# Supplementary Material for 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

## 1. Gradient Analysis

In the main paper, we give the analysis about $\mathcal{L}_1$, $\mathcal{L}_2$, $\mathcal{L}_3$, $\mathcal{L}_4$ with regard to the embedding $\phi(\mathbf{x})$. In this section, we give the full analysis about gradients, including the gradients with regard to $W$, and the scenario where $g(\cdot)$ is not an identity function.

With a bit of redundancy, we first revisit the notations defined in the main paper. We define the model output as $f_v(\mathbf{x}) = [W, P_v]^\top \phi(\mathbf{x})$, where $P_v = [\boldsymbol{p}_1, \cdots, \boldsymbol{p}_V] \in \mathbb{R}^{d\times V}$ is the collection of virtual prototypes. The output probability after softmax operation is denoted as: $\boldsymbol{a} = \text{Softmax}\left([W, P_v]^\top \phi\left(\mathbf{x}\right)\right) = [a_1, \cdots, a_{|Y_0|+V}]$. We decouple the embedding module into two parts, *i.e.*, $\phi(\mathbf{x}) = g(h(\mathbf{x}))$.

The final loss is defined as: $\mathcal{L} = \mathcal{L}_v + \mathcal{L}_f$, where

$$\mathcal{L}_v(\mathbf{x}, y) = \underbrace{\ell(f_v(\mathbf{x}), y)}_{\mathcal{L}_1} + \gamma \underbrace{\ell\left(\text{Mask}(f_v(\mathbf{x}), y), \hat{y}\right)}_{\mathcal{L}_2} \tag{1}$$

$$\mathcal{L}_f(\boldsymbol{z}) = \underbrace{\ell(f_v(\boldsymbol{z}), \hat{y})}_{\mathcal{L}_3} + \gamma \underbrace{\ell\left(\text{Mask}(f_v(\boldsymbol{z}), \hat{y}), \hat{y}\right)}_{\mathcal{L}_4} . \tag{2}$$

In Eq. 2, $\boldsymbol{z}$ is the manifold mixup instance from two different classes, *i.e.*, $\boldsymbol{z} = g\left[\lambda h(\mathbf{x}_i) + (1-\lambda)h(\mathbf{x}_j)\right], y_i \neq y_j$. In the following, we analyze the gradient using cross-entropy as the loss function $\ell(\cdot,\cdot)$.

Following the assumption in the main paper, in the analysis of gradients, we treat the classifier $[W, P_v]$ as a unified classifier, *i.e.*, denote $[W, P_v] = [\boldsymbol{w}_1, \cdots, \boldsymbol{w}_{|Y_0|+V}]$ for ease of discussion.

### 1.1. Supplementary Analysis for the Main Paper

**Analyzing $\mathcal{L}_1$:**

The optimization target of $\mathcal{L}_1$ corresponds to:

$$\mathcal{L}_1 = -\log a_y ,$$

where the output probability is defined as:

$$a_i = \frac{\exp\left(\boldsymbol{w}_i^\top \phi(\mathbf{x})\right)}{\sum_{j=1}^{|Y_0|+V} \exp\left(\boldsymbol{w}_j^\top \phi(\mathbf{x})\right)} . \tag{3}$$

Hence, we can obtain the negative gradient w.r.t. the embedding $\phi(\mathbf{x})$:

$$-\nabla_{\phi(\mathbf{x})}\mathcal{L}_1 = \boldsymbol{w}_y - \sum_{i=1}^{|Y_0|+V} a_i \boldsymbol{w}_i . \tag{4}$$

Eq. 4 indicates that optimizing $\mathcal{L}_1$ will push the embedding $\phi(\mathbf{x})$ towards the direction of $\boldsymbol{w}_y$, and away from other prototypes. It is the classical loss function, which helps to acquire the classification ability and discriminability among known classes.

We can also obtain the negative gradient w.r.t. the prototype $\boldsymbol{w}_i$:

$$-\nabla_{\boldsymbol{w}_i}\mathcal{L}_1 = \begin{cases} (1-a_i)\phi(\mathbf{x}), & i = y \\ -a_i\phi(\mathbf{x}), & otherwise \end{cases} . \tag{5}$$

Note that $a_i \in (0, 1)$, which reflects the similarity between $\boldsymbol{w}_i$ and $\phi(\mathbf{x})$. As a result, for the prototype weight from the ground truth label, *i.e.*, $\boldsymbol{w}_y$, the smaller $a_y$ is, the larger the gradient norm is. For the prototype from non-target class, *i.e.*, $\boldsymbol{w}_k, k \neq y$, the larger $a_k$ is, the larger the gradient norm is. Eq. 5 means that optimizing $\mathcal{L}_1$ will push the class prototype of the target class, *i.e.*, $\boldsymbol{w}_y$ towards the embedding of $\phi(\mathbf{x})$, and push the non-target class prototypes away from it. The effect of Eq. 5 is consistent with that of Eq. 4, which helps to classify $(\mathbf{x}, y)$ correctly.

**Analyzing $\mathcal{L}_2$**

We denote the pseudo label assigned to $\mathbf{x}$ is $\hat{y}$, and $\mathcal{L}_2$ equals to:

$$\mathcal{L}_2 = -\log a_{\hat{y}} .$$

Note that when calculating $\mathcal{L}_2$, the output probability is masked out for the ground-truth class $y$, which yields:

$$a_i = \begin{cases} 0, & i = y \\ \frac{\exp\left(\boldsymbol{w}_i^\top \phi(\mathbf{x})\right)}{\sum_{j=1}^{|Y_0|+V}\left(\exp\left(\boldsymbol{w}_j^\top \phi(\mathbf{x})\right)\right) - \exp\left(\boldsymbol{w}_y^\top \phi(\mathbf{x})\right)}, & otherwise \end{cases} . \tag{6}$$

Hence, we can obtain the negative gradient w.r.t. the embedding $\phi(\mathbf{x})$:

$$-\nabla_{\phi(\mathbf{x})}\mathcal{L}_2 = \boldsymbol{w}_{\hat{y}} - \sum_{i=1}^{|Y_0|+V} a_i \boldsymbol{w}_i \,. \tag{7}$$

Eq. 7 indicates that optimizing $\mathcal{L}_2$ will push the embedding $\phi(\mathbf{x})$ towards the direction of $\boldsymbol{w}_{\hat{y}}$, and away from other prototypes. As a result, we reserve the embedding space for new classes explicitly by pushing other prototypes away to make the model growable and forward compatible. Note that $a_y = 0$ and the push effect will not influence the ground truth class $y$, *i.e.*, optimizing Eq. 7 will not weaken the classification performance on known classes.

We can also obtain the negative gradient w.r.t. the prototype $\boldsymbol{w}_i$:

$$-\nabla_{\boldsymbol{w}_i}\mathcal{L}_2 = \begin{cases} (1-a_i)\phi(\mathbf{x}), & i = \hat{y} \\ -a_i\phi(\mathbf{x}), & otherwise \end{cases} . \tag{8}$$

Eq. 8 means that optimizing $\mathcal{L}_2$ will push the class prototype of the target class, *i.e.*, $\boldsymbol{w}_{\hat{y}}$ towards the embedding of $\phi(\mathbf{x})$, and push the non-target class prototypes away from it. Note that the probability $a_y$ is 0 for the ground truth class, and optimizing $\mathcal{L}_2$ will not hurt the classification performance. The effect of Eq. 8 is consistent with that of Eq. 7, which helps reserve the embedding space for class $\hat{y}$ explicitly without harming the classification performance.

**Analyzing $\mathcal{L}_3$**

We denote the pseudo label assigned to $\boldsymbol{z}$ is $\hat{y}$, where $\boldsymbol{z}$ is the product of manifold mixup, *i.e.*, $\boldsymbol{z} = g\left[\lambda h(\mathbf{x}_i) + (1-\lambda)h(\mathbf{x}_j)\right], y_i \neq y_j$. Similar to the assumption made in the main paper, we assume $g(\cdot)$ is identity function and $h(\mathbf{x}) = \phi(\mathbf{x})$. We analyze the scenario when $g(\cdot)$ is not identity in Sec. 1.2. $\mathcal{L}_3$ equals to:

$$\mathcal{L}_3 = -\log a_{\hat{y}} \,,$$

where output probability on class $m$ is defined as:

$$a_m = \frac{\exp\left(\boldsymbol{w}_m^\top \boldsymbol{z}\right)}{\sum_{k=1}^{|Y_0|+V} \exp\left(\boldsymbol{w}_k^\top \boldsymbol{z}\right)} \,. \tag{9}$$

Hence, we can obtain the negative gradient w.r.t. the mixed embedding $\boldsymbol{z}$:

$$-\nabla_{\boldsymbol{z}}\mathcal{L}_3 = \boldsymbol{w}_{\hat{y}} - \sum_{k=1}^{|Y_0|+V} a_k \boldsymbol{w}_k \,. \tag{10}$$

Eq. 10 indicates that optimizing $\mathcal{L}_3$ will push the embedding $\boldsymbol{z}$ towards the direction of $\boldsymbol{w}_{\hat{y}}$, and away from other prototypes. Since $\boldsymbol{z}$ is a generated virtual instance, we reserve the embedding space for new classes explicitly by pushing other prototypes away to make the model provident and forward compatible.

We can also obtain the negative gradient w.r.t. the prototype $\boldsymbol{w}_k$:

$$-\nabla_{\boldsymbol{w}_k}\mathcal{L}_3 = \begin{cases} (1-a_k)\boldsymbol{z}, & k = \hat{y} \\ -a_k\boldsymbol{z}, & otherwise \end{cases} . \tag{11}$$

Eq. 11 means that optimizing $\mathcal{L}_3$ will push the class prototype of the target class, *i.e.*, $\boldsymbol{w}_{\hat{y}}$ towards the embedding of $\boldsymbol{z}$, and push the non-target class prototypes away from it. The effect of Eq. 11 is consistent with that of Eq. 10, which helps reserve the embedding space for class $\hat{y}$ explicitly by instance mixture.

Apart from the gradient w.r.t. the embedding $\boldsymbol{z}$, we also provide that for $\phi(\mathbf{x}_i)$ and $\phi(\mathbf{x}_j)$:

$$\begin{aligned} -\nabla_{\phi(\mathbf{x}_i)}\mathcal{L}_3 &= -\lambda\nabla_{\boldsymbol{z}}\mathcal{L}_3 \\ &= \lambda\left(\boldsymbol{w}_{\hat{y}} - \sum_{k=1}^{|Y_0|+V} a_k \boldsymbol{w}_k\right) \\ -\nabla_{\phi(\mathbf{x}_j)}\mathcal{L}_3 &= -(1-\lambda)\nabla_{\boldsymbol{z}}\mathcal{L}_3 \\ &= (1-\lambda)\left(\boldsymbol{w}_{\hat{y}} - \sum_{k=1}^{|Y_0|+V} a_k \boldsymbol{w}_k\right) , \end{aligned} \tag{12}$$

which indicates that the embedding of mixup components $\phi(\mathbf{x}_i)$ and $\phi(\mathbf{x}_j)$ will be pushed towards the prototype $\boldsymbol{w}_{\hat{y}}$, and away from other prototypes. Eq. 12 is similar to Eq. 7, which reserves the embedding space for new classes and make the model forward compatible.

**Analyzing $\mathcal{L}_4$**

The analysis of $\mathcal{L}_4$ is similar to that of $\mathcal{L}_3$. Assume the pseudo label assigned to the masked probability is $\hat{\hat{y}}$, $\mathcal{L}_4$ is defined as:

$$\mathcal{L}_4 = -\log a_{\hat{\hat{y}}} \,,$$

where the output probability for class $m$ is defined as:

$$a_m = \begin{cases} 0, & m = \hat{y} \\ \frac{\exp\left(\boldsymbol{w}_m^\top \boldsymbol{z}\right)}{\sum_{k=1}^{|Y_0|+V}\left(\exp\left(\boldsymbol{w}_k^\top \boldsymbol{z}\right)\right) - \exp\left(\boldsymbol{w}_{\hat{y}}^\top \boldsymbol{z}\right)}, & otherwise \end{cases} . \tag{13}$$

Similarly, we have the negative gradient w.r.t. the manifold mixup product $\boldsymbol{z}$:

$$-\nabla_{\boldsymbol{z}}\mathcal{L}_4 = \boldsymbol{w}_{\hat{\hat{y}}} - \sum_{k=1}^{|Y_0|+V} a_k \boldsymbol{w}_k \,. \tag{14}$$

The negative gradient w.r.t. the classifier weight yields:

$$-\nabla_{\boldsymbol{w}_k}\mathcal{L}_4 = \begin{cases} (1-a_k)\boldsymbol{z}, & k = \hat{\hat{y}} \\ -a_k\boldsymbol{z}, & otherwise \end{cases} . \tag{15}$$

The negative gradient w.r.t. the mixup components yields:

$$\begin{aligned} -\nabla_{\phi(\mathbf{x}_i)}\mathcal{L}_4 &= -\lambda\nabla_{\boldsymbol{z}}\mathcal{L}_4 \\ &= \lambda\left(\boldsymbol{w}_{\hat{y}} - \sum_{k=1}^{|Y_0|+V} a_k\boldsymbol{w}_k\right) \\ -\nabla_{\phi(\mathbf{x}_j)}\mathcal{L}_4 &= -(1-\lambda)\nabla_{\boldsymbol{z}}\mathcal{L}_4 \\ &= (1-\lambda)\left(\boldsymbol{w}_{\hat{y}} - \sum_{k=1}^{|Y_0|+V} a_k\boldsymbol{w}_k\right), \end{aligned} \tag{16}$$

To summarize, when considering $\mathcal{L}_4$, the final loss function becomes symmetric. The effect of $\mathcal{L}_4$ is similar to that of $\mathcal{L}_1$. Since $\mathcal{L}_2$ and $\mathcal{L}_3$ both reserve the embedding space for new classes and squeeze the space of known ones, we seek to trade-off between the squeeze process and avoid over-squeezing with such symmetric loss form. The ablations in the main paper validate the effectiveness of such regularization.

## 1.2. When $g(\cdot)$ is not Identity

In the former part and main paper, we analyze the gradients of $\mathcal{L}_3$ and $\mathcal{L}_4$ by assuming $g(\cdot)$ is the identity function. In this section, we analyze a more general scenario where $g(\cdot)$ is not identity. We first analyze the liner situation and then analyze the nonlinear situation. Note that the gradients of $\mathcal{L}_3$ and $\mathcal{L}_4$ are similar, and we only give the gradients of $\mathcal{L}_3$ as an example.

### 1.2.1 $g(\cdot)$ is Linear

Following the former analysis, if $g(\cdot)$ is a linear layer, we can parameterize it as $\mathbf{V}$, and we have $\phi(\mathbf{x}) = \mathbf{V}^\top h(\mathbf{x})$. Denote the mixed instance at middle layer as $\boldsymbol{b} = \lambda h(\mathbf{x}_i) + (1-\lambda)h(\mathbf{x}_j)$, then $\boldsymbol{z} = g(\boldsymbol{b}) = \mathbf{V}^\top\boldsymbol{b}$ indicates the embedding of mixed instance. Hence, the output probability for mixed instance on class $m$ is defined as:

$$a_m = \frac{\exp\left(\boldsymbol{w}_m^\top\mathbf{V}^\top\boldsymbol{b}\right)}{\sum_{k=1}^{|Y_0|+V}\exp\left(\boldsymbol{w}_k^\top\mathbf{V}^\top\boldsymbol{b}\right)}.$$

We can obtain the negative gradient w.r.t. the final embedding of mixed instance $g(\boldsymbol{b})$:

$$-\nabla_{g(\boldsymbol{b})}\mathcal{L}_3 = \boldsymbol{w}_{\hat{y}} - \sum_{k=1}^{|Y_0|+V} a_k\boldsymbol{w}_k\,, \tag{17}$$

which is same to Eq. 10. The gradients indicate that optimizing $\mathcal{L}_3$ will reserve the embedding space for class $\hat{y}$ by moving $\mathbf{V}^\top\boldsymbol{b}$ towards $\boldsymbol{w}_{\hat{y}}$, and away from other prototypes.

We can further obtain the negative gradients w.r.t. $\boldsymbol{b}$:

$$\begin{aligned} -\nabla_{\boldsymbol{b}}\mathcal{L}_3 &= -\mathbf{V}\nabla_{g(\boldsymbol{b})}\mathcal{L}_3 \\ &= \mathbf{V}\left(\boldsymbol{w}_{\hat{y}} - \sum_{k=1}^{|Y_0|+V} a_k\boldsymbol{w}_k\right), \end{aligned} \tag{18}$$

which indicates that the pushing effect also works in the middle layer (*i.e.*, the output layer of $h(\mathbf{x})$), encouraging the feature reserving in the same direction as the last layer. Eq. 18 verifies that the reserving process works from shallow to deep, which makes forward compatibility maintained holistically. Then we get the negative gradients w.r.t. the mixup components $h(\mathbf{x}_i)$ and $h(\mathbf{x}_j)$:

$$-\nabla_{h(\mathbf{x}_i)}\mathcal{L}_3 = \lambda\mathbf{V}\left(\boldsymbol{w}_{\hat{y}} - \sum_{k=1}^{|Y_0|+V} a_k\boldsymbol{w}_k\right)$$

$$-\nabla_{h(\mathbf{x}_j)}\mathcal{L}_3 = (1-\lambda)\mathbf{V}\left(\boldsymbol{w}_{\hat{y}} - \sum_{k=1}^{|Y_0|+V} a_k\boldsymbol{w}_k\right).$$

The conclusions are consistent with Eq. 12, which only adds an extra term in the gradient direction, and we can infer that even $g(\cdot)$ is a linear classifier, the forward compatibility is still maintained with $\mathcal{L}_3$.

### 1.2.2 $g(\cdot)$ is Nonlinear

Under a more common scenario, we assume $g(\cdot)$ is nonlinear (*e.g.*, by a neural network block), and we have:

$$a_m = \frac{\exp\left(\boldsymbol{w}_m^\top g(\boldsymbol{b})\right)}{\sum_{k=1}^{|Y_0|+V}\exp\left(\boldsymbol{w}_k^\top g(\boldsymbol{b})\right)}.$$

Similarly, we have the negative gradients w.r.t. the embedding $g(\boldsymbol{b})$:

$$-\nabla_{g(\boldsymbol{b})}\mathcal{L}_3 = \boldsymbol{w}_{\hat{y}} - \sum_{k=1}^{|Y_0|+V} a_k\boldsymbol{w}_k\,. \tag{19}$$

The negative gradient w.r.t. the mixed instance $\boldsymbol{b}$ can be obtained via:

$$\begin{aligned} -\nabla_{\boldsymbol{b}}\mathcal{L}_3 =& -\left(\nabla_{\boldsymbol{b}}\boldsymbol{J}\right)^\top\nabla_{g(\boldsymbol{b})}\mathcal{L}_3 \\ =& \left(\nabla_{\boldsymbol{b}}\boldsymbol{J}\right)^\top\left(\boldsymbol{w}_{\hat{y}} - \sum_{k=1}^{|Y_0|+V} a_k\boldsymbol{w}_k\right), \end{aligned} \tag{20}$$

where $\nabla_{\boldsymbol{b}}\boldsymbol{J}$ is the Jacobian matrix of $g(\boldsymbol{b})$ w.r.t. $\boldsymbol{b}$. Suppose that $\left(\nabla_{\boldsymbol{b}}\boldsymbol{J}\right)^\top\boldsymbol{w}_{\hat{y}}$ are pointing to similar directions (*e.g.*, with a high cosine similarity), then the conclusion above still

holds. Similarly, we have the negative gradient w.r.t. the mixup components $h(\mathbf{x}_i)$ and $h(\mathbf{x}_j)$:

$$-\nabla_{h(\mathbf{x}_i)}\mathcal{L}_3 = \lambda(\nabla_{\boldsymbol{b}}\boldsymbol{J})^\top \left(\boldsymbol{w}_{\hat{y}} - \sum_{k=1}^{|Y_0|+V} a_k \boldsymbol{w}_k\right)$$

$$-\nabla_{h(\mathbf{x}_j)}\mathcal{L}_3 = (1-\lambda)(\nabla_{\boldsymbol{b}}\boldsymbol{J})^\top \left(\boldsymbol{w}_{\hat{y}} - \sum_{k=1}^{|Y_0|+V} a_k \boldsymbol{w}_k\right).$$

The conclusions are consistent with Eq. 12, and we can infer that even $g(\cdot)$ is not a linear classifier, the forward compatibility is still maintained with $\mathcal{L}_3$. To conclude, $\mathcal{L}_3$ enhances the forward compatibility holistically.

## 2. Degradation Form of FACT

In this section, we give another inference form of FACT, which adopts another form of assumption and can be seen as a degradation form of FACT. With a bit of redundancy, we start from the law of total probability:

$$\begin{aligned} p\left(y_i|\phi(\mathbf{x})\right) &= p(\boldsymbol{w}_i|\phi(\mathbf{x})) \\ &= \sum_{\boldsymbol{p}_v \in P_v} p(\boldsymbol{w}_i|\boldsymbol{p}_v, \phi(\mathbf{x}))p(\boldsymbol{p}_v|\phi(\mathbf{x}))\,, \end{aligned} \tag{21}$$

where $p(\boldsymbol{p}_v|\phi(\mathbf{x})) = \frac{\exp\left(\boldsymbol{p}_v{}^\top \phi(\mathbf{x})\right)}{\sum_{\boldsymbol{p}_v \in P_v} \exp(\boldsymbol{p}_v{}^\top \phi(\mathbf{x}))}$. Eq. 21 implies that we can consider the possible influence of all informative virtual prototypes to get the final prediction. We still assume $p\left(\phi(\mathbf{x})|\boldsymbol{w}_i, \boldsymbol{p}_v\right) = \eta\mathcal{N}(\phi(\mathbf{x})|\boldsymbol{w}_i, \Sigma) + (1-\eta)\mathcal{N}(\phi(\mathbf{x})|\boldsymbol{p}_v, \Sigma)$, which follows a Gaussian mixture distribution. According to Bayes' Theorem, we have:

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

where $|\mathcal{Y}_b|$ is the number of classes seen before. In the main paper, we argue that $p(\boldsymbol{w}_i|\boldsymbol{p}_v)$ reflects the similarity between $\boldsymbol{w}_i$ and $\boldsymbol{p}_v$, and assume it follows a Gaussian distribution. However, we can also treat it as the class prior given virtual class, which can be discarded by assuming all classes follow a uniform distribution in few-shot class-incremental learning. Hence, We have:

$$\begin{aligned} p(\boldsymbol{w}_i|\boldsymbol{p}_v, \phi(\mathbf{x})) &= \frac{p(\phi(\mathbf{x})|\boldsymbol{w}_i, \boldsymbol{p}_v)}{\sum_{j=1}^{|\mathcal{Y}_b|} p(\phi(\mathbf{x})|\boldsymbol{w}_j, \boldsymbol{p}_v)} \\ &= \frac{\eta\boldsymbol{n}(\boldsymbol{w}_i) + (1-\eta)\boldsymbol{n}(\boldsymbol{p}_v)}{\eta\sum_{j=1}^{|\mathcal{Y}_b|}\boldsymbol{n}(\boldsymbol{w}_j) + (1-\eta)|\mathcal{Y}_b|\boldsymbol{n}(\boldsymbol{p}_v)}\,, \end{aligned} \tag{22}$$

where $\boldsymbol{n}(\boldsymbol{w}) = \exp\left(\left(\Sigma^{-1}\boldsymbol{w}\right)^\top \phi(\mathbf{x}) - \frac{1}{2}\boldsymbol{w}^\top\Sigma^{-1}\boldsymbol{w}\right)$. When $\boldsymbol{w}$ and $\boldsymbol{p}$ are normalized, $\Sigma = I, \eta = 1$, Eq. 22 degrades into:

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

which means the probability is irrelevant to the virtual prototype $\boldsymbol{p}_v$. Hence, Eq. 21 turns into:

$$\begin{aligned} p\left(y_i|\phi(\mathbf{x})\right) &= \sum_{\boldsymbol{p}_v \in P_v} p(\boldsymbol{w}_i|\boldsymbol{p}_v, \phi(\mathbf{x}))p(\boldsymbol{p}_v|\phi(\mathbf{x})) \\ &= \frac{\exp\left(\boldsymbol{w}_i^\top \phi(\mathbf{x})\right)}{\sum_{j=1}^{|\mathcal{Y}_b|}\exp\left(\boldsymbol{w}_j^\top \phi(\mathbf{x})\right)}\,. \end{aligned} \tag{23}$$

It degrades into ProtoNet (as discussed in Section 3.2 of the main paper), where we only consider the influence of known class prototypes and ignore the possible influence of virtual prototypes. However, in our ablation study (c.f. Section 5.3 of the main paper), we find that when training the model with the same loss function, the inference performance of our FACT is better than ProtoNet. It validates that the classification ability is encoded into these virtual prototypes, which can differentiate among all known classes and help build a stronger classifier.

## 3. Introduction about Compared Methods

In this section, we give a detailed introduction about the compared methods adopted in the main paper. They are listed as:

- **Finetune**: when facing the few-shot incremental session, it simply optimizes the cross-entropy over these few-shot items. It easily suffers catastrophic forgetting.
- **iCaRL** [8]: when training an incremental new task, it combines cross-entropy loss with knowledge distillation loss together. The knowledge distillation part can help the model maintain discrimination ability over former learned knowledge.
- **Pre-Allocated RPC** [6]: it is a class-incremental learning method based on data rehearsal. It first pre-allocates all the classifiers on a regular polytope and then optimizes the embedding of new classes to fit these pre-allocated classifiers. However, since FSCIL tasks do not save exemplars for rehearsal, we can only use the few-shot dataset to optimize the embedding.
- **EEIL** [1]: considers an extra balanced fine-tuning process over iCaRL, which uses a balanced dataset to fine-tune the model and alleviate bias.
- **Rebalancing** [3]: uses cosine normalization, feature-wise knowledge distillation and contrastive learning to augment the model and resist catastrophic forgetting.

Table 1. Comparison with the state-of-the-art on CIFAR100 dataset. We report the results of compared methods from [9] and [12]. FACT **outperforms the runner-up method by 2.96% in terms of the last accuracy, by 1.43% in terms of the performance decay.**

| Method | Accuracy in each session (%) ↑ | | | | | | | | | PD ↓ | Our relative |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | | improvement |
| Finetune | 64.10 | 39.61 | 15.37 | 9.80 | 6.67 | 3.80 | 3.70 | 3.14 | 2.65 | 61.45 | **+38.95** |
| Pre-Allocated RPC [6] | 64.50 | 54.93 | 45.54 | 30.45 | 17.35 | 14.31 | 10.58 | 8.17 | 5.14 | 59.36 | **+36.86** |
| iCaRL [8] | 64.10 | 53.28 | 41.69 | 34.13 | 27.93 | 25.06 | 20.41 | 15.48 | 13.73 | 50.37 | **+27.87** |
| EEIL [1] | 64.10 | 53.11 | 43.71 | 35.15 | 28.96 | 24.98 | 21.01 | 17.26 | 15.85 | 48.25 | **+25.75** |
| Rebalancing [3] | 64.10 | 53.05 | 43.96 | 36.97 | 31.61 | 26.73 | 21.23 | 16.78 | 13.54 | 50.56 | **+28.06** |
| TOPIC [9] | 64.10 | 55.88 | 47.07 | 45.16 | 40.11 | 36.38 | 33.96 | 31.55 | 29.37 | 34.73 | **+12.23** |
| Decoupled-NegCosine [4] | 74.36 | 68.23 | 62.84 | 59.24 | 55.32 | 52.88 | 50.86 | 48.98 | 46.66 | 27.70 | **+5.20** |
| Decoupled-Cosine [10] | 74.55 | 67.43 | 63.63 | 59.55 | 56.11 | 53.80 | 51.68 | 49.67 | 47.68 | 26.87 | **+4.37** |
| Decoupled-DeepEMD [11] | 69.75 | 65.06 | 61.20 | 57.21 | 53.88 | 51.40 | 48.80 | 46.84 | 44.41 | 25.34 | **+2.84** |
| CEC [12] | 73.07 | 68.88 | 65.26 | 61.19 | 58.09 | 55.57 | 53.22 | 51.34 | 49.14 | 23.93 | **+1.43** |
| FACT | **74.60** | **72.09** | **67.56** | **63.52** | **61.38** | **58.36** | **56.28** | **54.24** | **52.10** | **22.50** | |

Table 2. Comparison with the state-of-the-art on *mini*ImageNet dataset. We report the results of compared methods from [9] and [12]. FACT **outperforms the runner-up method by 2.86% in terms of the last accuracy, by 2.30% in terms of the performance decay.**

| Method | Accuracy in each session (%) ↑ | | | | | | | | | PD ↓ | Our relative |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | | improvement |
| Finetune | 61.31 | 27.22 | 16.37 | 6.08 | 2.54 | 1.56 | 1.93 | 2.60 | 1.40 | 59.91 | **+37.84** |
| Pre-Allocated RPC [6] | 61.25 | 31.93 | 18.92 | 13.90 | 14.37 | 15.57 | 16.15 | 12.33 | 12.28 | 48.97 | **+26.90** |
| iCaRL [8] | 61.31 | 46.32 | 42.94 | 37.63 | 30.49 | 24.00 | 20.89 | 18.80 | 17.21 | 44.10 | **+22.03** |
| EEIL [1] | 61.31 | 46.58 | 44.00 | 37.29 | 33.14 | 27.12 | 24.10 | 21.57 | 19.58 | 41.73 | **+19.66** |
| Rebalancing [3] | 61.31 | 47.80 | 39.31 | 31.91 | 25.68 | 21.35 | 18.67 | 17.24 | 14.17 | 47.14 | **+25.07** |
| TOPIC [9] | 61.31 | 50.09 | 45.17 | 41.16 | 37.48 | 35.52 | 32.19 | 29.46 | 24.42 | 36.89 | **+14.82** |
| Decoupled-NegCosine [4] | 71.68 | 66.64 | 62.57 | 58.82 | 55.91 | 52.88 | 49.41 | 47.50 | 45.81 | 25.87 | **+3.80** |
| Decoupled-Cosine [10] | 70.37 | 65.45 | 61.41 | 58.00 | 54.81 | 51.89 | 49.10 | 47.27 | 45.63 | 24.74 | **+2.67** |
| Decoupled-DeepEMD [11] | 69.77 | 64.59 | 60.21 | 56.63 | 53.16 | 50.13 | 47.79 | 45.42 | 43.41 | 26.36 | **+4.29** |
| CEC [12] | 72.00 | 66.83 | 62.97 | 59.43 | 56.70 | 53.73 | 51.19 | 49.24 | 47.63 | 24.37 | **+2.30** |
| FACT | **72.56** | **69.63** | **66.38** | **62.77** | **60.6** | **57.33** | **54.34** | **52.16** | **50.49** | **22.07** | |

- **TOPIC** [9]: tailors few-shot class-incremental learning task with neural gas network. It preserves the topology of the feature manifold formed by different classes.

- **Decoupled-DeepEMD** [11]: decouples the training process of embedding and classifier. After the embedding training process of the base session, it replaces the classifier of each class with the mean embedding of this class. When learning a new session, the same classifier replacement process is adopted for every new class. It adopts a DeepEMD distance [11] calculation between classifier and incoming queries during inference.

- **Decoupled-Cosine** [10]: Similar to Decoupled-DeepEMD, it decouples the training process of embedding and classifier. It adopts a cosine distance [10] calculation during inference.

- **Decoupled-NegCosine** [4]: Similar to Decoupled-Cosine, it decouples the training process of embedding and classifier. The difference between it and Decoupled-Cosine is that it uses a negative margin softmax function during model pretraining. It adopts a cosine distance [10] calculation during inference.

- **CEC** [12]: trains an extra graph model during base session with pseudo-incremental learning sampling. The graph model learns to adapt the embeddings of old class prototypes and new class prototypes, and such ability is generalizable to the incremental learning process.

Note that iCaRL, EEIL, Pre-Allocated RPC, and Rebalancing are traditional class-incremental algorithms. Our empirical experiments in the main paper indicate that these classical class-incremental methods are unsuitable for few-shot class-incremental learning scenarios. For other SOTA

methods of FSCIL, our proposed FACT consistently outperforms them by vast performance measures.

**Discussion about related compatible training methods:** Some other works aim to build a compact embedding space [13], which can be seen as enhancing forward compatibility implicitly. For example, [14] seeks to detect new classes by learning placeholders, [5] utilizes the embedding with large margin between classes, [7] encourages class-wise orthogonality for more compact embedding. These works, however, have a different goal from ours. Their ultimate goal is to obtain a compact embedding, which facilitates the one-stage or in-domain performance in anomaly detection or classification. By contrast, our training scheme aims to build an embedding that enhances performance in the future. In other words, we propose forward compatibility to tailor the characteristics of FSCIL with multiple tasks. Besides, there are other differences between ours and [14]. For example, we use a symmetric loss to balance the learning and reserving process, which is consistent with forward compatibility and proven efficient. Besides, we use the class prototype (average mean) as the classifier, which benefits the embedding learning process in FSCIL. Lastly, the former learned virtual prototypes are utilized during inference to boost forward compatibility, instead of dropped directly. There are other methods addressing virtual classes, *e.g.*, [2]. However, the setting in [2] is different from ours, where extra semantic information are available to synthesis new classes directly.

Pre-Allocated RPC and Decoupled-NegCosine can be viewed as encouraging forward compatibility. The former pre-assigns the classifier for new classes but lacks the ability for classifier matching in FSCIL process. The latter considers preventing new class embedding from being harmed by using a negative margin. However, they are validated ineffective in the FSCIL setting, verifying the effectiveness of our prospective training paradigm.

## 4. Detailed Incremental Performance

In the main paper, we show the incremental performance on benchmark datasets, and report the detailed accuracy of CUB200. We report the incremental performance on the other benchmark datasets, *i.e.*, CIFAR100 and *mini*ImageNet in Table 1 and Table 2. We can infer that our proposed FACT has the better top-1 accuracy and lower performance decay, indicating FACT forgets lower than other state-of-the-art methods. These conclusions are consistent with the main paper, verifying the best performance of FACT.

**Algorithm 1** Forward Compatible Training for FSCIL

**Input**: Base dataset: $\mathcal{D}^0$, Virtual class number: $V$;
**Output**: $W, P_v, \phi(\cdot)$;

Randomly initialize $W, P_v, \phi(\cdot)$;
**repeat**
  Get a mini-batch of training instances: $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$;
  Calculate the virtual loss $\mathcal{L}_v$;
  Randomly shuffle the dataset, denoted as $\{(\mathbf{x}_j, y_j)\}_{j=1}^n$;
  Mask out same class instances in $(\mathbf{x}_i, \mathbf{x}_j)$;
  Calculate the forecasting loss $\mathcal{L}_f$;
  Get the total loss $\mathcal{L} = \mathcal{L}_v + \mathcal{L}_f$;
  Obtain derivative and update the model;
**until** reaches predefined epoches

## References


[1] 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. 4, 5

[2] Yu-Ying Chou, Hsuan-Tien Lin, and Tyng-Luh Liu. Adaptive and generative zero-shot learning. In *ICLR*, 2020. 6

[3] 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. 4, 5

[4] 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. 5

[5] Weiyang Liu, Yandong Wen, Zhiding Yu, and Meng Yang. Large-margin softmax loss for convolutional neural networks. In *ICML*, 2016. 6

[6] 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. 4, 5

[7] Kanchana Ranasinghe, Muzammal Naseer, Munawar Hayat, Salman Khan, and Fahad Shahbaz Khan. Orthogonal projection loss. In *ICCV*, pages 12333–12343, 2021. 6

[8] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *CVPR*, pages 2001–2010, 2017. 4, 5

[9] Xiaoyu Tao, Xiaopeng Hong, Xinyuan Chang, Songlin Dong, Xing Wei, and Yihong Gong. Few-shot class-incremental learning. In *CVPR*, pages 12183–12192, 2020. 5

[10] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. *NIPS*, 29:3630–3638, 2016. 5

[11] 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. 5

[12] 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. 5

[13] Da-Wei Zhou, Han-Jia Ye, and De-Chuan Zhan. Co-transport for class-incremental learning. In *ACM MM*, pages 1645–1654, 2021. 6

[14] Da-Wei Zhou, Han-Jia Ye, and De-Chuan Zhan. Learning placeholders for open-set recognition. In *CVPR*, pages 4401–4410, 2021. 6