Knowledge Consolidation based Class Incremental Online Learning with Limited Data

Mohammed Asad Karim[1*] , **Vinay Kumar Verma**[2] , **Pravendra Singh**[3] ,
Vinay Namboodiri[1,4] and **Piyush Rai**[1]

[1]Indian Institute of Technology Kanpur, India
[2]Duke University, United States
[3]Indian Institute of Technology Roorkee, India
[4]University of Bath, United Kingdom

asadkarim0938@gmail.com, vinaykumar.verma@duke.edu, pravendra.singh@cs.iitr.ac.in,
vpn22@bath.ac.uk, piyush@cse.iitk.ac.in

Abstract

We propose a novel approach for class incremental online learning in a limited data setting. This problem setting is challenging because of the following constraints: (1) Classes are given incrementally, which necessitates a class incremental learning approach; (2) Data for each class is given in an online fashion, i.e., each training example is seen only once during training; (3) Each class has very few training examples; and (4) We do not use or assume access to any replay/memory to store data from previous classes. Therefore, in this setting, we have to handle twofold problems of catastrophic forgetting and overfitting. In our approach, we learn robust representations that are generalizable across tasks without suffering from the problems of catastrophic forgetting and overfitting to accommodate future classes with limited samples. Our proposed method leverages the meta-learning framework with knowledge consolidation. The meta-learning framework helps the model for rapid learning when samples appear in an online fashion. Simultaneously, knowledge consolidation helps to learn a robust representation against forgetting under online updates to facilitate future learning. Our approach significantly outperforms other methods on several benchmarks.

1 Introduction

Deep neural networks have achieved promising results on various tasks. However, these models suffer from the problem of catastrophic forgetting [Kirkpatrick *et al.*, 2017]. The most prominent reason for catastrophic forgetting is that the model is not trained to also remember the previous knowledge when acquiring new knowledge. In general, the model is trained to optimize its performance on the current task with no consideration of how the updated model will perform on earlier tasks. This greedy update overwrites the parameter values that may have been optimal for previous tasks. *Continual-learning (CL)*

(also sometimes referred to as lifelong/incremental learning) is a learning paradigm to address this issue in deep neural networks and has been gaining significant attention in recent work [Parisi *et al.*, 2019].

We present a novel approach for *class incremental* online learning problem in a limited data setting. This problem setting is more challenging than standard class incremental learning [Javed and White, 2019] due to additional constraints: (1) Data in each class appears in the online fashion, i.e., the model sees every training example exactly once; (2) The number of training examples in each class is very small; and (3) We do not use any replay/memory to store the training examples from previous classes. This is the most general setting for class incremental learning and various practical usage scenario can be obtained through this or a relaxed setting. For instance, in face recognition, it is common to have few examples per class but usually not in an online learning fashion, whereas for a robot navigating in an environment, the setting would also be online. We empirically show that learning a robust representation that can accommodate future tasks may be a potential solution to handle the problem mentioned above. Our proposed approach achieves this by leveraging the meta-learning [Finn *et al.*, 2017] framework with knowledge consolidation.

Meta-learning [Finn *et al.*, 2017] has proven to be an effective approach for learning generic feature representations that can be rapidly adapted to new tasks by fine-tuning using very few examples (and in some cases, even *without* fine-tuning [Vinyals *et al.*, 2016; Raghu *et al.*, 2020]). While such use of meta-learning might seem appealing and does indeed show some promising results in continual learning settings [Javed and White, 2019], in practice, this approach is still prone to the problem of catastrophic forgetting. One of the reasons for this is the *overparametrized* nature of deep neural networks, in which only a few neurons are activated/fired for all samples. As a result, the network is reliant only on a small set of parameters. Although this may not be a problem when learning only a single task, it can potentially be an issue in continual learning where we are required to learn a sequence of tasks and, while learning a new task, any changes to these parameters can drastically affect the performance on the older tasks.
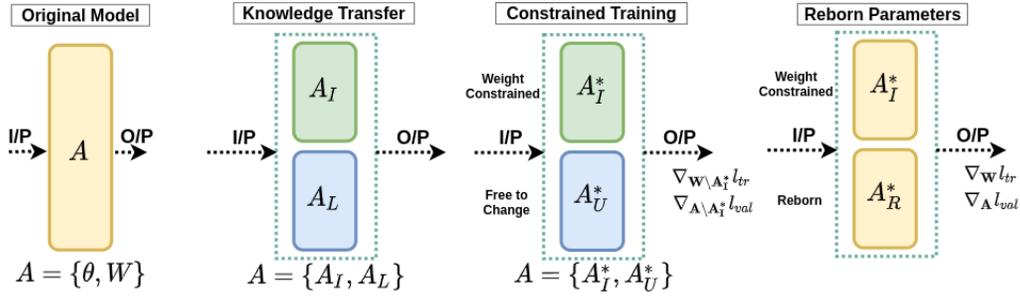
---

*Contact Author

Figure 1: The figure shows the various steps in knowledge consolidation. The original model is learned via a meta-learner. Thereafter, we transfer the model's knowledge into a subset of parameters and partition the model into important and unimportant parameters. Then we retrain the model by allowing $\mathbf{A}_U^*$ to change freely according to the loss function and also constraining $\mathbf{A}_I^*$ to preserve previous knowledge.

To address this, we present a *knowledge consolidation* based meta-learning approach. In our approach, during training, we identify and split the network parameters into two groups, called important and unimportant. The network's existing knowledge is squeezed into the set of important parameters, and the unimportant/dead parameters are freed, thereby expanding the network's capacity to accommodate the future learning trajectories. Briefly, a learning trajectory is a sequence of examples where examples from a particular class occur together in the sequence (we discuss this in detail later). The proposed strategy ensures that the knowledge from the old learning trajectories is preserved in a compressed form within a small set of important network parameters, which we identify and isolate, and *then* move on to adopt new learning trajectories. The extra knowledge obtained via learning from the new trajectories updates the unimportant parameters, and they also become important. In knowledge consolidation, we rejuvenate the dead neurons in the model and consolidate the knowledge of the previously preserved parameters and the new rejuvenated parameters. This helps to learn a robust representation. Therefore, the model capacity is fully utilized, and small changes in a few parameters strongly resist the catastrophic forgetting. The knowledge consolidation process overview is shown in Fig. 1.

Note that we use knowledge consolidation and meta-learning in the training phase. Training is done on a base class set using multiple learning trajectories. We strictly follow class incremental online learning setting within a learning trajectory. However, across learning trajectories, the class incremental learning setting is not used (since multiple learning trajectories can have the same set of classes). Therefore, we do not follow the incremental learning setting during training on the base class set (since training is done on multiple learning trajectories). We follow this to make the model's representation robust and facilitate future continual learning during evaluation time (Section 2), where we perform class incremental online learning with limited data on a *novel* class set. We use entirely different (disjoint) classes in the novel class set than the base class set, and during evaluation on the novel class set, we only use meta-learning for quick adaptation.

Our approach significantly outperforms other incremental learning methods by a significant margin. We show that a basic online updating strategy on representations learned by

our approach Knowledge Consolidation based Class Incremental Online Learning (KCCIOL) is better than memory-based rehearsal methods. Our approach can also be integrated with existing continual learning approaches such as MER, EWC, ER-Reservoir as shown in Section 7.

2 Problem Formulation and Evaluation Protocol for Novel Class Set Testing

Let $\{\tau_1, \tau_2 \dots \tau_k \dots \tau_l \dots\}$ denote a stream of learning-trajectories and $\forall i$, $\tau_i \sim P_{test}(\tau)$ where $P_{test}(\tau)$ denotes the trajectory distribution during testing from the novel class set. Each learning-trajectory τ_i is further split into two sets – train and validation, i.e., $\tau_i = \{\tau_{tr}, \tau_{val}\}$ where $\tau_{tr} = \{x_n, y_n\}_{n=1}^k$ and $\tau_{val} = \{x_n, y_n\}_{n=k+1}^{k+s}$ are the labeled samples, $\{k, s\} \in \mathbb{N}$. Here, $\forall n$, $(x_n, y_n) \in (\mathcal{X}_{test}, \mathcal{Y}_{test})$ denote the input and label pairs and the trajectory distribution $P_{test}(\tau)$ is defined over $(\mathcal{X}_{test}, \mathcal{Y}_{test})$ which is disjoint from base training class set $(\mathcal{X}_{train}, \mathcal{Y}_{train})$ i.e. $\mathcal{Y}_{train} \cap \mathcal{Y}_{test} = \varnothing$. Moreover, we assume $class(\tau_{tr}) = class(\tau_{val})$ i.e. classes of τ_{tr} are the same as classes of τ_{val}. The goal of continual learning is to minimize the loss on the unseen examples of classes learnt earlier in an incremental fashion, and can be written as: $\mathbb{E}_{\tau \sim P_{test}(\tau)}[\mathcal{L}(f(\tau_{val}^x|\theta, \mathbf{W}), \tau_{val}^y)]$ i.e model is evaluated on τ_{val}. Our evaluation protocol (Algorithm 4) is similar to the class-incremental setting [Javed and White, 2019], but samples within each class also arrive in an online manner. In particular, here are the key differences: 1) We assume availability of very few samples per class; 2) For a particular class, each sample is seen exactly once; and 3) We do not use any replay mechanism. These differences make our problem setting considerably more challenging than the standard class-incremental setting.

3 Training on Base Class Set using Meta Learning Approach

Following a similar notation as in Sec. 2, let $\{\tau_1, \tau_2 \dots \tau_k \dots \tau_l \dots\}$ denote a stream of learning trajectories and $\forall i$, $\tau_i \sim P_{train}(\tau)$ where $P_{train}(\tau)$ denotes the learning trajectory distribution during training. Following the model-agnostic meta learning (MAML) set-up [Finn *et al.*, 2017], we assume that the ith learning-trajectory's data τ_i is further split into two sets, meta-train and meta-val, i.e. $\tau_i = \{\tau_{tr}, \tau_{val}\}$ where $\tau_{tr} = \{x_n, y_n\}_{n=1}^k$ and

$\tau_{val} = \{x_n, y_n\}_{n=k+1}^{k+s}$ are the labeled samples, $\{k, s\} \in \mathbb{N}$. Here, $\forall n, (x_n, y_n) \in (\mathcal{X}_{train}, \mathcal{Y}_{train})$ denote the input and label pairs and the learning trajectory distribution $P_{train}(\tau)$ is defined over $(\mathcal{X}_{train}, \mathcal{Y}_{train})$. Moreover, we assume $class(\tau_{tr}) \subset class(\tau_{val})$ i.e. classes of τ_{tr} are a proper subset τ_{val}; therefore τ_{val} contains all the classes of τ_{tr} and some additional classes. Note that τ_{val} and τ_{tr} are the random trajectories of length k and s, respectively. In τ_{tr}, samples of each class occur together, i.e samples from class 2 occur after and before samples from class 1 and class 3, respectively. We sample a learning trajectory multiple times for training on a base class set. We randomly sample a learning trajectory by selecting a subset (proper) of classes (randomly) from the base class set. Therefore, every learning trajectory has a different class order.

We follow the MAML setting of continual learning [Javed and White, 2019]. In the inner loop, each class arrives in a sequential manner, and the boundary of each class is available in advance. In the outer-loop, the aim is to minimize the empirical risk over the unseen data, provided the model is optimized on the seen data in an online fashion with the continual learning constraint i.e. when learning from the training data of current class, we are not allowed to access the training data of previous classes. The overall loss function across all trajectories is defined as:

$$\mathbb{E}_{\tau \sim P_{train}(\tau)}[\mathcal{L}(f(\tau_{val}^x|\theta, \mathbf{W}), \tau_{val}^y)] \tag{1}$$

$\mathcal{L}(f(\tau_{val}^x|\theta, \mathbf{W}), \tau_{val}^y)$ denotes the loss on the model f for the validation trajectory of τ. For notational brevity (and slight abuse of notation), we use τ_{val}^x to refer to all the inputs of validation trajectory of task τ and τ_{val}^y to refer to the corresponding true labels.

The function $f : \mathcal{X} \to \mathcal{Y}$ is defined as $f(\tau^x|\theta, \mathbf{W}) = g(h(\tau^x|\theta)|\mathbf{W})$ where $h_\theta : \mathcal{X} \to \mathbb{R}^d$ is defined by parameter θ (representation learning parameters) and $g_{\mathbf{W}} : \mathbb{R}^d \to \mathcal{Y}$ is defined by $\mathbf{W}$. The classifier parameters $\mathbf{W}$ are learned using meta-train set (τ_{tr}), and representation learning parameters θ and $\mathbf{W}$ are jointly learned using the meta-val set (τ_{val}).

In the inner-loop of the meta-learner, which learns $\mathbf{W}$, the model is trained on the meta-train data τ_{tr}. The outer loop, which learns θ and $\mathbf{W}$ is trained using meta-val data τ_{val}. In the outer loop, the model's loss is also computed on novel classes not seen during the inner-loop training since the classes in τ_{tr} are a proper subset of classes in τ_{val}. Evaluation on both sets τ_{tr} and τ_{val} makes the model perform well on both current and previously learned classes. The optimization problems solved by the inner loop and the outer loop are given by:

$$\mathbf{W} = \underset{\mathbf{W}}{\arg\min}\, l_{tr}(\theta, \mathbf{W}) \stackrel{\text{def}}{=} \mathcal{L}(f(\tau_{tr}^x|\theta, \mathbf{W}), \tau_{tr}^y) \tag{2}$$

$$(\theta, \mathbf{W}) = \underset{\theta, \mathbf{W}}{\arg\min}\, l_{val}(\theta, \mathbf{W}) \stackrel{\text{def}}{=} \mathcal{L}(f(\tau_{val}^x|\theta, \mathbf{W}), \tau_{val}^y) \tag{3}$$

In Eq. 2, for notational simplicity, we use the entire training data but during training we perform an online update over τ_{tr}. The above two optimization problems are solved in an alternating fashion using τ_{tr} and τ_{val}, respectively, with the most recent parameter $\mathbf{W}$ obtained from Eq. 2 used in Eq. 3.

4 Knowledge Consolidation based Meta Learning for Training on Base Class Set

While the meta-learning based approach for continual learning described in the above section is promising, it is not particularly effective for our problem setting. One of the reasons for this is the *overparametrized* nature of deep neural networks, in which only a few neurons are activated/fired for all samples. As a result, the network is reliant only on a small set of parameters. Although this may not be a problem for a single task learning setting, it can potentially be an issue in continual learning where we are required to learn a sequence of tasks and, while learning a new task, any changes to these parameters can drastically affect the performance on the older tasks. We overcome this by using a knowledge consolidation based meta-learning approach. Our proposed approach identifies the important and unimportant/dead parameters, rejuvenates the dead parameters, and consolidates the knowledge of the important and reborn parameters (Fig. 1). Therefore, the model capacity is fully utilized, and small changes in a few parameters strongly resist the catastrophic forgetting.

A simple way to assess the importance of a parameter is to use its absolute value [Han *et al.*, 2015], often used in deep model compression. We can discard weights/parameters having small absolute values without sacrificing upon the model's performance [Han *et al.*, 2015]. We leverage this simple idea to identify the important parameters in the model effectively. The proposed approached modifies the meta-learning framework by introducing knowledge consolidation. We define $\mathbf{A} = [\theta, \mathbf{W}]$ as the joint set of parameters of the complete model. We partition model parameters $\mathbf{A}$ into two disjoint sets. The important parameters are denoted by $\mathbf{A}_I$ and the "less important" ones are denoted by $\mathbf{A}_L$, s.t., $\mathbf{A} = \{\mathbf{A}_I, \mathbf{A}_L\}$ and $\mathbf{A}_I \cap \mathbf{A}_L = \varnothing$.

Most of the model's knowledge is contained in $\mathbf{A}_I$. Our goal is to *preserve* the knowledge present in $\mathbf{A}_I$. We apply a weight-constrained regularization on $\mathbf{A}_I$ to ensure minimal changes when a new trajectory is learned. On the other hand, we let $\mathbf{A}_L$ be free to change in order to accommodate new trajectories. Therefore, while learning a new set of trajectories, the following regularized loss function is optimized:

$$\sum_{\tau \sim P_{train}(\tau)} \mathcal{L}(f(\tau_{val}^x|\theta, \mathbf{W}), \tau_{val}^y) + \mathcal{R}(\mathbf{A}_I) \tag{4}$$

One way to define the weight-constrained regularization $\mathcal{R}(\mathbf{A_I})$ would be $\lambda||\mathbf{A}_I^{t+1} - \mathbf{A}_I^t||_F$, where $\mathbf{A}_I^t$ is the important weights after t^{th} step. The large value of λ ensures minimal changes in the important weights.

Naïvely partitioning the model into $\{\mathbf{A}_I, \mathbf{A}_L\}$ (based on absolute value) often does not show any significant improvement since various techniques like dropout and batch normalization force the model's knowledge to be shared across all model parameters, which causes $\mathbf{A}_L$ to contain non-negligible knowledge. Ideally, the value of unimportant weights should be zero. However, in reality, this is not the case and the set $\mathbf{A}_L$ usually contains the non-negligible information. Therefore, we first distill the model's knowledge into a subset of parameters $\mathbf{A}_I^*$ (important parameters set) such that the remaining part $\mathbf{A}_U^*$ (unimportant parameters set) contains negligible information.

The weight-constrained regularization can now be imposed on $\mathbf{A}_I^*$, while the set $\mathbf{A}_U^*$ is free to be adapted for the new trajectories. To transfer/distill the model's knowledge to a subset of the parameters, we *finetune* the complete model with the following ℓ_1 regularized objective:

$$\sum_{\tau \sim P_{train}(\tau)} \mathcal{L}(f(\tau_{val}^x|\theta, \mathbf{W}), \tau_{val}^y) + \gamma||\mathbf{A}||_1 \qquad (5)$$

We can maintain model performance by using an appropriate hyperparameter γ. The ℓ_1 regularizer forces the model knowledge to be squeezed in a subset of model parameters $\mathbf{A}_I^*$. Rest of the model parameters $\mathbf{A}_U^*$ contain negligible information and therefore are free to change. Now, the set $\mathbf{A}$ can be split into important parameters $\mathbf{A}_I^*$ and unimportant parameters set $\mathbf{A}_U^*$, i.e., $\mathbf{A} = \{\mathbf{A}_I^*, \mathbf{A}_U^*\}$ and $\mathbf{A}_I^* \cap \mathbf{A}_U^* = \emptyset$. Given this updated set of important and unimportant/dead set of parameters, the outer-loop optimization of the meta-learner is given by (akin to Eq-4)

$$(\theta, \mathbf{W}) = \underset{\theta, \mathbf{W}}{\arg \min}\, l_{val}(\theta, \mathbf{W}) \overset{\text{def}}{=} \mathcal{L}(f(\tau_{val}^x|\theta, \mathbf{W}), \tau_{val}^y)$$
$$+ \lambda||\mathbf{A}_I^{*t} - \mathbf{A}_I^{*t+1}||_F \qquad (6)$$

$$\mathbf{A}_I^{*t+1} - \mathbf{A}_I^{*t} \approx \nabla_{\mathbf{A}_I^{*t}}(\mathcal{L}(f(\tau_{val}^x|\theta, \mathbf{W}), \tau_{val}^y)) \qquad (7)$$

To preserve the knowledge contained in $\mathbf{A}_I^*$, we apply weight-constrained regularization on $\mathbf{A}_I^*$ as above, which ensures that $\mathbf{A}_I^*$ do not change drastically when new learning-trajectories are encountered. Rest of the parameters $(\mathbf{A}_U^*)$ are free to change. Therefore, we "rejuvenate" the parameters in $\mathbf{A}_U^*$. Representing these rejuvenated set of parameters as $\mathbf{A}_R^*$, now the consolidated knowledge from both $\mathbf{A}_R^*$ and $\mathbf{A}_I^*$ provides a robust representation for our problem setting. Therefore, small changes in a few parameters strongly resist catastrophic forgetting because the model capacity is fully utilized, and model predictions are not reliant on a small set of parameters. As demonstrated by our experiments, such a parameter rejuvenation and knowledge consolidation significantly enhance the performance of a meta-learner based model for class incremental online learning. For a summarized algorithmic description of our approach, please refer to the Algorithms 1, 2, 3, 4.

5 Related Work

Incremental Learning Methods: Rehearsal based incremental learning methods [Rebuffi *et al.*, 2017; Shin *et al.*, 2017; Chaudhry *et al.*, 2019; Isele and Cosgun, 2018; Rolnick *et al.*, 2019; Chaudhry *et al.*, 2021] store a part of training data and re-train on this stored data while training on new tasks. Regularization based incremental learning methods [Lopez-Paz and Ranzato, 2017; Kirkpatrick *et al.*, 2017; Chaudhry *et al.*, 2019] add an additional regularization term in the loss function, which prevents the weights from changing drastically when moving from one task to the next task. DER [Buzzega *et al.*, 2020] uses both rehearsal and regularization. PODNet [Douillard *et al.*, 2020] uses a spatial distillation-loss along with a representation made of proxy vectors from each class. The dynamic network methods [Singh *et al.*, 2020; Singh *et al.*, 2021] are also proposed for incremental learning.

Algorithm 1 Training Algorithm

Require: γ: Learning coefficient for L1 loss
Require: λ: Learning coefficient for constraint loss
Require: δ: Fraction of model parameters ($\frac{|A_I^*|}{|A|}$)
Require: $\{\alpha_i, \beta_i\}_{i=1}^3$: Inner and outer loop lr (learning rate)
Require: $\{steps_i\}_{i=1}^3$: No of steps for KCCIOL algorithm

1: Randomly Initialize model parameters Θ,
2: $mask = 0$
3: $\Theta = \textbf{KCCIOL}(\alpha_1, \beta_1, 0, 0, mask, \Theta, steps_1)$
4: $\Theta = \textbf{KCCIOL}(\alpha_2, \beta_2, 0, \gamma, mask, \Theta, steps_2)$
5: $mask = \textbf{GetMask}(\Theta, \delta)$
6: $\Theta = \textbf{KCCIOL}(\alpha_3, \beta_3, \lambda, 0, mask, \Theta, steps_3)$

Algorithm 2 KCCIOL

Require : $p_{train}(\tau)$: Distribution over learning trajectories
Require : $mask$: Index matrix
Require : Θ: Model parameters

1: **for** $i = 1, \ldots, steps$ **do**
2:     Sample learning trajectory $\tau_i \sim p_{train}(\tau)$
3:     $\{\tau_{tr}, \tau_{val}\} = \tau_i$
4:     $\{\theta, W\} = \Theta$
5:     $W_0 = W$
6:     **for** $j=1, 2, \ldots, k$ **do**
7:         $(X_j, Y_j) = (\tau_{tr}^x[j], \tau_{tr}^y[j])$
8:         $W_j = W_{j-1} - \alpha \nabla_{W_{j-1}}[\mathcal{L}(f(X_j|\theta, W_{j-1}), Y_j)]$
9:     $l_{meta} = [\mathcal{L}(f(\tau_{val}^x|\theta, W_k), \tau_{val}^y)]$
10:    $l_{constraint} = ||mask * \nabla_{\theta, W} l_{meta}||_2^2$
11:    $l_1 = ||\Theta||_1$
12:    Update $\Theta \leftarrow \Theta - \beta \nabla_{\theta, W}(l_{meta} + \lambda l_{constraint} + \gamma l_1)$
13: **return** Θ

Algorithm 3 Mask Calculation

Require: Θ: Model Parameters
Require: δ: Fraction of model parameters ($\frac{|A_I^*|}{|A|}$)

1: **procedure** GETMASK(Θ, δ)
2:    $threshold = percentile(|\Theta|, 1 - \delta)$
3:    $mask = zeros(len(\Theta))$
4:    $index = 0$
5:    **while** $index <= len(\Theta)$ **do**
6:       **if** $mask[index] >= threshold$ **then**
7:          $mask[index] = 1$
8:          $index + +$
9:    **return** $mask$

Algorithm 4 Evaluation Protocol

Require : $p_{test}(\tau)$: Distribution over learning trajectories
Require : θ: Representation Learning Parameters

1: Randomly Initialize W
2: Sample learning trajectory $\tau_i \sim p_{test}(\tau)$
3: $\{\tau_{tr}, \tau_{val}\} = \tau_i$
4: **for** $j=1, \ldots, k$ **do**
5:    $(X_j, Y_j) = (\tau_{tr}^x[j], \tau_{tr}^y[j])$
6:    $W_j = W_{j-1} - \alpha \nabla_{W_{j-1}}[\mathcal{L}(f(X_j|\theta, W_{j-1}), Y_j)]$
7: **return** $Accuracy(f(\tau_{val}^x|\theta, W_k), \tau_{val}^y)$

Meta-Learning Methods: MER [Riemer *et al.*, 2019] uses meta-learning to learn parameters that prevent interference and enable knowledge transfer based on future gradients. OML [Javed and White, 2019] is a meta-learning approach that focuses on learning a generic representation that prevents catastrophic forgetting during online learning. OSAKA [Caccia *et al.*, 2020] is a general approach to continual learning where the agent must be able to solve previously unseen distribution tasks with minimal forgetting on previous tasks. OSAKA is different from the standard incremental learning setting as seen tasks can be revisited, and online average accuracy is reported at the end of the training instead of reporting accuracy on all seen tasks. Therefore, it would be unfair to compare our method with OSAKA.

Online Continual Learning Methods: MERLIN [K J and Balasubramanian, 2020] is a replay based method for online continual learning which learns a meta-distribution from which task specific parameters are sampled at the time of inference. [von Oswald *et al.*, 2020] proposed a method similar to MERLIN where deterministic task specific weights are generated using hypernetworks [Ha *et al.*, 2017]. GSS uses constrained optimization for sampling samples for replay instead of random sampling for online continual learning. Incremental learning in online scenario [He *et al.*, 2020] tackles the problem of catastrophic forgetting in the online scenario under different setting than ours. [He *et al.*, 2020] is a memory based method where future data consists of samples from new classes as well as unseen samples from old classes, whereas, in our approach, data from one class are seen together. CTN [Pham *et al.*, 2021] is a bi-level optimization network that uses a context network to model task-specific features which address catastrophic forgetting and knowledge transfer. But the context network needs task-specific knowledge and semantic memory to function, whereas our method does use replay and is task-free.

6 Experiments

We evaluate our approach via extensive experiments across various datasets. We follow the evaluation protocol where the model is updated in an online fashion and later evaluated on the unseen data (Section 2). We compare the performance of our model (KCCIOL) against several baselines.

Baselines: In the *Scratch* baseline, we evaluate the performance of a randomly initialized network. In *Pretrained* baseline, the network is pretrained on the train set. The *SRNN* approach uses a Set-KL method proposed by [Liu *et al.*, 2019] to learn a sparse representation using train set. *MRCL* is a recent approach originally proposed by [Javed and White, 2019] to train a model in a meta-learning setup for continual learning tasks. *OML* is a modified version of MRCL proposed by [Javed and White, 2019] where the classifier parameters are randomly re-initialized at each step of the training. *MAML-Rep* is also a MAML [Finn *et al.*, 2017] based algorithm similar to OML and MRCL, where batch updates are performed in the inner-loop.

6.1 Experiments on Omniglot Dataset:

Implementation Details: The Omniglot dataset [Lake *et al.*, 2015] contains 1623 classes of different handwritten char-

Classes	Scratch	Pretrained	SRNN	MRCL	MAML-Rep	OML	Ours
10	15.9 ± 3.5	42.6 ± 10	70.4	83.8 ± 6.2	86.1	92.6 ± 3.5	$\mathbf{95.4 \pm 3.4}$
50	2.4 ± 1.4	24.4 ± 4.3	53.9	66.5 ± 4.0	71.3	81.3 ± 2.4	$\mathbf{85.8 \pm 2.5}$
100	1.5 ± 0.3	15.5 ± 1.9	44.3	51.8 ± 2.6	70.0	76.1 ± 2.0	$\mathbf{81.4 \pm 2.3}$
150	1.2 ± 0.5	11.7 ± 1.1	27.0	42.8 ± 2.5	53.0	65.2 ± 2.2	$\mathbf{77.1 \pm 1.6}$
200	0.8 ± 0.5	8.0 ± 1.1	18.3	33.8 ± 1.8	35.7	59.3 ± 1.8	$\mathbf{72.6 \pm 1.5}$

Table 1: Classification accuracy (mean±std) on the omniglot dataset averaged across 50 test trajectories randomly sampled from the meta-test set. Classes column refers to the total number of classes in the sampled trajectory.

acters from 50 different alphabets. Each class contains 20 samples with 15/5 as the train/test split. The first 963 classes constitute the $(\mathcal{X}_{train}, \mathcal{Y}_{train})$ and the remaining classes are used as $(\mathcal{X}_{test}, \mathcal{Y}_{test})$. For learning trajectory during training, τ_{tr} consists of 10 samples from a class randomly sampled from the training set. τ_{val} consists of 10+1 samples where ten samples are randomly sampled from the train set, and the 11th sample belongs to the class used in τ_{tr}. During the evaluation, each learning trajectory consists of an equal number of classes in both τ_{tr} and τ_{val} which are sampled from the test set. We use 15/5 samples per class for τ_{tr}/τ_{val} during evaluation. We use Adam optimizer, and the first six layers are used for learning representation.

Hyperparameter Settings: We train our model using hyperparameters: $\beta_1 = $ 1e-4, $\alpha_1 = $ 1e-2, $steps_1 = 20000$, $\beta_2 = $ 1e-4, $\alpha_2 = $ 1e-2 , $\gamma = $5e-5, $steps_2 = 15000$, $\beta_3 = $ 1e-4, $\alpha_3 = $ 1e-2 , $\lambda = $5e-4, $steps_3 = 4000$, $\delta = 0.5$.

Model Architecture: We use six convolutional layers followed by two fully connected layers, and each convolutional layer contains 256 filters of 3×3 kernel size with (2, 1, 2, 1, 2, 2) strides (same as used in [Javed and White, 2019]). ReLU activation function is used for the non-linearity.

The proposed approach KCCIOL performs better by a significant margin on a wide range of classes as compared to the other baselines. We achieve a significant performance boost of 13.3 %, when 200 classes are learnt continually as shown in Table 1.

6.2 Experiments on Mini-Imagenet Dataset

Implementation Details: Vinyals et al. [Vinyals *et al.*, 2016] proposed the mini-imagenet dataset, which is a subset of the imagenet dataset. There are a total of 100 classes with 600 colored images of size 84×84. We use 64 classes for training and 20 classes for testing. For learning trajectory during training, τ_{tr} consists of 10 samples from a class randomly sampled from the training set. τ_{val} consists of 15 samples where 10 samples are randomly sampled from the training set, and 5 samples belong to the class used in τ_{tr}. During the evaluation, we sample classes from the test set for creating a learning trajectory. We use 30 samples per class for τ_{tr} and τ_{val}. We use Adam optimizer, and the first six layers are used for learning representations.

Hyperparameter Settings: We train our model using hyperparameters: $\beta_1 = $ 1e-4, $\alpha_1 = $ 1e-1, $steps_1 = 26000$, $\beta_2 = $ 1e-4, $\alpha_2 = $ 1e-1 , $\gamma = $1e-4, $steps_2 = 26000$, $\beta_3 = $ 1e-4, $\alpha_3 = 0.5$, $\lambda = $1e-4, $steps_3 = 22000$, $\delta = 0.5$.

Model Architecture: The model architecture for the Mini-Imagenet is same as used in the Omniglot dataset.

Classes	Scratch	Pretrained	SRNN	MRCL	OML	Ours
6	19.3 ± 4.0	16.9	34.2	41.8 ± 9.0	37.0 ± 6.8	$\mathbf{49.4 \pm 6.5}$
8	12.9 ± 1.3	13.0	29.2	36.8 ± 7.1	34.6 ± 4.2	$\mathbf{45.3 \pm 4.9}$
10	10.1 ± 0.9	10.0	25.8	34.4 ± 6.5	33.2 ± 4.6	$\mathbf{42.7 \pm 4.6}$
12	8.5 ± 0.9	8.5	23.1	30.6 ± 4.9	29.5 ± 3.2	$\mathbf{39.4 \pm 3.0}$
14	9.1 ± 1.9	8.8	20.4	29.6 ± 4.5	27.8 ± 2.5	$\mathbf{37.0 \pm 3.7}$
16	7.4 ± 1.8	10.8	18.1	28.0 ± 3.5	25.2 ± 2.0	$\mathbf{33.4 \pm 2.1}$
18	5.5 ± 0.7	10.4	17.7	26.3 ± 3.3	23.9 ± 2.1	$\mathbf{31.9 \pm 2.2}$
20	6.2 ± 1.4	10.0	17.3	25.5 ± 2.8	22.9 ± 1.6	$\mathbf{29.7 \pm 1.4}$

Table 2: Classification accuracy (mean±std) on mini-imagenet dataset averaged across 50 test trajectories randomly sampled from the meta-test set. Classes column refers to the total number of classes in the sampled trajectory.

Method	Standard	Pre-Training	OML	Ours
Online	4.64 ± 2.61	21.16 ± 2.71	64.72 ± 2.57	$\mathbf{85.68 \pm 2.10}$
Approx. IID	53.95 ± 5.50	54.29 ± 3.48	75.12 ± 3.24	$\mathbf{88.66 \pm 2.10}$
MER	54.88 ± 4.12	62.76 ± 2.16	76.00 ± 2.07	$\mathbf{91.28 \pm 1.38}$
EWC	5.08 ± 2.47	18.72 ± 3.97	64.44 ± 3.13	$\mathbf{87.10 \pm 1.40}$
ER-Reservoir	52.56 ± 2.12	36.72 ± 3.06	68.16 ± 3.12	$\mathbf{90.10 \pm 1.35}$

Table 3: KCCIOL combined with existing continual learning methods on the omniglot dataset. We use 50 tasks with 1 class per task. The accuracies are averaged over 10 runs.

Our method KCCIOL consistently outperforms others as shown in Table 2. We get a performance boost upto 12.4 % over the OML.

7 KCCIOL Complements Existing Continual Learning Methods

We have demonstrated the efficiency of KCCIOL when learning-trajectories are learnt in a continual online fashion. In this section, we will demonstrate that the representation generated by KCCIOL not only performs well on online updates but it also greatly improves the performance of other continual learning methods such as MER [Rolnick *et al.*, 2019], EWC [Kirkpatrick *et al.*, 2017], ER-Reservoir [Rolnick *et al.*, 2019] when they use our model as base network. Online, Approx IID baselines have been taken from [Javed and White, 2019].

We evaluate the model's performance under four different settings for each continual learning method, which are Standard, Pre-Training, OML, and KCCIOL. In OML and KCCIOL, we use the θ generated by these methods as the base of the model and do not update the θ parameters. In the Pre-Training setting, we train the model independently on the training data and again keep the θ fixed during the process. The difference between the Pre-Training and Standard setting is that all the parameters of the model are updated in the case of the Standard setting, while in the pre-training, only $\mathbf{W}$ are updated. In the standard setting, the model is pretrained independently on train data to prevent other settings from having an unfair advantage of being trained on the training set. From Tables-3, 4 we can observe that KCCIOL with just online updates significantly outperforms other continual learning methods even when they are combined with OML. KCCIOL can do so because it can learn a generic robust feature representation that is generalizable across all tasks, enabling the model to perform future learning. For the train/test split, we use 15/5 samples per class. Even without using any mem-

Method	Standard	Pre-Training	OML	Ours
Online	1.40 ± 0.43	11.80 ± 1.92	55.32 ± 2.25	$\mathbf{80.10 \pm 1.71}$
Approx. IID	48.02 ± 5.67	46.02 ± 2.83	67.03 ± 2.10	$\mathbf{85.90 \pm 1.76}$
MER	29.02 ± 4.01	42.05 ± 3.71	62.05 ± 2.19	$\mathbf{83.42 \pm 1.67}$
EWC	2.04 ± 0.35	10.03 ± 1.53	56.03 ± 3.20	$\mathbf{82.90 \pm 1.27}$
ER-Reservoir	24.32 ± 5.37	37.44 ± 1.67	60.92 ± 2.41	$\mathbf{84.76 \pm 1.12}$

Table 4: KCCIOL combined with existing continual learning methods on omniglot. We use 100 tasks with 5 classes per task. The accuracies are averaged over 10 runs.

Classes	DER	DER++	HAL	MERLIN	GSS	PODNet	Ours
6	27.22 ± 6.39	36.77 ± 5.73	26.33 ± 3.44	22.78	35.00	42.68 ± 2.33	$\mathbf{49.4 \pm 5.5}$
8	22.29 ± 5.42	28.13 ± 3.46	20.63 ± 3.11	14.16	30.00	36.10 ± 2.00	$\mathbf{45.3 \pm 4.9}$
10	19.0 ± 2.69	23.5 ± 4.14	15.93 ± 4.14	11.24	24.33	31.48 ± 1.56	$\mathbf{42.7 \pm 4.6}$
12	15.58 ± 3.35	19.14 ± 3.51	14.86 ± 2.45	9.42	17.78	27.94 ± 1.42	$\mathbf{39.4 \pm 3.0}$
14	13.93 ± 2.18	17.14 ± 3.27	12.36 ± 1.58	7.53	17.14	25.15 ± 1.29	$\mathbf{37.0 \pm 3.7}$
16	13.33 ± 2.83	14.21 ± 2.83	10.75 ± 1.56	6.83	11.87	22.93 ± 1.29	$\mathbf{33.4 \pm 2.1}$
18	11.48 ± 1.97	13.15 ± 2.96	9.46 ± 1.55	6.16	12.41	21.14 ± 1.22	$\mathbf{31.9 \pm 2.2}$
20	9.68 ± 3.07	12.48 ± 2.08	8.9 ± 1.50	6.11	13.00	19.63 ± 1.14	$\mathbf{29.7 \pm 1.4}$

Table 5: Performance comparison with recent SOTA incremental learning methods on the mini-imagenet dataset. Classification accuracy (mean±std) is averaged across 50 test trajectories randomly sampled from the meta-test set. Classes column refers to the total number of classes in the sampled trajectory.

ory/replay, KCCIOL can outperform the replay based methods by a significant margin.

8 Performance Comparison with Recent SOTA Incremental Learning Methods

We also compare our method KCCIOL with the recent state-of-the-art incremental learning methods on the mini-imagenet dataset under incremental online learning setting. From Table 5, we can observe that recent incremental learning methods such as DER [Buzzega *et al.*, 2020], PODNet [Douillard *et al.*, 2020], HAL [Chaudhry *et al.*, 2021] perform poorly in an incremental online setting. Even SOTA methods for online learning, such as GSS [Aljundi *et al.*, 2019], MERLIN [K J and Balasubramanian, 2020] suffer from catastrophic forgetting when training data is limited. We only use thirty samples per class for experiments on the mini-imagenet dataset. We observe that KCCIOL outperforms the next best method by more than 10% when ten classes are learnt continually. Also, all of the above-mentioned methods are replay based approaches, while KCCIOL does not use any replay/memory.

9 Conclusion

We propose a novel approach to learn robust feature representations that are generalizable across future learning classes. Our approach uses a meta-learning framework with knowledge consolidation for learning generic feature representations that can be rapidly adapted for future classes without forgetting the previous classes under online updates to facilitate future learning. Our approach shows a significant improvement for class incremental online learning on several benchmark datasets.

Acknowledgements

References

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

[Buzzega *et al.*, 2020] Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. In *Advances in Neural Information Processing Systems*, 2020.

[Caccia *et al.*, 2020] Massimo Caccia, Pau Rodriguez, Oleksiy Ostapenko, Fabrice Normandin, Min Lin, Lucas Caccia, Issam Laradji, Irina Rish, Alexande Lacoste, David Vazquez, et al. Online fast adaptation and knowledge accumulation: a new approach to continual learning. In *Advances in Neural Information Processing Systems*, 2020.

[Chaudhry *et al.*, 2019] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-GEM. In *International Conference on Learning Representations*, 2019.

[Chaudhry *et al.*, 2021] Arslan Chaudhry, Albert Gordo, Puneet K Dokania, Philip Torr, and David Lopez-Paz. Using hindsight to anchor past knowledge in continual learning. In *Thirty-Fifth AAAI conference on artificial intelligence*, 2021.

[Douillard *et al.*, 2020] Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. Podnet: Pooled outputs distillation for small-tasks incremental learning. In *Proceedings of the IEEE European Conference on Computer Vision (ECCV)*, 2020.

[Finn *et al.*, 2017] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org, 2017.

[Ha *et al.*, 2017] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. In *International Conference on Learning Representations*, 2017.

[Han *et al.*, 2015] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pages 1135–1143, 2015.

[He *et al.*, 2020] Jiangpeng He, Runyu Mao, Zeman Shao, and Fengqing Zhu. Incremental learning in online scenario. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[Isele and Cosgun, 2018] David Isele and Akansel Cosgun. Selective experience replay for lifelong learning. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

[Javed and White, 2019] Khurram Javed and Martha White. Meta-learning representations for continual learning. In *Advances in Neural Information Processing Systems*, pages 1818–1828, 2019.

[K J and Balasubramanian, 2020] Joseph K J and Vineeth Balasubramanian. Meta-consolidation for continual learning. In *Advances in Neural Information Processing Systems*, 2020.

[Kirkpatrick *et al.*, 2017] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, pages 3521–3526, 2017.

[Lake *et al.*, 2015] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, pages 1332–1338, 2015.

[Liu *et al.*, 2019] Vincent Liu, Raksha Kumaraswamy, Lei Le, and Martha White. The utility of sparse representations for control in reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4384–4391, 2019.

[Lopez-Paz and Ranzato, 2017] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, pages 6467–6476, 2017.

[Parisi *et al.*, 2019] German Ignacio Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual Lifelong Learning with Neural Networks: A Review. *Neural Networks*, 2019.

[Pham *et al.*, 2021] Quang Pham, Chenghao Liu, Doyen Sahoo, and Steven CH Hoi. Contextual transformation networks for online continual learning. In *International Conference on Learning Representations*, 2021.

[Raghu *et al.*, 2020] Aniruddh Raghu, Maithra Raghu, Samy Bengio, and Oriol Vinyals. Rapid learning or feature reuse? towards understanding the effectiveness of maml. In *International Conference on Learning Representations*, 2020.

[Rebuffi *et al.*, 2017] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.

[Riemer *et al.*, 2019] Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. In *International Conference on Learning Representations*, 2019.

[Rolnick *et al.*, 2019] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. In *Advances in Neural Information Processing Systems*, pages 348–358, 2019.

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

[Singh *et al.*, 2020] Pravendra Singh, Vinay Kumar Verma, Pratik Mazumder, Lawrence Carin, and Piyush Rai. Calibrating cnns for lifelong learning. *Advances in Neural Information Processing Systems*, 33, 2020.

[Singh *et al.*, 2021] Pravendra Singh, Pratik Mazumder, Piyush Rai, and Vinay P Namboodiri. Rectification-based knowledge retention for continual learning. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[Vinyals *et al.*, 2016] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in neural information processing systems*, pages 3630–3638, 2016.

[von Oswald *et al.*, 2020] Johannes von Oswald, Christian Henning, João Sacramento, and Benjamin F Grewe. Continual learning with hypernetworks. In *International Conference on Learning Representations*, 2020.