Feature Directions Matter: Long-Tailed Learning via Rotated Balanced Representation

Gao Peifeng [1] **Qianqian Xu** [2] **Peisong Wen** [1 2] **Zhiyong Yang** [3 4] **Huiyang Shao** [1 2] **Qingming Huang** [1 2 5 6]

[1]School of Computer Science and Technology, UCAS, Beijing, China. [2]Key Laboratory of Intelligent Information Processing, Inst. of Comput. Tech., CAS, Beijing, China. [3]State Key Laboratory of Info. Security (SKLOIS), Inst. of Info. Engin., CAS, Beijing, China. [4]School of Cyber Security, UCAS, Beijing, China. [5]BDKM, UCAS, Beijing, China. [6]Peng Cheng Laboratory, Shenzhen, China. Correspondence to: Qianqian Xu <xuqianqian@ict.ac.cn>, Qingming Huang <qmhuang@ucas.ac.cn>.

ICML 2023

2023.9.14

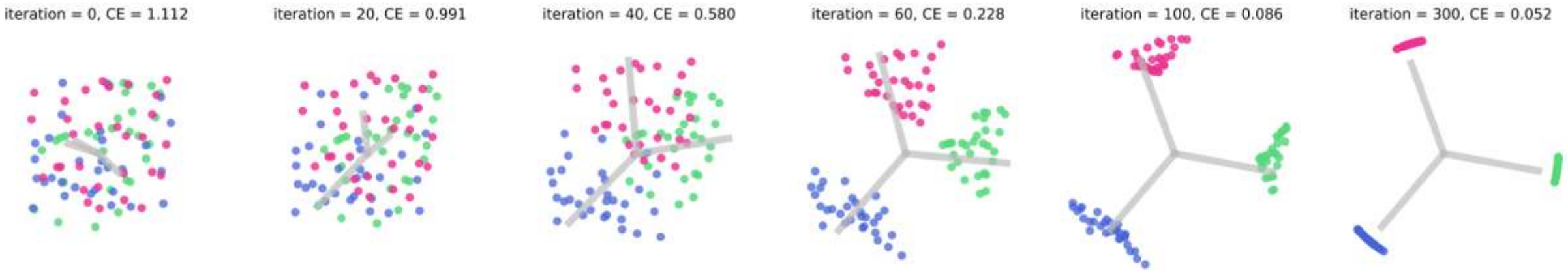
Neural Collapse

NC1 Variability Collapse All samples belonging to the same class converge to the class mean: $\|f(\boldsymbol{x}_{y,i}) - \overline{f(\boldsymbol{x}_y)}\| \to 0, \forall y, \forall i$ where $\overline{f(\boldsymbol{x}_y)} = \text{Ave}_i\left(f(\boldsymbol{x}_{y,i})\right)$ denotes the class-center of y-th class;

NC2 Convergence to Self Duality The samples and classifier belonging to the same class converge to the same: $\|f(\boldsymbol{x}_{y,i}) - M_y\| \to 0, \forall y, \forall i$;

NC3 Convergence to Simplex ETF The classifier weight converges to the vertices of Simplex Equiangular Tight Frame (ETF);

NC4 Nearest Classification The learned classifier behaves like the nearest classifier, *i.e.*, given any sample $\boldsymbol{x}$ in dataset, $\arg\max_y \langle M_y, f(\boldsymbol{x})\rangle \to \arg\min_y \|f(\boldsymbol{x}) - \overline{f(\boldsymbol{x}_y)}\|$.



(a) The *NeurCol* phenomenon. There are 30 samples for each class. A GIF animation can be found HERE.

Simplex ETF is a symmetric structure where points lie on a hypersphere, is linearly separable, and the distances between them are maximized.

Definition 1 (Simplex Equiangular Tight Frame) *A collection of vectors* $\mathbf{m}_i \in \mathbb{R}^d$, $i = 1, 2, \cdots, K$, $d \geq K-1$, *is said to be a simplex equiangular tight frame if:*

$$\mathbf{M} = \sqrt{\frac{K}{K-1}}\mathbf{U}\left(\mathbf{I}_K - \frac{1}{K}\mathbf{1}_K\mathbf{1}_K^T\right), \tag{1}$$

where $\mathbf{M} = [\mathbf{m}_1, \cdots, \mathbf{m}_K] \in \mathbb{R}^{d\times K}$, $\mathbf{U} \in \mathbb{R}^{d\times K}$ *allows a rotation and satisfies* $\mathbf{U}^T\mathbf{U} = \mathbf{I}_K$, $\mathbf{I}_K$ *is the identity matrix, and* $\mathbf{1}_K$ *is an all-ones vector.*

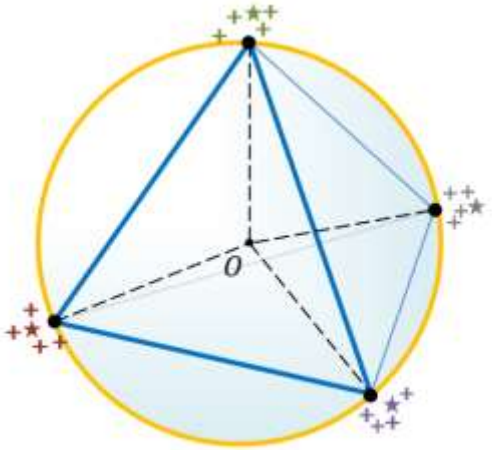


Figure 1: An illustration of a simplex equiangular tight frame when $d = 3$ and $K = 4$. The black spheres are the vertices of the ETF. The "+" and "⋆" signs in different colors refer to features and classifier vectors of different classes, respectively. Neural collapse indicates that the features and classifier vectors are aligned with the same simplex ETF.

Inducing Neural Collapse in Imbalanced Learning: Do We Really Need a Learnable Classifier at the End of Deep Neural Network?

Yibo Yang[1], Shixiang Chen[1], Xiangtai Li[2], Liang Xie[3], Zhouchen Lin[2,4,5*], Dacheng Tao[1]

[1]JD Explore Academy, Beijing, China
[2]Key Lab. of Machine Perception (MoE), School of Intelligence Science and Technology, Peking University
[3]State Key Lab of CAD&CG, Zhejiang University
[4]Institute for Artificial Intelligence, Peking University
[5]Pazhou Laboratory, Guangzhou, China

Simplex ETF is a symmetric structure where points lie on a hypersphere, is linearly separable, and the distances between them are maximized.

$$\mathbf{M} = \sqrt{\frac{K}{K-1}}\mathbf{U}\left(\mathbf{I}_K - \frac{1}{K}\mathbf{1}_K\mathbf{1}_K^T\right),$$

Definition 2.1 (Simplex Equiangular Tight Frame (Papyan et al., 2020)**).** A Simplex ETF is a collection of points in $\mathbb{R}^C$ specified by the columns of

$$M^\star = \alpha R\sqrt{\frac{C}{C-1}}\left(I - \frac{1}{C}\mathbb{I}\mathbb{I}^T\right)$$

where $I \in \mathbb{R}^{C\times C}$ is the identity matrix, $\mathbb{I} \in \mathbb{R}^C$ is the all-one vector, $R \in \mathbb{R}^{d\times C}(d \geq C)$ is an orthogonal projection matrix, $\alpha \in \mathbb{R}$ is a scale factor.

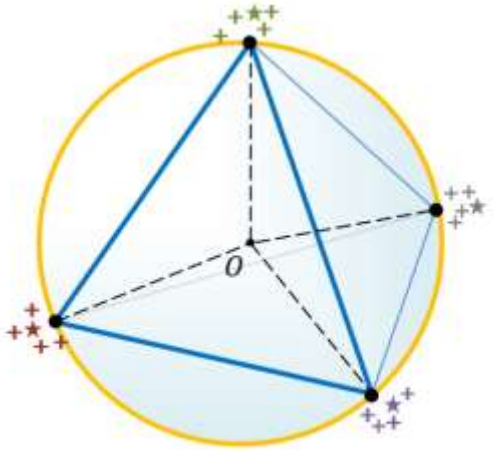


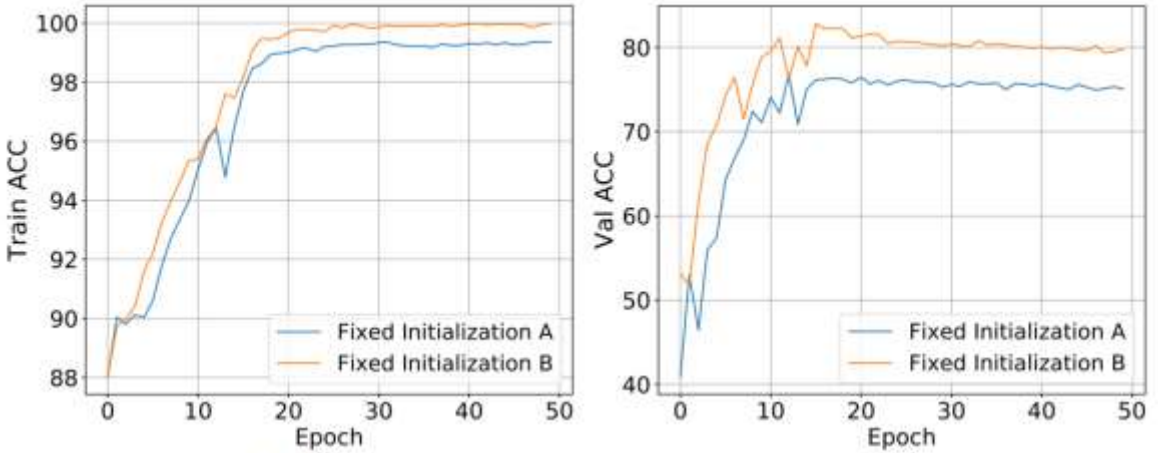
Figure 1: An illustration of a simplex equiangular tight frame when $d = 3$ and $K = 4$. The black spheres are the vertices of the ETF. The "+" and "$\star$" signs in different colors refer to features and classifier vectors of different classes, respectively. Neural collapse indicates that the features and classifier vectors are aligned with the same simplex ETF.

Motivation

ing. Recent work proposes to learn the balanced representation by fixing the linear classifier as *Equiangular Tight Frame* (ETF), since they argue what matters in classification is the structure of the feature, instead of their directions. Hold-

pletely symmetric. To avoid this issue, we propose *Representation-Balanced Learning* Framework (RBL), which introduces orthogonal matrices to learn directions while maintaining the geometric structure of ETF. Theoretically, our

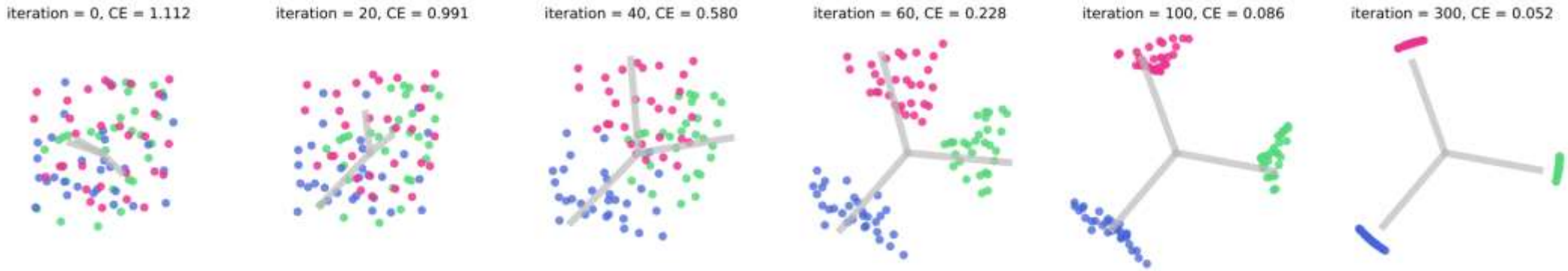
Concretely, we initialize the linear classifier $\mathbf{W}$ as a random simplex ETF by Eq. (1)



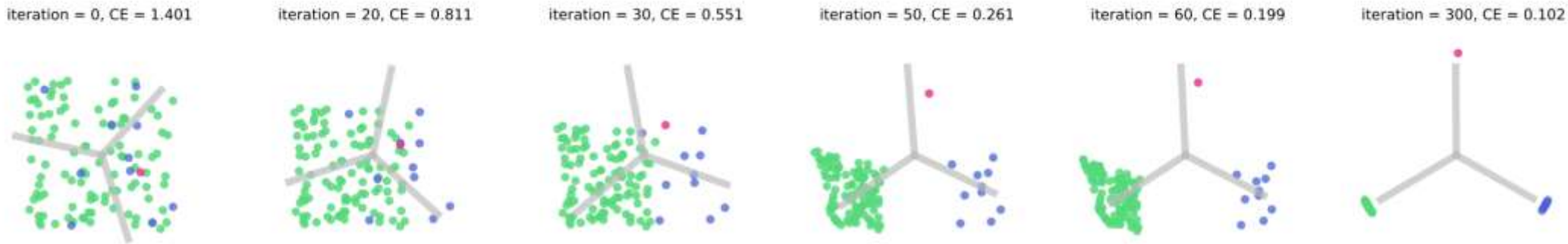
(a) The training and validation accuracies.

(b) Features visualization on training set at epoch 30. Left: initialization A. Right: initialization B.

Figure 1: Two toy experiments to illustrate feature learning and generalization of *Fixed*. A two-layer deep model and a linear classifier are trained to solve a long-tailed classification problem with 2-dimensional feature and 3 classes. In (b), points and lines indicate sample feature and model weight respectively.



(a) The *NeurCol* phenomenon. There are 30 samples for each class. A GIF animation can be found HERE.



(b) The feature learning of our framework. Each class has $100, 10, 1$ samples respectively. A GIF animation can be found HERE.

Figure 2: Two numerical simulation experiments to illustrate the feature learning of classification. In experiments, a linear classifier was trained to solve a classification problem with 2-dimensional feature and 3 categories. To simulate the model with infinite fitting ability, we directly update the features in $\mathbb{R}^2$. The pictures from left to right record the location of the model weight and sample feature in $\mathbb{R}^2$ during the optimization process. In each picture, points and lines indicate sample feature and model weight respectively. Implementation details of experiments in this figure could be found in Appendices.

Definition 2.1 (**Simplex Equiangular Tight Frame** (Papyan et al., 2020)). A Simplex ETF is a collection of points in $\mathbb{R}^C$ specified by the columns of

$$M^\star = \alpha R \sqrt{\frac{C}{C-1}} \left(I - \frac{1}{C} \mathbb{I}\mathbb{I}^T \right)$$

where $I \in \mathbb{R}^{C \times C}$ is the identity matrix, $\mathbb{I} \in \mathbb{R}^C$ is the all-one vector, $R \in \mathbb{R}^{d \times C} (d \geq C)$ is an orthogonal projection matrix, $\alpha \in \mathbb{R}$ is a scale factor.

Definition 3.1 (**Trivial Equiangular Tight Frame**). A trivial ETF is a collection of points in $\mathbb{R}^C$

$$M^\star = \sqrt{\frac{C}{C-1}} \left(I - \frac{1}{C} \mathbb{I}\mathbb{I}^T \right)$$

where I is the identity matrix and $\mathbb{I}$ is the ones vector.

Obviously, a C-dimensional trivial ETF can be seen as a Simplex ETF that has C vectors in $\mathbb{R}^C$. In classification problems, if the number of classes C is smaller than feature dimension d, we select C vectors in d-dimensional trivial ETF as the balanced feature. It is reasonable because the subset of ETF still meets *equiangular* condition. In another case that $C > d$, we directly generate trivial ETF in $\mathbb{R}^C$. When performing feature learning, we use linear transformation to transform the feature dimension d into C. In this way, we obtain features that satisfy *equiangular* property for any number of classes and feature dimension.

3.2. Representation-Balanced Learning

$$\begin{aligned} \min_{\boldsymbol{w},R} \quad & \mathcal{L}(\boldsymbol{w},R,S) := -\frac{1}{N}\sum_{i=1}^{N}\log\frac{\exp([\text{logit}(\boldsymbol{x})]_{y_i})}{\sum_{y=1}^{C}\exp([\text{logit}(\boldsymbol{x})]_{y})} \\ s.t. \quad & \text{logit}(\boldsymbol{x}_i) = M^{\star}Rf(\boldsymbol{x}_i;\boldsymbol{w}) \end{aligned} \tag{1}$$

where $M^{\star}$ is the balanced feature we generate in advance and R is the orthogonal matrix. According to the number of categories C and feature dimension d, the sizes of $M^{\star}$ and R are designed specifically. If $C \leq d$, $M^{\star}$ is a $C \times d$ matrix and R is $d \times d$ orthogonal matrix, where every row of $M^{\star}$ is the vector in d-dimensional trivial ETF. When $C > d$, we generate C-dimensional trivial ETF as $M^{\star}$ and let R be the $C \times d$ orthogonal projection matrix. The R is for preservation of *equiangular* of M^{*}. In this way, $f(\boldsymbol{x};\boldsymbol{w})$ is no longer only learning from M^{*}, but learning from M^{*}'s direction.

Post-Hoc Logit Adjustment In the analysis of the next section, we would prove that optimization framework (1) could lead to *NeurCol* even in the long-tailed scenarios. However, feature learning of our framework can only learn balanced features, and it still requires proper decision-making for full play to its abilities (Kang et al., 2019). To this end, we perform Logit Adjustment when the model performs classification.

$$\arg\max_{i \in \mathcal{Y}} \left[M^{\star} R f(\boldsymbol{x}; \boldsymbol{w}) - margin \right]_i \tag{2}$$

Here, we follow the configuration of Balanced Softmax (Ren et al., 2020), set *margin* as $[\log(N_1/N), \ldots, \log(N_C/N)]^T$, where N_i is the number of samples in category i of training set.

3.3. Optimization in Lie Group optimize an orthogonal matrix.

(1) 旋转矩阵 - 特殊正交群 $\mathrm{SO(n)}$

旋转矩阵（定义为R）描述了旋转前后同一个向量的坐标变换关系。旋转矩阵是一个行列式为1的正交矩阵，反之，行列式为1的正交矩阵也是一个旋转矩阵。我们把旋转矩阵的集合称为**特殊正交群（Special Orthogonal Group）**，定义为:

$$\mathrm{SO(n)} = \{\mathrm{R} \in \mathbb{R}^{\mathrm{n}\times\mathrm{n}} | \mathrm{RR}^{\mathrm{T}} = \mathrm{I}, \det(\mathrm{R}) = 1\} \tag{1-1}$$

旋转矩阵还有一些正交矩阵的特殊性质，如$\mathrm{R}^{-1} = \mathrm{R}^{\mathrm{T}}$等，旋转矩阵的逆或转置描述了一个相反的旋转。

1.3 李群（Lie Group）

李群是指具有连续（光滑）性质的群，是群也是流形。刚体在空间中的运动是连续的，用于描述该运动的$\mathrm{SO(3)}$和$\mathrm{SE(3)}$都是李群；整数群$\mathbb{Z}$由于没有连续性质（是离散的），因此不是李群。

optimize an orthogonal matrix. Suppose we need a matrix that lies in $SO(d)$ to represent rotation. $SO(d)$ is the special orthogonal group, *i.e.*, Lie Group

$$SO(d) = \{A \in \mathbb{R}^{d\times d} | A^T A = I, \det A = 1\}$$

Consider the Lie Algebra $\mathfrak{so}(d)$ formed by skew-matrices

$$\mathfrak{so}(d) = \{A \in \mathbb{R}^{d\times d} | A + A^T = 0\}$$

is a homomorphism of Lie Group $SO(d)$. The mapping exponential of matrix $\exp(\cdot)$ is defined as

$$\exp(A) = I + A + \frac{A^2}{2} + \dots$$

Therefore, the optimization in $SO(d)$ could be transformed into optimization in $\mathfrak{so}(d)$:

$$\min_{A\in SO(d)} \text{loss}(A) \overset{A=\exp\{B\}}{\Longleftrightarrow} \min_{B\in\mathfrak{so}(d)} \text{loss}(\exp\{B\}) \quad (3)$$

$\exp(B)$. Furthermore, the Lie Algebra $\mathfrak{so}(d)$ is isomorphic to a linear space. The isomorphism mapping is given by $\phi(A) : A \mapsto A - A^T$. Consequently, the constraint of $SO(d)$ could be eliminated.

$$\min_{A\in SO(d)} \text{loss}(A) \overset{A=\exp(B-B^T)}{\Longleftrightarrow} \min_{B\in\mathbb{R}^{d\times d}} \text{loss}(\exp(B - B^T)) \quad (4)$$

In the above formulation, the optimization with orthogonal constraint is transformed into the optimization in $\mathbb{R}^{d\times d}$. For the right side of (4), we could use standard optimization techniques such as SGD and Adam.

```python
class RBL(nn.Module):
    r"""
    Args:
        backbone (nn.Module) : deep model for feature
        feature_num (int) : backbone's feature dimension
        class_num (int) : the number of classes
        _cls_num_list (list) : numbers of sample in each class

    Examples:
        >>> import torchvision.models as models
        >>> feature_dim = 512
        >>> class_dim = 1000
        >>> resnet18 = models.resnet18(num_classes=feature_dim).cuda()
        >>> model = RBL(resnet18, feature_dim, class_dim, torch.arange(class_dim, 0, -1)).cuda()
        >>> pred = model(torch.randn(1, 3, 224, 224).cuda())
        >>> print(pred.shape)
        torch.Size([1, 1000])
    """
    def __init__(self, backbone, feature_num, class_num, _cls_num_list):
        super(RBL, self).__init__()
        self.feature_num = feature_num
        self.class_num = class_num
        self.backbone = backbone
        self.margin = torch.log(torch.Tensor(_cls_num_list) / sum(_cls_num_list)).cuda()

        if feature_num < class_num:
            self.rotate = nn.Linear(class_num, feature_num, bias=False)
            self.register_buffer("ETF", self.generate_ETF(dim=class_num))
        else:
            self.rotate = nn.Linear(feature_num, feature_num, bias=False)
            self.register_buffer("ETF", \
                self.generate_ETF(dim=feature_num)[:, :self.class_num])
        geotorch.orthogonal(self.rotate, "weight")

    def generate_ETF(self, dim):
        return torch.eye(dim, dim) - torch.ones(dim, dim) / dim

    def forward(self, x):
        logit = self.backbone(x) @ self.rotate.weight @ self.ETF
        return logit if self.training else logit - self.margin
```

Code 1: PyTorch implementation of our framework using `geotorch` library (Lezcano-Casado, 2019).

```
class PLPostHocModel(nn.Module):
def __init__(self, backbone, triv, feature_num, class_num, _cls_num_list):
    super(PLPostHocModel, self).__init__()
    self.feature_num = feature_num
    self.class_num = class_num
    self.backbone = backbone
    _cls_num_list = torch.Tensor(_cls_num_list)
    self.margin = torch.log(_cls_num_list / torch.sum(_cls_num_list)).cuda()

    if feature_num < class_num:
        self.register_buffer("ETF", self.generate_ETF(dim=class_num))
        self.rotate = nn.Linear(class_num, class_num, bias=False)
    else:
        self.register_buffer("ETF", self.generate_ETF(dim=feature_num)\
            [:, :self.class_num])
        self.rotate = nn.Linear(feature_num, feature_num, bias=False)

def generate_ETF(self, dim):
    return torch.eye(dim, dim) - torch.ones(dim, dim) / dim

def encode_rotate(self):
    if self.feature_num < self.class_num:
        return torch.linalg.matrix_exp(self.rotate.weight - self.rotate.weight.T)\
            [:self.feature_num, :]
    return torch.linalg.matrix_exp(self.rotate.weight - self.rotate.weight.T)

def forward(self, x):
    logit = self.backbone(x) @ self.encode_rotate() @ self.ETF
    return logit if self.training else logit - self.margin

def forward_feature(self, x):
    return self.backbone(x)
```

Code 2: PyTorch implementation of our framework without third-party libraries.

Table 1: Test accuracies on CIFAR10/100-LT. The best and second best results are marked as **bold** and underline. Rows with † denote results borrowed from (Wang et al., 2021c). Results of other competitors are taken from original papers.

Method	CIFAR-10			CIFAR-100		
	50	100	200	50	100	200
CB	79.3	74.6	68.9	45.3	39.6	36.2
LADE	-	-	-	50.5	45.4	-
Calibrated	84.3	82.8	78.5	51.1	45.5	42.1
cRT†	-	82.0	76.6	-	50.0	44.5
LWS†	-	83.7	78.1	-	50.5	45.3
BS†	-	83.1	79.0	-	50.3	45.9
MARC	-	**85.3**	81.1	-	50.8	47.4
HCL	85.4	81.4	-	51.9	46.7	-
TSC	82.9	79.7	-	47.4	43.8	-
Fixed	87.1	84.0	80.2	56.2	52.3	47.2
RBL	**87.6**	84.7	**81.2**	**57.2**	**53.1**	**48.9**

Table 2: Test accuracies on ImageNet-LT. The best and second best results are marked as **bold** and underline. Rows with † denote results borrowed from (Wang et al., 2021c). Results of other competitors are taken from original papers.

Method	Many	Medium	Few	All
Calibrated	-	-	-	48.4
cRT	61.8	46.2	27.4	49.6
LWS	60.2	47.2	30.3	49.9
Seesaw	**67.1**	45.2	21.4	50.4
BS†	62.2	48.8	29.8	51.4
MARC	60.4	50.3	**36.6**	52.3
LADE	65.1	48.9	33.4	53.0
KCL	61.8	49.4	30.9	51.5
TSC	63.5	**49.7**	30.4	52.4
Fixed	64.3	47.6	27.2	51.2
RBL	64.8	49.6	34.2	**53.3**

Table 3: Test accuracies on Places-LT. The best and second best results are marked as **bold** and underline. Results of other competitors are taken from original papers.

Method	Many	Medium	Few	All
NCM	40.4	37.1	27.3	36.4
cRT	42.0	37.6	24.9	36.7
LWS	40.6	39.1	28.6	37.6
τ-norm	37.8	**40.7**	31.8	37.9
Marc	39.9	39.8	**32.6**	38.4
BS	41.2	39.8	31.6	38.7
LADE	42.8	39.0	31.2	**38.8**
Fixed	43.7	39.7	23.9	38.0
RBL	**44.1**	**40.7**	24.4	**38.8**

Table 4: Ablation study on CIFAR100-LT measured by test accuracies.

Method	CIFAR100-LT		
	200	100	50
CE (Baseline)	42.7	46.7	51.8
Fixed Direction	41.7	46.5	50.5
Learnable Direction	43.4	47.7	52.9
LD	46.6	51.4	55.1
Fixed Direction + LD	47.2	52.3	56.2
Learnable Direction + LD (RBL)	**48.9**	**53.1**	**57.2**

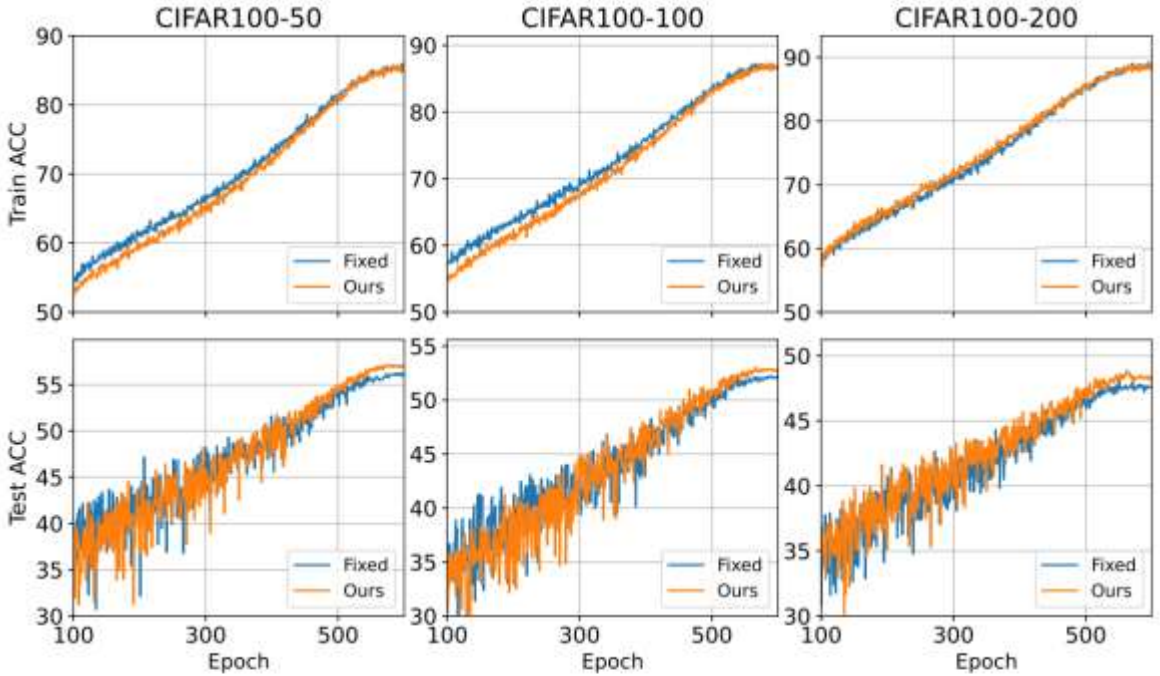


Figure 3: Generalization analysis on CIFAR100. The two rows show the accuracies of *Fixed* and our method on training set and test set in every epoch respectively.

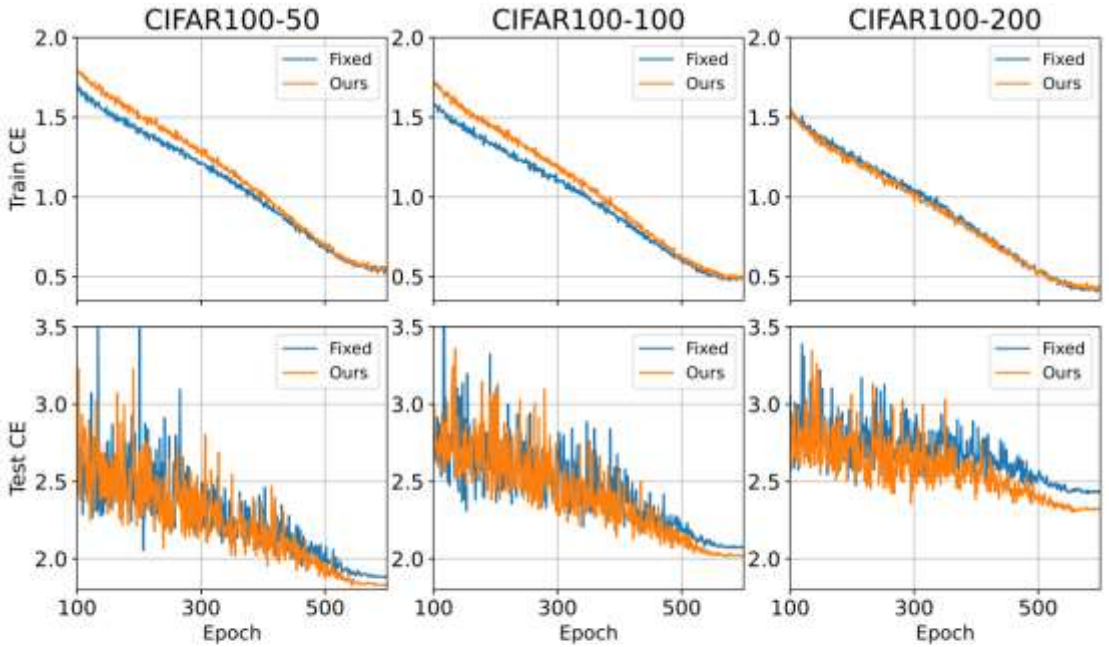


Figure 4: Generalization analysis on CIFAR100. The two rows show the cross entropy loss of *Fixed* and our method on training set and test set in every epoch respectively.

NEURAL COLLAPSE INSPIRED FEATURE-CLASSIFIER ALIGNMENT FOR FEW-SHOT CLASS INCREMENTAL LEARNING

Yibo Yang[1*†], **Haobo Yuan**[2*], **Xiangtai Li**[3], **Zhouchen Lin**[3,4,5†], **Philip Torr**[6], **Dacheng Tao**[1]

[1]JD Explore Academy [2]School of Computer Science, Wuhan University
[3]National Key Lab. of General Artificial Intelligence, School of Intelligence Science and Technology, Peking University
[4]Institute for Artificial Intelligence, Peking University [5]Peng Cheng Laboratory [6]University of Oxford
†: corresponding authors; *: equal contribution

ICLR 2023

3.1 FEW-SHOT CLASS-INCREMENTAL LEARNING (FSCIL)

In real-world applications, one often needs to adapt a model to data coming from a new label space with only a few labeled samples. FSCIL trains a model incrementally on a sequence of training datasets $\{\mathcal{D}^{(0)}, \mathcal{D}^{(1)}, \ldots, \mathcal{D}^{(T)}\}$, where $\mathcal{D}^{(t)} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{|\mathcal{D}^{(t)}|}$, $\mathcal{D}^{(0)}$ is the base session, and T the number of incremental sessions. The base session $\mathcal{D}^{(0)}$ usually contains a large label space $\mathcal{C}^{(0)}$ and sufficient training images for each class $c \in \mathcal{C}^{(0)}$. In each incremental session $\mathcal{D}^{(t)}$, $t > 0$, there are only a few labeled images and we have $|\mathcal{D}^{(t)}| = pq$, where p is the number of classes and q is the number samples per novel class, known as p-way q-shot. The label space $\mathcal{C}^{(t)}$ has no overlap with any other session, *i.e.,* $\mathcal{C}^{(t)} \cap \mathcal{C}^{(t')} = \emptyset$, $\forall t' \neq t$. For any incremental session $t > 0$, we only have access to the data in $\mathcal{D}^{(t)}$, and the training sets of the previous sessions are not available. For evaluation in session t, the test dataset comes from all the encountered classes in the previous and current sessions [1], *i.e.* the label space of $\cup_{i=0}^{t} \mathcal{C}^{(i)}$.

Therefore, FSCIL suffers from severe data scarcity and imbalance. It requires a model to be adaptable to novel classes, and meanwhile keep the ability on old classes.

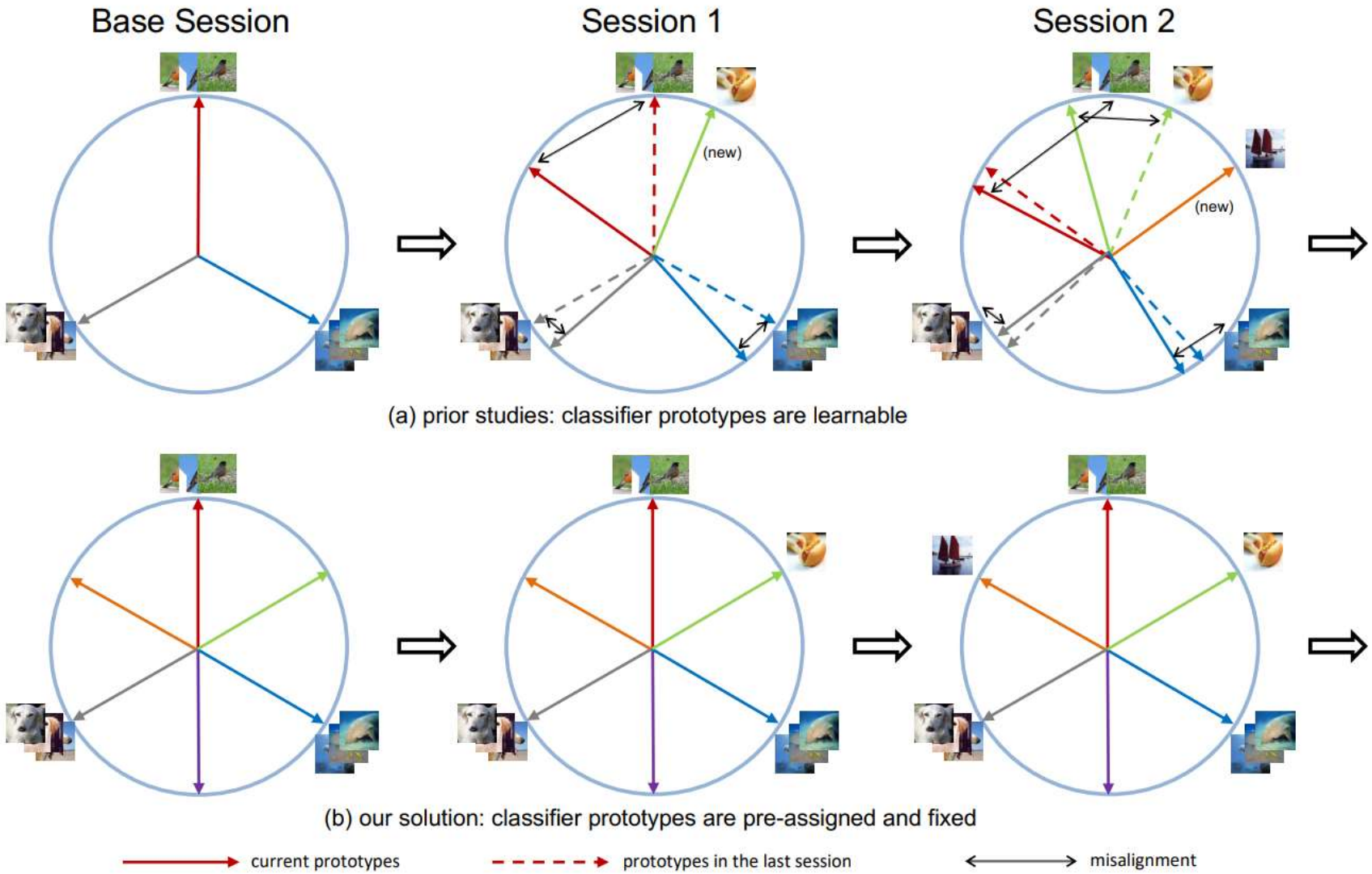


Figure 1: A popular choice in prior studies is to evolve the old-class prototypes via delicate design of loss or regularizer to keep them separated from novel-class prototypes, but will cause misalignment. As a comparison, we pre-assign and fix an optimal feature-classifier alignment, and then train a model towards the same neural collapse optimality in each session to avoid target conflict.

4.1 ETF CLASSIFIER

Assume that the base session contains a label space of K_0 classes, each incremental session has p classes, and we have T incremental sessions in total. The whole label space of this FSCIL problem has $K_0 + K'$ classes, where $K' = Tp$, *i.e.,* we need to learn a model that can recognize samples from $K_0 + K'$ classes. We denote a backbone network as f, and then we have $\boldsymbol{\mu} = f(\boldsymbol{x}, \theta_f)$, where $\boldsymbol{\mu} \in \mathbb{R}^d$ is the output feature of input $\boldsymbol{x}$, and θ_f is the backbone network parameters.

Definition 1 (Simplex Equiangular Tight Frame) *A simplex Equiangular Tight Frame (ETF)*

All column vectors in $\mathbf{E}$ *have the same* ℓ_2 *norm and any pair has an inner produce of* $-\frac{1}{K-1}$, i.e.,

$$\mathbf{e}_{k_1}^T \mathbf{e}_{k_2} = \frac{K}{K-1}\delta_{k_1,k_2} - \frac{1}{K-1}, \quad \forall k_1, k_2 \in [1, K], \tag{2}$$

where $\delta_{k_1,k_2} = 1$ *when* $k_1 = k_2$*, and 0 otherwise.*

Since neural collapse describes an optimal geometric structure of the last-layer feature and classifier, we pre-assign such an optimality by fixing a learnable classifier as the structure instructed by neural collapse. Following Yang et al. (2022b), we adopt an ETF classifier that initializes a classifier as a simplex ETF and fixes it during training. The difference lies in that we use it in an incremental fashion. Concretely, we randomly initialize classifier prototypes $\hat{\mathbf{W}}_{\mathrm{ETF}} \in \mathbb{R}^{d\times(K_0+K')}$ by Eq. (1) for the whole label space, *i.e.,* the union of classes in all session, $\cup_{i=0}^{T}\mathcal{C}^{(i)}$. We have $K_0 = |\mathcal{C}^{(0)}|$ and $K' = \sum_{i=1}^{T}|\mathcal{C}^{(i)}| = Tp$. Then any pair (k_1, k_2) of classifier prototypes in $\hat{\mathbf{W}}_{\mathrm{ETF}}$ satisfies:

$$\hat{\mathbf{w}}_{k_1}^T \hat{\mathbf{w}}_{k_2} = \frac{K_0 + K'}{K_0 + K' - 1}\delta_{k_1,k_2} - \frac{1}{K_0 + K' - 1}, \quad \forall k_1, k_2 \in [1, K_0 + K'], \tag{3}$$

where $\hat{\mathbf{w}}_{k_1}$ and $\hat{\mathbf{w}}_{k_2}$ are two column vectors in $\hat{\mathbf{W}}_{\mathrm{ETF}}$. Our ETF classifier ensures that the prototypes

4.2 DOT-REGRESSION LOSS

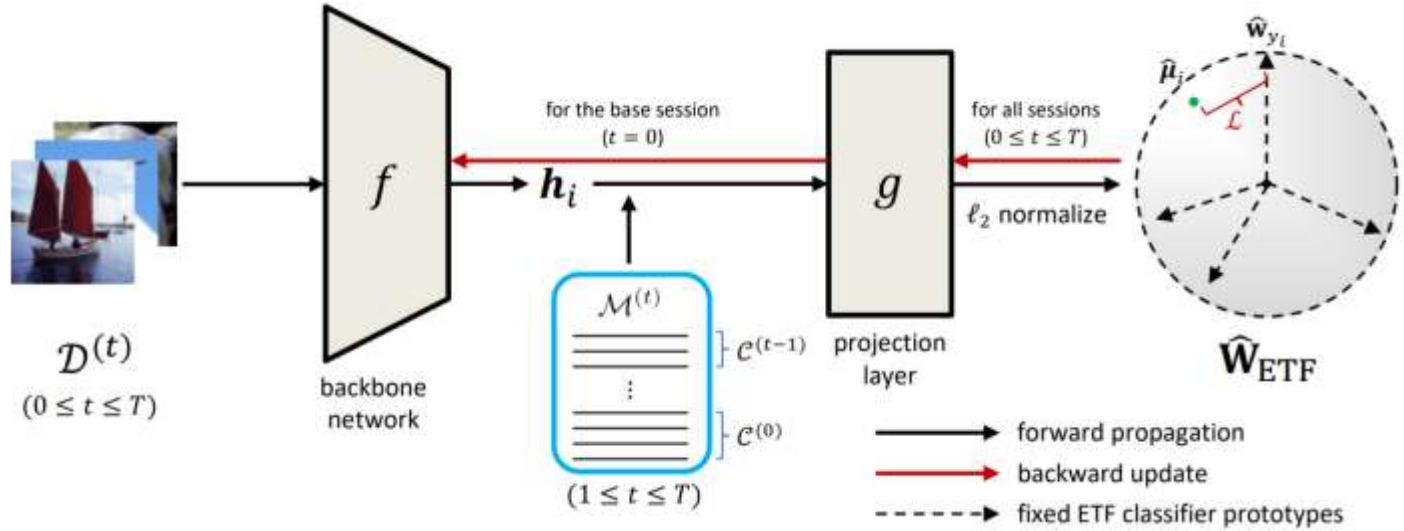
solution, and we can drop the `push` gradient that may be inaccurate. Accordingly, we adopt a novel loss named dot-regression (DR) loss that can be formulated as (Yang et al., 2022b):

$$\mathcal{L}\left(\hat{\boldsymbol{\mu}}_i, \hat{\mathbf{W}}_{\mathrm{ETF}}\right) = \frac{1}{2}\left(\hat{\mathbf{w}}_{y_i}^T \hat{\boldsymbol{\mu}}_i - 1\right)^2, \tag{4}$$

where $\hat{\boldsymbol{\mu}}_i$ is the normalized feature, *i.e.*, $\hat{\boldsymbol{\mu}}_i = \boldsymbol{\mu}_i / \|\boldsymbol{\mu}_i\|$, $\boldsymbol{\mu}_i = f(\boldsymbol{x}_i, \theta_f)$, y_i is the label of input $\boldsymbol{x}_i$, $\hat{\mathbf{w}}_{y_i}$ is the fixed prototype in $\hat{\mathbf{W}}_{\mathrm{ETF}}$ for class y_i, and we have $\|\hat{\mathbf{w}}_{y_i}\| = 1$ by Eq. (3). The total loss is an average over a batch of input $\boldsymbol{x}_i$. The gradient of Eq. (4) with respect to $\hat{\boldsymbol{\mu}}_i$ takes the form of:

$\partial\mathcal{L}/\partial\hat{\boldsymbol{\mu}}_i = -(1 - \cos\angle(\hat{\boldsymbol{\mu}}_i, \hat{\mathbf{w}}_{y_i}))\hat{\mathbf{w}}_{y_i}$. It is shown that the gradient pulls feature $\hat{\boldsymbol{\mu}}_i$ towards the direction of $\hat{\mathbf{w}}_{y_i}$, which is a pre-assigned target prototype. Finally, the converged features will be aligned with $\hat{\mathbf{W}}_{\mathrm{ETF}}$, and thus the geometric structure instructed by neural collapse is attained. The

4.3 NC-FSCIL



(2022). It projects the intermediate feature $\boldsymbol{h}_i$ into $\boldsymbol{\mu}_i$. Finally, we perform an ℓ_2 normalization on $\boldsymbol{\mu}_i$ to get the output feature $\hat{\boldsymbol{\mu}}_i$, *i.e.*,

$$\hat{\boldsymbol{\mu}}_i = \boldsymbol{\mu}_i / \|\boldsymbol{\mu}_i\|, \quad \boldsymbol{\mu}_i = g(\boldsymbol{h}_i, \theta_g), \quad \boldsymbol{h}_i = f(\boldsymbol{x}_i, \theta_f), \tag{5}$$

where θ_f and θ_g denote the parameters of the backbone network and the projection layer, respectively. We use the normalized output feature $\hat{\boldsymbol{\mu}}_i$ to compute error signal by Eq. (4).

In the base session $t = 0$, we jointly train both f and g using the base session data. The empirical risk to minimize in the base session can be formulated as:

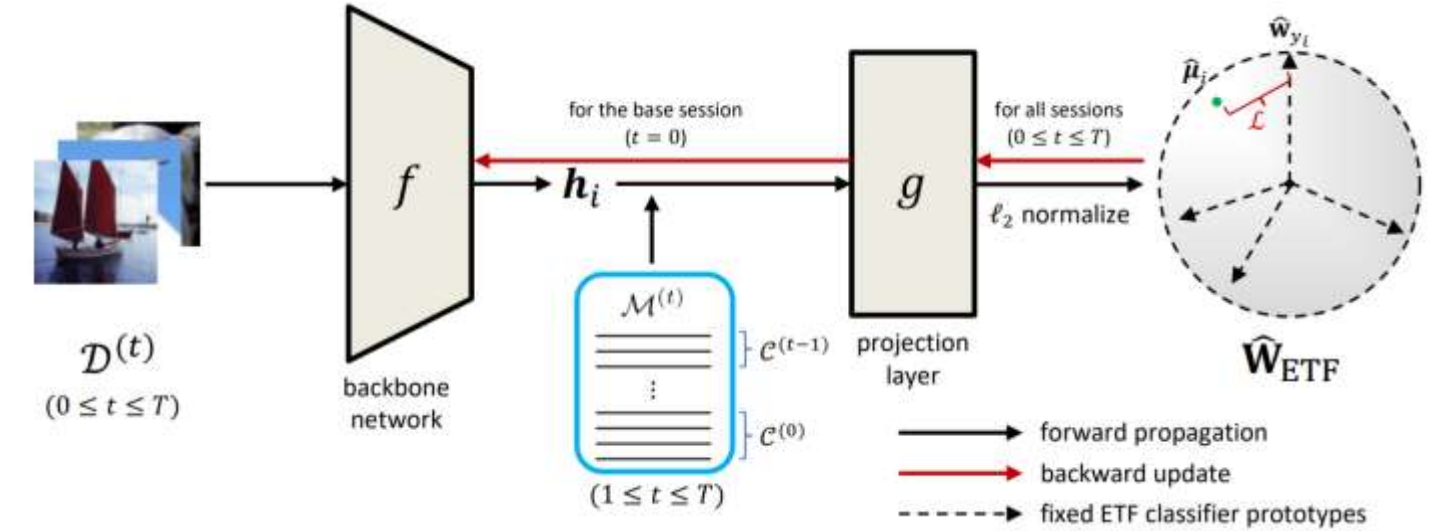
$$\min_{\theta_f, \theta_g} \quad \frac{1}{|\mathcal{D}^{(0)}|} \sum_{(\boldsymbol{x}_i, y_i) \in \mathcal{D}^{(0)}} \mathcal{L}\left(\hat{\boldsymbol{\mu}}_i, \hat{\mathbf{W}}_{\text{ETF}}\right), \tag{6}$$

Following Hersche et al. (2022), we only keep a memory $\mathcal{M}^{(t)}$ of the mean intermediate feature $\boldsymbol{h}_c$ for each old class c. Concretely, we have,

$$\mathcal{M}^{(t)} = \{\boldsymbol{h}_c | c \in \cup_{j=0}^{t-1} \mathcal{C}^{(j)}\}, \quad \boldsymbol{h}_c = \text{Avg}_i\{f(\boldsymbol{x}_i, \theta_f) | y_i = c\}, \quad 1 \le t \le T, \tag{7}$$

where f has been fixed after the base session. Then we use $\mathcal{D}^{(t)}$ as the input of f, and $\mathcal{M}^{(t)}$ as the input of g to finetune the projection layer g. The empirical risk to minimize in incremental sessions

4.3 NC-FSCIL



$$\min_{\theta_g} \quad \frac{1}{|\mathcal{D}^{(t)}| + |\mathcal{M}^{(t)}|} \left(\sum_{(\boldsymbol{x}_i, y_i) \in \mathcal{D}^{(t)}} \mathcal{L}\left(\hat{\boldsymbol{\mu}}_i, \hat{\mathbf{W}}_{\text{ETF}}\right) + \sum_{(\boldsymbol{h}_c, y_c) \in \mathcal{M}^{(t)}} \mathcal{L}\left(\hat{\boldsymbol{\mu}}_c, \hat{\mathbf{W}}_{\text{ETF}}\right) \right), \tag{8}$$

where $\hat{\boldsymbol{\mu}}_i$, and $\hat{\boldsymbol{\mu}}_c$ are the output features of $\boldsymbol{x}_i$ and $\boldsymbol{h}_c$, respectively, $|\mathcal{D}^{(t)}|$ is the number of training samples in session t, and we have $|\mathcal{M}^{(t)}| = \sum_{j=0}^{t-1} |\mathcal{C}^{(j)}|$. **Thanks to our pre-assigned alignment, we do not rely on any regularizer in our training**.

In the evaluation of session t, we predict an input $\boldsymbol{x}$ based on the inner product between its output feature $\hat{\boldsymbol{\mu}}$ and the ETF classifier prototypes: $\arg\max_k \langle \hat{\boldsymbol{\mu}}, \hat{\mathbf{w}}_k \rangle$, $\forall 1 \le k \le K_0 + K'$.

Table 1: Performance of FSCIL in each session on miniImageNet and comparison with other studies. The top rows list class-incremental learning and few-shot learning results implemented by Tao et al. (2020b); Zhang et al. (2021) in the FSCIL setting. "Average Acc." is the average accuracy of all sessions. "Final Improv." calculates the improvement of our method in the last session. * indicates that the method saves the within-class feature mean of each class for training or inference.

Methods	**Accuracy in each session (%) ↑**									**Average**	**Final**
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**Acc.**	**Improv.**
iCaRL (Rebuffi et al., 2017)	61.31	46.32	42.94	37.63	30.49	24.00	20.89	18.80	17.21	33.29	**+41.1**
NCM (Hou et al., 2019)	61.31	47.80	39.30	31.90	25.70	21.40	18.70	17.20	14.17	30.83	**+44.14**
D-Cosine (Vinyals et al., 2016)	70.37	65.45	61.41	58.00	54.81	51.89	49.10	47.27	45.63	55.99	**+12.68**
*TOPIC (Tao et al., 2020b)	61.31	50.09	45.17	41.16	37.48	35.52	32.19	29.46	24.42	39.64	**+33.89**
*IDLVQ (Chen & Lee, 2021)	64.77	59.87	55.93	52.62	49.88	47.55	44.83	43.14	41.84	51.16	**+16.47**
Self-promoted (Zhu et al., 2021a)	61.45	63.80	59.53	55.53	52.50	49.60	46.69	43.79	41.92	52.76	**+16.39**
CEC (Zhang et al., 2021)	72.00	66.83	62.97	59.43	56.70	53.73	51.19	49.24	47.63	57.75	**+10.68**
*LIMIT (Zhou et al., 2022b)	72.32	68.47	64.30	60.78	57.95	55.07	52.70	50.72	49.19	59.06	**+9.12**
*Regularizer (Akyürek et al., 2022)	80.37	74.68	69.39	65.51	62.38	59.03	56.36	53.95	51.73	63.71	**+6.58**
MetaFSCIL (Chi et al., 2022)	72.04	67.94	63.77	60.29	57.58	55.16	52.90	50.79	49.19	58.85	**+9.12**
*C-FSCIL (Hersche et al., 2022)	76.40	71.14	66.46	63.29	60.42	57.46	54.78	53.11	51.41	61.61	**+6.90**
Data-free Replay (Liu et al., 2022)	71.84	67.12	63.21	59.77	57.01	53.95	51.55	49.52	48.21	58.02	**+10.10**
*ALICE (Peng et al., 2022)	80.60	70.60	67.40	64.50	62.50	60.00	57.80	56.80	55.70	63.99	**+2.61**
***NC-FSCIL (ours)**	**84.02**	**76.80**	**72.00**	**67.83**	**66.35**	**64.04**	**61.46**	**59.54**	**58.31**	**67.82**	
Improvement over ALICE	+3.42	+6.20	+4.60	+3.33	+3.85	+4.04	+3.66	+2.74	+2.61	**+3.83**	

Table 2: Performance of FSCIL in each session on CIFAR-100 and comparison with other studies. The top rows list class-incremental learning and few-shot learning results implemented by Tao et al. (2020b); Zhang et al. (2021) in the FSCIL setting. "Average Acc." is the average accuracy of all sessions. "Final Improv." calculates the improvement of our method in the last session.

Methods	**Accuracy in each session (%) ↑**									**Average**	**Final**
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**Acc.**	**Improv.**
iCaRL (Rebuffi et al., 2017)	64.10	53.28	41.69	34.13	27.93	25.06	20.41	15.48	13.73	32.87	**+42.38**
NCM (Hou et al., 2019)	64.10	53.05	43.96	36.97	31.61	26.73	21.23	16.78	13.54	34.22	**+42.57**
D-Cosine (Vinyals et al., 2016)	74.55	67.43	63.63	59.55	56.11	53.80	51.68	49.67	47.68	58.23	**+8.43**
*TOPIC (Tao et al., 2020b)	64.10	55.88	47.07	45.16	40.11	36.38	33.96	31.55	29.37	42.62	**+26.74**
Self-promoted (Zhu et al., 2021a)	64.10	65.86	61.36	57.45	53.69	50.75	48.58	45.66	43.25	54.52	**+12.86**
CEC (Zhang et al., 2021)	73.07	68.88	65.26	61.19	58.09	55.57	53.22	51.34	49.14	59.53	**+6.97**
DSN (Yang et al., 2022a)	73.00	68.83	64.82	62.64	59.36	56.96	54.04	51.57	50.00	60.14	**+6.11**
*LIMIT (Zhou et al., 2022b)	73.81	72.09	67.87	63.89	60.70	57.77	55.67	53.52	51.23	61.84	**+4.88**
MetaFSCIL (Chi et al., 2022)	74.50	70.10	66.84	62.77	59.48	56.52	54.36	52.56	49.97	60.79	**+6.14**
*C-FSCIL (Hersche et al., 2022)	77.47	72.40	67.47	63.25	59.84	56.95	54.42	52.47	50.47	61.64	**+ 5.64**
Data-free Replay (Liu et al., 2022)	74.40	70.20	66.54	62.51	59.71	56.58	54.52	52.39	50.14	60.78	**+5.97**
*ALICE (Peng et al., 2022)	79.00	70.50	67.10	63.40	61.20	59.20	58.10	56.30	54.10	63.21	**+2.01**
***NC-FSCIL (ours)**	**82.52**	**76.82**	**73.34**	**69.68**	**66.19**	**62.85**	**60.96**	**59.02**	**56.11**	**67.50**	
Improvement over ALICE	+3.52	+6.32	+6.24	+6.28	+4.99	+3.65	+2.86	+2.72	+2.01	**+4.29**	

Table 3: Ablation studies on three datasets to investigate the effects of ETF classifier and DR loss. "Learnable+CE" uses a learnable classifier and the CE loss; "ETF+CE" adopts our ETF classifier with the CE loss; "ETF+DR" uses both ETF classifier and DR loss. "FINAL" refers to the accuracy of the last session; "AVERAGE" is the average accuracy of all sessions; "PD" denotes the performance drop, *i.e.,* the accuracy difference between the first and the last sessions.

Methods	miniImageNet			CIFAR-100			CUB-200		
	FINAL↑	AVERAGE↑	PD↓	FINAL↑	AVERAGE↑	PD↓	FINAL↑	AVERAGE↑	PD↓
Learnable+CE	50.04	61.30	34.53	52.13	62.68	30.14	50.38	59.58	29.19
ETF+CE	56.66	68.23	28.21	54.42	64.00	27.36	56.83	65.51	23.27
ETF+DR	58.31	67.82	25.71	56.11	67.50	26.41	59.44	67.28	21.01

THANKS