Online Continual Learning Via Candidates Voting

Jiangpeng He
he416@purdue.edu

Fengqing Zhu
zhu0@purdue.edu

School of Electrical and Computer Engineering, Purdue University, West Lafayette, Indiana USA

Abstract

Continual learning in online scenario aims to learn a sequence of new tasks from data stream using each data only once for training, which is more realistic than in offline mode assuming data from new task are all available. However, this problem is still under-explored for the challenging class-incremental setting in which the model classifies all classes seen so far during inference. Particularly, performance struggles with increased number of tasks or additional classes to learn for each task. In addition, most existing methods require storing original data as exemplars for knowledge replay, which may not be feasible for certain applications with limited memory budget or privacy concerns. In this work, we introduce an effective and memory-efficient method for online continual learning under class-incremental setting through candidates selection from each learned task together with prior incorporation using stored feature embeddings instead of original data as exemplars. Our proposed method implemented for image classification task achieves the best results under different benchmark datasets for online continual learning including CIFAR-10, CIFAR-100 and CORE-50 while requiring much less memory resource compared with existing works.

1. Introduction

Continual learning, a promising future learning strategy, is able to learn from a sequence of tasks incrementally using less computation and memory resource compared with retraining from scratch whenever observing a new task. However, it suffers from catastrophic forgetting [32], in which the model quickly forgets already learned knowledge due to the unavailability of old data. Existing methods address this problem under different scenarios including (1) *task-incremental* vs. *class-incremental* depending on whether task index is available and (2) *offline* vs. *online* depending on how many passes are allowed to use each new data. In general, *online class-incremental* methods use each data once to update the model and employs a single-head classifier [31] to test on all classes encountered so far during inference. This setting is more closer to real life learning environment where new classes come in as data streams with limited adaptation time and storage capacity allowed for processing [30]. Unfortunately, class-incremental learning in online scenario is not well-studied compared with offline setting. In addition, existing online methods [8, 37, 34, 13, 2] all require original data from each learned task as exemplars, which restricts their deployment for certain applications (*e.g.*, healthcare and medial research) with memory constraints or privacy concerns. Therefore, an effective online continual learning method is needed to address the above challenges for real world deployment and to improve the performance of online methods.

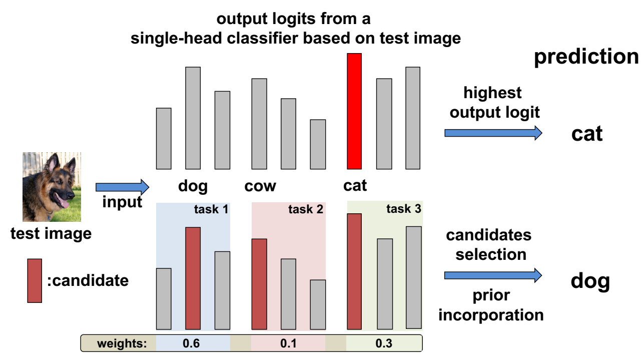


Figure 1: **Illustration of the difference between our proposed method and other methods to make prediction based on output of a single-head classifier.** With single-head classifier, the output is associated with the largest value of the output logits. In contrast, our method makes prediction by first selecting candidates from each learned task and then incorporating the corresponding weights.

For class-incremental methods using a single-head classifier, the prediction result is always associated with the largest value of output logits. However, during continual learning, the output logits become biased towards new task due to the unavailability of old task data [42], *i.e.*, the output logits of new task are much larger than those of old tasks.

This results in the corresponding biased prediction on new tasks, which is a significant contributing factor for catastrophic forgetting. Our method is motivated by the observation that the model is still able to maintain its discriminability for classes within each task [45] despite the bias issue, *i.e.*, the correct class label can be drawn from the candidate prediction given by each learned task during inference. Therefore, our method aims to treat the class label associated with the largest output logit for each learned task as a candidate and the final prediction is based on the weighted votes of all selected candidates. Figure 1 illustrates the main difference between our method and others to make prediction based on the output of a single-head classifier.

To achieve this goal, there are two associated questions we need to address: (1) How to obtain the largest logits as candidates from the output of each learned task using a single-head classifier without knowing the task index? (2) How to generate the weight for each selected candidate to determine the final prediction? In this work, we address both problems by leveraging exemplar set [35], where a small number of old task data is stored for replay during continual learning. However, different from existing methods [8, 37, 34, 13, 2] which use original data as exemplar, we apply a feature extractor and store only feature embeddings, which is more memory-efficient and privacy-preserving. We argue that the task index can be stored together with selected exemplars while learning each new task. Therefore, during inference phase, we can directly obtain the output logits for each learned task from the single-head classifier based on stored task index in the exemplar set and extract the largest output logits. We refer to this as the **candidates selection** process. In addition, we design a probabilistic neural networks [39] leveraging all stored feature embeddings to generate the probability distribution of learned task that the input test data belongs to, and use it as the weights to decide the final prediction. We denote this step as **prior incorporation**. The main contributions are summarized as follows.

- We propose a novel and efficient framework for online continual learning through candidates selection and prior incorporation without requiring original data to reduce the memory burden and address privacy issue for real world applications.
- An online sampler is designed to select exemplars from sequentially available data stream through dynamic mean update criteria and we further study exemplar augmentation in feature space to achieve improved performance
- We conduct extensive experiments on benchmark datasets including CIFAR-10 [24], CIFAR-100 [24] and CORE-50 [28] and show significant improvements compared with existing online methods while requiring the least storage.
- We further show that our online method outperforms state-of-the-art offline continual learning approaches on CIFAR-100 [24] dataset, at the same time it alleviates the weight bias problem and reduces the memory storage consumption compared with existing works.

2. Related Work

Continual learning is studied under different learning scenarios. In general, it can be divided into (1) class-incremental (2) task-incremental and (3) domain-incremental as discussed in [20]. Instead of using a single-head classifier [31] for all classes seen so far in class-incremental setting, methods for task-incremental problem apply a multi-head classifier [1] for each independent task and domain-incremental methods aim to learn the label shift rather than new classes. In addition, depending on whether each data is used more than once to update model, it can be categorized as (1) online learning that use each data once and (2) offline learning with no epoch restriction. In this work, we study the continual learning under online and class-incremental setting, where the model observes each data once and perform classification within all seen classes during inference phase. In this section, we review existing continual learning works related to our method in two categories including (1) Regularization-based and (2) Replay-based methods.

Regularization-based methods restrict the impact of learning new tasks on the parameters that are important for learned tasks. Representative methods include freezing part of layers [21, 23] and using distillation loss or its variants [26, 13, 6, 35, 18, 19, 25, 14]. However, they also limit the model's ability to learn new task and can even harm the performance if the teacher model used by distillation [17] is not learned on large balanced data [5]. Our method applies a fixed backbone model that is pre-trained on large scale datasets to extract feature embeddings of new data as input and uses cross-entropy to learn a discriminative classifier for each new task. Therefore, even though we freeze the parameters for learned tasks in the classifier, it has minimum impact on extracted features to learn new task. Recent studies [42, 45] also found that the bias of model weights towards new classes is one of the reasons for catastrophic forgetting. Therefore, Wu *et al.* [42] proposed to correct the weights by applying an additional linear model. Then Weight Aligning is proposed in [45] to directly correct the biased weights in the FC layer without requiring additional parameters. However, none of these methods are designed for online scenario where each data is only allowed to use once for training. In this work we propose to tackle this problem from a novel perspective by selecting candidates for each learned task and then use the weighted score for final prediction, which effectively addresses catastrophic for-

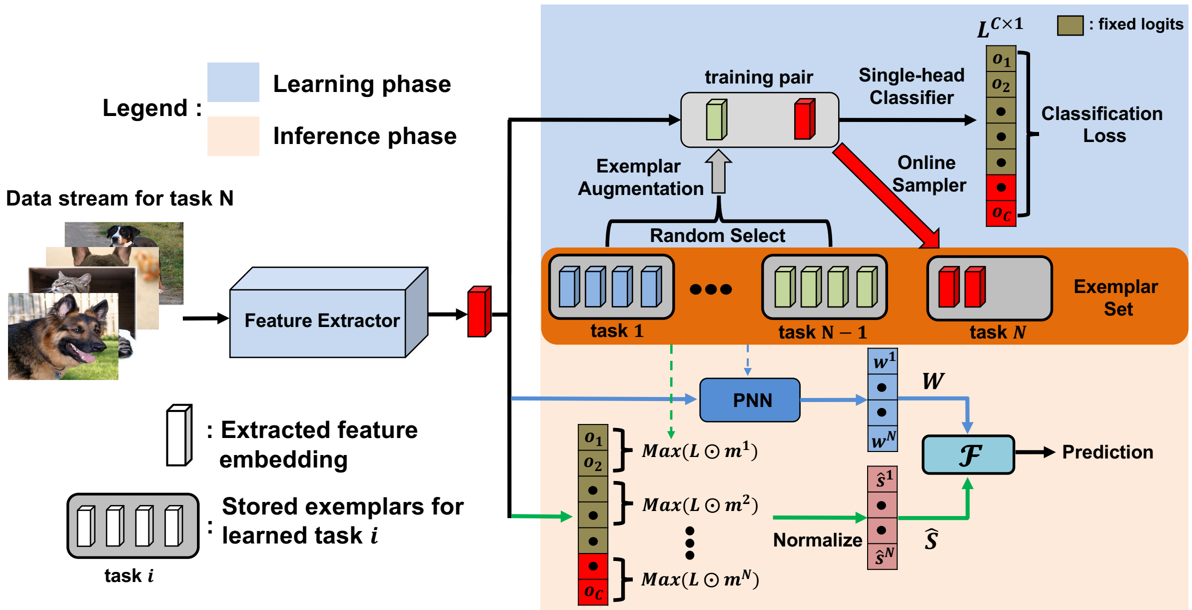


Figure 2: **Overview of our proposed online continual learning method to learn a new task N.** The upper half shows the learning phase where we pair the extracted feature of new data with an exemplar to train the single-head classifier. L denotes the output logits for all classes C seen so far. The parameters for each learned task in the classifier are fixed to maximally maintain its discriminability and an online sampler is designed to select exemplars for current task N. The lower half shows the inference phase where the candidates selection and prior incorporation are denoted by green and blue arrows, respectively. The output logits for each learned task is obtained using element-wise product on classifier output L and binary mask $\{m^i, i = 1, 2, ...N\}$ generated from exemplar set and we treat the highest logits for each task as candidates. A probabilistic neural network (PNN) is designed using all stored exemplars to provide the prior information of which task index the input data belongs to during inference, which can be regarded as weights for selected candidates to obtain the final prediction using our proposed function $\mathcal{F}$. (Best viewed in color)

getting in online case.

Replay-based methods are shown to be effective for maintaining learned knowledge by either using the original data as exemplars [35, 34, 29, 6, 27, 33, 2, 3, 8, 37, 9, 7] or synthetic data and statistics [38, 40, 43, 22]. However, using original data may not be feasible for certain applications due to privacy concerns and also it may require large storage depending on the size of input data. In addition, using synthetic data or data statistic require training a generative model [11] during learning phase, which is not feasible in online scenario. Therefore, we propose to use feature embeddings as exemplars for rehearsal to mitigate forgetting in online case. Besides, we also utilize the stored feature to (1) generate binary masks for each learned task to select candidates and (2) provide prior information as weights to obtain final prediction. We argue that both information are valuable to explore, particularly under the online continual learning context when available resource is limited.

Among these methods, only a few are studied for online mode [29, 34, 2, 3, 37, 8, 9, 7] with even less work under class-incremental setting [34, 3, 2, 37], which is more challenging but also worth investigating as it closely relates to applications in real world scenario.

3. Our Method

The overview of our method is illustrated in Figure 2, including a learning phase to learn new task from a data stream and an inference phase to test for all tasks seen so far. Our method applies a fixed backbone network to extract feature embedding as input, which is more discriminative, memory-efficient and also privacy-preserving compared with using original data. We freeze the parameters in the classifier after learning each new task to maximally maintain its discriminability. We emphasize that our method still uses a single-head classifier but restricts the update of parameters corresponding to all learned tasks.

3.1. Learning Phase

The upper half of Figure 2 shows the learning phase in online scenario where we train the classifier by pairing each extracted feature embedding of the new data with one exemplar randomly selected from exemplar set into the training batch. Cross-entropy is used as the classification loss to up-

Algorithm 1 Online Sampler

Input: Data stream for task N: $\{(\mathbf{x}_1, y_1)^N, (\mathbf{x}_2, y_2)^N, ...\}$
Require: Backbone feature extractor $\mathcal{F}$
Output: Updated exemplar set: $E^{N-1} \rightarrow E^N$

for i = 1, 2, ... **do**
  $v_i \leftarrow \mathcal{F}(\mathbf{x}_i)$ ▷ Extract feature embedding
  $f_m^{(y_i)} \leftarrow \frac{n_{y_i}}{n_{y_i}+1} f_m^{(y_i)} + \frac{1}{n_{y_i}+1} v_i$ ▷ online mean update
  $n_{y_i} \leftarrow n_{y_i} + 1$ ▷ total number of seen data
  if $|E^N(y_i)| < q$ **then** ▷ exemplars for class y_i not full
    $E^N(y_i) \leftarrow E^N(y_i) \cup (\mathbf{v}_i, y_i)^N$
  else
    $I_{max} \leftarrow \mathit{argmax}(||v_j - f_m^{(y_i)}||^2, j \in i \cup E^N(y_i))$
  if $I_{max} \neq i$ **then**
    Remove $(\mathbf{v}_{I_{max}}, y_i)^N$ from $E^N(y_i)$
    $E^N(y_i) \leftarrow E^N(y_i) \cup (\mathbf{v}_i, y_i)^N$
  else
    Continue

date the model, which generates a more discriminative classifier as no regularization term on learned tasks is used. It also does not require additional memory to store the output logits compared with using knowledge distillation loss [17].

Online sampler: There are two necessary conditions we need to satisfy when designing the online sampler for our method: (1) it should be able to select exemplars from sequentially available data in online scenario, (2) the selected exemplars should near the class mean as we will leverage stored features to provide prior information using distance-based metric during inference phase, which is described later in Section 3.2. However, none of the existing exemplar selection algorithms satisfy both conditions. In addition, although Herding [41] is widely applied to select exemplars based on class mean, it only works in offline scenario assuming the data from new task is all available. Therefore, we propose to use an online dynamic class mean update criteria [12] for exemplar selection, which does not require knowing the total number of data beforehand as shown in Equation 1.

$$\mathbf{v}_{mean} = \frac{n}{n+1}\mathbf{v}_{mean} + \frac{1}{n+1}\mathbf{v}_n \tag{1}$$

where n refers to the number of data seen so far in this class and $\mathbf{v}_n$ denotes a new observation. Algorithm 1 illustrates the exemplar selection process for a new task N, where $q = \frac{Q}{|class|}$ denotes the number of allowable exemplars per class given total capacity Q and $f_m^{(y_i)}$ is the mean vector for total n_{y_i} data seen so far for class label y_i. The exemplar set can be expressed as $E = \{(\mathbf{v}_1, y_1)^1, (\mathbf{v}_2, y_2)^1, ..., (\mathbf{v}_1, y_1)^N, (\mathbf{v}_2, y_2)^N, ...\}$, where $(\mathbf{v}_j, y_j)^k$ denotes the j-th stored exemplar for the k-th learned task and $k \in \{1, 2, ..., N\}$. Each stored exemplar contains extracted feature $\mathbf{v}$, class label y and task index k.

Exemplar augmentation in feature space: Although exemplars help to remember learned tasks by knowledge reply during continual learning, the model performance greatly depends on the size of the exemplar set, *i.e.,* the larger the better, which is challenging given a limited memory budget particularly in online scenario. Therefore, we also study the exemplar augmentation techniques in this work to help improve the performance without requiring additional storage. Since we store feature embedding as exemplar, common data augmentation methods that are typically applied to image data such as rotation, flip and random crop cannot be used directly in feature space. Therefore, we adopt random perturbation for feature augmentation [10].

Random perturbation: We generate pseudo feature exemplar by adding a random vector P drawn from a Gaussian distribution with zero mean and per-element standard deviation σ as shown in Equation 2

$$\tilde{\mathbf{v}}_i = \mathbf{v}_i + \alpha_r P, \quad P \sim N(0, \sigma_i) \tag{2}$$

where $\mathbf{v}_i$ refers to the stored feature in exemplar set, and $\tilde{\mathbf{v}}_i$ denotes the augmented feature. α_r is a constant which controls the scale of noise, and is set to $\alpha_r = 1$ in our implementation. We emphasize that we do not need to store augmented feature in exemplar set and the exemplar augmentation is randomly implemented when pairing the extracted feature of new data.

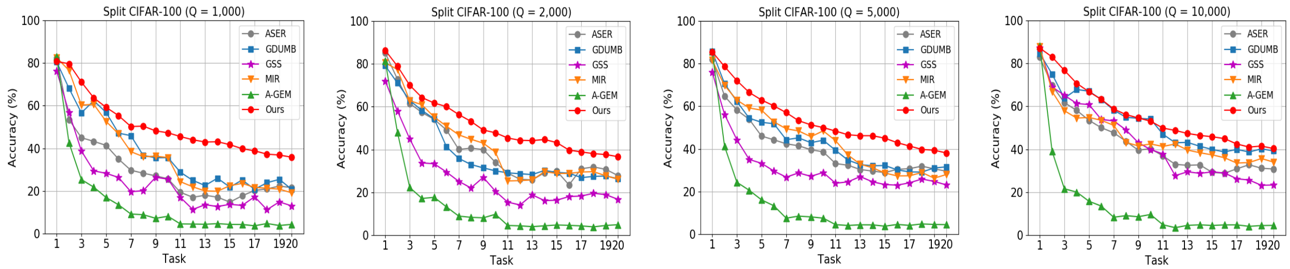
3.2. Inference Phase

The lower half of Figure 2 shows inference phase, which comprises of two key components: candidates selection and prior incorporation. The stored exemplars along with their task indexes are used to generate binary mask to obtain the corresponding output logits for each learned task during inference. We extract the highest output as candidates and a variant of probabilistic neural network (PNN) [39] using all stored exemplars is designed to provide prior information as weights for selected candidates to vote for final prediction, which will be described in detail below.

Candidates selection: We denote $L = \{o^1, o^2, ..., o^C\}$ as the output logits from the single-head classifier where C refers to the total number of seen classes belonging to N learned tasks so far. During inference phase, the exemplar set generates a binary mask $m^k \in \{0, 1\}^C$ for task k by assigning the i-th entry m_i^k as 1 if class label i belongs to task k and as 0 if not, so we have $\sum_{i=1}^{C} m_i^k = C^k$, where C^k is the number of classes belonging to task k. Thus, the candidate output logit from each learned task is selected by

$$s^k = \mathit{Max}\{L \odot m^k\}, \quad k = 1, 2, ..., N \tag{3}$$

where $\odot$ refers to element-wise product. We then perform normalization step for the extracted candidate logits by using the corresponding norm of weight vectors in classifier. Specifically, for each selected candidate s^k, let



(a) (b) (c) (d)

Figure 3: **Results on Split CIFAR-100** by comparing with existing online methods with different exemplar size Q. The accuracy is measured after learning of each task on all tasks seen so far. (Best viewed in color)

Datasets	Split CIFAR-10								CORE-50							
Size of exemplar set	$Q=1,000$		$Q=2,000$		$Q=5,000$		$Q=10,000$		$Q=1,000$		$Q=2,000$		$Q=5,000$		$Q=10,000$	
Accuracy(%)	Avg	Last	Avg	Last	Avg	Last	Avg	Last	Avg	Last	Avg	Last	Avg	Last	Avg	Last
A-GEM [8]	43.0	17.5	59.1	38.3	74.0	59.0	74.7	62.5	20.7	8.4	21.9	10.3	22.9	11.5	24.6	12.0
MIR [2]	67.3	52.2	80.2	66.2	83.4	74.8	86.0	78.4	33.9	21.1	37.1	24.5	38.1	27.7	41.1	31.8
GSS [3]	70.3	56.7	73.6	56.3	79.3	64.4	79.7	67.1	27.8	17.8	31.0	18.9	31.8	21.1	33.6	22.6
ASER [37]	63.4	46.4	78.2	59.3	83.3	73.1	86.5	79.3	24.3	12.2	30.8	17.4	32.5	18.5	34.1	21.8
GDUMB [34]	73.8	57.7	83.8	72.4	85.3	75.9	87.7	82.3	41.2	23.6	48.4	32.7	54.3	41.6	56.1	45.5
Ours	**76.0**	**62.9**	**84.9**	**74.1**	**86.1**	**77.0**	**88.3**	**82.7**	**45.1**	**26.5**	**50.7**	**34.5**	**56.3**	**43.1**	**57.5**	**46.2**

Table 1: **Average accuracy and Last step accuracy on Split CIFAR-10 and CORE-50**. Best results marked in bold.

$W^k \in \mathcal{R}^{d_m \times 1}$ and $|W^k|$ denotes the weight vector in classifier and its norm respectively where d_m is the input dimension. Then we normalize each candidate with

$$\hat{s}^k = \frac{1}{|W^k|} \frac{s^k - Min\{s^1, ...s^N\}}{\epsilon_n + \sum_{j=1}^{N}(s^j - Min\{s^1, ...s^N\})}$$

where ϵ_n is for regularization and larger $\hat{s}$ can reflect higher probability as prediction. Finally, the normalized selected candidates for N learned tasks can be expressed as $\hat{S} = \{\hat{s}^1, \hat{s}^2, ..., \hat{s}^N\}$ with corresponding extracted candidate class labels $Y = \{y^1, y^2, ..., y^N\}$.

Prior incorporation: We apply PNN to generate prior probability distribution of which learned task index the test data belongs to. PNN computes class conditional probabilities using all stored features in the exemplar set. Specifically, it calculates the probability that an input feature vector $\mathbf{f}$ belongs to task k as formulated in Equation 4 below.

$$\begin{aligned} P(k|\mathbf{f}) &= \frac{\alpha^k}{\sum_{i=1}^{N} \alpha^i} \\ \alpha^k &= (\epsilon_r + Min_j ||\mathbf{f} - \mathbf{v}_j^k||_2))^{-1} \end{aligned} \tag{4}$$

where $\epsilon_r > 0$ is used for regularization and $\mathbf{v}_j^k$ denotes the j-th stored feature in exemplar set for learned task k.

The output of PNN is a N dimension prior vector $W = (w^1, w^2, ..., w^N)$ and we use it as the weights to combine with the normalized candidates $\hat{S}$ to get final predicted class label $\hat{y}$ using Equation 5.

$$\hat{y} = \underset{y^i \in Y}{argmax}(\hat{s}^i + e^{(\gamma-1)} \times w^i) \tag{5}$$

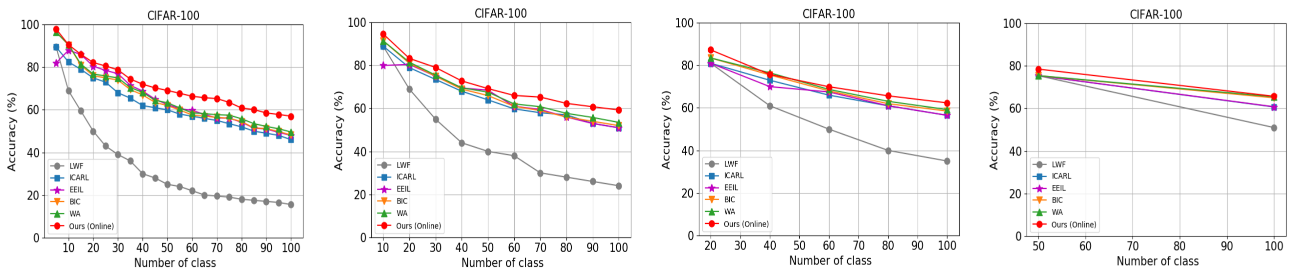
where $\gamma = \frac{Max(W) - Min(W)}{\beta}$ is a dynamic hyper-parameter used for incorporation determined by calculating difference between maximum and minimum value in prior vector. $\beta \in (0, 1)$ is a normalization constant. In this work, we show the effectiveness of our method by using a fixed $\beta = 0.5$ for all experiments.

4. Experimental Results

To show the effectiveness of our proposed approach, we compare with both the state-of-the-art *online methods* following experiment setting similar in [29, 8], and *offline continual learning methods* as well under benchmark protocol [35] by varying the incremental step size, which are illustrated in Section 4.2 and Section 4.3, respectively. In Section 4.4, we conduct ablation experiments to validate each component of our propose method. Finally, we study the weight bias problem in online scenario and analyze the storage consumption in Section 4.5.

4.1. Evaluation Metrics

We focus on continual learning under class-incremental setting as illustrated in Section 2. During inference, the model is evaluated to classify all classes seen so far. We use commonly applied evaluation metrics such as average accuracy (*Avg*) and last step accuracy (*Last*) in this section where *Avg* is calculated by averaging all the accuracy obtained after learning of each task, which shows the overall performance for the entire continual learning procedure. The *Last* accuracy shows the performance after the continual learning for all seen classes. No task index is provided during inference and we ran each experiment five times and



(a) (b) (c) (d)

Figure 4: **Results on CIFAR-100** by comparing with offline approaches with step size (a) 5, (b) 10, (c) 20 and (d) 50. Note that only our method is implemented in online. (Best viewed in color)

report the average Top-1 classification results.

4.2. Compare With Online Methods

We compare our method with existing *replay-based* online approaches including A-GEM [8], GSS [3], MIR [2], ASER [37] and GDUMB [34].

Dataset: We use Split CIFAR-10 [4], Split CIFAR-100 [44] and CORE-50 [28] for evaluation in this part.

- **Split CIFAR-10** splits CIFAR-10 dataset [24] into 5 tasks with each contains 2 disjoint classes. Each class contains 6,000, 32×32 RGB images with originally divided 5,000 for training and 1,000 for testing.
- **Split CIFAR-100** contains 20 tasks with non-overlapping classes constructed using CIFAR-100 [24]. Each task contains 2,500 training images and 500 test images corresponding to 5 classes.
- **CORE-50** is another benchmark dataset for continual learning. For class incremental setting, it is divided into 9 tasks and has a total of 50 classes with 10 classes in the first task and 5 classes in the subsequent 8 tasks. Each class has around 2,400, 128×128 RGB training images and 900 testing images.

Implementation detail: A small version of ResNet-18 (reduced ResNet-18) [29, 8] pretrained on ImageNet [36] is applied as the backbone model for all the compared methods. The ResNet implementation follows the setting as suggested in [16]. We emphasize that only our method freeze the parameters in backbone network while others do not. We apply SGD optimizer with a mini-batch size of 10 and a fixed learning rate of 0.1. We vary the size of exemplar set for $Q \in \{1000, 2000, 5000, 10000\}$ for comparisons.

4.2.1 Results on Benchmark Datasets

The average accuracy (*Avg*) and last step accuracy *Last* on Split CIFAR-10 and CORE-50 are summarized in Table 1. Given different exemplar size Q, our method outperforms existing online approaches, especially when Q is smaller by a larger margin, *i.e.*, our method performs better even with limited storage capacity. The reason is that our approach does not solely rely on exemplars to retain old knowledge but maintains the classifier's discriminability for each learned task and makes the prediction through candidates selection and prior incorporation. In addition, our method includes the exemplar augmentation step, which is more effective given limited number of exemplars as analyzed in Section 4.4. In addition, Figure 3 visualizes the results for continual learning of 20 tasks on Split CIFAR-100. The model is evaluated after learning each task on test data belonging to all classes seen far. Our method achieves the best performance for each step and we observe that A-GEM [8] does not work well under class-incremental setting, which only use stored exemplars to restrict the update of corresponding parameters while others perform knowledge replay by combining with new class data.

4.3. Compare With Offline Methods

While focusing on online continual learning, we also compare our method with offline continual learning approaches that use each data multiple times to update the model. Although it is widely acknowledged that performance in the online scenario is worse than offline as discussed in [29, 34] due to the limited number of available new data and each data is observed only once by the model, we show that our method implemented in online scenario is also effective to achieve comparable performance with state-of-the-arts offline approaches including LWF [26], ICARL [35], EEIL [6], BIC [42] and WA [45] following the benchmark protocol similar in [35].

Datasets: We use CIFAR-100 [24] for evaluation and arrange it into splits of 5, 10, 20, and 50 non-overlapped classes, resulting in 20, 10, 5, and 2 tasks, respectively.

Implementation detail: For experiments on CIFAR-100, we apply ResNet-50 [16] pretrained on ImageNet [36] as the backbone model. We apply SGD optimization with mini-batch size of 10 and a fixed learning rate of 0.1 for our method implemented in online scenario. For all the experiments, we arrange classes using identical random seed [35] and use fixed size of exemplar set as $Q = 2,000$.

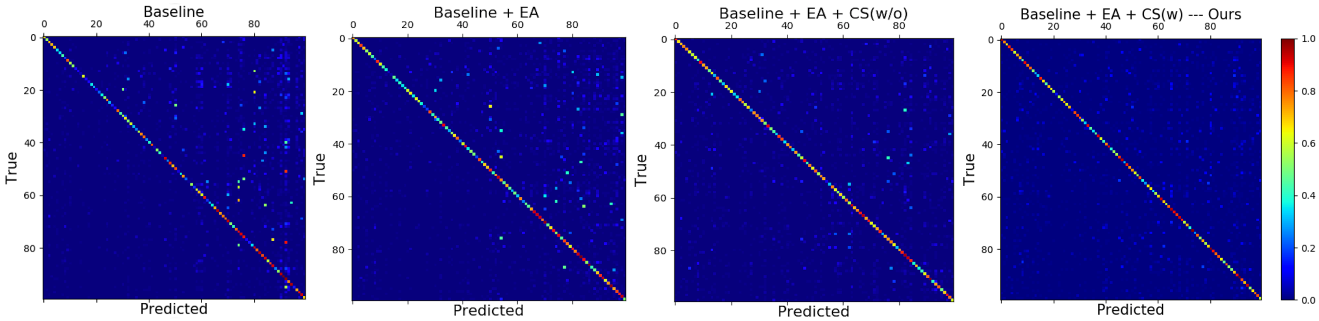


Figure 5: **Confusion matrices on Split CIFAR-100** for different variants in ablation study. (Best viewed in color)

Method	CIFAR-10	CIFAR-100	CORE-50
Baseline	56.2	16.7	19.8
Baseline + EA	58.9	20.1	22.4
Baseline + EA + CS(w/o)	81.7	49.6	43.9
Baseline + EA + CS(w) - Ours	**84.9**	**52.0**	**50.7**
Upper-bound	92.2	70.7	67.9

Table 2: **Average accuracy (%) for ablation study on Split CIFAR-10, Split CIFAR-100 and CORE-50**. Best results (except upper-bound) are marked in bold.

Method	CIFAR-10	CIFAR-100	CORE-50
Baseline (Q=1,000)	46.6	13.9	17.2
Baseline + EA	49.8 (**+3.2**)	18.5 (**+4.6**)	20.6 (**+3.4**)
Baseline (Q=5,000)	54.9	23.8	25.4
Baseline + EA	56.2 (+1.3)	25.4 (+1.6)	26.9 (+1.5)
Baseline (Q=10,000)	57.2	26.8	31.4
Baseline + EA	58.1 (+0.9)	27.4 (+0.6)	31.9 (+0.5)

Table 3: **Performance of exemplar augmentation step** for the exemplar size $Q \in \{1000, 5000, 10000\}$. Average accuracy (%) and the corresponding improvements compared with baseline are reported. Highest improvements are marked in bold for each dataset.

4.3.1 Results on CIFAR-100

We implement our proposed method in online scenario to use each data only once for training (except for the first task, which is learned in offline under this protocol), while all the compared existing methods are implemented in offline for all tasks. The results on CIFAR-100 for each incremental step are shown in Figure 4. Our method still achieves the best results for all incremental step sizes particularly for smaller step size. One of the reasons is that the weight bias problem becomes more severe with smaller incremental step size (more incremental steps) especially in offline case where the model is updated multiple times for each step, which is analyzed in Section 4.5. However, this problem is alleviated in online scenario by our proposed learning strategies to pair each new data with an exemplar as described in Section 3.1. Furthermore, our method for inference further mitigate the bias problem by selecting candidates and incorporating prior information using stored exemplars, which is illustrated later in Section 4.4.

4.4. Ablation Study

We also conduct ablation study to analyze the effectiveness of each component in our proposed method including *exemplar augmentation in feature space* (EA) and *candidates selection with prior incorporation* (CS) as illustrated in Section 3.1 and 3.2, respectively. Specifically, we consider the following variants of our method.

- **Baseline:** remove both CS and EA from our method while keeping exemplar set
- **Baseline + EA:** perform exemplar augmentation
- **Baseline + EA + CS(w/o):** select candidates using stored exemplar but without prior incorporation, which completely trusts the result of PNN by assigning the class of the closest store example as final prediction
- **Baseline + EA + CS(w):** Our proposed method with prior incorporation using Equation 5

We also include **Upper-bound** for comparison, which is obtained by training a model in non-incremental setting using all training samples from all classes together. We fix the size of exemplar set for $Q = 2,000$ and the average accuracy are summarized in Table 2. We observe large improvements by adding candidates selection step and our proposed prior incorporation method outperforms directly using PNN output as prediction. The main reason is that the stored feature embeddings extracted by a fixed pre-trained model may not be discriminative enough to make decision especially when there exists obvious distribution difference between the training and testing data as in CORE-50 [28], where the data are collected in distinct sessions (such as indoor or outdoor). Therefore, our proposed prior incorporation step mitigate this problem and achieves the best performance. In addition, we also provide confusion matrices as shown in Figure 5 to analyze the results in detail where the **Baseline** tends to predict new classes more frequently and ours is able to treat new classes and old classes more fairly. Finally, we analyze the exemplar augmentation (EA) by varying exemplar size Q and results are summarized in Table 3.

Our EA works more efficiently given limited storage capacity, which is one of the most significant constraints to apply continual learning in real world applications.

4.5. Weight Bias And Storage Consumption

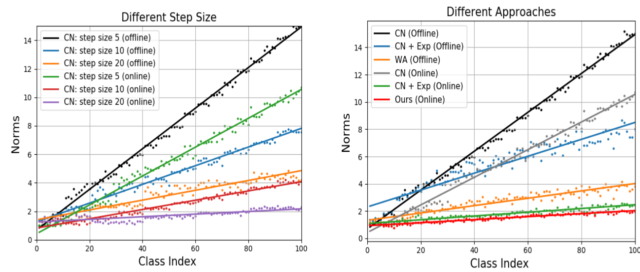
In this section, we implement additional experiments to show the advantages of our proposed method in online scenario including the analysis of norms of weight vectors in classifier and the comparisons of storage consumption.

Norms of weight vectors: One of the main reasons for catastrophic forgetting is the weights in trained model's FC layer are heavily biased towards new classes, which is already discussed in offline mode [42, 45] but lacks sufficient study in online scenario. Therefore, we provide analysis for the impact on biased weights in online and offline scenarios by (1) varying incremental step size and (2) with or without using exemplar set (Exp). For generality, we consider **CN** and **CN + Exp** as two baseline methods using regular cross entropy for continual learning without and with exemplars, respectively. We use CIFAR-100 with step size 5, 10 and 20 for experiments. We train 70 epochs in offline as in [35, 6] and 1 epoch in online scenario for each learning step. Results are shown in Figure 6. Each dot corresponds to the norm of the weight vectors in FC layer for each class. For better visualization, we fit the dots using linear least square to show the trend of each method when new classes are added sequentially.

We observe that the weight bias problem is getting more severe when the number of incremental steps increases, especially in offline case since we repeatedly update model using only new class data. The overall performance in online scenario is much better than offline as each data is used only once for training.

Next, we show that using exemplars is effective to correct biased weights in both online and offline scenario as indicated by **CN+EXP** compared to **CN**. We additionally compare baseline methods with our methods **Ours** and applying Weight Aligning [45] denoted as **WA** for bias correction. The performance of using exemplars in online scenario is even better than applying WA in offline case and our proposed strategy further alleviate this problem. Both analysis explain the larger gains we achieved for smaller step size on CIFAR-100 as discussed in Section 4.3.1. The comparison between online and offline results also show the potential to address catastrophic forgetting in online scenario with the benefit of reduced weight bias problem.

Storage consumption: Storage requirement poses significant constrains for continual learning in online mode. If we can store all data seen so far without considering storage requirement in real world scenario, then we can easily update the model using all available data. Therefore, we compare the storage consumption of our method with existing approaches to show the significant reduction in storage requirement. Let S denote the image size, C denote the number of total classes seen so far, Q refers to the number of data stored in exemplar set for each class and D denotes the dimension of extracted feature embedding. (1) For methods using original data as exemplars [34, 2, 3, 29, 8, 9, 7, 35, 6, 42, 45, 13], the storage requirement for storing data in exemplar set is $O(3 \times S^2 \times Q \times C)$. (2) For methods which store statistics of old classes and conduct pseudo rehearsal [22, 43], the total cost is $O(D^2 \times C)$ (3) For our method that store feature embeddings as exemplars, the total storage is $O(D \times C \times Q)$. Therefore, as $Q \ll D < 3 \times S^2$, our method requires the least storage while still achieving the best performance.



(a) (b)

Figure 6: **Norms of the weight vectors** for (a) the impact of different step size 5, 10, and 20. (b) Impact of different methods using step size 5. The solid line is obtained by linear least square to show the trend for each case.

5. Conclusion

In summary, we propose a novel and effective method for continual learning in online scenario under class-incremental setting by maintaining the classifier's discriminability for classes within each learned task and make final prediction through candidates selection together with prior incorporation using stored exemplars selected by our online sampler. Feature embedding instead of original data is stored as exemplars, which are both memory-efficient and privacy-preserving for real life applications and we further explore exemplar augmentation in feature space to achieve improved performance especially when given very limited storage capacity. Our method achieves best performance compared with existing online approaches on benchmark datasets including Split CIFAR10, Split CIFAR100 and CORE-50. In addition, we vary the incremental step size and achieves comparable performance even with offline approaches on CIFAR-100. Finally, our analysis on norms of weight vectors in the classifier also shows great potential for addressing catastrophic forgetting in online scenario that can significantly reduce the weight bias problem. Our future work will focus on unsupervised continual learning, which is more realistic and one possible solution is to use pseudo label as recently introduced in [15].

References

[1] Davide Abati, Jakub Tomczak, Tijmen Blankevoort, Simone Calderara, Rita Cucchiara, and Babak Ehteshami Bejnordi. Conditional channel gated networks for task-aware continual learning. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3931–3940, 2020.

[2] Rahaf Aljundi, Eugene Belilovsky, Tinne Tuytelaars, Laurent Charlin, Massimo Caccia, Min Lin, and Lucas Page-Caccia. Online continual learning with maximal interfered retrieval. *Advances in Neural Information Processing Systems*, pages 11849–11860, 2019.

[3] Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. *Advances in Neural Information Processing Systems*, 32, 2019.

[4] Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Online continual learning with no task boundaries. *arXiv preprint arXiv:1903.08671*, 2019.

[5] Eden Belouadah and Adrian Popescu. Il2m: Class incremental learning with dual memory. *Proceedings of the IEEE International Conference on Computer Vision*, pages 583–592, 2019.

[6] Francisco M. Castro, Manuel J. Marin-Jimenez, Nicolas Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. *Proceedings of the European Conference on Computer Vision*, September 2018.

[7] Arslan Chaudhry, Albert Gordo, Puneet K Dokania, Philip Torr, and David Lopez-Paz. Using hindsight to anchor past knowledge in continual learning. *arXiv preprint arXiv:2002.08165*, 2020.

[8] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420*, 2018.

[9] Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*, 2019.

[10] Terrance DeVries and Graham W. Taylor. Dataset augmentation in feature space. *arXiv preprint arXiv:1702.05538*, 2017.

[11] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Proceedings of the Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[12] Samantha Guerriero, Barbara Caputo, and Thomas Mensink. Deepncm: Deep nearest class mean classifiers. 2018.

[13] Jiangpeng He, Runyu Mao, Zeman Shao, and Fengqing Zhu. Incremental learning in online scenario. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 13926–13935, 2020.

[14] Jiangpeng He and Fengqing Zhu. Online continual learning for visual food classification. *arXiv preprint arXiv:2108.06781*, 2021.

[15] Jiangpeng He and Fengqing Zhu. Unsupervised continual learning via pseudo labels. *arXiv preprint arXiv:2104.07164*, 2021.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[17] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *Proceedings of the NIPS Deep Learning and Representation Learning Workshop*, 2015.

[18] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Lifelong learning via progressive distillation and retrospection. *Proceedings of the European Conference on Computer Vision*, pages 437–452, 2018.

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

[20] Yen-Chang Hsu, Yen-Cheng Liu, Anita Ramasamy, and Zsolt Kira. Re-evaluating continual learning scenarios: A categorization and case for strong baselines. *arXiv preprint arXiv:1810.12488*, 2018.

[21] Heechul Jung, Jeongwoo Ju, Minju Jung, and Junmo Kim. Less-forgetting learning in deep neural networks. *arXiv preprint arXiv:1607.00122*, 2016.

[22] Ronald Kemker and Christopher Kanan. Fearnet: Brain-inspired model for incremental learning. *arXiv preprint arXiv:1711.10563*, 2017.

[23] 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. *The National Academy of Sciences*, 114(13):3521–3526, 2017.

[24] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[25] Kibok Lee, Kimin Lee, Jinwoo Shin, and Honglak Lee. Overcoming catastrophic forgetting with unlabeled data in the wild. *Proceedings of the IEEE International Conference on Computer Vision*, pages 312–321, 2019.

[26] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, 2017.

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

[28] V Lomanco and Davide Maltoni. Core50: a new dataset and benchmark for continual object recognition. *Proceedings of the 1st Annual Conference on Robot Learning*, pages 17–26, 2017.

[29] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, pages 6467–6476, 2017.

[30] Zheda Mai, Ruiwen Li, Jihwan Jeong, David Quispe, Hyunwoo Kim, and Scott Sanner. Online continual learning in image classification: An empirical survey. *arXiv preprint arXiv:2101.10423*, 2021.

[31] Davide Maltoni and Vincenzo Lomonaco. Continuous learning in single-incremental-task scenarios. *Neural Networks*, 116:56–73, 2019.

[32] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of Learning and Motivation*, volume 24, pages 109–165. Elsevier, 1989.

[33] Jary Pomponi, Simone Scardapane, Vincenzo Lomonaco, and Aurelio Uncini. Efficient continual learning in neural networks with embedding regularization. *Neurocomputing*, 397:139–148, 2020.

[34] Ameya Prabhu, Philip HS Torr, and Puneet K Dokania. Gdumb: A simple approach that questions our progress in continual learning. *European Conference on Computer Vision*, pages 524–540, 2020.

[35] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. iCaRL: Incremental classifier and representation learning. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, July 2017.

[36] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[37] Dongsub Shim, Zheda Mai, Jihwan Jeong, Scott Sanner, Hyunwoo Kim, and Jongseong Jang. Online class-incremental continual learning with adversarial shapley value. *arXiv preprint arXiv:2009.00093*, 2020.

[38] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. *Proceedings of the Advances in Neural Information Processing Systems*, pages 2990–2999, 2017.

[39] Donald F Specht. Probabilistic neural networks. *Neural networks*, 3(1):109–118, 1990.

[40] Ragav Venkatesan, Hemanth Venkateswara, Sethuraman Panchanathan, and Baoxin Li. A strategy for an uncompromising incremental learner. *arXiv preprint arXiv:1705.00744*, 2017.

[41] Max Welling. Herding dynamical weights to learn. *Proceedings of the International Conference on Machine Learning*, pages 1121–1128, 2009.

[42] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 2019.

[43] Ye Xiang, Ying Fu, Pan Ji, and Hua Huang. Incremental learning using conditional adversarial networks. *Proceedings of the IEEE International Conference on Computer Vision*, pages 6619–6628, 2019.

[44] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. *International Conference on Machine Learning*, pages 3987–3995, 2017.

[45] Bowen Zhao, Xi Xiao, Guojun Gan, Bin Zhang, and Shu-Tao Xia. Maintaining discrimination and fairness in class incremental learning. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 13208–13217, 2020.