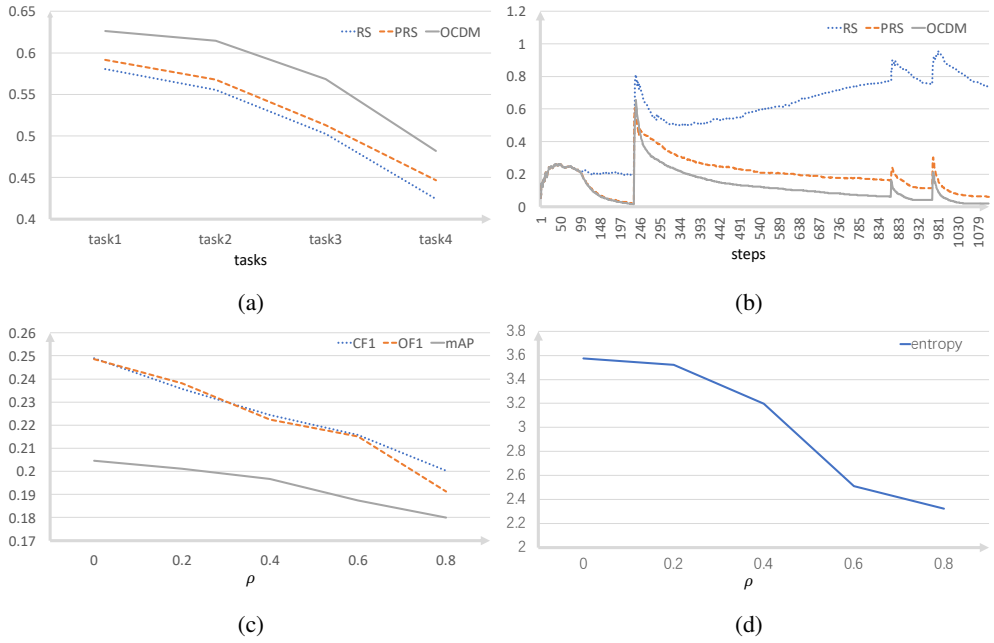


Figure 3: (a) change of mAP after each task is over on MSCOCO; (b) change of distance between class distribution in memory and target distribution during training on MSCOCO; (c) change of model's performance on NUSWIDE with difference values of ρ; (d) entropy of class distribution in memory after training on NUSWIDE with different values of ρ.

Distance Change during Training We show the distance change between class distribution in memory and target distribution during the whole training on MSCOCO in Figure 3b, where the KL-difference is used as distance function. There are several sharp points in the figure because at these points the class distribution of the data stream has changed, that is, new classes of samples have appeared. In RS, the distance does not decrease with the increase of the number of steps. On the contrary, sometimes the distance in RS will gradually increase. Using PRS and our method OCDM can make the distance decrease gradually, which means the class distribution in the memory gradually converges to uniform distribution. Furthermore, OCDM can make the class distribution in memory become closer to the target distribution than PRS. The results on NUSWIDE is similar and we show it in appendix. The above results show that OCDM can control the class distribution in memory better than other methods.

Changing the Value of ρ We change the parameter ρ of the target distribution in (4) to get different target class distribution in memory, and then study the influence on the model's performance. We set ρ to the five values of 0.0, 0.2, 0.4, 0.6, and 0.8, and show the model's performance on NUSWIDE in Figure 3c. It can be seen that as ρ gradually increases, the mAP, CF1 and OF1 of the model gradually decrease. Figure 3d shows the entropy of class distribution in memory, we can find the entropy decreases with the increase of ρ. This shows that maintaining a balanced distribution of different classes in memory is important for the model to perform well in the imbalanced multi-label data stream.

5 Conclusion

In this paper, we propose a novel replay-based method, called OCDM, for online multi-label continual learning problems without task boundaries and task labels. Experimental results on real datasets show that OCDM can well control the class distribution in memory and outperform other methods to achieve state-of-the-art performance.

REFERENCES

Hongjoon Ahn, Sungmin Cha, Donggyu Lee, and Taesup Moon. Uncertainty-based continual learning with adaptive regularization. In *Advances in Neural Information Processing Systems*, pp. 4392–4402, 2019.

Rahaf Aljundi, Lucas Caccia, Eugene Belilovsky, Massimo Caccia, Min Lin, Laurent Charlin, and Tinne Tuytelaars. Online continual learning with maximally interfered retrieval. *arXiv preprint arXiv:1908.04742*, 2019a.

Rahaf Aljundi, Klaas Kelchtermans, and Tinne Tuytelaars. Task-free continual learning. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pp. 11254–11263, 2019b.

Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. In *Advances in Neural Information Processing Systems*, pp. 11816–11825, 2019c.

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. *arXiv preprint arXiv:2004.07211*, 2020.

Arslan Chaudhry, Naeemullah Khan, Puneet K Dokania, and Philip HS Torr. Continual learning in low-rank orthogonal subspaces. *arXiv preprint arXiv:2010.11635*, 2020.

Aristotelis Chrysakis and Marie-Francine Moens. Online continual learning from imbalanced data. In *Proceedings of the International Conference on Machine Learning*, pp. 1952–1961, 2020.

Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yantao Zheng. Nus-wide: a real-world web image database from national university of singapore. In *Proceedings of the International Conference on Image and Video Retrieval*, pp. 1–9, 2009.

Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *arXiv preprint arXiv:1909.08383*, 2019.

Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013.

Yunhui Guo, Mingrui Liu, Tianbao Yang, and Tajana Rosing. Improved schemes for episodic memory-based lifelong learning. *arXiv preprint arXiv:1909.11763*, 2019.

Gunshi Gupta, Karmesh Yadav, and Liam Paull. La-maml: Look-ahead meta learning for continual learning. *arXiv preprint arXiv:2007.13904*, 2020.

Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pp. 831–839, 2019.

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. *Advances in Neural Information Processing Systems*, 32:13669–13679, 2019.

Chris Dongjoo Kim, Jinseo Jeong, and Gunhee Kim. Imbalanced continual learning with partitioning reservoir sampling. In *Proceedings of the European Conference on Computer Vision*, pp. 411–428, 2020.

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.

Xilai Li, Yingbo Zhou, Tianfu Wu, Richard Socher, and Caiming Xiong. Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. In *Proceedings of the International Conference on Machine Learning*, pp. 3925–3934, 2019.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Proceedings of the European Conference on Computer Vision*, pp. 740–755, 2014.

Yaoyao Liu, Yuting Su, An-An Liu, Bernt Schiele, and Qianru Sun. Mnemonics training: Multi-class incremental learning without forgetting. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pp. 12245–12254, 2020.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, pp. 6470–6479, 2017.

German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019.

Jathushan Rajasegaran, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Mubarak Shah. itaml: An incremental task-agnostic meta-learning approach. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pp. 13588–13597, 2020.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pp. 2001–2010, 2017.

Jeffrey S Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software*, 11(1):37–57, 1985.

Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pp. 374–382, 2019.

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

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *Proceedings of the International Conference on Machine Learning*, pp. 3987–3995, 2017.

A APPENDIX

A.1 DETAILS OF DATASET

Previous method (Kim et al., 2020) constructed non-stationary multi-label data stream with imbalanced distribution. However, we find that the proportion of samples with more than one label in their datasets MSCOCO and NUSWIDEseq are too low, which means the model may only needs to pay attention to samples with only one label to control the class distribution in memory. In this work, we construct two new multi-label datasets which contain higher proportion of samples with more than one label. Specifically, we first use the labels with more than one label to construct the training sets of several tasks. Following the previous work (Kim et al., 2020), each task is constructed to have exclusive classes. Then, the remaining samples which are not included in training sets will be used to constructed test sets for each tasks. The test sets are constructed to have equal number of samples for each class like the previous datasets MSCOCOseq and NUSWIDEseq. During this process, some classes will be removed because of their insufficient number of samples for testing, and some samples in training sets may have only one label due to the removed classes. We showed the proportion of the samples with more than one label used in the previous work (Kim et al., 2020) and this work in the Table 4. Figure 4 and Figure 5 show the class distribution in each task of our datasets.

Table 4: The proportion of the samples with more than one label in the datasets we constructed and the previous datasets.

	MSCOCO		NUSWIDE	
	Previous	Ours	Previous	Ours
task1	20.38%	34.40%	8.34%	99.59%
task2	38.07%	80.56%	45.91%	87.40%
task3	8.02%	88.53%	31.58%	91.13%
task4	5.95%	91.12%	5.76%	96.27%
task5	-	-	1.98%	87.89%
task6	-	-	34.40%	-

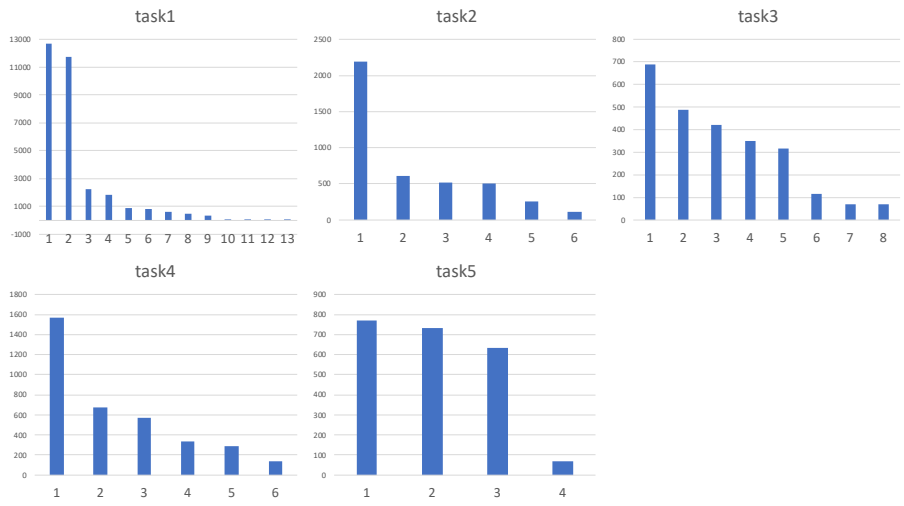


Figure 4: The class distribution in memory after training for NUSWIDE

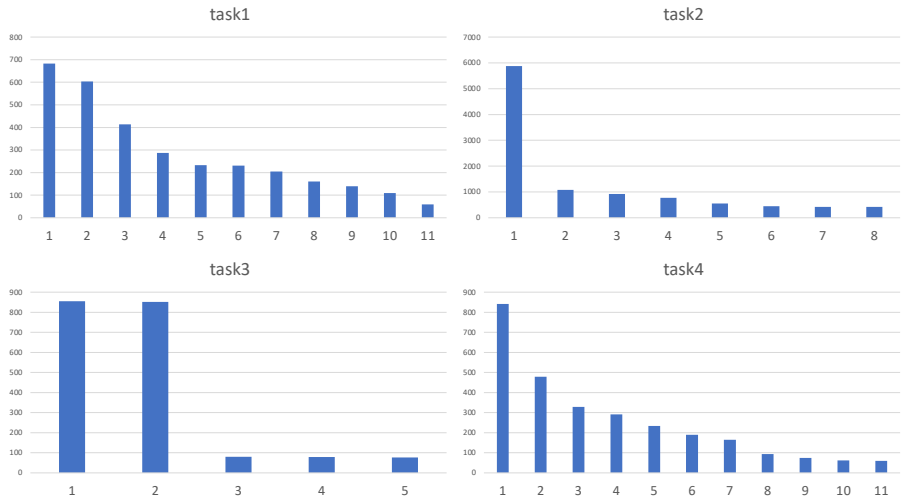


Figure 5: The class distribution for each task on MSCOCO

A.2 Some Additional Experimental Results

We give the class distribution in memory after training on MSCOCO in Figure 6. The class distribution in memory for RS and GSS-Greedy is highly imbalanced just like NUSWIDE. PRS and our method OCDM could keep class distribution balanced in memory. While the class distribution in memory for OCDM is more balanced than PRS. Specifically, in PRS there are 21 classes with more than 40 samples and 6 classes with less than 20 samples in memory. While for our method OCDM, there are only 2 classes with more than 40 samples and no classes with less than 20 samples in memory.

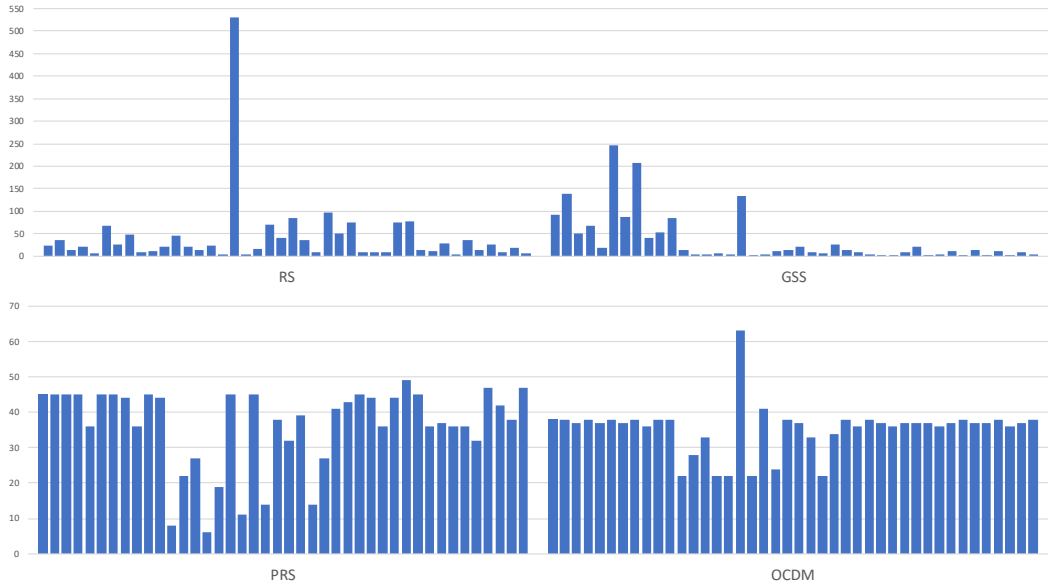
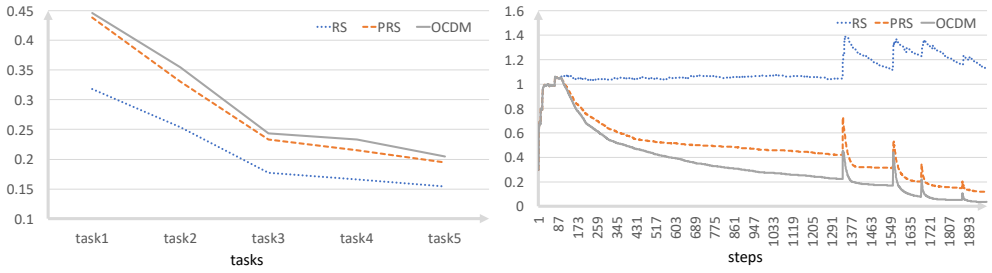


Figure 6: The class distribution for each task on MSCOCO

Figure 7a shows the changes in mAP after each task ends when training on NUSWIDE. The results are similar to MSCOCO. Specifically, the mAP of all methods gradually decreases as the number of tasks increases, but our method is consistently better than PRS and RS.

Figure 7b show the distance change between class distribution in memory and target distribution during the whole training on NUSWIDE, where the KL-difference is used as distance function. The result is similar to the MSCOCO results. Specifically, OCDM can make the class distribution in memory become closer to the target distribution than PRS and RS.



(a) (b)

Figure 7: (a) change of mAP after each task is over on NUSWIDE; (b) change of distance between class distribution in memory and target distribution during training on NUSWIDE.

A.3 Experiment Results on Previous Datasets

We also perform on MSCOCOseq and NUSWIDEseq which is used in the previous work (Kim et al., 2020) and show the mAP results in Table 5. In the Figure 8 and Figure 9 we give the class distribution after training on MSCOCOseq and NUSWIDEseq. The training details and the model architecture are the same with section 4.1. We can find that in these datasets, our method and PRS get similar mAP and similar memory distribution. However, the total number of labels keeped in memory for PRS and OCDM is 1060 and 1010 respectively, which means more than 940 and 990 samples in memory have only one label. While in the experiment on our dataset NUSWIDE, the memory size is also set to 1000 and the number of labels in memory is 1977 and 1911 for PRS and OCDM respectively. This shows that our dataset has a high proportion of samples with multiple labels to some extent.

Table 5: Results of the continual learning method on NUSWIDEseq and MSCOCOseq.

	MSCOCOseq				
Methods	Majority	Moderate	Minority	Total	Time (per 1000 steps)
RS	69.21	46.00	19.04	46.58	576.7
PRS	68.77	50.88	32.08	51.72	1342.7
OCDM (ours)	71.56	51.97	33.59	53.46	599.6
	NUSWIDEseq				
Methods	Majority	Moderate	Minority	Total	Time (per 1000 steps)
RS	27.60	21.06	17.88	21.03	359.9
PRS	26.61	22.29	27.94	25.59	828.0
OCDM (ours)	23.31	22.63	26.64	24.49	368.9

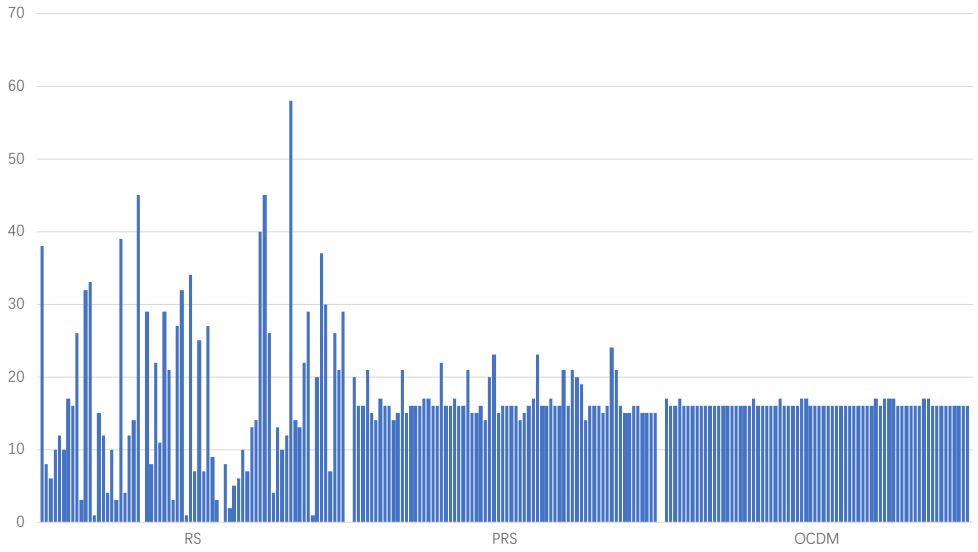


Figure 8: The class distribution in memory after training on MSCOCOseq

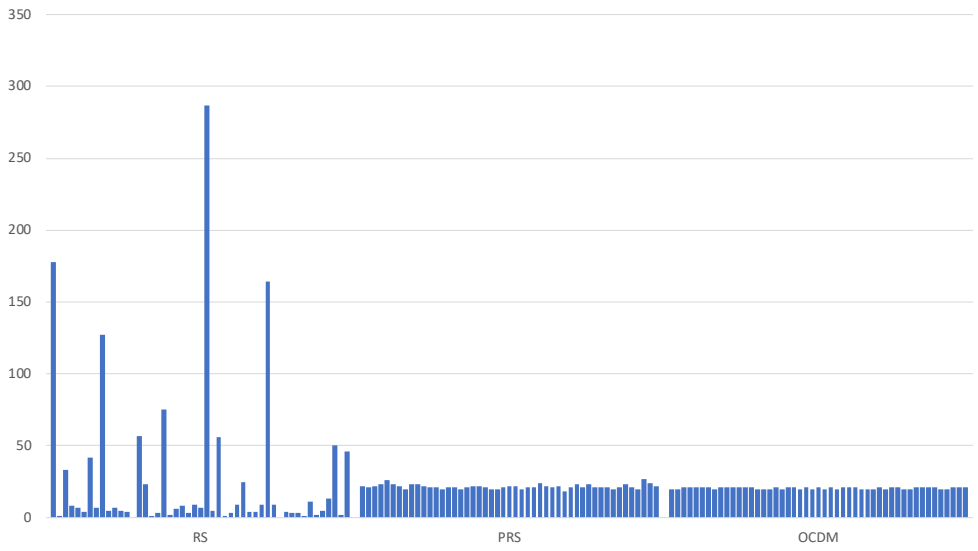


Figure 9: The class distribution in memory after training on NUSWIDEseq