More Classifiers, Less Forgetting: A Generic Multi-classifier Paradigm for Incremental Learning

Yu Liu[1], Sarah Parisot[2,3], Gregory Slabaugh[2], Xu Jia[2], Ales Leonardis[2], and Tinne Tuytelaars[1]

[1]KU Leuven [2]Huawei Noah's Ark Lab [3]Mila
{firstname.lastname}@kuleuven.be {firstname.lastname}@huawei.com

Abstract. Overcoming catastrophic forgetting in neural networks is a long-standing and core research objective for incremental learning. Notable studies have shown regularization strategies enable the network to remember previously acquired knowledge devoid of heavy forgetting. Since those regularization strategies are mostly associated with classifier outputs, we propose a MUlti-Classifier (MUC) incremental learning paradigm that integrates an ensemble of auxiliary classifiers to estimate more effective regularization constraints. Additionally, we extend two common methods, focusing on parameter and activation regularization, from the conventional single-classifier paradigm to MUC. Our classifier ensemble promotes regularizing network parameters or activations when moving to learn the next task. Under the setting of task-agnostic evaluation, our experimental results on CIFAR-100 and Tiny ImageNet incremental benchmarks show that our method outperforms other baselines. Specifically, MUC obtains 3%∼5% accuracy boost and 4%∼5% decline of forgetting ratio, compared with MAS and LwF. Our code is available at https://github.com/Liuy8/MUC.

Keywords: Incremental learning; Regularization; Classifier ensemble

1 Introduction

Incremental learning dates back decades, but has recently shown an increased popularity due to the renewed interest in deep neural networks [33,20]. Unlike standard multi-task learning, the tasks during incremental learning arrive sequentially, and the data of previous tasks is not accessible anymore (*e.g.* , due to memory limits or privacy issues). Here, we consider the *class-incremental learning* setup [35,16], in which each new task learns a set of classes disjoint from the old tasks. The network needs to learn feature representations for classifying the images of old and new classes. Besides, we adopt a *task-agnostic* evaluation: at test time it is unknown which task an image sample belongs to.

The major challenge in incremental learning is the so-called *catastrophic forgetting* [29], a phenomenon where previously acquired knowledge is lost from the network after it is trained on the newly incoming task. To reduce forgetting,

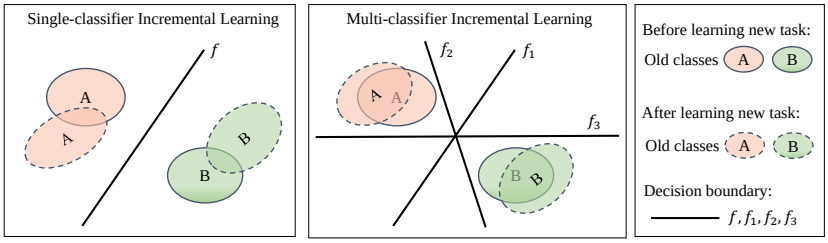


Fig. 1: Conceptual comparison between single-classifier and multi-classifier incremental learning. Our multi-classifier paradigm is better at regularizing the feature distributions of old classes than the single-classifier paradigm.

a large set of methods exploit *regularization strategies* to constrain changes of network parameters or activations. When learning a new task, the network is updated by combining the regularization loss with the standard classification loss. The objective is to find the optimal trade-off between the *adaptation* to new tasks and the *preservation* on previous tasks. Most regularization strategies are closely associated with *classifier outputs*: (1) *parameter regularization* methods such as EWC [18] and MAS [1] estimate an importance weight for each parameter in the network and penalize changes to important parameters. The computation of those importance weights is based on the loss or output of the classifier. (2) *activation regularization* methods like LwF [24] introduce a knowledge distillation based regularization that enforces the classifier outputs of the new model to be close to those of the old model. In both regularization methods, the classifier is crucial not only for classifying new tasks, but also for regularizing old ones. However, these existing methods learn a single classifier only for each task and their regularization strategies are heavily limited by the output of one single classifier. Motivated by the above finding, our work aims to address the question: *How to exploit more classifiers to improve the effectiveness of the regularization strategies for incremental learning?*

To this end, we propose a MUlti-Classifier (MUC) paradigm that integrate *classifiers ensemble* to estimate more effective regularization constraints. First, we train a standard neural network with in-distribution data of current task. Then, we construct upon the network a set of new and auxiliary classifiers which are trained on out-of-distribution data irrelevant to current task. To enhance the discrepancy among those classifiers, we train a *classifier discrepancy loss* to maximize prediction disagreement on the out-of-distribution data and agreement on the in-distribution data. Despite that those classifiers make different decision boundaries for the same classification objective, they help to produce complementary and robust information to regularize forgetting of previously learned classes. We show in Fig. 1 how MUC work differently from conventional single-classifier paradigm. MUC is a generic method and can be integrated with most pre-existing regularization strategies. Additionally, we show MUC leverages multiple classifiers for improving two common incremental learning methods, focusing on parameter and activation regularization, respectively.

The contributions of this paper are summarized below:

- We propose a novel and generic multi-classifier incremental learning paradigm, coined MUC. which demonstrates the effectiveness of taking into account the role of the classifier for reducing forgetting. This work is the first to exploit the classifier discrepancy for incremental learning.
- We introduce two instantiations based on MUC, by extending parameter and activation regularization, respectively. It suggests improving existing regularization strategies is also important for incremental learning.
- In the setting of class-incremental learning, we experiment with CIFAR-100 and Tiny ImageNet incremental benchmarks, where MUC achieves considerable and promising improvements over the single-classifier paradigm. Extensive analysis additionally verifies the strengths of MUC.

2 Related Work

In recent years, incremental learning has become one of the most critical yet challenging directions in a broad spectrum of application domains, including image classification [24,35], object detection [39,10] and semantic segmentation [30,5]. Due to the "*stability-plasticity*" dilemma in neural networks [3,29], incremental learners perform well on the latest task but witness a dramatic degradation of performance on previous tasks. To alleviate such a forgetting issue, extensive regularization strategies have been proposed in the literature, which can be grouped into two main categories below.

The first category is normally called *parameter regularization* [18,48,23,1,6,25] that penalizes drastic updates of important parameters when the network is learning a new task. The intuition is that keeping the important parameters for old tasks intact can reduce forgetting while the remaining parameters learn to adapt to the incoming new task. Being one of the most representative approaches, Elastic Weight Consolidation (EWC) [18] estimated the parameters' importance to the change in loss function by the diagonal of the Fisher information matrix (FIM). Memory Aware Synapses (MAS) [1] presented a new importance weight through the gradient of the L2-normalization outputs *w.r.t.* the parameter. Nevertheless, devising a robust manner to formulate importance weights is still an open and challenging problem. The second category is *activation regularization* that imposes regularization constraints on the feature activations in the network rather than on the parameters themselves. Learning without Forgetting (LwF) [24], being a fundamental approach in this category, fed the data of the new task into the stored model and recorded the output probabilities as soft targets. Then a knowledge distillation loss [13] was used to encourage the newly updated model to produce similar predictions as the soft labels. Upon LwF, many approaches have been proposed to improve the regularization based on knowledge distillation [43,7,50,22,49]. For instance, LwM [7] considered adding knowledge distillation on the feature activations of intermediate layers. Instead of designing a specific regularization strategy, our work presents a generic paradigm in which existing strategies can be improved to further reduce forgetting.

Next to the above regularization, other rehearsal based methods [35,26,17,15,2] store some old data to make the network remember previous tasks, albeit violating the motivation of incremental learning to some extent. iCaRL [35] used an external memory to store a subset of data samples (*a.k.a.* exemplars) for old classes. It also employed a knowledge distillation loss to help regularize the update of the network. Additionally, some research efforts are made to address the data imbalance between a large amount of new classes samples and a small budget of old classes samples [4,14,44]. Inspired by the success of Generative Adversarial Networks (GANs), a few works [38,11,45] proposed training generative networks to produce pseudo-rehearsal samples instead of storing the original and real data. Unlike these works, our MUC is not limited by the need of storing and re-using old data. Nevertheless, we empirically in the experiment show its effectiveness under the rehearsal-based scenario.

Multi-classifier learners have been studied in several vision tasks [42,21,46]. On the one hand, the research objective is to maximize the *consensus* of outputs from multiple classifiers, to consolidate the transfer learning from source domain to target domain [27,8]. On the other hand, the objective is to maximize the *discrepancy* of the classifiers' predictions. For example, the approach in [47] enlarged the discrepancy between two classifiers to separate in-distribution samples from out-of-distribution samples. Focusing on unsupervised domain adaptation, the work in [36] combined the above two objectives in an adversarial learning manner. It first maximized the discrepancy for the target samples and then minimized the discrepancy for feature generation. However, our work aims to exploit the classifier discrepancy for incremental learning.

3 Proposed Method

Overall idea. We focus on the class-incremental learning (CIL) setup, in which the model continually learns more classes from new tasks while retaining the recognition of old classes from previous tasks. Note that, we mainly follow the standard setting, without re-using the image samples from previous tasks. First of all, we introduce how the MUC paradigm is trained in a two-stage fashion (in Sec. 3.1). In the first stage, we add a main classifier (*i.e.* the last layer of the network) on top of the feature extractor (*i.e.* the earlier layers of the network) and update the entire network to correctly classify the newly incremental classes. During the second stage, we freeze the feature extractor and train a set of additional side classifiers (*i.e.* newly auxiliary layers) in parallel to the main classifier. Since most CIL methods heavily rely on a single classifier, we then extend two instantiations based on MUC (Sec. 3.2 and Sec. 3.3), by utilizing multiple classifiers for more robust and effective regularization.

Problem notation. Suppose that there are T of sequential tasks together with their data $\{X^t, Y^t\}_{t=1}^{T}$. $X^t = \{x_i^t\}_{i=1}^{N^t}$ and $Y^t = \{y_i^t\}_{i=1}^{N^t}$ are the input images and their ground-truth labels, where N^t denotes the number of image samples in task t. Task t contains a number of C^t classes, and the classes from different tasks should be disjoint. The feature extractor F is shared across all tasks, but

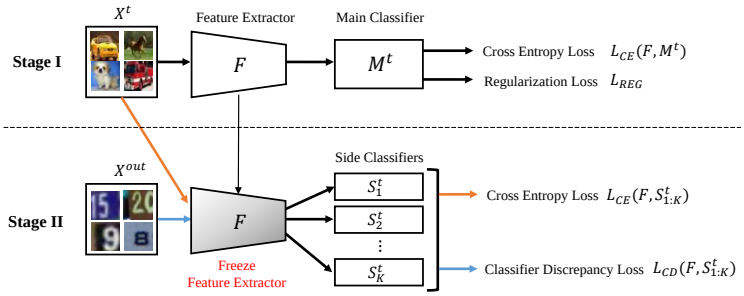


Fig. 2: Pipeline of training MUC in a two-stage fashion. The two stages optimize the cross-entropy loss $\mathcal{L}_{CE}$ for a main classifier M^t and a set of K side classifiers $S_{1:K}^t$, respectively. The regularization term $\mathcal{L}_{REG}$ in `Stage I` is used to reduce forgetting for previous tasks. The classifier discrepancy loss $\mathcal{L}_{CD}$ in `Stage II` aims to diversify the side classifiers.

is updated continually by the data. The main classifier M^t and K side classifiers $S_{1:K}^t = \{S_k^t\}_{k=1}^K$ are associated with task t. After training all tasks, the sets of main classifiers and side classifiers are denoted as $\{M^t\}_{t=1}^T$ and $\{S_{1:K}^t\}_{t=1}^T$.

3.1 Multi-classifier Incremental Learning

Our MUC for incrementally learning tasks is performed in two stages (Fig. 2).

Stage I: train feature extractor and main classifier. During incremental learning, the feature extractor is trained from scratch for the first task and is then updated continually by subsequent tasks. For each task, its main classifier is randomly initialized and newly trained. Given the training data $\{X^t, Y^t\}$ for the new task t, we minimize the standard cross-entropy (CE) loss and optimize the feature extractor F and the main classifier M^t simultaneously. Additionally, it is crucially necessary for incremental learning to impose a regularization loss $\mathcal{L}_{REG}$ that is used to constrain the updates of important parameters associated with previous tasks. The objective in this stage becomes

$$\mathcal{L}_{\text{stage}_\text{I}} = \mathcal{L}_{CE}(F, M^t) + \lambda \mathcal{L}_{REG} = \sum_{i=1}^{N^t} -\log[p(y_i^t|x_i^t)] + \lambda \mathcal{L}_{REG}, \tag{1}$$

where $p(y_i^t|x_i^t)$ is the Softmax probability for the ground-truth label y_i^t. λ is a trade-off hyper-parameter. We will detail $\mathcal{L}_{REG}$ in later subsections.

Stage II: freeze feature extractor and train side classifiers. This stage seeks to learn new side classifiers for task t. Specifically, the same feature extractor F transferred from `Stage I` is frozen during the training of `Stage II`. We develop upon the feature extractor a set of K side classifiers, each of which learns to correctly classify the same C^t classes like M^t. To jointly train these side classifiers, we accumulate their CE loss to be

$$\mathcal{L}_{CE}(F, S_{1:K}^t) = \sum_{i=1}^{N^t} \sum_{k=1}^{K} -\log[p_k(y_i^t|x_i^t)], \tag{2}$$

where $p_k(y_i^t|x_i^t)$ is the prediction probability from the k-th side classifier S_k^t.

However, training only with a classification objective leads to nearly identical side classifiers. The side classifiers learn similar parameters including weights and bias. Consequently, the identical classifiers produce the same regularization terms that have no benefit for further reducing the forgetting ratio on the old tasks. To make the side classifiers learn different parameters, we additionally maximize the *classifier discrepancy* (or *disagreement*) when training the side classifiers $S_{1:K}^t$. Maximum classifier discrepancy (MCD) has been used in other areas [47,36], but this work is the first to exploit it for incremental learning. First, we need to choose an *out-of-distribution* (OOD) dataset X^{out}, which contains N^{out} samples that are totally different from the in-distribution classes in the tasks. The OOD samples can be unlabeled, as the classifier discrepancy loss does not need to use their labels. Given any OOD sample $x^{out} \in X^{out}$, we compute the classifier discrepancy with the side classifiers' probabilistic vectors. The classifier discrepancy between any two probabilistic vectors is the L1-norm distance of their absolute difference

$$d(\mathbf{p}_m(y|x^{out}), \mathbf{p}_n(y|x^{out})) = |\mathbf{p}_m(y|x^{out}) - \mathbf{p}_n(y|x^{out})|, \tag{3}$$

where $\mathbf{p}_m(y|x^{out})$ and $\mathbf{p}_n(y|x^{out})$ represent the C^t-dimensional probabilistic vectors predicted by S_m^t and S_n^t, respectively. For K side classifiers, there are $\binom{K}{2}$ many possible pairs. The total classifier discrepancy loss is denoted by

$$\mathcal{L}_{CD}(F, S_{1:K}^t) = \sum_{i=1}^{N^{out}} \sum_{m=1}^{K} \sum_{n=m+1}^{K} d(\mathbf{p}_m(y|x_i^{out}), \mathbf{p}_n(y|x_i^{out})). \tag{4}$$

Finally, the objective of `Stage II` is to minimize the classification cost and at the same time maximize the classifier discrepancy

$$\mathcal{L}_{\text{stage}_{\text{II}}} = \mathcal{L}_{CE}(F, S^t) - \mathcal{L}_{CD}(F, S_{1:K}^t). \tag{5}$$

Consequently, these side classifiers become distinct by learning different parameters for the same task. In addition, they retain the *agreement* on the samples of task t while increasing the *disagreement* on the samples of OOD dataset. The core in incremental learning is how to impose an extra regularization term to consolidate previous knowledge when learning the next task. In the following two subsections, we extend pre-existing regularization strategies from the single-classifier paradigm to the MUC paradigm.

3.2 MUC with Parameter Regularization

Here, we present how to perform the parameter regularization (PR) methods in our MUC paradigm. Without loss of generality, we employ the importance weight defined in Memory aware synapses (MAS) [1], while MUC can also handle with the importance weights in other PR methods. Our method, namely MUC-MAS, enables to estimate importance weights from not only the main classifier, but also additional side classifiers. To be specific, after training the task $t-1$ in `Stage I`, the importance weight per parameter is denoted by

$$\alpha_j^{t-1} = \frac{1}{N^{t-1}} \sum_{i=1}^{N^{t-1}} \left|\left| \frac{\partial[l_2^2(M^{t-1}(F(x_i^{t-1})))]}{\partial \theta_j} \right|\right|, \tag{6}$$

where $\theta_j \in \boldsymbol{\theta}$ are the parameters in the feature extractor, and $M^{t-1}(F(\cdot))$ is the output before the Softmax function. We do not compute importance weights for the parameters of the classifiers, because they will be fixed once the network starts to learn the next task. Likewise, the side classifiers learned in `Stage II` are also used to estimate more importance weights. The feature extractor is fixed during `Stage II`, the side classifiers, however, are able to provide diverse outputs due to their different parameters. Thereby, the importance weight $\delta_{j,k}^{t-1}$ based on the k-th side classifier S_k^{t-1} becomes

$$\delta_{j,k}^{t-1} = \frac{1}{N^{t-1}} \sum_{i=1}^{N^{t-1}} \left|\left| \frac{\partial [l_2^2(S_k^{t-1}(F(x_i^{t-1})))]}{\partial \theta_j} \right|\right|. \tag{7}$$

We further average the importance weights from K side classifiers by

$$\delta_j^{t-1} = \frac{1}{K} \sum_{k=1}^{K} \delta_{j,k}^{t-1}. \tag{8}$$

Moreover, we propose a new property called *stability factor*, which allows us to assess how stable the parameters in the network are. For each parameter, if its importance weights from the side classifiers are close with that from the main classifier, it shows that this parameter is robust and stable to different classifiers. In this case, we assign this parameter with a larger stability factor. To be specific, we compute the standard deviation *w.r.t.* the importance weights

$$\mathrm{std}(\theta_\mathrm{j}) = \frac{1}{\alpha_\mathrm{j}^{\mathrm{t-1}}} \sqrt{\frac{\sum_{\mathrm{k=1}}^{\mathrm{K}} (\delta_{\mathrm{j,k}}^{\mathrm{t-1}} - \alpha_\mathrm{j}^{\mathrm{t-1}})^2}{\mathrm{K}}}. \tag{9}$$

This standard deviation $\mathrm{std}(\theta_\mathrm{j})$ quantifies the differences of the importance weights between the main classifier and the side classifiers. Based on the standard deviation, we define the *stability factor* with

$$\gamma_j^{t-1} = e^{1-\mathrm{std}(\theta_\mathrm{j})} \in (0, e]. \tag{10}$$

The stability factor will be multiplied with δ_j^{t-1}, to adjust the impact of the importance weights. Finally, the parameter regularization loss in MUC-MAS for learning the t task is formulated by

$$\mathcal{L}_{REG}^{PR} = \underbrace{\sum_{j}^{|\boldsymbol{\theta}|} \alpha_j^{t-1}(\theta_j - \tilde{\theta}_j)^2}_{\text{main classifier}} + \underbrace{\sum_{j}^{|\boldsymbol{\theta}|} \gamma_j^{t-1}\delta_j^{t-1}(\theta_j - \tilde{\theta}_j)^2}_{\text{side classifiers}}, \tag{11}$$

where $\tilde{\theta}_j$ is the corresponding parameter weight stored in the old network.

3.3 MUC with Activation Regularization

Activation regularization (AR), which aims to compare the activations between old and new networks, is driven by the idea of knowledge distillation [13]. In LwF [24], the activations refer to the probability predictions, which act as soft labels to constrain the updates of the network. Here, we demonstrate MUC-LwF by extending LwF to the MUC paradigm. First, we compute the AR loss based

on the main classifier as follows

$$\mathcal{L}_{AR}(F, M^{1:t-1}) = \frac{1}{N^t}\sum_{i=1}^{N^t} \mathrm{KD}\left(\log\left[\sigma\left(\frac{Q(x_i^t)}{ts}\right)\right], \sigma\left(\frac{\tilde{Q}(x_i^t)}{ts}\right)\right), \tag{12}$$

where KD is the function for computing the knowledge distillation term; σ is the Softmax function; the temperature scalar ts is normally fixed with 2. Taking as input a sample x_i^t into the network, $Q(x_i^t) = M^{1:t-1}(F(x_i^t))$ represents a concatenation vector output from the main classifiers corresponding to the previous $t-1$ tasks. Likewise, $\tilde{Q}(x_i^t) = \tilde{M}^{1:t-1}(\tilde{F}(x_i^t))$ is the vector derived from the old network. Accordingly, we further accumulate the AR loss for K side classifiers by

$$\mathcal{L}_{AR}(F, S_{1:K}^{1:t-1}) = \frac{1}{N^t}\frac{1}{K}\sum_{i=1}^{N^t}\sum_{k=1}^{K} \mathrm{KD}\left(\log\left[\sigma\left(\frac{Q_k(x_i^t)}{ts}\right)\right], \sigma\left(\frac{\tilde{Q}_k(x_i^t)}{ts}\right)\right), \tag{13}$$

where $Q_k(x_i^t) = S_k^{1:t-1}(F(x_i^t))$ is the concatenation vector of the k-th side classifier towards task 1 to task $t-1$. $\tilde{Q}_k(x_i^t) = \tilde{S}_k^{1:t-1}(\tilde{F}(x_i^t))$ is the corresponding vector extracted from the old model. Lastly, the total activation regularization loss in MUC-LwF when the network learns the t-th task becomes

$$\mathcal{L}_{REG}^{AR} = \underbrace{\mathcal{L}_{AR}(F, M^{1:t-1})}_{\text{main classifier}} + \underbrace{\mathcal{L}_{AR}(F, S_{1:K}^{1:t-1})}_{\text{side classifiers}}. \tag{14}$$

It is worthy mentioning that our MUC is a generic framework for many incremental learning methods, but is not limited to MAS and LwF. Particularly, in the experiment we empirically demonstrate its effectiveness under the rehearsal based scenario (Sec. 4.5).

4 Experiments

4.1 Datasets and Evaluation Metrics

We conducted the experiments on two widely-used benchmarks, `CIFAR-100` [19] and `Tiny ImageNet` [41]. `CIFAR-100` contains 100 classes, each of which has 500 training images and 100 test images of size 32×32. In `Tiny ImageNet`, there are 200 classes and each class contains 500 training images, 50 validation images and 50 test images of size 64×64. Since the class labels of test images in `Tiny ImageNet` are not available, the performance is generally evaluated on the validation set. Regarding the out-of-distribution dataset, we use the `SVHN` dataset [31] that contains only digits classes and is different from `CIFAR-100` and `Tiny ImageNet`. In the setting of class-incremental learning, we split the classes with $g = 10$ or 20 for `CIFAR-100`, and $g = 20$ or 40 for `Tiny ImageNet`, where g indicates the number of classes in each task. This setting results in $T = 10$ or 5 tasks for both datasets. The first evaluation metric we use is the standard top-1 classification accuracy. In addition, we report the forgetting ratio which was defined in [37]. The ratio belongs to $[-1, 0]$ and less negative ratios mean less forgetting. Normally, it is unnecessary to compare the performance of the first task, as it has no incremental learning yet.

4.2 Implementation Details

For a fair comparison with previous works, the network architecture we use is ResNet-32 [12]. We train the network from scratch for the first task and then update it continually for subsequent tasks. We downsample images of `Tiny ImageNet` to 32×32, so that they can use the same network as `CIFAR-100`. During each incremental session, we train the network with 200 epochs. The learning rate starts from 0.1 and decays with a factor of 10 after 120, 160 and 180 epochs. We optimize the network using SGD with a momentum of 0.9 and a weight decay of 5e-4. We use a batch size of 128 for all experiments. We use the same hyper-parameters to train `Stage II` but terminate the training after 80 epochs. Like iCaRL [35], we run the experiments several times and report the average performance. At test time, we use the predictions from the main classifier to compute the performance. We also test the predictions from the side classifiers, and they have the similar performance as the main classifier. We employ the '*single-head*' (*i.e.* task-agnostic) evaluation which is more practical than the '*multi-head*' (*i.e.* task-conditioned) evaluation [9].

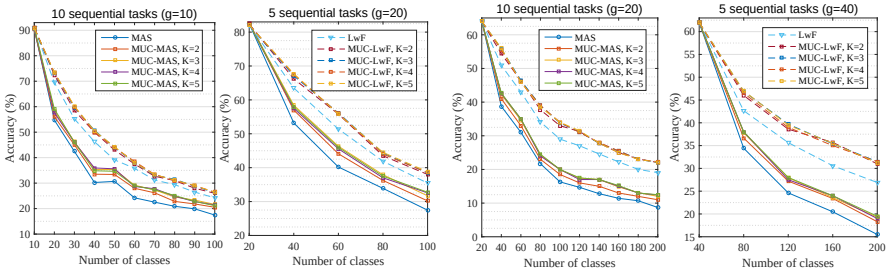
Notably, the parameter λ in Eq. 1 is significant for balancing the two loss terms during incremental learning. After learning more tasks, it is needed to increase the importance of the regularization loss, so as to avoid incessant forgetting on old tasks. Specifically, we set $\lambda = t-1$ for MUC-LwF, similar with the setting in BiC [44]. However, λ is fixed for MUC-MAS, because its regularization loss has already accumulated new and old importance weights, as suggested in MAS [1]. We set λ to be 0.01 for `CIFAR-100` and 0.005 for `Tiny ImageNet`.

4.3 Comparison and Discussion

We implement two baseline methods, including MAS [1] and LwF [24], because our MUC-MAS and MUC-LwF are build with their regularization. The same hyper-parameters are used to train our methods and the baselines for a fair comparison. In addition, we assess our methods with varying numbers (*i.e.* K) of side classifiers. Figure 3 presents the accuracy results on the two datasets.

Results of parameter regularization. Compared with the baseline MAS, the best accuracy from MUC-MAS achieves about **4%∼5%** gains on `CIFAR-100` and **3%∼4%** gains on `Tiny ImageNet`. The comparison demonstrates the benefit of exploiting side classifiers for parameter regularization. In terms of the number of side classifiers, the MUC-MAS variant with $K = 3$ has about **1.5%** improvement over that with $K = 2$. When K reaches to 4 or 5, the accuracy results are close with those when $K = 3$. This finding is consistent with prior works [47,36], where they used only two classifiers and achieved promising performance. To maintain the efficiency, we use the MUC-MAS variant with $K = 3$.

Results of activation regularization. It suggests in prior works [49,34] that LwF performs better than MAS in the context of class-incremental learning. Nevertheless, MUC-LwF surpasses LwF with a margin of **3%∼5%** gains on both datasets. Likewise, we also evaluate MUC-LwF with different numbers of side classifiers. By comparing those MUC-MAS variants, the one with $K = 3$ is



(a) Results on `CIFAR-100` (b) Results on `Tiny ImageNet`

Fig. 3: Results of classification accuracy on two datasets, each of which contains 10 or 5 sequential tasks. Our methods (MUC-MAS and MUC-LwF) outperform the corresponding baselines (MAS and LwF) across tasks and datasets.

Table 1: Results of forgetting ratio (less negative indicates less forgetting) on `CIFAR-100` (top table) and `Tiny ImageNet` (bottom table). The average forgetting ratio (Avg.) excludes the first task as it has no incremental learning. MUC methods exhibit less forgetting than the baselines.

`CIFAR-100`	Number of classes (10 tasks)											Number of classes (5 tasks)					
Method	10	20	30	40	50	60	70	80	90	100	Avg.	20	40	60	80	100	Avg.
MAS	-	-0.36	-0.47	-0.62	-0.61	-0.68	-0.69	-0.75	-0.75	-0.77	-0.55	-	-0.21	-0.45	-0.53	-0.57	-0.44
MUC-MAS	-	-0.31	-0.42	-0.55	-0.56	-0.61	-0.62	-0.71	-0.70	-0.72	**-0.50**	-	-0.17	-0.39	-0.48	-0.54	**-0.39**
LwF	-	-0.10	-0.35	-0.29	-0.34	-0.39	-0.44	-0.51	-0.56	-0.61	-0.33	-	-0.10	-0.26	-0.33	-0.41	-0.28
MUC-LwF	-	-0.08	-0.31	-0.24	-0.29	-0.32	-0.39	-0.45	-0.51	-0.55	**-0.29**	-	-0.07	-0.22	-0.30	-0.38	**-0.24**

`Tiny ImageNet`	Number of classes (10 tasks)											Number of classes (5 tasks)					
Method	20	40	60	80	100	120	140	160	180	200	Avg.	40	80	120	160	200	Avg.
MAS	-	-0.43	-0.53	-0.65	-0.72	-0.75	-0.79	-0.80	-0.82	-0.85	-0.71	-	-0.44	-0.58	-0.63	-0.70	-0.59
MUC-MAS	-	-0.40	-0.48	-0.60	-0.67	-0.71	-0.75	-0.76	-0.78	-0.81	**-0.66**	-	-0.41	-0.53	-0.58	-0.66	**-0.54**
LwF	-	-0.21	-0.31	-0.44	-0.48	-0.53	-0.62	-0.63	-0.65	-0.68	-0.51	-	-0.34	-0.41	-0.50	-0.52	-0.44
MUC-LwF	-	-0.17	-0.28	-0.40	-0.44	-0.49	-0.58	-0.60	-0.61	-0.65	**-0.47**	-	-0.28	-0.37	-0.45	-0.49	**-0.40**

slightly better than others. For consistency and generalization, we also use three side classifiers for MUC-LwF.

Results of forgetting ratio. We further report the forgetting ratio results in Table 1. It shows that our methods outperform the baselines with an average decline of **4%∼5%** forgetting ratios. The results support our motivation: *using more classifiers leads to less forgetting*.

Complexity analysis. Despite the fact that the side classifiers impose extra computational cost, however, the number of their parameters is a small fraction with respect to the number of all the parameters in the network. For the case when $K = 3$ and $g = 20$ on `CIFAR-100`, the final network consumes about 20,000 extra parameters due to adding the side classifiers, while they are only **4%** of the total parameters. It suggests that MUC is a practical and efficient method for incremental learning, without violating the memory limit much.

Table 2: Euclidean distances among the parameter vectors of three side classifiers. By using the classifier discrepancy loss, the parameters of the classifiers become dissimilar.

Side classifiers	w/o $\mathcal{L}_{CD}$	with $\mathcal{L}_{CD}$
S_1 v.s. S_2	0.057	5.284
S_1 v.s. S_3	0.059	5.110
S_2 v.s. S_3	0.054	5.662

Table 3: Accuracy results of one-stage and two-stage training on `CIFAR-100`. The one-stage training has higher accuracy at the beginning, while largely underperforms the two-stage training for subsequent tasks.

Method	Training	20	40	60	80	100
MUC-MAS	one-stage	**83.5**	**58.9**	44.8	35.2	28.8
MUC-MAS	two-stage	82.6	58.4	**46.3**	**37.9**	**31.7**
MUC-LwF	one-stage	**83.5**	**70.8**	58.0	47.6	38.4
MUC-LwF	two-stage	82.6	69.6	**59.4**	**49.5**	**41.6**

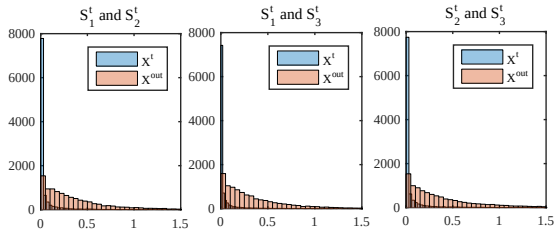


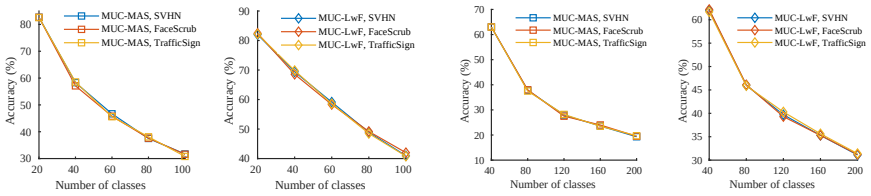
Fig. 4: Histogram statistics of prediction disagreements for X^t and X^{out}. X axis indicates the L1-norm discrepancy distance and Y axis counts the number of samples.

4.4 Component Analysis

Analysis of classifier discrepancy. This study is to show how the the classifier discrepancy loss $\mathcal{L}_{CD}$ in `Stage II` diversify the side classifiers. Specifically, we reshape the parameters (weights and bias) of each side classifier into a one-dimensional vector, and then compute the Euclidean distance between a pair of those parameter vectors. Table 2 reports the distances with or without using the classifier discrepancy loss, in terms of $K = 3$ on `CIFAR-100`. Notably, this loss succeeds in increasing the disagreement among the side classifiers in case that they learn nearly identical parameters.

Comparison between in-distribution and OOD samples. Recall that the objective of `Stage II` is to make the side classifiers produce consistent predictions for in-distribution samples but distinct predictions for OOD samples. We use the L1-norm distance in Eq. 3 to quantify the discrepancy among the predictions (Fig. 4). It can be seen that the disagreements for most in-distribution samples are close to 0, while the OOD samples has much larger disagreements. In this example, we show the results when $t = 1$ on `CIFAR-100`, while similar behavior is observed as well for subsequent tasks.

Evaluation of different OOD datasets. This experiment shows the performance when we choose OOD samples from different datasets. Apart from the SVHN dataset, we additionally use another two datasets including FaceScrub [32] and TrafficSign [40]. The results in Fig. 5 depict that our MUC is robust to different OOD datasets. We choose to use SVHN due to its popularity in the field.



(a) 5 sequential tasks on `CIFAR-100` (b) 5 sequential tasks on `Tiny ImageNet`

Fig. 5: Performance of MUC-MAS and MUC-LwF by using different OOD datasets. Our results are consistent across three OOD datasets.

Table 4: Effect of increasing the trade-off parameter λ for incrementally learning 5 tasks on `CIFAR-100`.

Method	Parameter	20	40	60	80	100
MAS	$\lambda = 0.01$	82.6	53.2	41.2	33.9	27.5
MAS	$\lambda = 0.02$	82.6	57.6	45.4	36.7	30.8
MUC-MAS	$\lambda = 0.01$	82.6	58.4	46.3	37.9	31.7
MUC-MAS	$\lambda = 0.02$	82.6	59.5	48.2	40.8	34.4

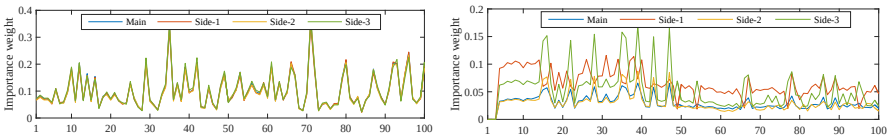
Table 5: Analyzing the stability factor in MUC-MAS. This comparison is performed on the five tasks of `CIFAR-100`. The fixed factor is 1.5.

MUC-MAS	20	40	60	80	100
w/o stability factor	82.6	57.6	45.4	36.7	30.8
with fixed factor	82.6	57.0	45.6	37.0	31.0
with stability factor	82.6	58.4	46.3	37.9	31.7

Two-stage *versus* one-stage. We aim to show the advantage of the two-stage fashion for training MUC. To this end, we also implement a one-stage training fashion, in which the main and side classifiers are trained simultaneously. As such, the total objective is composed of three terms: cross-entropy loss, classifier discrepancy loss and regularization loss. We report the comparison results in Table 3, where the two-stage training performs better than the one-stage training for a larger number of tasks. The main reason is that the classifier discrepancy loss has a negative effect on the regularization loss in the one-stage training. Hence, we decouple these two loss terms in two stages.

Effect of the trade-off parameter λ. We fix λ to be 0.01 for MAS and MUC-MAS, while it is encouraged to test the performance by increasing λ. In Table 4, we compare the results when $\lambda = 0.01$ and $\lambda = 0.02$. First, both MAS and MUC-MAS yield considerable gains due to a larger λ. Importantly, the results of MUC-MAS with $\lambda = 0.01$ are even better than those of MAS with $\lambda = 0.02$. The reason is that MUC-MAS learns complementary regularization terms derived from the side classifiers, rather than simply increasing λ for the regularization term from the main classifier. However, when $\lambda = 0.02$, we find an impractical trend that the accuracy of new tasks is lower than that of old tasks. In other words, the method tends to trade new tasks accuracy for higher accuracy of old tasks. To avoid this issue, we instead choose to use $\lambda = 0.01$.

Effect of stability factors. Regarding MUC-MAS, we discuss the results with or without using stability factors (Table 5). Using stability factors brings about 1% gains across the tasks. In addition, we consider using a fixed factor for all parameters and compare it with our parameter-adaptive stability factor. For



(a) Stable parameters (b) Unstable parameters

Fig. 6: Importance weights derived from the main classifier (Main) and three side classifiers (Side-1, Side-2 and Side-3). Based on the stability factors, we choose to show (a) 100 stable parameters and (b) 100 unstable parameters.

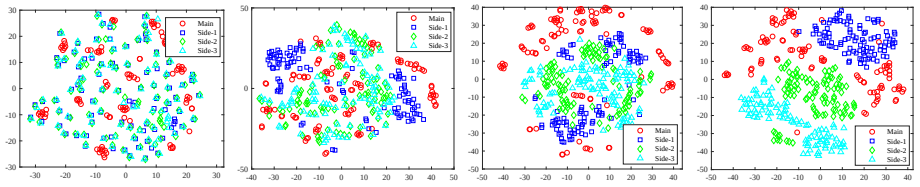
fairness, The fixed factor is set with 1.5 which is the average value of all stability factors. We see that the performance with fixed factors has no considerable gains. It suggests the advantage of our stability factor that adapts to each individual parameter. Although the performance gains are not significant, our stability factor provides a new degree to analyze the parameters in the network.

Comparison of importance weights. For MUC-MAS, we investigate important weights captured from the main classifier and side classifiers. Instead of choosing the parameters randomly or manually, we provide a robust selection based on the stability factors. To be specific, we rank all the parameters by their stability factors, and choose 100 stable parameters that have the largest factors and 100 unstable parameters being with the smallest factors. Figure 6 visually compares the importance weights from the main classifier and side classifiers. For 100 stable parameters, the importance weights from the side classifiers are almost the same as those from the main classifier. On the other hand, regarding 100 unstable parameters, each classifier produces a different importance weight. Our method allows to quantify the stability of the parameters and help to discover potentially stable parameters.

Visualization of soft labels. Regarding MUC-LwF, this experiment is to study the soft labels from the classifiers. Specifically, we pick up one class from task $t = 5$ and feed its data into four old models when $t = 1$ to $t = 4$. Then, we extract the soft labels from each old model and visualize the distributions with t-SNE [28] (Fig. 7). First, the distributions with three side classifiers are different from that with the main classifier. In addition, the distributions associated with three side classifiers tend to differ more largely from $t = 1$ to $t = 4$.

4.5 Learning with Exemplars

It is feasible to extend MUC to the scenario of storing some exemplars for old classes, even though it is not the core of our work. Following iCaRL [35], we store a fixed budget of 2000 exemplars. Instead of using the herding algorithm in iCaRL, we select an equal number of samples for new and old classes, and additionally run a balanced fine-tuning stage. As suggested in recent work [4,49], this simple fine-tuning stage achieves competitive performance with iCaRL. We adapt exemplars to the methods including MAS, LwF, MUC-MAS and MUC-LwF (Fig. 8). In the case of using exemplars, MUC-LwF* yields 3% gains against



(a) t=1 (b) t=2 (c) t=3 (d) t=4

Fig. 7: Visualization of soft labels extracted from the main classifier (M) and three side classifiers (Side-1, Side-2 and Side-3). Given the image samples from one class, it visually shows how their soft labels from each classifier change over a sequence of tasks (more details when zoomed in).

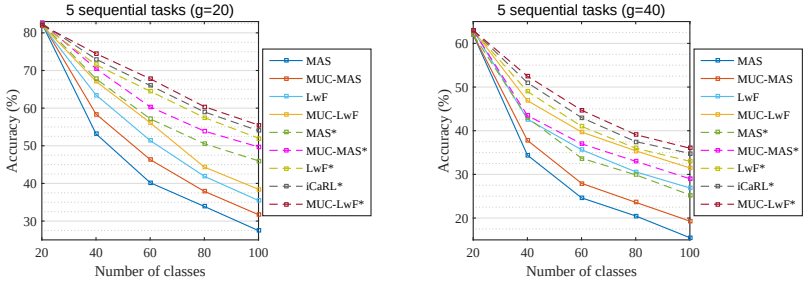


Fig. 8: Results of incrementally learning 5 tasks in the setting of using exemplars for (Left) `CIFAR-100` and (Right) `Tiny ImageNet`. The methods with '*' store a fixed budget of 2000 exemplars for old tasks; otherwise are exemplar-free.

LwF* and 1.5% gains against iCaRL*. Despite the fact that the performance gap becomes slight due to using exemplars, it will be a promising direction about how to fully leverage exemplars in the MUC paradigm.

5 Conclusion

We have proposed a generic multi-classifier incremental learning paradigm, based on which we further develop two instantiations to improve the effectiveness of parameter and activation regularization, respectively. Compared with the single-classifier methods, our MUC has achieved higher accuracy and less forgetting across tasks and datasets. Through additional component analysis, MUC demonstrated more insights which were not shown in the single-classifier paradigm. This work makes us realize that the classifiers play a crucial role in the scenario of incrementally learning tasks. In the future, it is promising to exploit MUC for other vision applications in the context of incremental learning, such as object detection and semantic segmentation.

Acknowledgements

This research was funded by Huawei as part of an HIRP Open project and by the FWO project "Structure from Semantics" (grant number G086617N).

References

1. Aljundi, R., Babiloni, F., Elhoseiny, M., Rohrbach, M., Tuytelaars, T.: Memory aware synapses: Learning what (not) to forget. In: ECCV. pp. 144–161 (2018)
2. Belouadah, E., Popescu, A.: IL2M: class incremental learning with dual memory. In: ICCV. pp. 583–592 (2019)
3. Carpenter, G.A., Grossberg, S.: Art 2: self-organization of stable category recognition codes for analog input patterns. Appl. Opt. **26**(23), 4919–4930 (1987)
4. Castro, F.M., Marín-Jiménez, M.J., Guil, N., Schmid, C., Alahari, K.: End-to-end incremental learning. In: ECCV. pp. 241–257 (2018)
5. Cermelli, F., Mancini, M., Bulò, S.R., Ricci, E., Caputo, B.: Modeling the background for incremental learning in semantic segmentation. CoRR **abs/2002.00718** (2020)
6. Chaudhry, A., Dokania, P.K., Ajanthan, T., Torr, P.H.S.: Riemannian walk for incremental learning: Understanding forgetting and intransigence. In: ECCV. pp. 556–572 (2018)
7. Dhar, P., Singh, R.V., Peng, K., Wu, Z., Chellappa, R.: Learning without memorizing. In: CVPR. pp. 5138–5146 (2019)
8. Duan, L., Tsang, I.W., Xu, D., Chua, T.: Domain adaptation from multiple sources via auxiliary classifiers. In: ICML. pp. 289–296 (2009)
9. Farquhar, S., Gal, Y.: Towards robust evaluations of continual learning. CoRR **abs/1805.09733** (2018)
10. Hao, Y., Fu, Y., Jiang, Y., Tian, Q.: An end-to-end architecture for class-incremental object detection with knowledge distillation. In: ICME. pp. 1–6 (2019)
11. He, C., Wang, R., Shan, S., Chen, X.: Exemplar-supported generative reproduction for class incremental learning. In: BMVC. p. 98 (2018)
12. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. pp. 770–778 (2016)
13. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. In: NIPS Deep Learning and Representation Learning Workshop (2015)
14. Hou, S., Pan, X., Loy, C.C., Wang, Z., Lin, D.: Lifelong learning via progressive distillation and retrospection. In: ECCV. pp. 452–467 (2018)
15. Hou, S., Pan, X., Loy, C.C., Wang, Z., Lin, D.: Learning a unified classifier incrementally via rebalancing. In: CVPR. pp. 831–839 (2019)
16. Hsu, Y., Liu, Y., Kira, Z.: Re-evaluating continual learning scenarios: A categorization and case for strong baselines. CoRR **abs/1810.12488** (2018)
17. Kemker, R., Kanan, C.: Fearnet: Brain-inspired model for incremental learning. In: ICLR (2018)
18. Kirkpatrick, J., Pascanu, R., Rabinowitz, N.C., Veness, J., Desjardins, G., Rusu, A.A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., Hadsell, R.: Overcoming catastrophic forgetting in neural networks. PNAS **114 13**, 3521–3526 (2016)
19. Krizhevsky, A.: Learning multiple layers of features from tiny images. Master's thesis, Department of Computer Science, University of Toronto. (2009)
20. Lange, M.D., Aljundi, R., Masana, M., Parisot, S., Jia, X., Leonardis, A., Slabaugh, G.G., Tuytelaars, T.: Continual learning: A comparative study on how to defy forgetting in classification tasks. CoRR **abs/1909.08383** (2019)
21. Lee, C., Xie, S., Gallagher, P., Zhang, Z., Tu, Z.: Deeply-supervised nets. In: AISTATS (2015)

22. Lee, K., Lee, K., Shin, J., Lee, H.: Overcoming catastrophic forgetting with unlabeled data in the wild. In: ICCV. pp. 312–321 (2019)
23. Lee, S., Kim, J., Jun, J., Ha, J., Zhang, B.: Overcoming catastrophic forgetting by incremental moment matching. In: NIPS. pp. 4652–4662 (2017)
24. Li, Z., Hoiem, D.: Learning without forgetting. In: ECCV. pp. 614–629 (2016)
25. Liu, X., Masana, M., Herranz, L., van de Weijer, J., López, A.M., Bagdanov, A.D.: Rotate your networks: Better weight consolidation and less catastrophic forgetting. In: ICPR. pp. 2262–2268 (2018)
26. Lopez-Paz, D., Ranzato, M.: Gradient episodic memory for continual learning. In: NIPS. pp. 6467–6476 (2017)
27. Luo, P., Zhuang, F., Xiong, H., Xiong, Y., He, Q.: Transfer learning from multiple source domains via consensus regularization. In: CIKM. pp. 103–112 (2008)
28. van der Maaten, L., Hinton, G.: Visualizing high-dimensional data using t-sne. JMLR **9**, 2579–2605 (2008)
29. McCloskey, M., Cohen, N.J.: Catastrophic interference in connectionist networks: The sequential learning problem. Psychology of Learning and Motivation, vol. 24, pp. 109 – 165. Academic Press (1989)
30. Michieli, U., Zanuttigh, P.: Incremental learning techniques for semantic segmentation. In: ICCV, Workshop on TASK-CV (2019)
31. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y.: Reading digits in natural images with unsupervised feature learning. In: NIPS Workshop on Deep Learning and Unsupervised Feature Learning (2011)
32. Ng, H., Winkler, S.: A data-driven approach to cleaning large face datasets. In: International Conference on Image Processing. pp. 343–347 (2014)
33. Parisi, G.I., Kemker, R., Part, J.L., Kanan, C., Wermter, S.: Continual lifelong learning with neural networks: A review. Neural Networks **113**, 54–71 (2019)
34. Rajasegaran, J., Hayat, M., Khan, S., Khan, F.S., Shao, L.: Random path selection for incremental learning. Advances in Neural Information Processing Systems (2019)
35. Rebuffi, S., Kolesnikov, A., Sperl, G., Lampert, C.H.: icarl: Incremental classifier and representation learning. In: CVPR. pp. 5533–5542 (2017)
36. Saito, K., Watanabe, K., Ushiku, Y., Harada, T.: Maximum classifier discrepancy for unsupervised domain adaptation. In: CVPR. pp. 3723–3732 (2018)
37. Serrà, J., Suris, D., Miron, M., Karatzoglou, A.: Overcoming catastrophic forgetting with hard attention to the task. In: ICML. pp. 4555–4564 (2018)
38. Shin, H., Lee, J.K., Kim, J., Kim, J.: Continual learning with deep generative replay. In: NIPS. pp. 2990–2999 (2017)
39. Shmelkov, K., Schmid, C., Alahari, K.: Incremental learning of object detectors without catastrophic forgetting. In: ICCV. pp. 3420–3429 (2017)
40. Stallkamp, J., Schlipsing, M., Salmen, J., Igel, C.: The German Traffic Sign Recognition Benchmark: A multi-class classification competition. In: IEEE International Joint Conference on Neural Networks. pp. 1453–1460 (2011)
41. Stanford: Tiny imagenet challenge, cs231n course. https://tiny-imagenet.herokuapp.com/
42. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: CVPR. pp. 1–9 (2015)
43. Triki, A.R., Aljundi, R., Blaschko, M.B., Tuytelaars, T.: Encoder based lifelong learning. In: ICCV. pp. 1329–1337 (2017)
44. Wu, Y., Chen, Y., Wang, L., Ye, Y., Liu, Z., Guo, Y., Fu, Y.: Large scale incremental learning. In: CVPR. pp. 374–382 (2019)

45. Xiang, Y., Fu, Y., Ji, P., Huang, H.: Incremental learning using conditional adversarial networks. In: ICCV. pp. 6618–6627 (2019)
46. Xie, S., Tu, Z.: Holistically-nested edge detection. In: ICCV. pp. 1395–1403 (2015)
47. Yu, Q., Aizawa, K.: Unsupervised out-of-distribution detection by maximum classifier discrepancy. In: ICCV. pp. 9517–9525 (2019)
48. Zenke, F., Poole, B., Ganguli, S.: Continual learning through synaptic intelligence. In: ICML. pp. 3987–3995 (2017)
49. Zhang, J., Zhang, J., Ghosh, S., Li, D., Tasci, S., Heck, L.P., Zhang, H., Kuo, C.J.: Class-incremental learning via deep model consolidation. CoRR **abs/1903.07864** (2019)
50. Zhou, P., Mai, L., Zhang, J., Xu, N., Wu, Z., Davis, L.S.: M2KD: multi-model and multi-level knowledge distillation for incremental learning. CoRR **abs/1904.01769** (2019)