Data Efficient Incremental Learning via Attentive Knowledge Replay

Yi-Lun Lee† Dian-Shan Chen† Chen-Yu Lee‡ Yi-Hsuan Tsai‡ Wei-Chen Chiu†
†*Department of Computer Science, National Yang Ming Chiao Tung University, Hsinchu, Taiwan* ‡*Google*
†{yilunlee850704.cs09, dianshan14.c, walon}@nycu.edu.tw
‡{chenyulee260, wasidennis}@google.com

***Abstract*— Class-incremental learning (CIL) tackles the problem of continuously optimizing a classification model to support growing number of classes, where the data of novel classes arrive in streams. Recent works propose to use representative exemplars of learnt classes, and replay the knowledge of them afterward under certain memory constraints. However, training on a fixed set of exemplars with an imbalanced proportion to the new data leads to strong biases in the trained models. In this paper, we propose an attentive knowledge replay framework to refresh the knowledge of previously learnt classes during incremental learning, which generates virtual training samples by blending between pairs of data. Particularly, we design an attention module that learns to predict the adaptive blending weights in accordance with their relative importance to the overall objective, where the importance is derived from the change of the image features over incremental phases. Our strategy of attentive knowledge replay encourages the model to learn smoother decision boundaries and thus improves its generalization beyond memorizing the exemplars. We validate our design in a standard class-incremental learning setup and demonstrate its flexibility in various settings.**

I. Introduction

The ability of incrementally learning new knowledge is essential for real-world learning agents. In recent years, we have witnessed the great capacity of deep networks with their learning abilities being driven by large-scale training data. However, most deep networks still suffer from the issue of "catastrophic forgetting" – the knowledge learnt from new classes tends to override the old one learnt from previously seen classes due to the lack of access to the old data, leading to performance decrease of old classes.

To mitigate the forgetting, many works have been proposed [1], [2], [3], [4], [5], [6] lately: LwF [1] adopts knowledge distillation to retain the posterior distributions of old classes while learning the new classes; iCaRL [2] proposes to explicitly set up a replay buffer which stores subsets of data from learnt classes as *exemplars* to preserve the previously acquired knowledge. Such a scheme can largely alleviate the forgetting issue, by equipping a replay buffer that updates the classification model on both the exemplars and the data samples of novel classes, in which it becomes a widely-used protocol for class-incremental learning nowadays.

However, as the replay buffer is with limited size in practice, it would inevitably result in the imbalance on the number of training samples between new and old classes, causing the trained classifier to have biased weights among classes and prone to over-memorize exemplars stored in the buffer. Here we perform a simple case study to better illustrate the data imbalance issue, as shown in Figure 1. We follow the iCaRL [2] approach to build a base model, and then we evaluate the top-1 accuracy on exemplars and testing data of an old class in each incremental phase (i.e., when novel classes arrive), as well as the average L2 norm of classifier weights of the final model (cf. Figure 1 (a) and (c) respectively). As a reference, we also include the performance of the base model trained with full training data as the upper bound for comparison. As the training proceeds along the incremental phases, the model is able to perform extremely well on the exemplars stored in the buffer, which are used as training samples in all phases during incremental learning. Nevertheless, due to the extreme imbalance between the exemplars and the data of new classes (see Figure 1 (b)), the classifier is biased to new classes and tends to forget old classes. As a result, the model has a significant performance decrease on the test set for the old classes. The phenomenon indicates that the common exemplars replay is not sufficient for the model to preserve knowledge of old classes. Therefore, how to replay exemplars both efficiently and effectively, which avoids over-memorizing exemplars and the biased classifier, is the main issue we address in this paper.

To this end, we propose a simple yet effective attentive knowledge replay framework that linearly performs blending between exemplars, or the ones across exemplars and newly arriving training samples to better retain the knowledge of learnt classes. Instead of randomly sampling weights for blending, we design a novel attention module to learn suitable blending weights of the given pairs of data. By leveraging the change of image features extracted by the old model (trained in the previous incremental phase) and the current model, our attention module learns to estimate the proper blending weights for the image pair according to their relative importance to the objective of class-incremental learning. The mechanism of dynamically replaying exemplars helps the classification model learn smoother decision boundaries to avoid over-memorizing the stored exemplars. Moreover, our proposed framework is able to benefit the classification performance under different settings of class-incremental learning, showing the efficacy and applicability of our method.

II. Related Work

We group class-incremental learning methods into following three categories and review them briefly:
Parameter restriction methods [7], [8] estimate the importance of model parameters for different classes. While

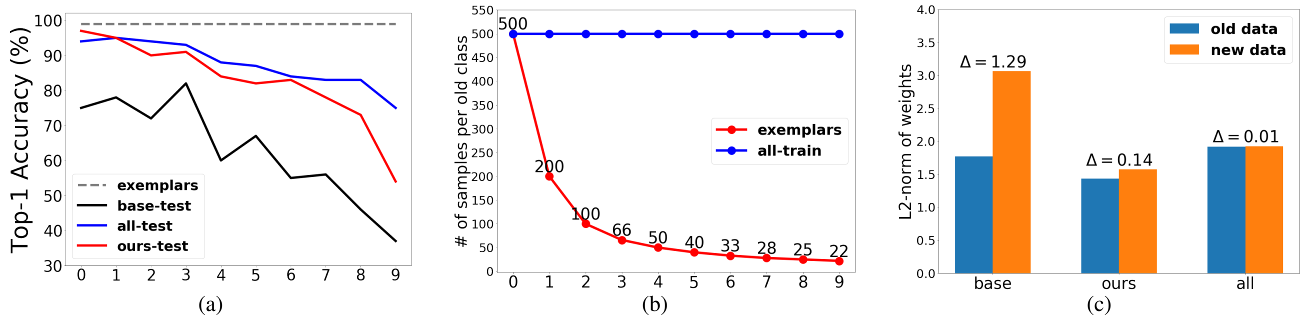

Fig. 1: Study for the issue of over-memorizing exemplars. (a) shows the incremental accuracy of an old class which is learnt at the beginning. The dashed and solid lines represent the accuracy on the exemplars and testing data (-test) respectively. The line in black color is for the base model which follows the similar model design of iCaRL, while the one in red color is for our method. The line in blue color is the base model which is trained with all the training data of old class instead of using only exemplars. (b) shows that as the size of replay buffer is limited, the number of exemplars for each class decreases along the incremental phases from 500 images in the beginning to 22 images at the end. (c) compares the average L_2 norm of the classification weight vectors for old and new classes after training. We see that the base model suffers from memorizing the exemplars and results in large performance gap between the exemplars and testing data, while our proposed method (i.e. equipping base model with our attentive knowledge replay framework) can better alleviate such issue and have smaller performance gap.

some parameters are identified as important ones to the previously learnt classes, a relatively large penalty is imposed on the change of those parameters when learning new classes. On the other hand, the other parameters would receive more flexibility to be updated to support new classes. The difference among this line of work is the way to identify the parameter importance for classes. For instance, EWC [7] adopts the Fisher information matrix, while MAS [8] uses the gradients of the l_2 norm of model outputs with respect to the parameters. However, the computation of identifying important parameters and performing regularization is typically costly.

Experience replay methods [2], [4], [6], [3], [5] train the model by simultaneously leveraging the exemplars of old classes and the training samples of new classes to replay the previous experience and learn the novel concept respectively, where the exemplars are maintained in a replay buffer with limited size. For instance, iCaRL [2] proposes to use herding algorithm for selecting exemplars and adopt the knowledge distillation to retain the previously learnt knowledge. Instead of having image-level exemplars, Memory-Efficient [9] proposes to preserve feature-level exemplars for reducing the memory footprint. Moreover, Mnemonics [6] advance to turn the exemplars learnable, where the model learning and exemplars updating are formulated into a bi-level optimization program, thus resulting in more informative exemplars and further boosting the performance. Nevertheless, as the existence of data imbalance between the exemplars of old classes and the training samples of new classes, the classification models typically would have biased weights among classes (i.e. tending to predict the test samples as new classes). It is worth noting that, another group of methods [10], [11], [12] shares the similar high-level idea as experience replay but with different ways to realize it: these works adopt generative models (e.g. GANs) to train the generators for each learnt class, such that the experience replay is achieved by using generated samples of old classes as the exemplars. However, the efforts for keeping generators usually cost more than using the replay buffer.

Class balance methods [3], [5] mainly tackle the issue of biased classifier stemming from data imbalance. For instance, BiC [3] splits a small validation set from the training set to train an additional bias correction layer, while WA [5] aligns the norm of the weight vectors in the final fully connected layer between new classes. However, as the data imbalance affects the entire model which includes both the classifier and the feature extractor, these class balance methods which only focus on correcting the bias on the classifier would potentially leave the whole model still suffering from forgetting.

Being parallel with incremental learning, data augmentation is one of popular techniques for dealing with limited amount of training data, in which it also helps to improve the generalizability of the models. Lately, Mixup [13] is proposed to generate augmented data by mixing pairs of training examples and their corresponding labels, which alleviates the issues of memorization and sensitivity to adversarial examples. This easy-to-implement yet effective method rapidly receives a lot of attention. Different variations are further proposed [14], [15], [16], e.g., Manifold Mixup [14] performs feature-level mixup, while CutMix [15] combines the idea of Cutout [16] and Mixup to synthesize augmented data.

Inspired by data augmentation, we propose an attentive knowledge replay framework to effectively refresh the knowledge of previously learnt classes and ease the catastrophic forgetting. The attentive module dynamically blends two training samples optimized over the incremental learning loss.

III. Proposed Method

A. A Base CIL Framework

The training data in the class-incremental learning (CIL) setting comes in as a stream, where different classes arrive at different time steps (i.e. incremental phases), and the training samples of a class arrive as a pack. Assume there are N incremental phases and the training data arriving at the respective phases is denoted as $\{D^1, \cdots, D^N\}$. In the n-th incremental phase, model is expected to learn the new classes C^n_{new} in D^n and meanwhile preserve the knowledge of the old classes C^n_{old} in $\{D^1, \cdots, D^{n-1}\}$. To be more specific, we denote the new training data as $X^n_{new} \in D^n$ and the old ones as $X^n_{old} \in \{D^1, \cdots, D^{n-1}\}$. Due to the limited size of replay buffer, the model can not access all the old data X^n_{old} for training but only the exemplars X^n_{exp}, which are selected from old data via the herding algorithm in the previous $n-1$ phases. Thus, the available training data $X^n_{all} = X^n_{new} \cup X^n_{exp}$ is used to update the model in the the n-th incremental phase. The model takes a training sample $x_i \in X^n_{all}$ as input and predicts the posterior $\sigma(x_i) = \{\sigma_0(x_i), \cdots, \sigma_c(x_i)\}$, where c denotes the number of all the seen classes up to the current phase. Our method is built upon a base model, following the similar design as iCaRL [2] to have two typical losses.

Classification loss $\mathcal{L}_{cls}$. We adopt the binary cross entropy as our classification objective, where the base model is encouraged to learn to classify the new classes as well as the old classes at the same time:

$$\mathcal{L}_{cls}(x_i) = -\sum_{c=1}^{C^n_{all}} (\delta_{c=y_i} \log \sigma_c(x_i) + \delta_{c \neq y_i} \log(1 - \sigma_c(x_i))), \tag{1}$$

where y_i is the groundtruth label of x_i. We follow recent CIL methods [4], [3], [5] that consider all the classes in $\mathcal{L}_{cls}(x_i)$.

Distillation loss $\mathcal{L}_{dist}$. For encouraging our base model to preserve the knowledge of old classes C^n_{old}, we utilize the knowledge distillation technique as [2], [3], [4], [5], [6], [9] to perform the regularization on the base model, such that given an input x_i, the prediction $\sigma(x_i)$ of the current model should be similar to the prediction $q(x_i)$ of the previous model (i.e. learnt at the previous incremental phase).

$$\begin{aligned}\mathcal{L}_{dist}(x_i) = -\sum_{c=1}^{C^n_{old}} \big[&q_c(x_i) \log \sigma_c(x_i) \\ &+ (1 - q_c(x_i)) \log(1 - \sigma_c(x_i))\big].\end{aligned} \tag{2}$$

Limitation of base CIL framework. Even though the replay buffer is able to help the model to preserve knowledge of old classes, the limited size would cause the data imbalance between the exemplars of old classes and the training samples of new classes. Especially, such imbalance becomes severe along the incremental learning phases, making it more and more difficult for model to retain the performance on the old classes and eventually lead to catastrophic forgetting. Moreover, the base CIL model tends to over-memorize the exemplars as demonstrated in Figure 1. The exemplars are perfectly classified in each incremental phase, but the classification accuracy on the testing data of old classes keeps decreasing. Therefore, we propose an attentive knowledge replay framework which addresses the forgetting issue in the base CIL framework. The proposed attention module learns to enlarge the training exemplars and replay them efficiently in each incremental phases, thus alleviating the forgetting of old classes. The details are presented in the next section.

B. Attentive Knowledge Replay

To replay exemplars in a simple yet efficient fashion, we propose an attentive knowledge replay framework (cf. Figure 2), which generates numerous virtual training samples via the linear interpolation between training pairs (sampled from X^n_{all} at the n-th incremental phase) and their labels. These virtual training samples encourage the classification model to learn smooth decision boundaries among all classes, and thus improve its generalizability. Over incremental phases, the model refreshes the knowledge of old classes via mixing exemplars with the training data of new classes, and also learns to recognize new classes via mixing among their training samples.

In order to blend a pair of data effectively, giving proper blending weights plays a critical role in our framework. We design an attention module, which is composed of importance estimation sub-networks $\mathbb{S}$ and a normalization layer, learning to predict the weights for blending a given pair of data (x_i, x_j). The importance estimation sub-network $\mathbb{S}$ aims to estimate the importance of the input sample according to the change of its feature representations over the incremental phases, as shown in Figure 3. To be specific, given a sample x_i, the normalized feature difference f^{diff}_i between its features f^p_i extracted by the old classification model (learnt in the previous incremental phase) and f^c_i extracted by the model at the current phase is leveraged to predict the importance score s_i of x_i via a multiple-layer perceptron MLP, where

$$s_i = \text{MLP}(f^{\text{diff}}_i) \quad \text{and} \quad f^{\text{diff}}_i = \frac{f^c_i}{|f^c_i|} - \frac{f^p_i}{|f^p_i|}. \tag{3}$$

The importance scores s_i and s_j of the image pair (x_i, x_j) are then considered jointly to obtain the blending weights a_i and a_j respectively: $a_i = s_i/(s_i + s_j), a_j = s_j/(s_i + s_j)$.

The virtual training sample $\tilde{x}$ and its corresponding label $\tilde{y}$ are then generated by the following linear combination:

$$\tilde{x} = a_i x_i + a_j x_j, \quad \tilde{y} = a_i y_i + a_j y_j. \tag{4}$$

As the virtual training samples generated by our attentive knowledge replay framework are involved in the overall incremental learning scheme, the attention module would be automatically optimized to weigh the important training samples more in order to boost the performance of the target classification model (for both old or new classes). That is, our attention module learns to predict the adaptive blending weights according to the relative importance of the given pair of training samples for the incremental learning objective.

The objectives for jointly learning the proposed attentive knowledge replay module and our target classifier include:

Blending Classification loss $\mathcal{L}^{blend}_{cls}$. Different from the classification loss in our base model (see Section III-A), the blending loss here is composed of two binary cross entropy losses, as the label of the virtual sample $\tilde{x}$ is the linear combination over y_i and y_j, with weights a_i and a_j

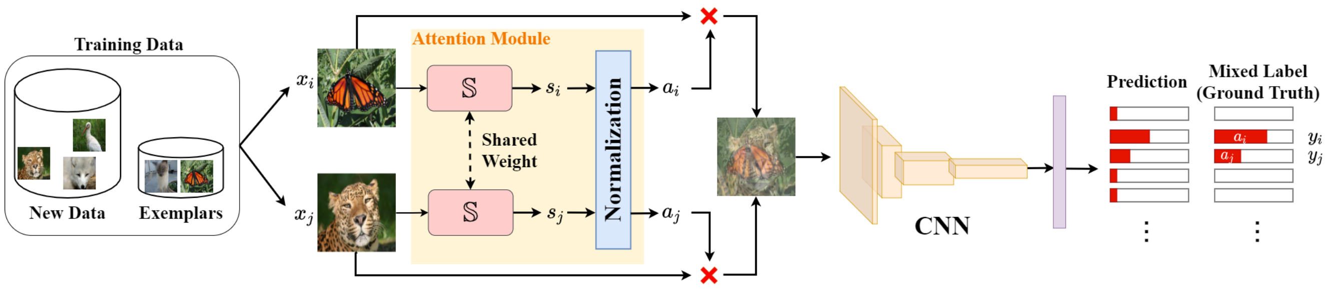

Fig. 2: Overview of our **Attentive Knowledge Replay**. During training, a pair of training samples x_i and x_j are fed into the attention module simultaneously to estimate their attentive blending weights a_i and a_j respectively, where $a_i + a_j = 1$. The training samples (x_i, x_j) and their labels (y_i, y_j) are blended accordingly by the weights a_i and a_j to produce the virtual training data, which is further utilized for training the classification model via incremental-learning objectives. The symbol of red cross denotes the multiplication operation.

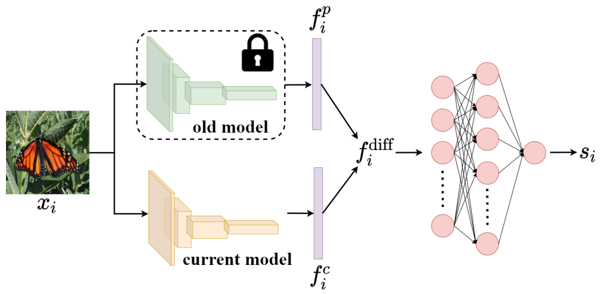

Fig. 3: Importance estimation sub-network $\mathbb{S}$ used in the attention module. Image features f_i^p and f_i^c of the sample x_i are first extracted from the old and current classification models respectively, where their normalized difference f_i^{diff} is then passed through a multiple-layer perceptron to predict the importance s_i of the input sample x_i.

respectively. The blending loss $\mathcal{L}_{cls}^{blend}$ is thus defined as:

$$\mathcal{L}_{cls}^{blend}(\tilde{x}) = -a_i \sum_{c=1}^{C_{all}^n} (\delta_{c=y_i} \log \sigma_c(\tilde{x}) + \delta_{c \neq y_i} \log(1 - \sigma_c(\tilde{x}))) \\ - a_j \sum_{c=1}^{C_{all}^n} (\delta_{c=y_j} \log \sigma_c(\tilde{x}) + \delta_{c \neq y_j} \log(1 - \sigma_c(\tilde{x}))). \tag{5}$$

Blending Distillation loss $\mathcal{L}_{dist}^{blend}$. The distillation loss here is similar to the one used in the base model, but now the training samples are instead the virtual ones $\tilde{x}$ generated by the attention module. Given the virtual samples $\tilde{x}$, the prediction $\sigma(\tilde{x})$ of current model should be similar to the prediction $q(\tilde{x})$ of the previous model. Blending distillation loss $\mathcal{L}_{cls}^{blend}$ is:

$$\mathcal{L}_{dist}^{blend}(\tilde{x}) = -\sum_{c=1}^{C_{old}^n} \left[q_c(\tilde{x}) \log \sigma_c(\tilde{x}) + (1 - q_c(\tilde{x})) \log(1 - \sigma_c(\tilde{x})) \right]. \tag{6}$$

Original Loss $\mathcal{L}_{org}$. To stabilize the training of the attention module, we follow Peng *et al.* [17] to keep the original objective in the base model (i.e. samples are utilized without blending) as a regularization term in our final objective. The original objective $\mathcal{L}_{org}$ consists of the classification and distillation losses described in Section III-A: $\mathcal{L}_{org} = \mathcal{L}_{cls} + \lambda \mathcal{L}_{dist}$. Finally, the overall objective of our incremental learning $\mathcal{L}_{total}$ is defined as:

$$\mathcal{L}_{total} = (1 - \lambda_r)(\mathcal{L}_{cls}^{blend} + \lambda \mathcal{L}_{dist}^{blend}) + \lambda_r \mathcal{L}_{org}, \tag{7}$$

where λ is used to control the weight of the distillation-related losses and λ_r is the weight of the regularization term (i.e. original loss $\mathcal{L}_{org}$). In our experiments, we follow Peng *et al.* [17] to set the λ_r to 0.25, and the setting of λ is described later in Section IV.

IV. Experimental Results

Dataset. Our experiments are based on CIFAR-100 and ImageNet [18] datasets, which are widely used for the evaluation of class-incremental learning methods. CIFAR-100 contains 60,000 color images of size $32 \times 32 \times 3$ from 100 different classes, where each class has 500 images for training and 100 images for testing. ImageNet is a large-scale dataset with 1000 classes, which consists of around 1.2 millions color images of size $224 \times 224 \times 3$ for training and another 50,000 for testing. We follow the setting as [2] to sample 100 classes from ImageNet in our experiments, denoted as ImageNet-subset. The total size of the replay buffer is set to support 2,000 images (hence the number of exemplars per old class decreases gradually). We apply the herding algorithm from iCaRL [2] to determine which images to be kept in exemplars.

Experimental Settings. There are several settings adopted in the prior works [2], [3], [4], [6], [5] of class-incremental learning for evaluation, which can be grouped into two main schemes: (1) Training from scratch, and (2) Training form half.

– *Training from scratch:* The classification model is learnt from scratch, i.e. no pre-training. We divide all classes into N incremental phases, where each phase has equal number of new classes to be learned. iCaRL [2], BiC [3] and WA [5] are originally proposed and evaluated under this scheme.

– *Training from half:* Half of the total classes are used to pre-train the model first, while the remaining half is divided into N incremental phases, where each phase has equal number of new classes. LUCIR [4] and Mnemonics [6] are originally proposed under this setting. As pre-training is performed, the model starts the incremental learning with already having

TABLE I: Average incremental accuracy on CIFAR-100 under two training schemes with different incremental phases, i.e. N=5, 10, 20 for training from scratch (TFS) and N=5, 10, 25 for training from half (TFH). "Origin-Set" represents the original training scheme of baselines. **Bold** number indicates the best performance and <u>underlined</u> number represents the second best one.

CIFAR-100	Origin-Set		Traing From Scratch			Traing From Half		
	TFS	TFH	N=5	10	20	5	10	25
iCaRL [2]	✓		61.45	56.57	53.77	61.93	59.22	55.48
BiC [3]	✓		66.70	61.90	60.32	62.15	58.84	53.82
LUCIR [4]		✓	64.82	60.26	57.97	62.18	60.22	57.73
WA [5]	✓		<u>70.00</u>	67.25	64.33	63.28	55.29	41.47
Mnemonics [6]		✓	60.78	56.18	54.03	63.38	<u>63.89</u>	**64.68**
Base	✓	✓	67.24	63.53	59.14	61.88	57.84	53.57
Mixup	✓	✓	<u>69.96</u>	<u>68.14</u>	<u>66.04</u>	<u>66.97</u>	63.84	59.68
Ours	✓	✓	**70.38**	**69.11**	**67.40**	**67.73**	**65.15**	<u>60.38</u>

a strong and discriminative feature extractor; these methods however tend to not perform well while training from scratch.

We follow the standard CIL evaluation metrics in [2] to compare the proposed approach with recent competing methods using the incremental accuracy and the average incremental accuracy. *Incremental accuracy* is the accuracy for classifying all the seen classes at the end of each incremental phase. *Average incremental accuracy* is the average of incremental accuracy from the first phase to the last phase.

A. Implementation Details

A proper training procedure matters a lot in incremental learning, while most off-the-shelf methods ignore it. We point out the problem of using the same learning procedure for each incremental phase and adjust the overall incremental learning framework with three important *training techniques*: warm up, phase-wise learning rate decay, and adjusted weight of distillation-related loss.

Warm up (WP). The change between incremental phases is large. The new data in the previous phase is reduced to a few exemplars and meanwhile a relatively large amount of samples of new classes are added in. In results, a large learning rate at the beginning of training would make the model learn the new data quickly and destroy the decision boundary of previous classes. Moreover, as we proposed an attentive module to predict meaningful weights for blending images, the model could face an unstable training state when using the same learning rate at first. Thus, we apply gradual warmup [19] before model training with the given learning rate to avoid dramatic changes of previous knowledge. For each phase, we add additional 20 epochs to warm up the learning rate. The learning rate starts with 1/20 of the given learning rate and increases gradually to it in 20 epochs.

Phase-wise learning rate decay (LRD). Most incremental learning works apply the same initial learning rate for each incremental phase. However, having the initial learning rate for later incremental phase as large as the one used in the early incremental phase would cause a strong alteration of model parameters, especially when the data imbalance becomes more severe in the later phases. Thus, we design a phase-wise learning rate decay strategy to alleviate the large change of the model. We first denote the number of base classes as N_{base}, in which its is the number of classes used for pre-training for the setting of training from half, and is the number of classes learnt in the first incremental phase for the setting of training from scratch. With denoting the number of new classes in each incremental phase as N_{new}, for the i-th phase, its initial learning rate lr_i is set as $lr_0 * \sqrt{N_{\text{new}}/(N_{\text{base}} \cdot i)}$. As we gradually decrease the initial learning rate for each incremental phase, the model would not change dramatically along the phases and keep more knowledge of old data consequently.

Adjusted weight of distillation-related losses (AD). We set a hyper-parameter λ to weigh the importance of distillation-related losses (cf. Equation 7). As we expect that our method should perform well under both training schemes, this weight λ is used to increase the importance of preserving old knowledge in the setting of training from half, and defined as $\lambda = \sqrt{(N_{\text{base}})/(N_{\text{new}})}$. When training from scratch, the weight is equal to 1 since $N_{\text{base}} = N_{\text{new}}$.

Our models are implemented with Pytorch [20]. We follow the iCaRL [2] to adopt a 32-layer ResNet [21] for CIFAR-100 and a 18-layer ResNet for ImageNet-subset. We use SGD optimizer to train the model and set the initial learning rate to 2 for all the settings. For CIFAR-100, each incremental phase has 90 epochs, which consist of 20 epochs at the beginning for gradual warm up. Within each incremental phase, the learning rate reduces to 1/5 of the previous learning rate after 70 and 84 epochs. The weight decay is set to 0.00001 and the batch size is 128. While for ImageNet-subset, each incremental phase has 120 (respectively, 110) epochs, with learning rate being reduced to 1/5 of the previous learning rate after 50, 80, and 100 (respectively, 50 and 80), and the batch size set to 256 (respectively, 128) for the setting of training from scratch (respectively, training from half), where the first 20 epochs are for gradual warm up and the weight decay is set to 0.00001. where the first 20 epochs are for gradual warm up. Within each incremental phase, the learning rate reduces to 1/5 of the previous learning rate after 50, 60, and 100 (50 and 80) epochs. The weight decay is set to 0.00001 and the batch size is 256 (128). For all the settings, we also apply random cropping, horizontal flip, and normalization to augment training images.

B. Quantitative Results

We compare our method with several baselines [2], [3], [4], [6], [5] on CIFAR-100 under two training schemes with different amounts of incremental phases. For simplicity, we

TABLE II: Average incremental accuracy on ImageNet-subset under two training schemes with different total incremental phases (N=5, 10, 20 for training from scratch and N=5, 10, 25 for training from half).

ImageNet-subset	Origin-Set		Traing From Scratch			Traing From Half		
	TFS	TFH	N=5	10	20	5	10	25
iCaRL [2]	✓		73.96	69.10	63.40	68.03	59.44	50.35
BiC [3]	✓		72.48	66.42	61.24	65.42	57.82	46.33
LUCIR [4]		✓	66.26	59.45	54.64	70.78	68.62	66.48
WA [5]	✓		71.50	67.93	62.15	70.15	63.44	64.64
Mnemonics [6]		✓	59.96	51.88	44.98	70.63	70.24	**70.45**
Base	✓	✓	78.88	73.32	65.07	72.55	66.82	61.00
Mixup	✓	✓	78.54	72.75	65.50	73.43	68.71	63.88
Ours	✓	✓	**79.04**	**74.53**	**67.37**	**74.75**	**70.85**	65.11

TABLE III: Ablation study for the learning rate policy with and without our attentive knowledge replay, conducted on the CIFAR-100 dataset, where the performance gain of our method is due to the attentive knowledge replay instead of the learning rate policy.

		TFS, N=10		TFH, N=10	
WP	LRD	Base	Ours	Base	Ours
		56.50	61.48	54.50	61.49
	✓	55.74	60.16	54.78	63.65
✓		64.03	69.20	55.94	61.40
✓	✓	63.53	69.11	56.21	63.75

TABLE IV: Comparison with baselines which apply our training techniques (cf. Section IV-A). Obviously, the training tricks somewhat improve the baselines, but our method still outperforms them.

CIFAR-100	Traing From Scratch			Traing From Half		
	N=5	10	20	5	10	25
BiC	66.70	61.90	60.32	62.15	58.84	53.82
BiC+WP/LRD/AD	67.76	64.93	60.44	61.93	58.63	48.40
WA	70.00	67.25	64.33	63.28	55.29	41.47
WA+WP/LRD/AD	70.18	68.09	64.23	63.47	55.87	43.64
Ours	**70.38**	**69.11**	**67.40**	**67.73**	**65.15**	**60.38**

denote "training from scratch" as "TFS" and "training from half" as "TFH" in the following. All the baseline methods are reproduced by following their own implementation details.

The results of quantitative comparison of our method with other works on CIFAR-100 are provided in Table I. Here we compare our proposed method with two base models: **Base** is our base model (see Section III-A); **Mixup** is the base model with replaying the examplars via the mixup technique [13], where the blending weight λ is sampled from the prior Beta distribution $Beta(\alpha, \alpha)$ with α set to 1.0; **Ours** is the model with our proposed framework of attentive exemplars replay. The results show that, the baselines which are originally proposed under the setting of training from half (TFH) (i.e. Mnemonics [6] and LUCIR [4]) do have good performance, but instead have significantly worse performance while being trained from scratch (TFS). On the contrary, the baselines which are originally proposed under the setting of training from scratch (i.e. iCaRL [2], BiC [3], and WA [5]) outperform other two TFH baselines on the TFS scheme, but are worse when being training from half. In comparison to the baselines which have the issue of generalization across training schemes, our methods utilizing exemplars in a data-efficient manner via blending training samples (i.e. Mixup and Ours) focus on alleviating the over-memorization issue, thus leading to better or comparable performance with respect to the baselines on both TFS and TFH schemes. In particular, with the attentive knowledge replay which predicts the proper weight for blending training samples according to the incremental learning objective, our proposed method on average reaches the best performance.

The experimental results on ImageNet-subset are provided in Table II. We observe that the issue of having poor generalization across training schemes becomes more significant for baselines. For example, under the TFS scheme, Mnemonics [6] and LUCIR [4] only have 44.98% and 54.64% average incremental accuracy as N=20, while other methods are all above 60%. This evidence reveals that these methods [6], [4] cannot generalize to the training scheme of training from scratch as they need a stronger model pre-trained on half of classes to start with. In contrast, our method outperforms the baselines in most of the settings, showing the generalization ability under different training schemes, especially for the large gains in the TFS scheme. However, in the TFH settings with N set to 25, our method does not perform better than Mnemonics. The reason is that in such a setting, there are only 2 new classes in each incremental phase, consequently less training data than exemplars, which causes the model to overfit on old classes rather than learning new classes.

In Table III, we show that our performance gain mainly results from the proposed attentive knowledge replay framework, while the learning rate policy is used to help training the incremental learning models more effectively. In the experiments, we follow [22] to design our learning rate policy to include learning rate decay (LRD) and warmup (WP), which are essential for successful attention module training. LRD is mainly designed for the Train-from-half (TFH) setting since there is a robust base model pre-trained on half classes and needs not such large learning rate for model training in the following incremental phases. Experimental results in Table III show that the main improvement is due to our attentive knowledge replay framework over the baseline method (Base, a variation of iCaRL [2]), and the learning rate policy provides a corresponding suitable training scheme to reach the convergence. Note that the technique of adjusted weights of distillation loss (AD) is not applied here for clear comparisons.

Besides, we also apply our training techniques to other baselines (e.g. BiC [3] and WA [5] here), as shown in

TABLE V: Ablation study of training techniques (cf. Section IV-A) on CIFAR-100 under the scheme of training from scratch (**TFS**). Note that, as λ for weighting the distillation-related losses is equal to 1 under TFS scheme, here only the techniques of warm up (WP) and phase-wise learning decay (LRD) are investigated.

WP	LRD	N=5	10	20
		67.00	61.48	59.68
	✓	66.90	60.16	55.10
✓		**70.62**	69.20	**67.44**
✓	✓	70.38	**69.11**	67.40

TABLE VI: Ablation study of training techniques (cf. Section IV-A), i.e. warm up (WP), phase-wise learning decay (LRD), and adjusted weight of distillation-related losses (AD), on CIFAR-100 under the scheme of training from half (**TFH**).

WP	LRD	AD	N=5	10	25
			65.98	61.49	56.60
	✓		66.60	63.65	60.11
		✓	66.53	63.37	58.38
	✓	✓	67.13	64.34	59.90
✓			65.97	61.40	56.31
✓	✓		66.83	63.75	60.09
✓		✓	66.55	63.52	58.65
✓	✓	✓	**67.73**	**65.15**	**60.38**

Table IV. Obviously, the training techniques indeed boost the performance of baselines, but the improvement is not significant. The reason is that the training techniques are specifically designed to stable our attentive module training, and may not bring the same improvement to other baselines. Moreover, these baselines also have their own training tricks to reach the best performance, such as having different designs of weighting the distillation loss [3], [5]. Such experiment in Table IV also help to verify that our performance gain is not solely stemmed from the training techniques, but clearly contributed by our proposed attentive knowledge replay mechanism.

C. Ablation Study

Effects of warm up (WP): We observe that there is a significant performance drop without the warmup process in the TFS scheme (shown in Table V). This reveals the importance of using a smaller learning rate at the beginning of each incremental phase to ease up the change of decision boundary and preserve more previous knowledge. Besides, a dramatic performance drop occurs under the TFS scheme with the number of incremental phases $N = 20$, when warmup is not adopted. This phenomenon is caused by a failure on learning attentive weights. At the beginning of each incremental phase, the attention module would face a large amount of new data. If the learning rate is large, it would be quickly occupied by the new data and fail to consider the exemplars. Therefore, a warmup process is crucial in our methods to provide a buffer time for models to learn with new data.

Effects of phase-wise learning rate decay (LRD): We can see that in the TFH scheme (shown in Table VI), the methods with LRD can boost the performance by about 1$\sim$3% under different settings of N, while in the TFS scheme (shown

TABLE VII: Ablation study on different design choices for representing the feature changes f_i^{diff} of a given sample x_i, over the incremental phases (please refer to the second paragraph of Section IV-C for more details). Experiments are conducted on the CIFAR-100 dataset. In most of experimental settings, the design of using **subtraction**, as what we have now in our attention module, reaches the best performance.

CIFAR-100	Traing From Scratch			Traing From Half		
	N=5	10	20	5	10	25
Subtraction	**70.38**	69.11	67.40	**67.73**	**65.15**	**60.38**
L1 Distance	70.16	**69.58**	**68.11**	67.13	64.99	59.65
Schur Product	70.22	69.31	67.89	67.17	65.13	59.53
Concatenation	69.76	69.56	67.55	67.65	65.12	60.13

in Table V) the methods with LRD may not have such improvement. The reason is that, in the TFH scheme, the model learns much robust features with half of the dataset at first, and it only needs a small learning rate in the following incremental phases to fine-tune the model with new data. However, in the TFS scheme, the model is trained from scratch and is not strong enough at the beginning. Then, turning into a small learning rate does not gain such a significant benefit.

Effects of adjusted weight of distillation-related loss (AD): The design of the adjusted weight λ of distillation-related loss is to tackle the gap between the base classes (i.e. N_{base}) and new classes (i.e. N_{new}) in the first incremental phase. Please note that, as in the TFS scheme $\lambda = 1$, we only perform study in the TFH scheme. As shown in Table VI, the methods with AD are better than those without AD by about 1$\sim$4% under different numbers of incremental phases N. In particular, when N is increased, the performance boost brought by AD also increases. This phenomenon indicates that the importance of distillation-related loss becomes larger when the ratio between N_{base} and N_{new} is increased. Our AD technique is a simple way to adjust the weight of distillation-related loss, and we believe there could exist a more effective adjustment and leave it as future work.

Design choices for predicting the importance score. Regarding our current model design of predicting the importance score s_i for a given sample x_i in the attention module, we adopt the simple **subtraction** between the normalized f_i^p and f_i^c (cf. Equation 3), which are extracted by the old and new classification models respectively, to indicate the feature changes f_i^{diff} of x_i over the incremental phases. Here we conduct an ablation study to experiment different designs for representing such feature changes, including the **L1 distance**, the **Schur product**, and the **concatenation** between the normalized f_i^p and f_i^c. The results of different design choices are provided in Table VII. We can see that the performance differences among these designs are not that large (only about 1% at most in terms of average incremental accuracy), while our current design of using subtraction reaches the best performance in most of the experimental settings.

Effects of different blending strategies. As the training samples include new data and exemplars (old data), there are three kinds of pairs being blended to generate virtual training sample: new-versus-new, old-versus-old, and new-

TABLE VIII: Ablation study of blending strategies on CIFAR-100. Base: base model without blending; Intra-Blend: blending with new-versus-new and old-versus-old pairs. Inter-Blend: blending with new-versus-old pairs. Ours: using both Intra-Blend and Inter-Blend.

CIFAR-100	Traing From Scratch			Traing From Half		
	N=5	10	20	5	10	25
Base	67.24	63.53	59.14	61.88	57.84	53.57
Intra-Blend	66.68	65.73	63.43	66.87	64.41	59.45
Inter-Blend	67.92	66.50	65.40	64.92	62.13	55.92
Ours	**70.38**	**69.11**	**67.40**	**67.73**	**65.15**	**60.38**

TABLE IX: The experimental results of applying our attentive knowledge replay framework on other CIL methods, conducted on the CIFAR-100 dataset. Our attentive knowledge replay is able to consistently benefit the performance of other CIL methods under both Train-from-scratch (TFS) and Train-from-half (TFH) settings.

CIFAR-100	Traing From Scratch			Traing From Half		
	N=5	10	20	5	10	25
BiC	66.70	61.90	60.32	62.15	58.84	53.82
BiC+Ours	**70.36**	**68.71**	**65.12**	**62.72**	**59.03**	**53.88**
WA	70.00	67.25	64.33	63.28	55.29	41.47
WA+Ours	**70.78**	**68.79**	**66.84**	**63.37**	**59.35**	**52.39**

versus-old. We divide them into two strategies: intra-blend and inter-blend, where the former consists of new-versus-new and old-versus-old pairs while the latter consists of new-versus-old pairs. To analyze the effect of different blending strategies, we conduct the ablation study of blending strategies in Table VIII. Results show that having intra-blend or inter-blend strategy individually can already boost the performance. While further jointly using both intra-blend and inter-blend strategies, our full model achieves the best average incremental accuracy. The improvement indicates that both inter-blend and intra-blend strategies are essential for effective exemplars replay.

Applications of attentive knowledge replay on other CIL methods. As our framework can be easily integrated into other CIL methods, we also provide its application on other two CIL methods: BiC [3] and WA [5]. In Table IX, our attentive knowledge replay consistently brings improvements to these two CIL methods. However, these improvements are not as large as the base model (similar to iCaRL [2]) has. The reason is that our attentive knowledge replay contributes to alleviate the influence of imbalance amount of new and old data, while other CIL methods also propose some designs to tackle this issue (e.g. bias correction in BiC [3] and weight alignment in WA [5]). Therefore, applying our attentive knowledge replay to other methods may not have such significant improvement as applying to iCaRL. However, our method still brings improvement to other CIL methods in most of settings, showing the efficacy and complementary property of our method.

V. Conclusions

In this work, we introduce a simple yet effective data-efficient method to refresh the knowledge of previously experienced classes during class-incremental learning. We show that our Attentive Knowledge Replay framework can adaptively blend pairs of data coming from the exemplar buffer and new classes, in order to obtain high performance for old and new classes. In addition, our method performs favorably against state-of-the-art methods in different CIL settings, as well as improving upon existing CIL pipelines.

Acknowledgement This work is supported by NSTC 111-2628-EA49-018-MY4 and NSTC 111-2636-E-A49-003.

References

[1] Zhizhong Li and Derek Hoiem, "Learning without forgetting," *IEEE T-PAMI*, 2017.

[2] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert, "icarl: Incremental classifier and representation learning," in *CVPR*, 2017.

[3] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu, "Large scale incremental learning," in *CVPR*, 2019.

[4] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin, "Learning a unified classifier incrementally via rebalancing," in *CVPR*, 2019.

[5] Bowen Zhao, Xi Xiao, Guojun Gan, Bin Zhang, and Shu-Tao Xia, "Maintaining discrimination and fairness in class incremental learning," in *CVPR*, 2020.

[6] Yaoyao Liu, Yuting Su, An-An Liu, Bernt Schiele, and Qianru Sun, "Mnemonics training: Multi-class incremental learning without forgetting," in *CVPR*, 2020.

[7] James Kirkpatrick et al., "Overcoming catastrophic forgetting in neural networks," *PNAS*, 2017.

[8] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars, "Memory aware synapses: Learning what (not) to forget," in *ECCV*, 2018.

[9] Ahmet Iscen, Jeffrey Zhang, Svetlana Lazebnik, and Cordelia Schmid, "Memory-efficient incremental learning through feature adaptation," in *ECCV*, 2020.

[10] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim, "Continual learning with deep generative replay," in *NeurIPS*, 2017.

[11] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, Zhengyou Zhang, and Yun Fu, "Incremental classifier learning with generative adversarial networks," *ArXiv:1802.00853*, 2018.

[12] Oleksiy Ostapenko, Mihai Puscas, Tassilo Klein, Patrick Jahnichen, and Moin Nabi, "Learning to remember: A synaptic plasticity driven framework for continual learning," in *CVPR*, 2019.

[13] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz, "mixup: Beyond empirical risk minimization," in *ICLR*, 2018.

[14] Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio, "Manifold mixup: Better representations by interpolating hidden states," in *ICML*, 2019.

[15] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo, "Cutmix: Regularization strategy to train strong classifiers with localizable features," in *ICCV*, 2019.

[16] Terrance DeVries and Graham W Taylor, "Improved regularization of convolutional neural networks with cutout," *ArXiv:1708.04552*, 2017.

[17] Xiaojiang Peng, Kai Wang, Zhaoyang Zeng, Qing Li, Jianfei Yang, and Yu Qiao, "Suppressing mislabeled data via grouping and self-attention," in *ECCV*, 2020.

[18] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al., "Imagenet large scale visual recognition challenge," *IJCV*, 2015.

[19] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He, "Accurate, large minibatch sgd: Training imagenet in 1 hour," *arXiv preprint arXiv:1706.02677*, 2017.

[20] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer, "Automatic differentiation in pytorch," 2017.

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[22] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin, "Attention is all you need," in *NeurIPS*, 2017.