Forward Compatible Few-Shot Class-Incremental Learning

Da-Wei Zhou[1], Fu-Yun Wang[1], Han-Jia Ye[1][†], Liang Ma[2], Shiliang Pu[2], De-Chuan Zhan[1]

[1] State Key Laboratory for Novel Software Technology, Nanjing University [2] Hikvision Research Institute

{zhoudw, yehj, zhandc}@lamda.nju.edu.cn, wangfuyun@smail.nju.edu.cn, {maliang6, pushiliang.hri}@hikvision.com

Abstract

Novel classes frequently arise in our dynamically changing world, e.g., new users in the authentication system, and a machine learning model should recognize new classes without forgetting old ones. This scenario becomes more challenging when new class instances are insufficient, which is called few-shot class-incremental learning (FSCIL). Current methods handle incremental learning retrospectively by making the updated model similar to the old one. By contrast, we suggest learning prospectively to prepare for future updates, and propose ForwArd Compatible Training (FACT) for FSCIL. Forward compatibility requires future new classes to be easily incorporated into the current model based on the current stage data, and we seek to realize it by reserving embedding space for future new classes. In detail, we assign virtual prototypes to squeeze the embedding of known classes and reserve for new ones. Besides, we forecast possible new classes and prepare for the updating process. The virtual prototypes allow the model to accept possible updates in the future, which act as proxies scattered among embedding space to build a stronger classifier during inference. FACT efficiently incorporates new classes with forward compatibility and meanwhile resists forgetting of old ones. Extensive experiments validate FACT's state-of-the-art performance. Code is available at: https://github.com/zhoudw-zdw/CVPR22-Fact

1. Introduction

Recent years have witnessed the significant breakthroughs of deep neuron networks in many vision tasks [14, 19, 36, 40, 54]. However, data often come in stream format [18] with emerging new classes [12, 63, 65] in real-world applications, *e.g.*, new types of products in e-commerce. It requires a model to incorporate new class knowledge incrementally, which is called Class-Incremental Learning (CIL). When updating the model with new classes, a fatal problem occurs, namely catastrophic forgetting [16] — the discriminability of
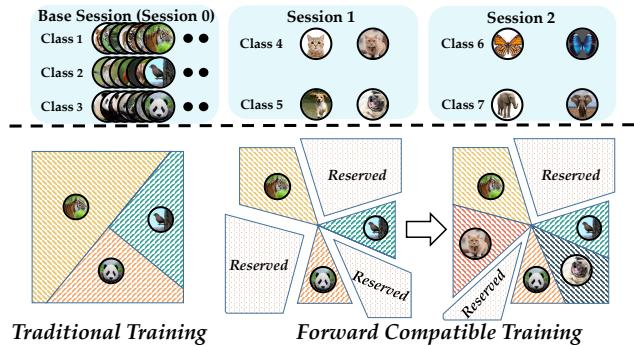
Figure 1. Top: the setting of FSCIL. We need to maintain a classifier covering all classes, where sessions with non-overlapping classes arrive sequentially. Ample training instances are available in the base session, while only few-shot instances in incremental sessions. The model should incorporate new classes without forgetting old ones. Bottom: forward compatible training scheme. Different from traditional training paradigm, we reserve the embedding space for new classes in the base session for future possible extensions.

old classes drastically declines. How to design effective CIL algorithms to overcome catastrophic forgetting has attracted much interest in the computer vision field [22, 25, 48, 50, 60].

Current CIL methods address the scenario where new classes are available with *sufficient* instances. However, the data collection and labeling cost can be relatively high in many applications. Take a rare bird classification model for an example. We can only use few-shot images to train the incremental model since they are hard to collect. This task is called Few-Shot Class-Incremental Learning (FSCIL), which is shown at the top of Figure 1. A model needs to sequentially incorporate new classes with *limited* instances without harming the discriminability of old classes. Apart from the forgetting problem, it also triggers overfitting on few-shot instances. As a result, some algorithms [58, 67] are proposed to solve FSCIL from the few-shot learning perspective, aiming to alleviate overfitting in model updating.

In FSCIL, a sequence of models should work together with harmony, *i.e.*, the updated model should maintain the discriminability of old classes. Such a learning process is similar to software development. The newer version software should accept the data that worked under the previous version, which is referred to as 'backward compatibil-

---

[†]Correspondence to: Han-Jia Ye (yehj@lamda.nju.edu.cn)

ity' [29,43]. It measures the capability of different systems to work together without adaptation. From this perspective, the ability to overcome forgetting represents model's backward compatibility — if an updated model is good at classifying former classes, it is compatible with the old model and does not suffer forgetting. Consequently, CIL methods seek to increase backward compatibility by maintaining discriminability of old classes, and FSCIL methods achieve this by fixing embedding module and incorporating new classes.

Current methods concentrate on backward compatibility, which shifts the burden of overcoming forgetting to the later model. However, if the former model works poorly, the latter model would degrade consequently. It is impossible to maintain backward compatibility with limited instances in incremental stages. Take software development for an example. If an early version is poorly designed, the later version needs to work hard putting patches to maintain backward compatibility. By contrast, a better solution is to consider future extensions in the early version and reserve the interface in advance. Consequently, another compatibility, namely *forward compatibility* is more proper for FSCIL, which prepares the model for possible future updates.

The model with forward compatibility should be *growable and provident*. On the one hand, growable means the model is aware of the incoming classes in the future and makes room for their embedding space. Hence, the model does not need to squeeze the space of former classes to make room for new ones when updating. Provident indicates the ability to forecast possible future classes. The model should anticipate the future and develop methods to minimize the effects of shocks and stresses of future events. Benefits from forward compatibility, the embedding space of old classes will be more compact, and new classes can be easily matched to the reserved space, as shown in Figure 1.

In this paper, we propose ForwArd Compatible Training (FACT) for FSCIL to prepare the model for future classes. To make the model growable, we pre-assign multiple virtual prototypes in the embedding space, pretending they are the reserved space. By optimizing these virtual prototypes, we push the instances from the same class to be closer and reserve more spaces for incoming new classes. Moreover, we also generate virtual instances via instance mixture to make the model provident. The virtual instances enable us to reserve the embedding space with explicit supervision. These reserved virtual prototypes can be seen as informative basis vectors during inference, with which we can build a strong classification model incrementally. Vast experiments on benchmark datasets under various settings are conducted, validating the effectiveness of FACT.

2. Related Work

Few-Shot Learning (FSL): aims to fit unseen classes with insufficient training instances [9,47]. Few-shot learning algorithms can be roughly divided into two groups: optimization-based methods and metric-based methods. Optimization-based methods try to enable fast model adaptation with few-shot data [3, 4, 15, 30]. Metric-based algorithms utilize a pretrained backbone for feature extraction, and employ proper distance metrics between support and query instances [26,37,39,45,52,53,57].

Class-Incremental Learning (CIL): aims to learn from a sequence of new classes without forgetting old ones, which is now widely discussed in various computer vision tasks [13, 51,62,64]. Current CIL algorithms can be roughly divided into three groups. The first group estimates the importance of each parameter and prevents important ones from being changed [2,22,56]. The second group utilizes knowledge distillation to maintain the model's discriminability [20,25, 33]. Other methods rehearsal former instances to overcome forgetting [6,48,60,66]. Pernici *et al.* [32] pre-allocates classifiers for future classes, which needs an extra memory for feature tuning and is unsuitable for FSCIL.

Few-Shot Class-Incremental Learning: is recently proposed to address the few-shot inputs in the incremental learning scenario [1,11,24,61]. TOPIC [41] uses the neural gas structure to preserve the topology of features between old and new classes to resist forgetting. Semantic-aware knowledge distillation [10] treats the word embedding as auxiliary information, and builds knowledge distillation terms to resist forgetting. To resist overfitting on few-shot inputs, FSLL [27] selects a few parameters to be updated at every incremental session. CEC [58] is the current state-of-the-art method, which utilizes an extra graph model to propagate context information between classifiers for adaptation.

Backward Compatible Learning: The concept 'compatibility' is a design characteristic considered in software engineering [17,29,43,49]. Forward compatibility allows a system to accept input intended for a later version of itself, and backward compatibility allows for interoperability with an older legacy system. They are introduced to the machine learning field in [5,38]. Recent work focuses on improving model's backward compatibility [7,28,35], while we are the first to address the model's forward compatibility in FSCIL.

3. From Old Classes to New Classes

In this section, we first describe the setting of FSCIL and then introduce the baseline methods and their limitations.

3.1. Few-Shot Class-Incremental Learning

Base Session: In FSCIL, a model first receives the training set $\mathcal{D}^0 = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n_0}$ with *sufficient* instances and evaluated with the testing set $\mathcal{D}_t^0 = \{(\mathbf{x}_j, y_j)\}_{j=1}^{m_0}$. $\mathcal{D}^0$ is called the base task.* $\mathbf{x}_i \in \mathbb{R}^D$ is a training instance of class $y_i \in Y_0$. Y_0 is the label space of base task. An algorithm fits

---

*We interchangeably use 'task' and 'session' in this paper.

a model $f(\mathbf{x})$ to minimize the empirical risk over testing set:

$$\sum_{(\mathbf{x}_j, y_j) \in \mathcal{D}_t^0} \ell\left(f\left(\mathbf{x}_j\right), y_j\right), \tag{1}$$

where $\ell(\cdot, \cdot)$ measures the discrepancy between prediction and ground-truth label. The model can be decomposed into embedding and linear classifier: $f(\mathbf{x}) = W^\top \phi(\mathbf{x})$, where $\phi(\cdot): \mathbb{R}^D \to \mathbb{R}^d$ and $W \in \mathbb{R}^{d \times |Y_0|}$. We denote the classifier for class k as $\boldsymbol{w}_k$: $W = [\boldsymbol{w}_1, \cdots, \boldsymbol{w}_{|Y_0|}]$.

Incremental Sessions: New classes often arrive incrementally with *insufficient* instances in practical applications, *i.e.*, a sequence of datasets $\{\mathcal{D}^1, \cdots, \mathcal{D}^B\}$ will emerge sequentially. $\mathcal{D}^b = \{(\mathbf{x}_i, y_i)\}_{i=1}^{NK}$, where $y_i \in Y_b$, and Y_b is the label space of task b. $Y_b \cap Y_{b'} = \varnothing$ for $b \neq b'$. We only have access to $\mathcal{D}^b$ when training task b. The limited instances in each dataset can be organized as N-way K-shot format, *i.e.*, there are N classes in the dataset, and each class has K examples. Facing a new dataset $\mathcal{D}^b$, a model should learn new classes and maintain performance over old classes, *i.e.*, minimize the empirical risk over all testing datasets:

$$\sum_{(\mathbf{x}_j, y_j) \in \mathcal{D}_t^0 \cup \cdots \mathcal{D}_t^b} \ell\left(f\left(\mathbf{x}_j\right), y_j\right). \tag{2}$$

3.2. Backward Compatible Training for FSCIL

Knowledge Distillation: To make the updated model still capable of classifying the old class instances, a common approach in CIL combines cross-entropy loss and knowledge distillation loss [20]. It builds a mapping between the former and the current model to maintain backward compatibility:

$$\begin{aligned}
\mathcal{L}(\mathbf{x}, y) &= (1 - \lambda)\mathcal{L}_{\text{CE}}(\mathbf{x}, y) + \lambda\mathcal{L}_{\text{KD}}(\mathbf{x}) \\
\mathcal{L}_{\text{KD}}(\mathbf{x}) &= \sum_{k=1}^{|\mathcal{Y}_{b-1}|} -\mathcal{S}_k(\bar{W}^\top \bar{\phi}(\mathbf{x})) \log \mathcal{S}_k(W^\top \phi(\mathbf{x})),
\end{aligned} \tag{3}$$

where $\mathcal{Y}_{b-1}$ denotes the set of old classes, $\mathcal{Y}_{b-1} = Y_0 \cup \cdots Y_{b-1}$, and $\mathcal{S}_k(\cdot)$ denotes the k-th class probability after softmax operation. $\bar{W}$ and $\bar{\phi}$ correspond to frozen classifier and embedding before learning $\mathcal{D}^b$. Aligning the output of old model and current model helps to maintain model's discriminability and encourages backward compatibility.

Prototypical Network: Knowledge distillation suffers from overfitting with few-shot inputs and works poorly. As a result, FSL algorithms are modified to conquer overfitting and encourage backward compatibility. ProtoNet [37] trains the model on base classes with cross-entropy loss. The embedding $\phi(\cdot)$ is then fixed and utilized to extract average embedding (*i.e.*, prototype) of each class:

$$\boldsymbol{p}_i = \frac{1}{K} \sum_{j=1}^{|\mathcal{D}^b|} \mathbb{I}(y_j = i)\phi(\mathbf{x}_j). \tag{4}$$

$\mathbb{I}(\cdot)$ is the indicator function. The averaged embedding represents the common pattern of each class, which is used as

the classifier, *i.e.*, $\boldsymbol{w}_i = \boldsymbol{p}_i$. The embedding is fixed throughout the incremental stages, and backward compatibility is maintained between old and new models.

Ignorance of Forward Compatibility: Eq. 3 and Eq. 4 focus on different aspects of backward compatibility. The former maintains discriminability by aligning the old and new models into the same scale, and the latter fixes the embedding to prevent it from shifting away. They treat incremental learning from a *retrospective* standpoint — we need to keep the model unchanged during updating. However, none of them consider the *forward compatibility*, *i.e.*, when training the model at the current session, it should prepare for the future possible updates. It is profitable to treat incremental learning from a *prospective* view — look ahead to prepare for possible updates. In other words, the training process should be '*future-proof*'. If we reserve the embedding space for possible new classes and anticipate their possible pattern, the adaptation cost will be released in the future.

4. Forward Compatible Training for FSCIL

Motivated by the potential of forward compatibility, we seek to enhance such characteristics in FSCIL. The training target is to look ahead and prepare for incoming classes in the base session, and we realize it from two aspects. On the one hand, to make the model growable, we seek to optimize the posterior probability into bimodal distribution — we assign each instance to an extra class except for the ground-truth class. The extra label stands for the reserved class space for new classes, which is optimized explicitly during training. On the other hand, to make the model provident, we try to foresee the possible distribution of new classes via instance mixture. The mimicked new instances help transform static training into incremental training. Forward compatibility is kept during this training paradigm.

We first introduce how to reserve embedding space, and then discuss how to use them for inference.

4.1. Pretrain with Virtual Prototypes

4.1.1 Allocating Virtual Prototypes

In the base session, a common pretrain method is to optimize the empirical loss over the training set, which takes no consideration of future incremental learning process and overspreads the embedding space. There are only $|Y_0|$ classes in $\mathcal{D}^0$, but the model needs to handle $|Y_0| + NB$ classes in the final session. As a result, the traditional training paradigm needs to squeeze the embedding of old classes to make room for new ones, which takes many rounds and is unsuitable for FSCIL. Our classification is based on the similarity between instance embedding and the class prototype (we replace the classifier weight with the class prototype), *i.e.*, $p(y|\mathbf{x}) \propto \text{sim}\langle \boldsymbol{w}_y, \phi(\mathbf{x})\rangle$, the more similar they are, the
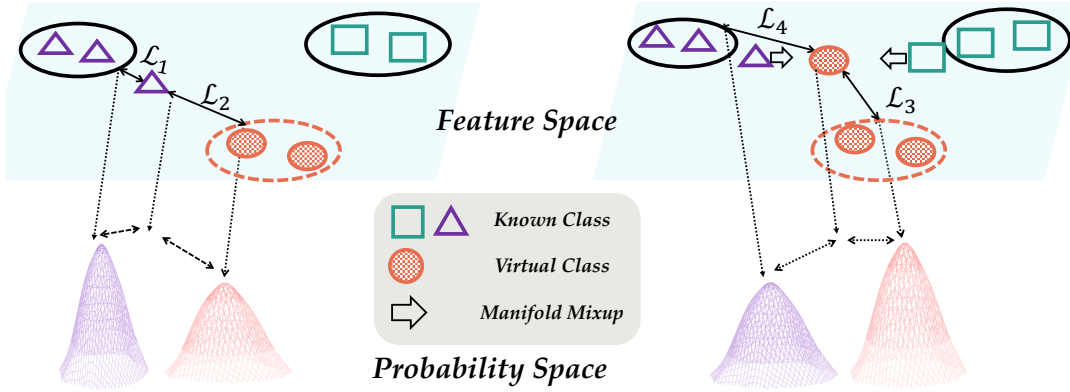
Figure 2. Illustration of FACT. Left: making the model growable. Apart from the cross-entropy loss ($\mathcal{L}_1$), the model also assigns an instance to a virtual class ($\mathcal{L}_2$), which reserves the space for new classes. Right: making the model provident. We first forecast virtual instances by manifold mixup (shown with arrow) and then conduct a symmetric reserving process by assigning it to the virtual class and known class. The training target is a bimodal distribution, which forces the instance to be assigned to different clusters and reserve embedding space.

more probably $\mathbf{x}$ belongs to class y. We use the cosine classifier as the similarity measure[†] : $f(\mathbf{x}) = (\frac{W}{\|W\|_2})^\top (\frac{\phi(\mathbf{x})}{\|\phi(\mathbf{x})\|_2})$. To enhance the forward compatibility, we pre-assign several virtual prototypes P_v in the embedding space, and treat them as 'virtual classes'. $P_v = [\boldsymbol{p}_1, \cdots, \boldsymbol{p}_V] \in \mathbb{R}^{d \times V}$, where V is the number of virtual classes. Denote the output of current model as $f_v(\mathbf{x}) = [W, P_v]^\top \phi(\mathbf{x})$, we seek to reserve the embedding space for these V classes:

$$\mathcal{L}_v(\mathbf{x}, y) = \underbrace{\ell(f_v(\mathbf{x}), y)}_{\mathcal{L}_1} + \gamma \underbrace{\ell(\text{Mask}(f_v(\mathbf{x}), y), \hat{y})}_{\mathcal{L}_2} \quad (5)$$
$$\text{Mask}(f_v(\mathbf{x}), y) = f_v(\mathbf{x}) \otimes (\mathbf{1} - \text{OneHot}(y)),$$

where $\hat{y} = \text{argmax}_v \, \boldsymbol{p}_v^\top \phi(\mathbf{x}) + |Y_0|$ is the virtual class with maximum logit, acting as the pseudo label. $\otimes$ is Hadamard product (element-wise multiplication), $\mathbf{1}$ is an all-ones vector. The first item in Eq. 5 corresponds to the vanilla training loss, which matches the output to its ground-truth label. The second term first masks out the ground-truth logit with function $\text{Mask}(\cdot, \cdot)$, and then matches the rest part to the pseudo label $\hat{y}$. Since $\hat{y}$ is the virtual class with maximum logit, Eq. 5 reserves the embedding space for $\hat{y}$ explicitly.

Effect of Virtual Prototypes: Eq. 5 forces the output of $f_v(\mathbf{x})$ to be bimodal, which is shown in left part of Figure 2. The first item forces an instance to be nearest to its ground-truth cluster, and the second term matches it to the nearest virtual cluster. By optimizing Eq. 5, all non-target class prototypes will be pushed away from the reserved virtual prototype, and the decision boundary will be pushed in the same direction. As a result, the embedding of other classes will be more compact, and the embedding spaces for virtual classes will be reserved. Hence, the model becomes growable and forward compatibility is enhanced. We set V to the number of new classes, *i.e.*, $V = NB$ as default.

4.1.2 Forecasting Virtual Instances

To make the model provident, we try to equip the model with 'future-proof' ability — if it has seen the novel patterns in the pretrain stage, the reserved space would be more suitable for incoming new classes. To this aim, we attempt to generate new classes by instance mixture and reserve the embedding space for these generated instances.

Motivated by the intuition that interpolations between two different clusters are often regions of low-confidence predictions [44], we seek to fuse two instances by manifold mixup [44] and treat the fused instance as a virtual new class. We decouple the embedding into two parts at the middle layer: $\phi(\mathbf{x}) = g(h(\mathbf{x}))$. For any instance couple from different classes in the mini-batch, say $(\mathbf{x}_i, \mathbf{x}_j), y_i \neq y_j$, we fuse the embedding of this pair as a virtual instance:

$$\boldsymbol{z} = g[\lambda h(\mathbf{x}_i) + (1 - \lambda)h(\mathbf{x}_j)], \quad (6)$$

where $\lambda \in [0, 1]$ is sampled from Beta distribution. Denote the current output for $\boldsymbol{z}$ is $f_v(\boldsymbol{z}) = [W, P_v]^\top \boldsymbol{z}$. Similar to the virtual loss in Eq. 5, we can build a *symmetric* loss for the virtual instance $\boldsymbol{z}$ to reserve embedding space:

$$\mathcal{L}_f(\boldsymbol{z}) = \underbrace{\ell(f_v(\boldsymbol{z}), \hat{y})}_{\mathcal{L}_3} + \gamma \underbrace{\ell(\text{Mask}(f_v(\boldsymbol{z}), \hat{y}), \hat{\hat{y}})}_{\mathcal{L}_4}, \quad (7)$$

where $\hat{y}$ is the same as in Eq. 5, which is the pseudo label among virtual classes, and $\hat{\hat{y}} = \text{argmax}_k \, \boldsymbol{w}_k^\top \boldsymbol{z}$ is the pseudo label among current known classes. Note that the trade-off parameter γ in Eq. 7 is the *same* as in Eq. 5.

Effect of Virtual Instances: Eq. 7 is the symmetric form of Eq. 5. Generating virtual classes forecasts the possible distribution of incoming new classes, which is shown in the right part of Figure 2. The first item pushes the mixed instance $\boldsymbol{z}$ towards a virtual prototype and away from other classes, reserving the space for the virtual class. Besides, the second term pushes the mixed instance towards the nearest known class, which trades-off between known and virtual

classes to prevent known classes from being over-squeezed. Hence, the model becomes provident by mimicking future instances, and forward compatibility is enhanced.

Why Virtual Loss Enhance Compatibility: The final loss is combined of Eq. 5 and Eq. 7, *i.e.*, $\mathcal{L} = \mathcal{L}_v + \mathcal{L}_f$, which has four loss terms. We analyze the gradient towards ϕ and W to find out the intrinsic intuition hidden in virtual prototypes. Denote the probability of class y after softmax operator as a_y, the first term in Eq. 5 corresponds to $-\log a_y$, whose negative gradient w.r.t. $\phi(\mathbf{x})$ is: $-\nabla_{\phi(\mathbf{x})}\mathcal{L}_1 = \boldsymbol{w}_y - \sum_{i=1}^{|Y_0|+V} a_i \boldsymbol{w}_i.^{\ddagger}$ It pushes the embedding towards the direction of ground-truth class center and away from other classes, including virtual classes. Similarly, we obtain the gradient $-\nabla_{\phi(\mathbf{x})}\mathcal{L}_2 = \boldsymbol{w}_{\hat{y}} - \sum_{i=1}^{|Y_0|+V} a_i \boldsymbol{w}_i$. Note that the logit a_y on ground-truth class is masked-out when optimizing $\mathcal{L}_2$, and thus we can push the embedding of $\phi(\mathbf{x})$ towards its nearest virtual prototype without hurting the classification performance. These conclusions are consistent with the bimodal distribution in Figure 2. We conduct the same analysis for $\mathcal{L}_3$ and $\mathcal{L}_4$. Assume $g(\mathbf{x})$ as an identity function, *i.e.*, $\phi(\mathbf{x}) = h(\mathbf{x})$, the gradient yields: $-\nabla_{\phi(\mathbf{x}_i)}\mathcal{L}_3 = \lambda \left(\boldsymbol{w}_{\hat{y}} - \sum_{k=1}^{|Y_0|+V} a_k \boldsymbol{w}_k \right)$, $-\nabla_{\phi(\mathbf{x}_j)}\mathcal{L}_3 = (1 - \lambda) \left(\boldsymbol{w}_{\hat{y}} - \sum_{k=1}^{|Y_0|+V} a_k \boldsymbol{w}_k \right)$. It is similar to the effect of $\mathcal{L}_2$, which pushes all other non-target class prototypes away from the mixed components. As a result, the embeddings of known classes are optimized to be more compact, facilitating forward compatibility.

Pseudo Code: We give the pseudo code of FACT in the supplementary. In each mini-batch, we first calculate the virtual loss as Eq. 5. Afterward, we shuffle the dataset and conduct manifold mixup to calculate the forecasting loss in Eq. 7. Note that we do not combine all possible pairs in the dataset (costs $\mathcal{O}(n*n)$), but instead, we only mix instances from different classes with the same index (costs $\mathcal{O}(n)$).

4.2. Incremental Inference with Virtual Prototypes

We have elaborated on the insight of virtual prototypes, and the left question is how to use them during inference. The main idea is to treat them as bases encoded in the embedding space, and consider the possible influence of these bases on the prediction results. Each time an incremental dataset $\mathcal{D}^b$ arrives, we extract the prototype of these new classes as Eq. 4, and expand our classifiers: $W = [W; \boldsymbol{w}_i, i \in Y_b]$. We treat class prototypes as the representation of each class. Based on the law of total probability, we have:

$$p\left(y_i|\phi(\mathbf{x})\right) = p(\boldsymbol{w}_i|\phi(\mathbf{x}))$$
$$= \sum_{\boldsymbol{p}_v \in P_v} p(\boldsymbol{w}_i|\boldsymbol{p}_v, \phi(\mathbf{x}))p(\boldsymbol{p}_v|\phi(\mathbf{x})), \quad (8)$$

---

‡We treat $[W, P_v]$ as a uniform classifier and do not differentiate $\boldsymbol{w}$ and $\boldsymbol{p}$ in the analysis for ease of discussion.

where $p(\boldsymbol{p}_v|\phi(\mathbf{x})) = \frac{\exp(\boldsymbol{p}_v^\top \phi(\mathbf{x}))}{\sum_{\boldsymbol{p}_v \in P_v} \exp(\boldsymbol{p}_v^\top \phi(\mathbf{x}))}$. Eq. 8 implies that we can consider the possible influence of all informative virtual prototypes to get the final prediction. For embedding $\phi(\mathbf{x})$ from class y_i with pseudo label $\hat{y} = v$, it shall follow a bimodal distribution between class $\boldsymbol{w}_i$ and $\boldsymbol{p}_v$, which is consistent with the training target. Hence, we assume $p(\phi(\mathbf{x})|\boldsymbol{w}_i, \boldsymbol{p}_v) = \eta \mathcal{N}(\phi(\mathbf{x})|\boldsymbol{w}_i, \Sigma) + (1 - \eta)\mathcal{N}(\phi(\mathbf{x})|\boldsymbol{p}_v, \Sigma)$. $\phi(\mathbf{x})$ follows a Gaussian mixture distribution, which is a linear superposition of $\mathcal{N}(\phi(\mathbf{x})|\boldsymbol{w}_i, \Sigma)$ and $\mathcal{N}(\phi(\mathbf{x})|\boldsymbol{p}_v, \Sigma)$. According to Bayes' Theorem:

$$p(\boldsymbol{w}_i|\boldsymbol{p}_v, \phi(\mathbf{x})) = \frac{p(\phi(\mathbf{x})|\boldsymbol{w}_i, \boldsymbol{p}_v)p(\boldsymbol{w}_i|\boldsymbol{p}_v)}{\sum_{j=1}^{|\mathcal{Y}_b|} p(\phi(\mathbf{x})|\boldsymbol{w}_j, \boldsymbol{p}_v)p(\boldsymbol{w}_j|\boldsymbol{p}_v)}. \quad (9)$$

$|\mathcal{Y}_b|$ is the number of classes seen before. The last term in denominator and numerator measures the similarity between class $\boldsymbol{w}_i$ and $\boldsymbol{p}_v$, and we assume it follows a Gaussian distribution, *i.e.*, $p(\boldsymbol{w}_i|\boldsymbol{p}_v) = \mathcal{N}(\boldsymbol{w}_i|\boldsymbol{p}_v, \Sigma)$. We have:

$$\begin{aligned} p(\boldsymbol{w}_i|\boldsymbol{p}_v, \phi(\mathbf{x})) &= \frac{p(\phi(\mathbf{x})|\boldsymbol{w}_i, \boldsymbol{p}_v)p(\boldsymbol{w}_i|\boldsymbol{p}_v)}{\sum_{j=1}^{|\mathcal{Y}_b|} p(\phi(\mathbf{x})|\boldsymbol{w}_j, \boldsymbol{p}_v)p(\boldsymbol{w}_j|\boldsymbol{p}_v)} \\ &= \frac{\text{Mix}_\eta\left(\boldsymbol{m}\left(\boldsymbol{w}_i, \phi\left(\mathbf{x}\right)\right), \boldsymbol{m}\left(\boldsymbol{p}_v, \phi\left(\mathbf{x}\right)\right)\right)\boldsymbol{m}(\boldsymbol{w}_i, \boldsymbol{p}_v)}{\sum_{j=1}^{|\mathcal{Y}_b|} \text{Mix}_\eta\left(\boldsymbol{m}\left(\boldsymbol{w}_j, \phi\left(\mathbf{x}\right)\right), \boldsymbol{m}\left(\boldsymbol{p}_v, \phi\left(\mathbf{x}\right)\right)\right)\boldsymbol{m}(\boldsymbol{w}_j, \boldsymbol{p}_v)}, \end{aligned} \quad (10)$$

where $\boldsymbol{m}(\boldsymbol{\mu}, \boldsymbol{t}) = \exp\left(\left(\Sigma^{-1}\boldsymbol{\mu}\right)^\top \boldsymbol{t} - \frac{1}{2}\boldsymbol{\mu}^\top \Sigma^{-1}\boldsymbol{\mu}\right)$, $\text{Mix}_\eta(a, b) = \eta a + (1 - \eta)b$. When $\phi(\mathbf{x}), \boldsymbol{p}, \boldsymbol{w}$ are normalized and $\Sigma = I$, Eq. 10 turns into:

$$\frac{\text{Mix}_\eta\left(\exp\left(\boldsymbol{w}_i^\top\left(\phi\left(\mathbf{x}\right) + \boldsymbol{p}_v\right)\right), \exp\left(\boldsymbol{p}_v^\top\left(\phi\left(\mathbf{x}\right) + \boldsymbol{w}_i\right)\right)\right)}{\sum_{j=1}^{|\mathcal{Y}_b|} \text{Mix}_\eta\left(\exp\left(\boldsymbol{w}_j^\top\left(\phi\left(\mathbf{x}\right) + \boldsymbol{p}_v\right)\right), \exp\left(\boldsymbol{p}_v^\top\left(\phi\left(\mathbf{x}\right) + \boldsymbol{w}_j\right)\right)\right)}.$$

Specifically, when $\eta = 1$, Eq. 10 further degrades into: $\frac{\exp(\boldsymbol{w}_i^\top(\phi(\mathbf{x}) + \boldsymbol{p}_v))}{\sum_{j=1}^{|\mathcal{Y}_b|} \exp(\boldsymbol{w}_j^\top(\phi(\mathbf{x}) + \boldsymbol{p}_v))}$. It turns into ProtoNet (Eq. 4) when $p(\boldsymbol{w}_i|\boldsymbol{p}_v)$ is a uniform distribution, which we analyze in the supplementary. In our implementation, we set $\eta = 0.5$ as default for simplification. The inference process builds a channel for virtual prototypes to be taken into account for the final prediction and reflects their influences.

To summarize, with the help of virtual prototypes learned in the pretraining stage, we are able to obtain a more informative distribution for $\phi(\mathbf{x})$, which helps build a stronger incremental classifier in the incremental stage. The inference process validates the forward compatibility of FACT.

5. Experiment

In this section, we compare FACT on benchmark FSCIL datasets with state-of-the-art methods and large scale dataset ImageNet. The ablations verify the effectiveness of forward compatible training, and we visualize the incremental process of FACT with new classes.

Table 1. Detailed accuracy of each incremental session on CUB200 dataset. The results of compared methods are cited from [41] and [58]. Methods with † are reproduced under FSCIL setting with the source code. Please refer to supplementary for results on other datasets.

Method	Accuracy in each session (%) ↑											PD ↓	Δ PD
	0	1	2	3	4	5	6	7	8	9	10		
Finetune	68.68	43.70	25.05	17.72	18.08	16.95	15.10	10.06	8.93	8.93	8.47	60.21	**+41.25**
Pre-Allocated RPC† [32]	68.47	51.00	45.42	40.76	35.90	33.18	27.23	24.24	21.18	17.34	16.20	52.27	**+33.31**
iCaRL [33]	68.68	52.65	48.61	44.16	36.62	29.52	27.83	26.26	24.01	23.89	21.16	47.52	**+28.56**
EEIL [8]	68.68	53.63	47.91	44.20	36.30	27.46	25.93	24.70	23.95	24.13	22.11	46.57	**+27.61**
Rebalancing [21]	68.68	57.12	44.21	28.78	26.71	25.66	24.62	21.52	20.12	20.06	19.87	48.81	**+29.85**
TOPIC [41]	68.68	62.49	54.81	49.99	45.25	41.40	38.35	35.36	32.22	28.31	26.26	42.40	**+23.44**
SPPR [67]	68.68	61.85	57.43	52.68	50.19	46.88	44.65	43.07	40.17	39.63	37.33	31.35	**+12.39**
Decoupled-NegCosine† [26]	74.96	70.57	66.62	61.32	60.09	56.06	55.03	52.78	51.50	50.08	48.47	26.49	**+7.53**
Decoupled-Cosine [45]	75.52	70.95	66.46	61.20	60.86	56.88	55.40	53.49	51.94	50.93	49.31	26.21	**+7.25**
Decoupled-DeepEMD [57]	75.35	70.69	66.68	62.34	59.76	56.54	54.61	52.52	50.73	49.20	47.60	27.75	**+8.79**
CEC [58]	75.85	71.94	68.50	63.50	62.43	58.27	57.73	55.81	54.83	53.52	52.28	23.57	**+4.61**
FACT	**75.90**	**73.23**	**70.84**	**66.13**	**65.56**	**62.15**	**61.74**	**59.83**	**58.41**	**57.89**	**56.94**	**18.96**	

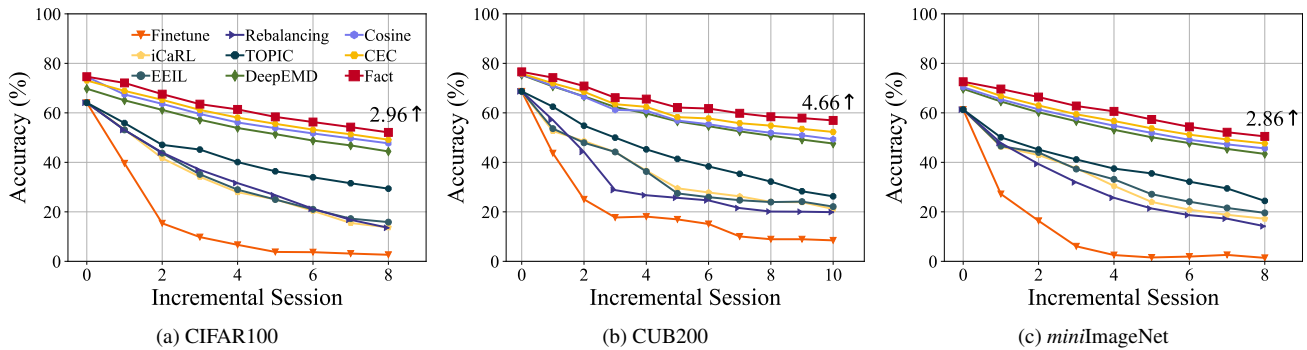(a) CIFAR100   (b) CUB200   (c) *mini*ImageNet

Figure 3. Top-1 accuracy of each incremental session. We show the legends in (a), and annotate the performance gap after the last session between FACT and the runner-up method at the end of each curve. Please refer to Table 1 and supplementary for detailed values.

5.1. Implementation Details

Dataset: Following the benchmark setting [41], we evaluate the performance on CIFAR100 [23], CUB200-2011 [46] and *mini*ImageNet [34]. We also experiment on the large scale dataset ImageNet ILSVRC2012 [14]. CIFAR100 contains 60,000 images from 100 classes. CUB200 is a fine-grained image classification task with 200 classes. *mini*ImageNet is a subset of ImageNet [14] with 100 classes. ImageNet1000 contains 1,000 classes, and we also sample a subset of 100 classes according to [48], denoted as ImageNet100.

Dataset Split: For CIFAR100, *mini*ImageNet and ImageNet100, 100 classes are divided into 60 base classes and 40 new classes. The new classes are formulated into eight *5-way 5-shot* incremental tasks. For CUB200, 200 classes are divided into 100 base classes and 100 incremental classes, and the new classes are formulated into ten 10-way 5-shot incremental tasks. For ImageNet1000, 600 classes are selected as base classes, and the other 400 classes are formulated into eight 50-way 5-shot tasks. We use the *identical* training splits [41] (including base and incremental sessions) for every compared method for a fair comparison. The testing set is the same as the original one to evaluate holistically.

Compared methods: We first compare to classical CIL methods iCaRL [33], EEIL [8] Pre-Allocated RPC [32] and Rebalancing [21]. Besides, we also compare to current SOTA FSCIL algorithms: TOPIC [41], SPPR [67],
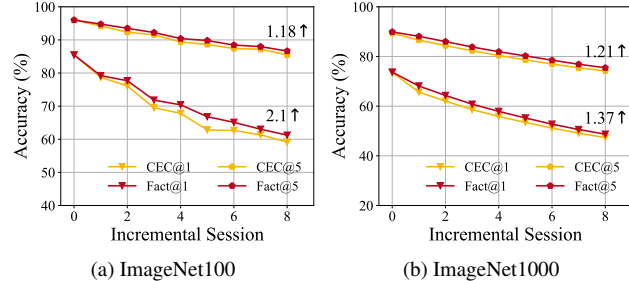

(a) ImageNet100   (b) ImageNet1000

Figure 4. Top-1 and Top-5 accuracy on ImageNet100/1000. The performance gap is annotated at the end of each curve.

Decoupled-DeepEMD/Cosine/NegCosine [26, 45, 57] and CEC [58]. We report the baseline method by finetuning the model with few-shot instances as 'finetune'.

Training details: All models are deployed with PyTorch [31]. We use the *same* network backbone [41] for *all* compared methods. For CIFAR100, we use ResNet20 [19], while for the others we use ResNet18. The model is trained with a batch size of 256 for 600 epochs, and we use SGD with momentum for optimization. The learning rate starts from 0.1 and decays with cosine annealing.

Evaluation protocol: Following [41], we denote the Top-1 accuracy after the i-th session as $\mathcal{A}_i$. We also quantitatively measure the forgetting phenomena with performance dropping rate (PD), *i.e.*, PD $= \mathcal{A}_0 - \mathcal{A}_B$, where $\mathcal{A}_0$ stands for the accuracy after base session, and $\mathcal{A}_B$ is the accuracy after
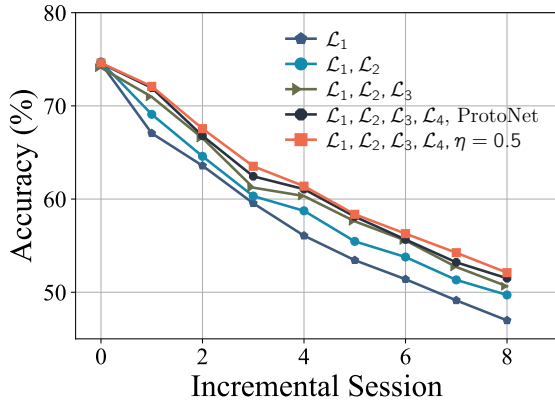
Figure 5. Ablation study. Reserving and forecasting the embedding space for new classes help enhance the forward compatibility. Every part in FACT improves the performance of FSCIL.

the last session.

5.2. Benchmark Comparison

We report the performance curve over benchmark datasets [41] *i.e.*, CIFAR100, CUB200, and *mini*ImageNet in Figure 3 and Table 1. We also report the results on large scale dataset, *i.e.*, ImageNet100/1000 in Figure 4.

We can infer from Figure 3 that FACT consistently outperforms the current SOTA method, *i.e.*, CEC, by 3-5% on benchmark datasets. The poor performance of CIL methods indicates that addressing the backward compatibility via output restriction is improper for FSCIL. The backward compatible tuning consumes numerous training instances, which is inconsistent with FSCIL scenario. FACT also outperforms FSCIL methods which enhance backward compatibility by embedding freezing, indicating that considering forward compatibility is more proper and suitable for FSCIL. Besides, FACT has better performance than Decoupled-NegCosine, which considers forward compatibility from another perspective. It reveals that our training scheme is more proper for FSCIL. We report the detailed value on CUB200 dataset in Table 1. It shows that FACT has a minimum degradation in terms of PD metric, indicating forward compatible training consistently resists forgetting in FSCIL.

Apart from the benchmark setting with small scale datasets, we also suggest conducting experiments with large scale datasets, *i.e.*, ImageNet. We report top-1 and top-5 accuracy of FACT and the runner-up method CEC in Figure 4. As can be seen from the figures, FACT still surpasses CEC by all metrics. To conclude, FACT consistently handles small scale and large scale FSCIL tasks with SOTA performance.

5.3. Ablation Study

There are four loss terms in FACT, and we conduct an ablation study to analyze the importance of each component in FACT on CIFAR100 dataset. The implementation details are the same as in the benchmark setting. We report the accuracy curve of different variants in Figure 5.
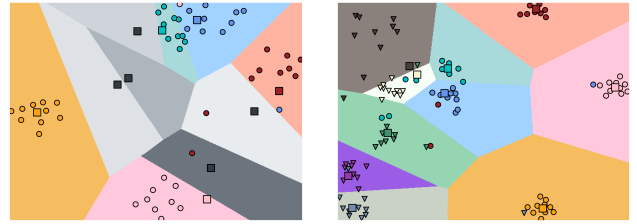
(a) Base session, 5 old classes & 5 virtual prototypes.

(b) Incremental session, 5 old classes & 5 new classes.
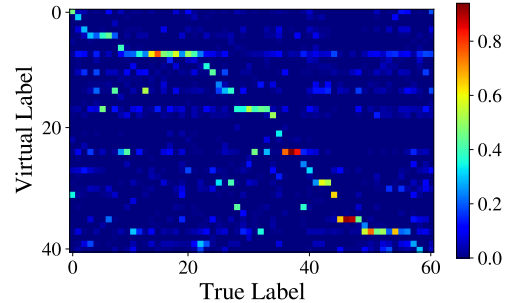
(c) Pseudo label $\hat{y}$ assignment of each known class.

Figure 6. Top: visualization of the decision boundary on CIFAR100 between 2 sessions. Old classes are shown in dots, and new classes are shown in triangles. The shadow region represents the decision boundary. Bottom: the pseudo label assignment for each known class. Each known class is assigned to one or more virtual classes.

The model trained with $\mathcal{L}_1$ gets the lowest accuracy among all variants. $\mathcal{L}_1$ does not consider the future possible new classes and overspreads the embedding space with known classes. As a result, the model trained with $\mathcal{L}_1$ lacks forward compatibility and works poorly. When equipped with $\mathcal{L}_2$, the model pre-assigns extra virtual prototypes, which reserves the embedding space for new classes by virtual loss. Consequently, the model becomes growable and increases the forward compatibility. We then add $\mathcal{L}_3$ to forecast the possible new classes during the pretraining process. By anticipating the future and reserving the corresponding space, the model becomes provident with better compatibility. Optimizing $\mathcal{L}_4$ makes the final loss symmetric, which also increases the compactness of embedding and facilitates FSCIL. Lastly, these virtual prototypes can be utilized as bases encoded in the embedding space, which helps inference via Eq. 10. The comparison between ProtoNet and Eq. 10 with $\eta = 0.5$ verifies the effectiveness of these virtual prototypes during inference. Ablations validate that forward compatible training is helpful for FSCIL.

5.4. Visualization of Incremental Sessions

We visualize the learned decision boundaries with t-SNE [42] on CIFAR100 dataset in Figure 6a,6b. Figure 6a stands for the decision boundary in the base session, where we train 5 old classes and 5 virtual prototypes. Known classes are shown in colorful dots, and their prototypes are represented by squares. Black squares represent virtual prototypes, and their embedding space is shown in gray with
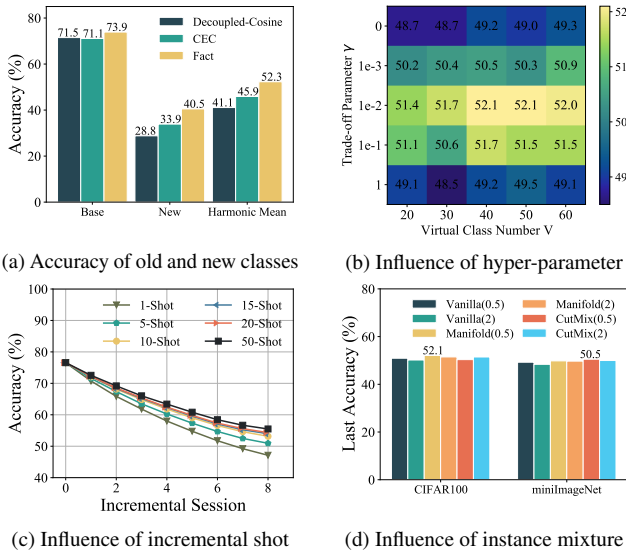
(a) Accuracy of old and new classes

(b) Influence of hyper-parameter

(c) Influence of incremental shot

(d) Influence of instance mixture

Figure 7. Analysis on different performance measure, influence of hyper-parameters, incremental shot and instance mixture method.

different shades. As we can infer from Figure 6a, forward compatible training reserves the embedding space for new classes (the gray parts) and makes the known class embedding more compact. When moving to Figure 6b, we show the incremental stage when 5 new classes emerge (represented by triangles). The reserved space benefits the incremental learning process, and new classes do not need to squeeze the embedding of old ones during incremental learning process.

We record the pseudo label assigned for each class during training and show the frequency matrix in Figure 6c. The matrix is *reconstructed* by switching lines to make similar classes close. As we can infer from the figure, each known class is assigned to one or more virtual classes. Virtual prototypes cover all the known classes and prepare for the possible update between them.

5.5. Further Analysis

Performance Measure: The accuracy measure covers both old and new classes. To explore the ability of new class learning and forgetting resistance, we report the accuracy of base classes and new classes, and the harmonic mean between them on CUB200. The other settings are the same as the benchmark, and the results are shown in Figure 7a. We also report the results of two competitive methods, *i.e.*, Decoupled-Cosine and CEC. We can infer that FACT has better performance on new classes, which verifies the effectiveness of forward compatible training.

Hyper-Parameter: There are two hyper-parameters in FACT, *i.e.*, trade-off parameter γ in Eq. 5 and the number of virtual prototypes V. We report the final accuracy on CIFAR100 varying the γ and V in Figure 7b. We fix the other settings the same as in the benchmark experiment, and change γ from $\{0, 1e-3, 1e-2, 1e-1, 1\}$. We also change the virtual class number from $\{20, 30, 40, 50, 60\}$, resulting in 25 compared results. We can infer from Figure 7b that

FACT prefers small γ and large V. However, since the increasing trend w.r.t. V is trivial for $V > 40$, we suggest to set $\gamma = 0.01$ and $V = NB$ as default, where NB indicates the number of new classes in total.

Incremental Shot: FACT relies on the N-way K-shot dataset to estimate the class center of new classes, and we change the shot number to find out the influence on the last accuracy. We fix the incremental way the same as in the benchmark setting, and change the shot number K among $\{1, 5, 10, 15, 20, 50\}$ on *mini*ImageNet. As we can infer from Figure 7c, with more instances per class, the estimation of prototypes will be more precise, and the performance will correspondingly improve. However, we also find that the improvement trend becomes trivial for $K > 20$.

Instance Mixture Method: Apart from manifold mixup [44], there are other ways for instance mixture, *e.g.*, vanilla mixup [59] and CutMix [55]. We explore the influence of the last accuracy by changing these mixture methods in Figure 7d. The accuracy is annotated for the method with the best performance. Since the mixture weight λ is sampled from Beta(α, α), we also change α from $\{0.5, 2\}$. Figure 7d implies that the final accuracy is robust to mixture methods, and we use manifold mixup with $\alpha = 0.5$ as default.

6. Conclusion

Few-shot class-incremental learning ability becomes an essential characteristic in real-world learning systems, which requires a model to learn few-shot new classes without forgetting old ones. In this paper, we address the importance of building forward compatible models for FSCIL. We pre-assign virtual prototypes in the embedding space to reserve the space for new classes from two aspects. The model becomes growable and provident by reserving and forecasting the future possible new classes, which releases the shock during model updating. These virtual prototypes act as bases in the embedding space and facilitate inference performance of FSCIL. FACT efficiently incorporates new knowledge into the current model and obtains SOTA performance.

Limitations: Possible limitations include the hypothesis of new class instances. It turns into CIL problem if sufficient new class instances are available, where forward and backward compatibility should be considered simultaneously.

References

[1] Idan Achituve, Aviv Navon, Yochai Yemini, Gal Chechik, and Ethan Fetaya. Gp-tree: A gaussian process classifier for few-shot incremental learning. In *ICML*, pages 54–65. PMLR, 2021. 2

[2] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *ECCV*, pages 139–154, 2018. 2

[3] Antreas Antoniou, Harrison Edwards, and Amos Storkey. How to train your maml. In *ICLR*, 2018. 2

[4] Sébastien Arnold, Shariq Iqbal, and Fei Sha. When maml can adapt fast and how to assist when it cannot. In *AISTATS*, pages 244–252, 2021. 2

[5] Gagan Bansal, Besmira Nushi, Ece Kamar, Daniel S Weld, Walter S Lasecki, and Eric Horvitz. Updates in human-ai teams: Understanding and addressing the performance/compatibility tradeoff. In *AAAI*, pages 2429–2437, 2019. 2

[6] Eden Belouadah and Adrian Popescu. Il2m: Class incremental learning with dual memory. In *ICCV*, pages 583–592, 2019. 2

[7] Mateusz Budnik and Yannis Avrithis. Asymmetric metric learning for knowledge transfer. In *CVPR*, pages 8228–8238, 2021. 2

[8] Francisco M Castro, Manuel J Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *ECCV*, pages 233–248, 2018. 6

[9] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. In *ICLR*, 2018. 2

[10] Ali Cheraghian, Shafin Rahman, Pengfei Fang, Soumava Kumar Roy, Lars Petersson, and Mehrtash Harandi. Semantic-aware knowledge distillation for few-shot class-incremental learning. In *CVPR*, pages 2534–2543, 2021. 2

[11] Ali Cheraghian, Shafin Rahman, Sameera Ramasinghe, Pengfei Fang, Christian Simon, Lars Petersson, and Mehrtash Harandi. Synthesized feature based few-shot class-incremental learning on a mixture of subspaces. In *ICCV*, pages 8661–8670, 2021. 2

[12] Yu-Ying Chou, Hsuan-Tien Lin, and Tyng-Luh Liu. Adaptive and generative zero-shot learning. In *ICLR*, 2020. 1

[13] Matthias Delange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Greg Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *TPAMI*, page In press., 2021. 2

[14] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009. 1, 6

[15] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, pages 1126–1135, 2017. 2

[16] Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999. 1

[17] Ovidiu Gheorghioiu, Alexandru Salcianu, and Martin Rinard. Interprocedural compatibility analysis for static object preallocation. In *POPL*, pages 273–284, 2003. 2

[18] Heitor Murilo Gomes, Jean Paul Barddal, Fabrício Enembreck, and Albert Bifet. A survey on ensemble learning for data stream classification. *CSUR*, 50(2):1–36, 2017. 1

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2015. 1, 6

[20] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 2, 3

[21] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *CVPR*, pages 831–839, 2019. 6

[22] 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. *PNAS*, 114(13):3521–3526, 2017. 1, 2

[23] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, 2009. 6

[24] Anna Kukleva, Hilde Kuehne, and Bernt Schiele. Generalized and incremental few-shot learning by explicit learning and calibration without forgetting. In *ICCV*, pages 9020–9029, 2021. 2

[25] Zhizhong Li and Derek Hoiem. Learning without forgetting. *TPAMI*, 40(12):2935–2947, 2017. 1, 2

[26] Bin Liu, Yue Cao, Yutong Lin, Qi Li, Zheng Zhang, Mingsheng Long, and Han Hu. Negative margin matters: Understanding margin in few-shot classification. In *ECCV*, pages 438–455, 2020. 2, 6

[27] Pratik Mazumder, Pravendra Singh, and Piyush Rai. Few-shot lifelong learning. In *AAAI*, pages 2337–2345, 2021. 2

[28] Qiang Meng, Chixiang Zhang, Xiaoqiang Xu, and Feng Zhou. Learning compatible embeddings. In *ICCV*, pages 9939–9948, 2021. 2

[29] Santosh Nagarakatte, Jianzhou Zhao, Milo MK Martin, and Steve Zdancewic. Softbound: Highly compatible and complete spatial memory safety for c. In *PLDI*, pages 245–258, 2009. 2

[30] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018. 2

[31] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, pages 8026–8037, 2019. 6

[32] Federico Pernici, Matteo Bruni, Claudio Baecchi, Francesco Turchini, and Alberto Del Bimbo. Class-incremental learning with pre-allocated fixed classifiers. In *ICPR*, pages 6259–6266, 2021. 2, 6

[33] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *CVPR*, pages 2001–2010, 2017. 2, 6

[34] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015. 6

[35] Yantao Shen, Yuanjun Xiong, Wei Xia, and Stefano Soatto. Towards backward-compatible representation learning. In *CVPR*, pages 6368–6377, 2020. 2

[36] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1

[37] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *NIPS*, pages 4080–4090, 2017. 2, 3

[38] Megha Srivastava, Besmira Nushi, Ece Kamar, Shital Shah, and Eric Horvitz. An empirical analysis of backward compatibility in machine learning systems. In *KDD*, pages 3272–3280, 2020. 2

[39] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *CVPR*, pages 1199–1208, 2018. 2

[40] Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. In *CVPR*, pages 10781–10790, 2020. 1

[41] Xiaoyu Tao, Xiaopeng Hong, Xinyuan Chang, Songlin Dong, Xing Wei, and Yihong Gong. Few-shot class-incremental learning. In *CVPR*, pages 12183–12192, 2020. 2, 6, 7

[42] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *JMLR*, 9(11), 2008. 7

[43] Pradeep Varma, Rudrapatna K Shyamasundar, and Harshit J Shah. Backward-compatible constant-time exception-protected memory. In *FSE/ESEC*, pages 71–80, 2009. 2

[44] Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. Manifold mixup: Better representations by interpolating hidden states. In *ICML*, pages 6438–6447, 2019. 4, 8

[45] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. *NIPS*, 29:3630–3638, 2016. 2, 6

[46] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011. 6

[47] Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. Generalizing from a few examples: A survey on few-shot learning. *CSUR*, 53(3):1–34, 2020. 2

[48] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *CVPR*, pages 374–382, 2019. 1, 2, 6

[49] Wei Xu, Daniel C DuVarney, and R Sekar. An efficient and backwards-compatible transformation to ensure memory safety of c programs. In *FSE*, pages 117–126, 2004. 2

[50] Shipeng Yan, Jiangwei Xie, and Xuming He. Der: Dynamically expandable representation for class incremental learning. In *CVPR*, pages 3014–3023, 2021. 1

[51] Yang Yang, Da-Wei Zhou, De-Chuan Zhan, Hui Xiong, Yuan Jiang, and Jian Yang. Cost-effective incremental deep model: Matching model capacity with the least sampling. *IEEE Transactions on Knowledge and Data Engineering*, 2021. 2

[52] Han-Jia Ye, Yi Shi, and De-Chuan Zhan. Identifying ambiguous similarity conditions via semantic matching. In *CVPR*, 2022. 2

[53] Han-Jia Ye, De-Chuan Zhan, Yuan Jiang, and Zhi-Hua Zhou. Heterogeneous few-shot model rectification with semantic mapping. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(11):3878–3891, 2020. 2

[54] Han-Jia Ye, Da-Wei Zhou, Lanqing Hong, Zhenguo Li, Xiu-Shen Wei, and De-Chuan Zhan. Contextualizing multiple tasks via learning to decompose. *arXiv preprint arXiv:2106.08112*, 2021. 1

[55] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *ICCV*, pages 6023–6032, 2019. 8

[56] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *ICML*, pages 3987–3995, 2017. 2

[57] Chi Zhang, Yujun Cai, Guosheng Lin, and Chunhua Shen. Deepemd: Few-shot image classification with differentiable earth mover's distance and structured classifiers. In *CVPR*, pages 12203–12213, 2020. 2, 6

[58] Chi Zhang, Nan Song, Guosheng Lin, Yun Zheng, Pan Pan, and Yinghui Xu. Few-shot incremental learning with continually evolved classifiers. In *CVPR*, pages 12455–12464, 2021. 1, 2, 6

[59] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2018. 8

[60] Bowen Zhao, Xi Xiao, Guojun Gan, Bin Zhang, and Shu-Tao Xia. Maintaining discrimination and fairness in class incremental learning. In *CVPR*, pages 13208–13217, 2020. 1, 2

[61] Hanbin Zhao, Yongjian Fu, Mintong Kang, Qi Tian, Fei Wu, and Xi Li. Mgsvf: Multi-grained slow vs. fast framework for few-shot class-incremental learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. 2

[62] Da-Wei Zhou, Fu-Yun Wang, Han-Jia Ye, and De-Chuan Zhan. Pycil: A python toolbox for class-incremental learning. *arXiv preprint arXiv:2112.12533*, 2021. 2

[63] Da-Wei Zhou, Yang Yang, and De-Chuan Zhan. Learning to classify with incremental new class. *IEEE Transactions on Neural Networks and Learning Systems*, 2021. 1

[64] Da-Wei Zhou, Han-Jia Ye, and De-Chuan Zhan. Co-transport for class-incremental learning. In *ACM MM*, pages 1645–1654, 2021. 2

[65] Da-Wei Zhou, Han-Jia Ye, and De-Chuan Zhan. Learning placeholders for open-set recognition. In *CVPR*, pages 4401–4410, 2021. 1

[66] Fei Zhu, Xu-Yao Zhang, Chuang Wang, Fei Yin, and Cheng-Lin Liu. Prototype augmentation and self-supervision for incremental learning. In *CVPR*, pages 5871–5880, 2021. 2

[67] Kai Zhu, Yang Cao, Wei Zhai, Jie Cheng, and Zheng-Jun Zha. Self-promoted prototype refinement for few-shot class-incremental learning. In *CVPR*, pages 6801–6810, 2021. 1, 6