IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 22 January 2026, accepted 29 January 2026, date of publication 6 February 2026, date of current version 11 February 2026.

Digital Object Identifier 10.1109/ACCESS.2026.3661148

RESEARCH ARTICLE

FFCIL: Fine-Grained Few-Shot Class Incremental Learning With Destruction-Construction Learning for TFT-LCD Defect Classification

ANINDITA SURYARASMI[1], CHIA-YU LIN[1], (Member, IEEE), LE-CHIEN CHU[2], WEI-JEN WANG[1], (Member, IEEE), AND DERON LIANG[1], (Member, IEEE)

[1]Department of Computer Science and Information Engineering, National Central University, Zhongli District, Taoyuan City 320317, Taiwan
[2]Department of Computer Science and Engineering, Yuan Ze University, Zhongli District, Taoyuan City 32003, Taiwan

Corresponding author: Chia-Yu Lin (sallylin0121@ncu.edu.tw)

This work was supported by the National Science and Technology Council (NSTC) under Project NSTC 114-2622-E-008-016 and Project NSTC 113-2221-E-008-107-MY2.

ABSTRACT Defect classification for thin-film transistor liquid crystal displays (TFT-LCD) poses significant challenges due to the fine-grained nature of microscopic defects and the limited availability of labeled data. This problem is particularly demanding in the context of few-shot class-incremental learning (FSCIL). FSCIL requires learning new defect classes from limited data while distinguishing between highly similar classes. At the same time, it must preserve knowledge of previously learned classes. We introduce Fine-grained Few-shot Class Incremental Learning with Destruction-construction Learning (FFCIL) to address these challenges. This approach employs a multistage training strategy that begins with destruction-construction learning. This stage captures intricate and fine-grained features during pretraining and is combined with manifold-mixup as a placeholder mechanism. Together, these components enable the seamless integration of new defect classes with minimal samples. Furthermore, a triplet loss is applied during pseudo-incremental learning to improve the model's ability to differentiate between base and pseudo-classes. This approach enhances the fine-grained classification performance in a few-shot class incremental learning scenarios. Experimental results on an industry-based TFT-LCD dataset demonstrate a notable 16.36% improvement in overall accuracy. In addition, FFCIL achieves the lowest performance dropping rate among all compared methods, indicating superior stability. Tests on public datasets further reveal an improvement of up to 39.58% in overall accuracy, underscoring the potential applicability of FFCIL to broader domains. Code is available at https://github.com/sallylinlab/ffcil

INDEX TERMS TFT-LCD, artificial intelligence, few-shot incremental learning, destruction-construction learning, fine-grained classification, defect classification.

I. INTRODUCTION

The TFT-LCD industry is rapidly advancing to meet consumer demands for various devices. To ensure the quality of the product is well maintained, automated optical inspection (AOI) is often implemented to detect defective products as they are produced automatically [1], [2], [3], [4]. In contrast to manual inspection, which is susceptible to subjectivity, deep learning-based AOIs provide more reliable results across various domains [5], [6], [7]. However, deep learning approaches require a large number of image samples to achieve optimal performance, which is difficult to obtain in real-world scenarios [8]. Furthermore, in practical manufacturing environments, defect data often does not become available at once. Only a limited set of defect classes may be known at the initial stage of deploying deep learning-based AOI. As the model is implemented and used over time, new types of defects are discovered, requiring the system to adapt to evolving data distributions in a phased manner (see Fig. 1 for an illustration of this progressive data availability).

Incremental learning in defect classification proves particularly useful in scenarios where the dataset is continuously

The associate editor coordinating the review of this manuscript and approving it for publication was Xin Sun.

growing. This approach allows models to adapt to newly introduced defect classes over time, enabling them to remain relevant and effective in dynamic environments [9], [10]. However, this process becomes significantly more challenging when only limited data for each new increment is available, as is often the case in real-world industrial settings. Limited data can lead to underfitting, making it harder for the model to generalize effectively to unseen samples [4], [11], [12].

To address the additional complexity of learning with minimal data, incremental learning has been extended to few-shot class incremental learning (FSCIL). The few-shot learning technique is specifically designed to enable the model to effectively learn new classes with minimal samples [13], [14], [15], [16]. This learning technique becomes valuable for industries where collecting extensive labeled datasets is costly or impractical. The challenge in FSCIL becomes more pronounced when newly introduced defect classes during the incremental stage share significant similarities with previously learned classes, transforming the problem into fine-grained defect classification [13]. In such cases, the model must effectively adapt to new data while precisely differentiating between closely related classes without succumbing to catastrophic forgetting, which becomes a common issue in incremental learning.

A promising approach to FSCIL implementation is the continuously evolved classifier (CEC) [13]. This method separates feature learning from classifier updating, allowing the model to accommodate multiple incremental learning tasks and learn new features from the previously trained model. While CEC [13] demonstrates strong performance in general incremental learning scenarios, it does not explicitly address the challenges posed by fine-grained defect classification.

To tackle this limitation, we propose fine-grained few-shot class incremental learning with destruction-construction learning (FFCIL), which combines several established techniques into an FSCIL training pipeline. Specifically, destruction and construction learning (DCL) [17] is employed during the base training stage to enhance discriminative features during the destruction phase and captures the semantic correlation among local regions through the reconstruction phase. To better accommodate future unseen classes in the incremental stage, manifold-mixup (MM) [18] is incorporated to regularize the feature space. Additionally, pseudo-incremental learning is introduced to simulate the incremental phase, thereby preparing the network to adapt to new classes. Furthermore, triplet loss [19] is introduced to improve class separability between base and incremental classes during few-shot updates. The proposed FFCIL focuses on combining these complementary techniques to fit the FSCIL requirements, where robust base representations, sufficient feature space for novel classes, and stable inter-class discrimination are all critical. Their joint use within the CEC [13] protocol enables the framework to better address fine-grained classification challenges while mitigating catastrophic forgetting during incremental learning.

When applied to a proprietary industry-based dataset, this study achieves a 16.36% improvement in overall accuracy, 18.86% gain in preventing knowledge forgetting, and a 14.08% increase in learning new information from limited fine-grained samples. When tested on two public datasets, FFCIL shows significant performance gains, with overall accuracy increasing by up to 39.58% and 31.81%; existing knowledge retention rising by up to 40.64% and 34.38%; and new knowledge acquisition improving by up to 46.52% and 35.89%, respectively. The key contributions of the study can be summarized as follows:

- The proposed FFCIL enhances the fine-grained classification performance in few-shot incremental learning by systematically combining DCL [17], manifold-mixup [18], and triplet loss [19] into the CEC [13] framework.
- FFCIL can reduce the performance dropping rate to as low as 15.13%, with higher new knowledge acquisition up to 46.52% on emerging defect classes.
- FFCIL holds potential for wide-ranging application across various domains, as evidenced by its strong performance on both private and public datasets from various fields.

The remainder of this paper is structured as follows: Section II reviews related works; Section III explains the details of the proposed method; Section IV describes the experiments; Section V presents the results and discussion; and Section VI provides the conclusions.

II. RELATED WORKS

Recent studies have explored various techniques for detecting mura and related defects in TFT-LCD displays. One line of work focuses on enhancing defect localization and segmentation, such as the SVDRNC framework [1], which integrates color normalization via singular value decomposition with a neutrosophic canny edge detector to address color inconsistencies during image acquisition. This method enables precise edge extraction for subsequent laser-cutting path generation and has demonstrated superior performance over several conventional techniques on industrial TFT-LCD images. Complementing this, another study targets the challenge of cross-process defect classification, proposing a multi-modal learning strategy that combines descriptive embedding generation with cross-modal contrastive learning [2]. By incorporating fine-grained textual descriptions alongside visual features, the method achieves notable accuracy improvements and has been validated in real manufacturing environments. Another deep learning has been introduced to enhance robustness under complex visual conditions, such as the YOLO-GA framework [3], which integrates Ghost modules and CBAM attention to improve lightweight feature extraction and vertical-line mura detection performance on high-resolution LCD images. However, as defect categories

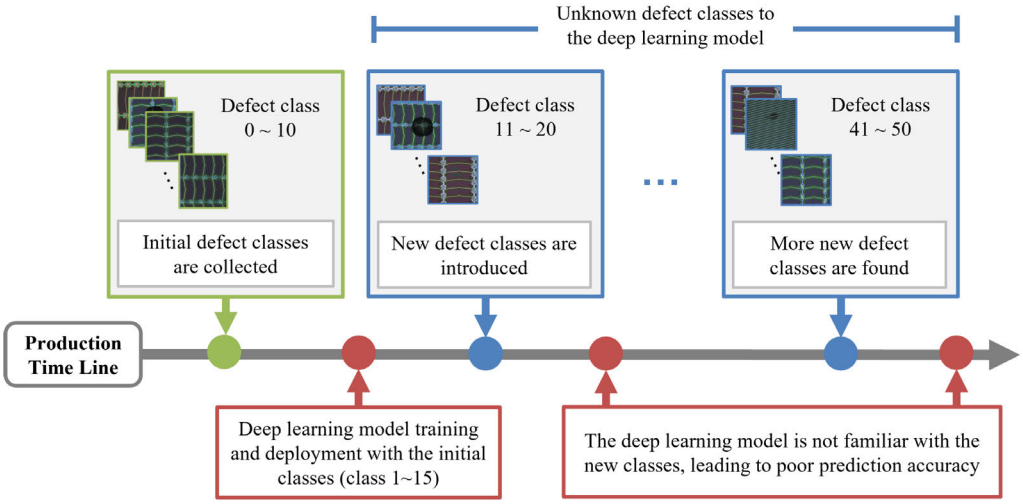


FIGURE 1. Several defect classes are initially introduced during the deep learning base model training and deployment. As production processes evolve, new defect classes emerge. These new classes often have few samples and exhibit high visual similarity to existing ones, yet they remain unknown to the deployed model.

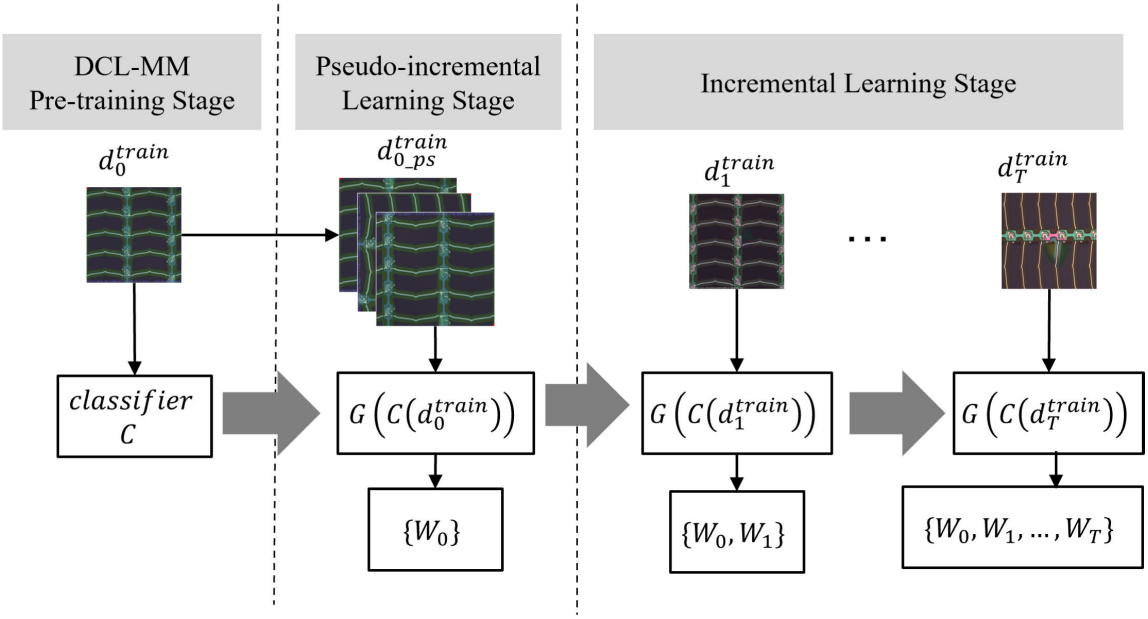


FIGURE 2. The proposed framework consists of three stages: the DCL-MM pretraining stage, the pseudo-incremental learning stage, and the incremental learning stage.

in real manufacturing environments continue to evolve and new types may emerge over time, these detection systems increasingly require learning capabilities that can adapt to new classes without retraining from scratch, motivating the use of class incremental learning (CIL).

Class incremental learning (CIL) aims to progressively acquire knowledge of new classes without forgetting the previously learned ones. Several approaches have been proposed to resolve the problem of knowledge forgetting in CIL. Zhou et al. [20] introduced a dual-consolidation scheme that merges backbone representations and aligns classifier weights across incremental domains, thereby stabilizing both feature and classifier drift. Zhou et al. [21] proposed the ENGINE method, which injects external knowledge into a CLIP-based model and then re-rank predictions at inference, thereby improving resilience to evolving classes. Finally, Wang et al. [22] presented Task-Specific and Universal Adapters for Pre-Trained Model-based Class-Incremental Learning, which combines task-specific adapters selected via an entropy-based mechanism with a universal adapter capturing shared knowledge, thereby blending specialized and general representations to improve incremental class learning.

While various strategies have been proposed to address knowledge forgetting in CIL, another practical challenge lies in the limited number of samples per class available

for training in each incremental task. This scarcity of data encourages the development of few-shot class incremental learning (FSCIL). Several studies have been conducted in the FSCIL domain. Tao et al. [14] built a topology-preserving knowledge incrementor network to stabilize the topology and enhance representation learning for few-shot new classes. Sun et al. [23] proposed a feature fusion framework that emphasizes key regions to capture discriminative local features, enabling effective recognition of visually similar categories from limited samples. By leveraging focus-area localization and high-order feature integration, their approach highlights the importance of modeling subtle intra-class variations and part-level interactions. These characteristics are especially relevant to FSCIL, where models must rely on fine-grained visual cues to generalize from scarce data. Ahmad et al. [9] built leveraging self-supervised features (FeSSSS) by training a classifier on the fusion of supervised and self-supervised models to solve the problems of overfitting and catastrophic forgetting in FSCIL. Zhou et al. [10] proposed forward-compatible training (FACT) to learn prospectively by reserving embedding space for future new classes. Zhu et al. [24] introduced an incremental prototype learning strategy called self-promoted prototype refinement (SPPR), which comprises a random episode selection strategy and a self-promoted prototype refinement mechanism to optimize representation and reorganize prototypes with minimal supervision. Finally, Zhang et al. [13] proposed continuously evolved classifiers (CECs), which adopt a decoupled learning strategy to prevent knowledge forgetting in class representations.

While the FSCIL approaches demonstrated effectiveness, new challenges arise when applied to fine-grained datasets. Several methods have been proposed to address this issue without relying on a few-shot CIL. Chen et al. [17] introduced the destruction and construction learning (DCL [17]) method, which partitions input images into local regions to refocus attention on discriminative regions. Schroff et al. [19] employed triplet loss by using matching and nonmatching data triplets to help the model differentiate between distinct categories. Verma et al. [18] developed a manifold-mixup (MM [18]), which encourages neural networks to remain uncertain when encountering data samples that differ from those seen during training, thus addressing the issue of incorrect yet overly confident predictions. By combining CEC [13] with triplet loss [19], and incorporating DCL [17] and MM [18] as the backbone feature extractor, the proposed FFCIL is constructed to address the few-shot classification problem in incremental-class fine-grained datasets.

III. METHODS

The proposed framework, illustrated in Fig. 2 comprises three main stages: the DCL-MM pretraining stage (the terms "pretraining stage" and "base stage" are used interchangeably throughout the study), the pseudo-incremental learning stage, and the incremental learning stage. The first stage trains the base classifier model C using destruction and construction learning (DCL) [17] and manifold-mixup (MM) [18] on the base dataset d_0^{train}. In the second stage, a graph attention network G generates weight vectors $\{W_0\}$, which can be expanded as new data groups are introduced. Finally, during the incremental learning stage, new weight vectors $\{W_t\}$ are generated for each new image group t, expanding the set into $\{W_0, .., W_t\}$. It allows the classifier to learn new data while retaining previous knowledge. The notations used in this study are summarized in Table 1.

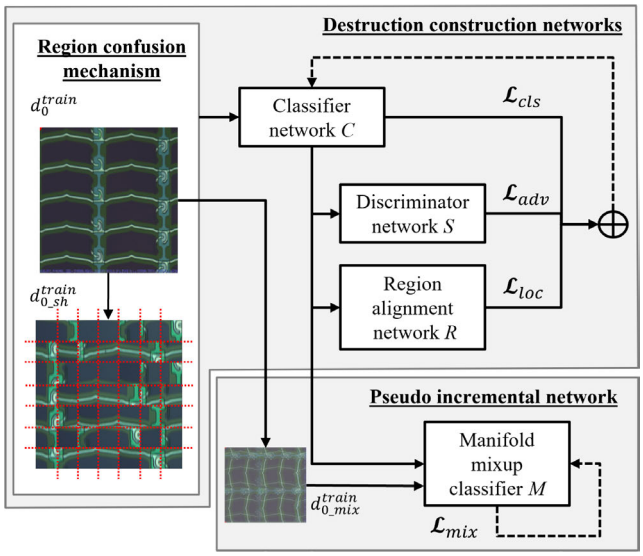


FIGURE 3. The pretraining and pseudo-incremental stage adopt a region confusion mechanism (RCM) and four networks, which include classification C, discriminator S, region alignment R, and manifold-mixup M. Each network is associated with specific losses, including classifier loss (L_{cls}), adversarial loss (L_{adv}), location loss (L_{loc}), and mixup loss (L_{mix}).

A. DCL-MM PRETRAINING STAGE

The DCL [17] identifies fine defective features through a destruction procedure that captures information from discriminative regions. This is followed by a construction process that learns the local details based on semantic correlation, along with the discriminator, classification, and manifold-mixup networks.

As illustrated in Fig. 3, the DCL-MM mainly consists of two blocks: the destruction-construction networks and the manifold-mixup (MM) network. In the first component, the region confusion mechanism (RCM) is applied to the input dataset d_0^{train}, producing a destructed sample $d_{0\_sh}^{train}$. This component is chosen based on the characteristics of the target dataset, where discriminative defect features are extremely small and localized. As a result, global spatial structures and large-scale patterns provide limited semantic information, whereas fine-grained local details play a dominant role in class discrimination.

To further illustrate this mechanism, Fig. 4 presents a visual example of RCM. The original input image (top left) is first partitioned into 7 × 7 non-overlapping patches (top right),

TABLE 1. Notation used in this study.

Symbol	Description	Symbol	Description	Symbol	Description
d_t^{train}	train split of image group d_t	$d_{0\_sh}^{train}$	shuffled patch image from the training group d_0^{train}	$d_{0\_mix}^{train}$	mixup image from the training group d_0^{train}
d_t^{test}	test split of image group d_t	C	classification network	S	discriminator network
R	region alignment network	G	graph attention network	W_t	weight vector of task t
θ_{cls}	learnable parameter of classifier network	θ_{adv}	learnable parameter of discriminator network	θ_{loc}	learnable parameter of region alignment network
θ_{mix}	learnable parameter of mixup network	$\mathcal{L}_{cls}$	classification loss	$\mathcal{L}_{loc}$	location loss
$\mathcal{L}_{adv}$	adversarial loss	$\mathcal{L}_{mix}$	manifold-mixup loss	$\mathcal{L}_{trp}$	triplet loss
α	weighting factor for $\mathcal{L}_{loc}$	β	weighting factor for $\mathcal{L}_{adv}$	γ	weighting factor for $\mathcal{L}_{loc}$
ρ	rotation parameter	$\mathrm{ACC_{all}}$	accuracy of d^{test_all}	$\mathrm{ACC_{prm}}$	accuracy of d^{test_prm}
$\mathrm{ACC_{add}}$	accuracy of d^{test_add}	$\mathrm{ACC_4^{d_t}}$	accuracy of image group task t at task 4		

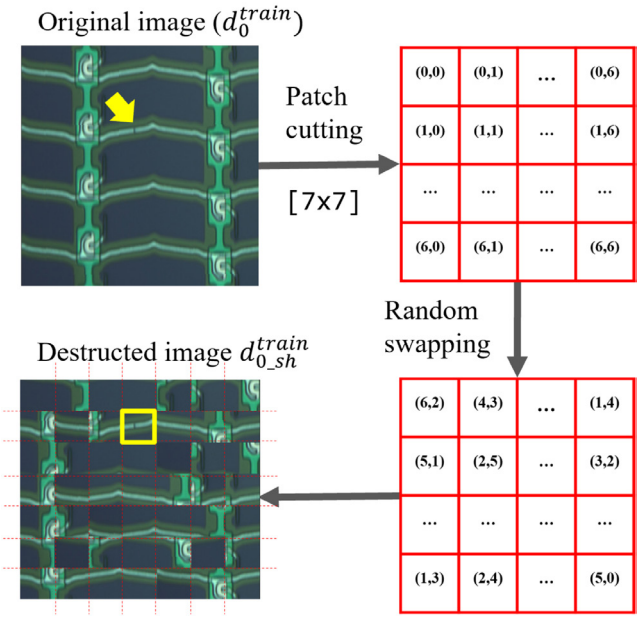


FIGURE 4. Example image (above) and the corresponding deconstructed images (below) from the shuffled patches in RCM [17].

followed by a random swapping operation that generates a shuffled configuration (bottom right and left). As indicated by the yellow arrow in the original image, the defect is confined to a small local region. After shuffling, the same defect remains confined within a single patch, highlighted by the yellow box, despite its spatial relocation. By disrupting the global layout while preserving local patch content, RCM encourages the model to learn that the discriminative information resides in fine-grained local details rather than in overall image structure.

The destructed images $d_{0\_sh}^{train}$ are then subsequently fed into a classification network C, that takes images and categorizes them into defective classes, as defined in Eq. 1. In this formulation, y denotes the class label and θ_{cls} is the learnable classification parameter. ResNet18 [25] architecture is selected as the backbone classification network with pretrained ImageNet weights. The feature vectors acquired from the classifier are used as input for the discriminator S (Eq. 3) that determines whether to retain or discard domain-specific patterns to mitigate the impact of detrimental visual noise introduced by the shuffled subregions. Consequently, two distinct losses are employed for the classifier and discriminator: the classifier loss L_{cls} [17] and the adversarial loss L_{adv} [17]. The classifier loss, defined in Eq. 2, uses x, $\phi(x)$, y, and C to represent the original image, destructed image, label for fine-grained categories, and probability distribution vector, respectively. The adversarial loss, defined in Eq. 4, uses S as the discriminator network and $\boldsymbol{s}$ as a one-hot vector, indicating whether destruction has occurred.

$$C(x) = \mathrm{softmax}(\theta_{cls}, f(x, y)) \tag{1}$$

$$\mathcal{L}_{cls} = -\sum_{x \in d_0^{train}} y \cdot \log\left[C(x)C(\phi(x))\right] \tag{2}$$

$$S(x) = \mathrm{softmax}(\theta_{adv} C(x)) \tag{3}$$

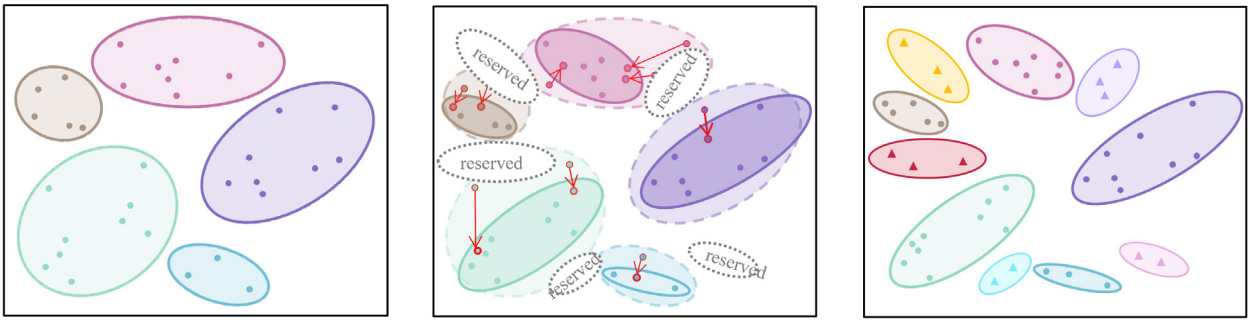
$$\mathcal{L}_{adv} = -\sum_{x \in d_0^{train}} \boldsymbol{s} \cdot \log[S(x)] + (1 - \boldsymbol{s}) \cdot \log[S(\phi(x))] \tag{4}$$

Concurrently, the region alignment network R [17] is established to restore the spatial layout. As defined in Eq. 5, it takes input of classifier features $C(x)$ to obtain a map sized $2 \times N \times N$, which corresponds to rows and columns of each region. While training, the location loss L_{loc} [17], as defined in Eq. 6, calculates the distance between the predicted coordinates and the original coordinates. $R_{\sigma(i,j)}(\phi(x))$ and $R_{i,j}(x)$ in Eq. 6 denote the predicted locations of the destroyed and original images, respectively, whereas i and j represent the ground truth locations. The total loss calculation for the destruction-contstruction block becomes the cumulative sum of three losses, as denoted in Eq. 9, where α, β, and γ, represent the respective weighting factors. In this work, we set all loss weights to one. An ablation of the value of the weight losses are presented in Section V-B.

$$R(x) = \mathrm{map}(\theta_{loc}, C(x)) \tag{5}$$

$$\mathcal{L}_{\mathrm{loc}} = \sum_{x \in d_0^{train}} \sum_{i=1}^{N} \sum_{j=1}^{N} \left| R_{\sigma(i,j)}(\phi(x)) - \begin{bmatrix} i \\ j \end{bmatrix} \right|_1 + \left| R_{i,j}(x) - \begin{bmatrix} i \\ j \end{bmatrix} \right|_1 \tag{6}$$

Furthermore, a manifold-mixup network M is employed to ensure the classifier can encompass all classes across



(a) (b) (c)

FIGURE 5. (a) The visualization of five base classes' initial feature distribution during the pretraining stage. (b) To provide space for future classes introduced during the incremental stage, the distance between each class is increased with the manifold-mixup. (c) The new classes (shown in triangles) arrive sequentially during the incremental stage and occupy the reserved spaces.

incremental tasks. Fig. 5 conceptually illustrates how M influences the feature space across different learning stages. In Fig. 5a, the initial feature distribution during pretraining is shown. Although the classes are separable, they are located close to one another, leaving limited room for accommodating future classes. The Fig. 5b depicts the effect of applying M, where the feature distributions of base classes become more compact, effectively increasing the inter-class margins and creating reserved regions in the feature space that are not occupied by base classes. Finally, in the Fig. 5c, new classes (triangular markers) are sequentially introduced during incremental learning. These classes occupy the previously reserved spaces, reducing interference with the base classes, which strengthens incremental learning. This also provides sufficient embedding space for novel classes, despite limited samples, thereby improving few-shot learning performance.

As defined in Eq. 7, the M network takes two image samples x and x' along with the corresponding labels y and y' to create a mixed classes data input, annotated as $d^{train}_{0\_mix}$. A mixup loss L_{mix} is implemented (Eq. 8) to this network by applying mixed ground truth labels y^{mix}_c and mixed class prediction $M(x^{mix}_c)$ to the standard cross-entropy loss. The pseudocode outlining the key stages of DCL-MM pretraining stage is shown in Algorithm 1.

$$M(x) = \text{softmax}(\theta_{mix}, \text{Mix}_\lambda(C(x), C(x')), \text{Mix}_\lambda(y, y'))) \tag{7}$$

$$\mathcal{L}_{mix} = -\sum_{c=1}^{C} y^{mix}_c \cdot \log(M(x^{mix}_c)) \tag{8}$$

$$\mathcal{L} = \alpha\mathcal{L}_{cls} + \beta\mathcal{L}_{loc} + \gamma\mathcal{L}_{adv} \tag{9}$$

Algorithm 1 DCL-MM Pretraining Stage Pseudo-Code

Input: Training data task 0 d^{train}_0
Output: Base classifier model C
$d^{train}_{0\_sh} \leftarrow$ shuffle d^{train}_0
Define C, S, R, and M networks
while $i < DCL\_epochs$ **do**
  Forward pass C with d^{train}_0 and $d^{train}_{0\_sh}$
  $\mathcal{L}_{cls} \leftarrow$ calculate classifier loss
  Forward pass S with with features from C
  $\mathcal{L}_{adv} \leftarrow$ calculate adversarial loss
  Forward pass R with features from C
  $\mathcal{L}_{loc} \leftarrow$ calculate location loss
  $\mathcal{L} \leftarrow \alpha\mathcal{L}_{cls} + \beta\mathcal{L}_{loc} + \gamma\mathcal{L}_{adv}$
  Update C, S, and R
end while
$M \leftarrow$ Load C network weights
while $i < base\_epochs$ **do**
  $d^{train}_{0\_mix} \leftarrow$ mix d^{train}_0
  Forward pass M with $d^{train}_{0\_mix}$
  $\mathcal{L}_{mix} \leftarrow$ calculate mixup loss
  Update M
end while
$C \leftarrow$ update weights from M

B. PSEUDO-INCREMENTAL LEARNING STAGE

This stage performs training with pseudo-classes to enhance the model's overall capability to generalize and adapt to new scenarios. Following the learning approach introduced in [13], a graph attention network (GAT), denoted as G, is constructed using pseudo-classes to simulate the incremental learning stage. As defined in Eq. 10, G takes classifier features from the base dataset d^{train}_0 and the pseudo-classes $d^{train}_{0\_ps}$ to obtain the weight vectors $\{W_0\}$. The pseudo-classes, described in Eq. 11, are generated by applying random rotations to d^{train}_0, controlled by the rotation parameter $\rho \in \{90°, 180°, 270°\}$. This constraint ensures that the rotated images remain axis-aligned and avoid diagonal transformation, following the original image layout.

During the optimization of G, triplet loss L_{trp} [19] is added alongside the standard cross-entropy loss to enhance the model's ability to discriminate between the base and the pseudo-class. Three types of samples are defined within

L_{trp}: the anchor sample, denoted as x^a; the positive sample, denoted as x^p; and the negative sample, denoted as x^n. The goal is to minimize the distance between embeddings of similar samples (x^a and x^p) while maximizing the distance between embeddings of dissimilar samples (x^a and x^n), as shown in Eq. 12, with $\epsilon = 1$ denoting the enforced margin. By promoting compact intra-class clustering and larger inter-class separation, L_{trp} strengthens incremental learning by reducing interference from newly introduced classes and simultaneously improves few-shot learning by making the limited novel samples more discriminative in the embedding space. The pseudocode of the proposed method is provided in Algorithm 2.

The multiple losses employed in the proposed framework each target a specific aspect of the learning objective. For example, L_{cls} enforces correct class prediction, L_{adv} enhances feature robustness, L_{loc} maintains spatial consistency, and L_{mix} encourages smooth feature-space interpolation. These losses serve as complementary supervisory signals that guide the optimization of the shared network components, including the classifier C, without introducing additional prediction heads or increasing model capacity. This design follows the precedent established in DCL [17], where multiple losses are used to improve feature generalization.

$$\{W_0\} = G(C(d_0^{train}, d_{0\_ps}^{train})) \tag{10}$$

$$d_{0\_ps}^{train} = \text{rotate}(d_0^{train}, \rho) \tag{11}$$

$$\mathcal{L}_{trp} = \text{Max}(|f(x^a) - f(x^p)|^2 - |f(x^a) - f(x^n)|^2 + \epsilon) \tag{12}$$

Algorithm 2 Pseudo-Incremental Learning Stage Pseudo-Code

Input: d_0^{train}, C
Output: Graph attention network G, weight vectors W_0
$d_{0\_ps}^{train} \leftarrow$ random rotation of d_0^{train}
Define initial G node features from $C(d_0^{train}, d_{0\_ps}^{train})$
while $i <$ *pseudo_incremental_epochs* **do**
    Forward pass G
    $\mathcal{L}_{trp} \leftarrow$ calculate triplet loss
    Update G
end while
$\{W_0\} \leftarrow G(C(d_0^{train}, d_{0\_ps}^{train}))$

C. INCREMENTAL LEARNING STAGE

In real-world scenarios, as illustrated in Fig. 1, the new group of data introduced after the model is deployed may include unseen classes with very few samples available, making retraining the model impractical. Therefore, as defined in Eq. 13, rather than retraining the model with newly introduced data and classes, the frozen classification model C is utilized to learn the features from the samples of the current incremental task t (denoted by d_t^{train}). The features are then input to the GAT G to obtain the weight vector W_t, which is subsequently appended via operation A to the previous set $\{W_0, \ldots, W_{t-1}\}$. Freezing C preserves the knowledge learned from earlier tasks, which is beneficial for incremental learning, while the newly produced weight vectors allow the classifier to accommodate new categories without disrupting the existing decision boundaries. The pseudocode of the incremental stage is presented in Algorithm 3.

$$\{W_0, \ldots, W_t\} = A(\{W_0, \ldots, W_{t-1}\}, G(C(d_t^{train}))) \tag{13}$$

TABLE 2. Hyperparameter settings for FFCIL training.

Variable	Value
Classifier	ResNet18
DCL learning rate	0.0008
DCL epoch	100
Pseudo incremental epoch	100
$\alpha, \beta, \gamma, \epsilon$	1
ρ	90, 180, 270

TABLE 3. Learning rate and epochs for the baselines and proposed methods.

Method	Learning rate	Epochs
CEC [13]	0.1	100
FACT [10]	0.0005	400
BiDist [26]	0.1	200
TEEN [27]	0.0004	100
AsyCLR [28]	0.1	100
FFCIL	0.001	100

Algorithm 3 Incremental Learning Stage Pseudo-Code

Input: d_t^{train}, C, G, $\{W_0\}$
Output: $\{W_0, \ldots, W_t\}$
$t \leftarrow 1$
while $t < T$ **do**
    $W_t \leftarrow G(C(d_t^{train}))$
    Append the set $\{W_0, \ldots, W_{t-1}\}$ with W_t
end while

D. TRAINING PROCESS

The proposed framework consists of three sequential training stages: DCL-based pretraining, pseudo-incremental training, and incremental learning. Although each stage is optimized separately, the stages are functionally dependent. First, the DCL script is run to produce a classifier model, which serves as the input to the second script, the pseudo-incremental stage. This second stage takes the trained DCL model and feature representations learned in the pretraining stage provide the initialization for the pseudo-incremental stage, which in turn generates a graph attention network and weight vectors used in the incremental learning stage. This sequential dependency ensures that knowledge acquired from the base dataset is retained and effectively propagated through all subsequent stages.

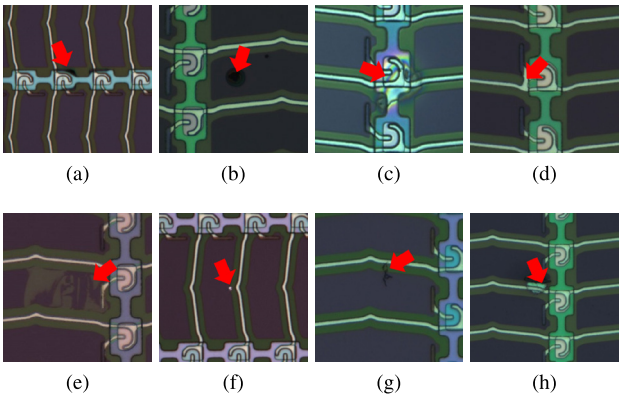


FIGURE 6. Representative examples of TFT-LCD defects in the proprietary A19 dataset: (a) brush defects affecting small areas along pattern edges; (b) pattern deformation caused by foreign material; (c) hole defects with halo artifacts produced by detached particles; (d) pattern deformation; (e) fragmented holes; (f) white aluminum residue; (g) debris on the pattern; and (h) laser repair marks.

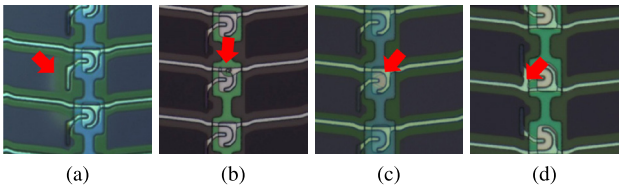


FIGURE 7. Examples of four defect samples from different classes that exhibit visual similarity. Each instance exhibits minor discoloration that closely matches the underlying pattern, allowing the defects to blend seamlessly with the background.

For the DCL pretraining stage, we use 100 epochs, a learning rate of 0.0008, 7×7 patch partitioning, and a ResNet backbone classifier. The resulting pretrained model is then used to initialize the pseudo-incremental training stage, which is trained for 100 epochs with a learning rate of 0.001. The primary hyperparameters for training FFCIL are reported in Table 2, while the losses and learning rates for FFCIL and the baseline models are provided in Table 3. The additional hyperparameters, such as network-specific settings and augmentation parameters, follow standard practices described in the corresponding papers.

IV. EXPERIMENT

We conduct comprehensive experiments on one proprietary TFT-LCD dataset (hereafter referred to as A19) and two publicly available datasets, CUB-200 [29] and Stanford Dogs [30]. Due to company confidentiality policies and signed agreements, the A19 dataset cannot be publicly released. For reproducibility, we also report results on publicly accessible benchmarks: the CUB-200 [29] and the Stanford Dogs [30] dataset.

The A19 dataset consists of 34,198 pairs of TFT-LCD RGB images and labels, with image resolutions ranging from 768 to 1380 pixels. It encompasses a total of 27 defect classes, which are labeled by domain experts in accordance with standard industrial inspection criteria. This dataset poses visual and distributional challenges. As seen in Fig. 6, the

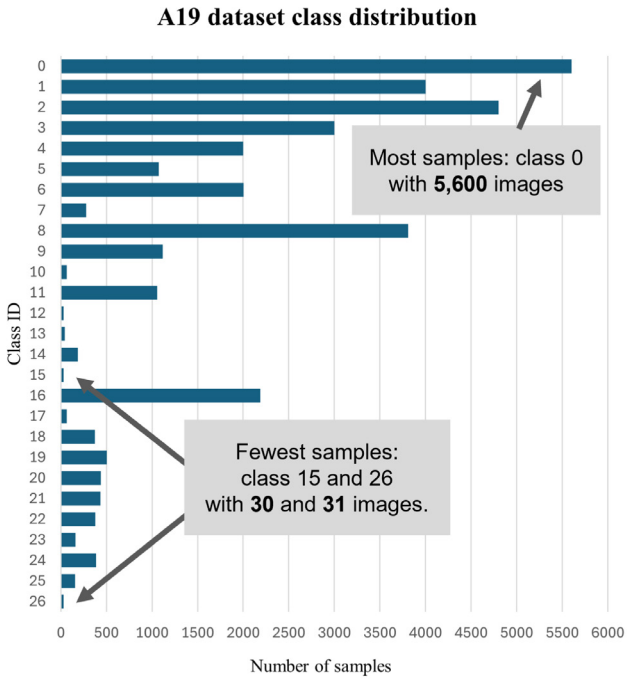


FIGURE 8. Across its 27 classes, the A19 dataset exhibits extreme imbalance, with sample sizes varying from 5,600 down to only 30, leading to a few-shot condition for the underrepresented classes.

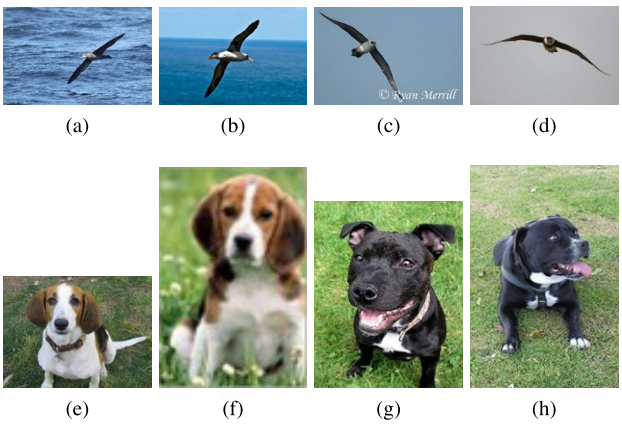


FIGURE 9. Samples from four different classes of the CUB-200 [29] dataset are shown in subfigures (a) to (d), while subfigures (e) to (f) display samples from the Stanford Dogs dataset. Despite representing different classes, the pairs (a)-(b), (c)-(d), (e)-(f), and (g)-(h) exhibit significant visual similarities, highlighting the challenge of fine-grained classification.

dominant pattern often overwhelms the appearance of the defects, rendering them subtle and difficult to perceive. Additionally, many defect regions share similar color tones with the surrounding pattern, causing them to blend in and resemble normal features. Furthermore, some classes exhibit highly similar visual characteristics, making class discrimination even more complicated. As shown in Fig. 7, four different classes share nearly indistinguishable appearances due to minor discolorations that match the pattern's color, thereby posing a fine-grained classification challenge. The dataset also suffers from severe class imbalance. Fig. 8 illustrates the

distribution of sample counts across the 27 classes, with the smallest classes containing only 30 samples and the largest up to 5,600. Among the 27 classes, only 11 classes have more than 1,000 samples, whereas the majority, particularly classes ID 10 to 26, contain fewer than 500 samples. This long-tailed distribution naturally leads to a few-shot learning scenario for underrepresented classes and provides strong motivation for adopting a few-shot class incremental learning setting, where base classes are trained with abundant data and later classes are introduced with limited samples.

Additionally, several defect classes exhibit similar visual characteristics, making them particularly challenging to distinguish. For instance, dark holes on Fig. 6d exhibit certain similarities to those observed in Fig. 6f and Fig. 6g. Furthermore, the public datasets indicated in Fig. 9, also demonstrate visual similarities between classes. Subfigures 9a-9d show samples from the CUB-200 dataset, where different bird species share highly similar appearances, such as white bodies and dark wings, making them difficult to distinguish at a glance. Subfigures 9e-9f display samples from the Stanford Dogs dataset, where dogs from different classes exhibit nearly identical color patterns, further emphasizing inter-class similarity. These examples demonstrate that finegrained classification challenges are not limited to industrial datasets but are also prevalent in widely used benchmarks, justifying the evaluation of the proposed method on both proprietary and public datasets.

A. EXPERIMENT SETUP

Throughout the experiments, the dataset is partitioned into multiple image groups d_t, where $0 \leq t \leq T$, and T denotes the number of incremental tasks for the dataset. The first data group d_0 is reserved for the base stage, whereas the subsequent image groups $\{d_1, \ldots, d_T\}$ are used for the corresponding incremental tasks t. To prevent data contamination during training, each data group is further split into training d_t^{train} and testing d_t^{test} subsets. The selection of the classes for the increment stage adheres to the original data setup of CEC [13], ensuring that the base dataset contains classes with sufficient training data availability, whereas the remaining incremental group data may have limited image samples.

During the experiments, each incremental task t undergoes three evaluations. The first evaluates the overall performance of the model, considering all testing data from all tasks $\{d_0^{test}, \ldots, d_t^{test}\}$, and is referred to as d_{all}^{test}. The second examines the impact of the new task on the performance of the model with respect to the previously known data, considering only the test data prior to the current task $\{d_0^{test}, \ldots, d_{t-1}^{test}\}$, and is denoted as d_{prm}^{test}. The last assesses the model performance on the newly introduced task, considering only the testing data from the current task $\{d_t^{test}\}$, and is referred to as d_{add}^{test}. These three evaluation metrics are used to compare the performance of five incremental learning models: CEC [13], FACT [10], BiDist [26], TEEN [27], AsyCLR [28], and the proposed FFCIL.

TABLE 4. Incremental data setup for A19, Stanford Dogs [30], and CUB-200 [29].

	A19	Stanford Dogs [30]	CUB-200 [29]
Incremental tasks	5	7	11
Base classes	15	60	100
Incremental classes per task	3	10	10
Total classes	27	120	200
Base samples	23,250	3,000	8,021
Incremental samples per task	15	50	50
Total samples	34,198	20,580	11,788

Following the scenarios outlined in the original FSCIL study [14], the CUB-200 dataset is partitioned such that the base class (d_0^{train}) contains 100 classes. Each increment (d_1^{train} to d_T^{train}) introduces ten new classes. Similarly, the Stanford Dogs dataset is partitioned into one base group (d_0^{train}) and six incremental groups (d_1^{train} to d_6^{train}), with each increment adding ten new classes. The A19 dataset follows a similar structure with one base group (d_0^{train}) and four incremental groups (d_1^{train} to d_4^{train}), with each increment adding three new classes. The number of samples for each base and incremental tasks in all datasets is listed in Table 4.

B. EVALUATION METRICS

For each testing data group, three multi-class classification accuracies are computed on d_{all}^{test}, d_{prm}^{test}, and d_{add}^{test}, denoted as ACC_{all}, ACC_{prm}, and ACC_{add}, respectively. The metric ACC_{all} represents the overall classification performance of the trained model C over all classes encountered up to task t, regardless of their order of occurrence. However, in the FSCIL setting, particularly under severe class imbalance, this aggregated accuracy alone may obscure the model's behavior toward different class groups. To address this, ACC_{prm} is introduced to measure the prediction accuracy on previously learned classes, thereby reflecting the model's ability to preserve knowledge acquired in earlier tasks, while ACC_{add} evaluates the recognition performance on newly introduced classes, explicitly capturing the model's plasticity when learning from limited samples. By disentangling overall performance, knowledge retention, and new-class adaptation, these complementary metrics provide a more representative and fair evaluation of FSCIL behavior than a single averaged accuracy. Furthermore, following [14], the performance dropping rate (PD) is calculated to quantify catastrophic forgetting by measuring the accuracy difference between the base task ($t = 0$) and the final task ($t = T$). Collectively, these evaluation metrics provide a comprehensive view of the model's response at each incremental task, both globally and with respect to the trade-offs between stability and plasticity.

V. RESULTS AND DISCUSSION

The study comprises two sets of experiments conducted with threefold cross-validation. The first set of experiments

evaluates the proposed approach against the state-of-the-art methods (CEC [13], FACT [10], BiDist [26], TEEN [27], AsyCLR [28]) with all three datasets (A19, Stanford Dogs [30], CUB-200 [29]). The second set of experiments conducts ablation studies using the A19 dataset, where one or more FFCIL components are removed, and various loss weight values are explored.

A. PERFORMANCE COMPARISON

Table 5 presents the overall accuracy ACC_{all} across the three datasets. On A19, FFCIL consistently surpasses most baseline methods throughout the incremental learning process. Notably, it demonstrates a substantial advantage over TEEN, with an average performance margin of up to 16.36%. When compared with FACT, FFCIL achieves comparable accuracy with a margin of 0.57%, averaged across all tasks. A similar pattern is observed on the Stanford Dogs dataset, where FFCIL remains competitive with FACT, showing an average gap of 0.76%, while significantly exceeding the performance of other baselines, particularly AsyCLR, by as much as 39.58%. On the CUB-200 dataset, even though TEEN achieves the highest accuracy at the base stage, its performance degrades more rapidly as the incremental tasks progress. In contrast, FFCIL maintains a more stable accuracy progression and consistently outperforms the remaining baseline methods across all incremental tasks by up to 31.81%. Collectively, these results underscore the robustness and general applicability of FFCIL across diverse datasets.

Although the overall accuracy ACC_{all} may appear moderate in comparison to certain baselines, additional factors provide further insight. The first is the performance dropping rate (PD), as detailed in Table 8. FFCIL achieves the lowest PD on both A19 (33.13%) and Stanford Dogs (15.16%) datasets, indicating superior resistance to catastrophic forgetting. In particular, the PD on Stanford Dogs is substantially lower than all baseline methods by 15.13%. On the CUB-200 dataset, FFCIL achieves a competitive PD of 20.64%, outperforming CEC, BiDist, TEEN, and AsyCLR, while remaining close to FACT. These results demonstrate that FFCIL effectively minimizes performance degradation when integrating new data, highlighting its robustness in preventing model performance loss.

The second aspect is the ACC_{prm} results. As presented in Table 6, on the A19 dataset, FFCIL consistently exceeds the performance of CEC (by 2.8%), BiDist (2.93%), TEEN (18.86%), and AsyCLR (16.43%), while having comparable results to FACT with only a 1.72% difference. A comparable trend is observed on the Stanford Dogs dataset, where FFCIL maintains high ACC_{prm} despite the increasing number of incremental tasks. The most pronounced improvement is observed relative to AsyCLR, reaching 44.68% at the final task and averaging 40.64% across all tasks. On the CUB-200 dataset, FFCIL consistently achieves the highest ACC_{prm} among all methods throughout the incremental process, with a maximum average margin of 34.48% over AsyCLR. These results demonstrate FFCIL's strong ability to retain knowledge from earlier tasks while accommodating newly introduced classes.

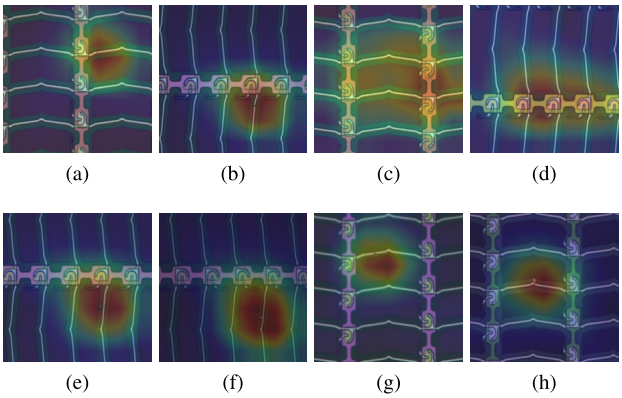


FIGURE 10. Heatmap visualizations of FFCIL predictions across three classes. Subfigures a and b show focused attention for correctly classified class 1 examples. Subfigures c and d illustrate misclassified instances of class 1 with scattered attention. Subfigures e, f, and subfigures g, h correspond to correctly classified examples from class 2 and class 8, respectively, highlighting the model's ability to distinguish visually similar classes by attending to discriminative regions.

Another indicator of FFCIL's generalization ability is its capability to acquire new knowledge during the incremental stage under the frozen classifier C, as measured by ACC_{add}. As shown in the Table. 7, FFCIL consistently delivers superior performance relative to nearly all baseline methods on the A19 dataset by up to 14.08% relative to TEEN. More pronounced gains are observed on the Stanford Dogs dataset, where FFCIL achieves an average improvement of 46.52% across all tasks. On the CUB-200, FFCIL maintains competitive performance relative to CEC and FACT, with average differences of 8.46% and 2.68%, respectively, while outperforming AsyCLR by an average margin of 35.89% across all tasks. These results highlight FFCIL's ability to extend learned knowledge throughout the incremental stage, even when dealing with limited and fine-grained samples.

Building on these observations, we further conducted paired t-tests to compare our FFCIL method against FACT, the strongest baseline across all metrics and datasets, evaluating ACC_{all}, ACC_{prm}, and ACC_{add}. On the A19 dataset, FFCIL significantly outperformed FACT on ACC_{add} ($p = 0.00043$), while improvements in ACC_{all} and ACC_{prm} showed positive trends but did not reach statistical significance ($p = 0.386$ and $p = 0.056$, respectively). For the CUB dataset, FFCIL achieved highly significant gains across all metrics, including ACC_{all} ($p = 4.8 \times 10^{-19}$), ACC_{prm} ($p = 7.6 \times 10^{-6}$), and ACC_{add} ($p = 0.00499$), demonstrating both superior base-class retention and incremental learning performance. On the Stanford dataset, FFCIL significantly outperformed FACT in ACC_{all} ($p = 0.0028$) and ACC_{add} ($p = 7.85 \times 10^{-6}$), with ACC_{prm} differences not reaching significance ($p = 0.322$). These results collectively indicate that FFCIL consistently improves over the most competitive baseline, particularly in the novel-class metrics (ACC_{add}),

TABLE 5. ACC_{all} comparison of five baseline methods and the proposed FFCIL.

t	0	1	2	3	4	5	6	7	8	9	10
Dataset: A19											
CEC [13]	91.77±1.4	79.21±2.1	69.51±1.2	62.00±1.1	56.00±0.7	–	–	–	–	–	–
FACT [10]	94.48±1.7	84.28±1.4	74.45±0.7	66.56±0.5	59.67±0.7	–	–	–	–	–	–
BiDist [26]	91.88±3.3	79.65±4.4	69.43±4.7	62.02±5.5	55.21±5.1	–	–	–	–	–	–
TEEN [27]	84.52±6.9	60.46±6.1	55.55±5.5	49.55±5.9	44.75±6.0	–	–	–	–	–	–
AsyCLR [28]	81.30±3.8	68.06±6.4	57.04±7.8	49.87±7.7	43.93±7.2	–	–	–	–	–	–
FFCIL	93.35±0.8	82.73±1.9	73.67±2.3	66.67±2.3	60.19±1.7	–	–	–	–	–	–
Dataset: Stanford Dogs [30]											
CEC [13]	74.91±0.7	67.15±0.8	62.29±1.2	57.69±1.1	54.57±1.6	51.87±1.6	48.74±1.3	–	–	–	–
FACT [10]	82.31±0.2	77.73±0.8	74.25±0.7	70.69±0.7	69.50±0.8	67.49±1.0	64.20±1.2	–	–	–	–
BiDist [26]	70.12±7.8	62.21±7.4	56.50±7.2	51.44±7.0	47.93±6.9	44.84±6.7	41.63±6.5	–	–	–	–
TEEN [27]	79.83±5.9	63.15±4.5	60.49±5.2	57.63±6.2	56.41±6.2	54.42±6.4	52.07±6.5	–	–	–	–
AsyCLR [28]	52.55±6.5	41.09±6.2	35.40±5.8	30.78±5.6	27.60±5.2	24.73±5.0	22.25±4.8	–	–	–	–
FFCIL	81.40±0.7	77.91±1.2	75.01±0.6	71.09±0.9	70.62±1.1	69.19±1.3	66.24±1.3	–	–	–	–
Dataset: CUB-200 [29]											
CEC [13]	74.69±0.5	70.49±0.6	66.85±0.7	63.16±1.1	60.58±0.9	57.79±0.8	56.10±0.8	54.62±0.9	52.47±0.6	51.88±0.8	50.32±0.7
FACT [10]	75.40±1.0	72.12±1.2	69.31±1.0	65.60±0.9	63.18±0.9	60.86±0.8	59.71±0.7	58.71±0.8	56.76±0.6	56.64±0.8	55.65±0.7
BiDist [26]	69.78±10.6	64.97±10.8	60.91±10.6	56.98±10.1	52.67±9.3	48.78±8.7	46.86±8.6	45.58±8.6	43.51±8.5	42.74±8.4	41.41±8.4
TEEN [27]	81.73±8.0	69.56±7.2	66.90±7.6	63.52±7.6	59.21±7.5	56.96±7.7	58.15±8.4	57.15±8.5	55.62±8.3	55.05±8.5	54.04±8.6
AsyCLR [28]	50.87±11.2	44.27±11.2	39.70±10.9	36.14±10.3	32.09±9.3	29.18±8.6	27.29±8.2	25.60±8.0	24.06±7.7	22.83±7.3	21.61±7.0
FFCIL	77.81±0.9	74.87±1.6	71.36±0.9	67.37±0.7	65.81±0.7	63.27±0.7	61.69±0.9	60.65±1.0	58.68±1.1	58.33±1.0	57.17±1.0

TABLE 6. ACC_{prm} comparison of five baseline methods and the proposed FFCIL.

t	1	2	3	4	5	6	7	8	9	10
Dataset: A19										
CEC [13]	89.85±1.6	78.03±1.8	68.53±1.2	60.96±0.9	–	–	–	–	–	–
FACT [10]	93.29±2.1	83.54±1.2	73.27±0.8	65.47±1.0	–	–	–	–	–	–
BiDist [26]	88.96±3.5	79.19±4.4	67.88±5.0	60.94±5.2	–	–	–	–	–	–
TEEN [27]	67.47±6.0	61.77±5.8	54.58±6.3	49.43±6.3	–	–	–	–	–	–
AsyCLR [28]	75.42±5.1	64.57±8.1	54.67±8.0	48.33±7.4	–	–	–	–	–	–
FFCIL	89.85±0.9	80.92±2.5	72.20±2.3	65.73±2.4	–	–	–	–	–	–
Dataset: Stanford Dogs [30]										
CEC [13]	72.07±0.7	65.45±1.1	60.82±1.2	56.32±1.2	53.57±1.5	50.74±1.4	–	–	–	–
FACT [10]	81.39±0.2	76.64±0.8	73.17±0.7	70.06±0.7	68.75±0.8	66.54±1.1	–	–	–	–
BiDist [26]	69.07±7.7	61.12±7.5	55.50±7.0	50.71±7.0	47.31±6.8	44.29±6.7	–	–	–	–
TEEN [27]	65.16±4.5	62.41±4.7	59.92±5.9	57.17±6.1	55.75±6.4	53.94±6.5	–	–	–	–
AsyCLR [28]	45.09±6.3	38.29±6.0	32.89±5.7	29.18±5.4	26.30±5.1	23.61±5.0	–	–	–	–
FFCIL	80.39±0.9	76.57±1.4	73.92±0.8	70.32±1.1	69.68±1.2	68.29±1.1	–	–	–	–
Dataset: CUB-200 [29]										
CEC [13]	72.46±0.6	69.64±0.7	66.09±0.7	62.30±0.9	59.82±1.0	57.38±0.8	55.38±0.8	53.94±0.9	51.96±0.7	51.34±0.8
FACT [10]	74.41±0.9	71.67±1.3	68.28±0.9	64.85±0.9	62.82±0.9	60.59±0.8	59.35±0.6	58.22±0.8	56.55±0.6	56.13±0.7
BiDist [26]	68.89±11.0	64.36±10.9	60.06±10.4	56.71±10.0	52.26±9.3	48.43±8.7	46.49±8.6	45.00±8.6	43.26±8.5	42.39±8.4
TEEN [27]	71.91±7.2	69.58±7.4	66.24±7.6	63.77±8.0	58.30±7.4	59.14±8.5	58.30±8.5	57.18±8.5	55.48±8.5	55.14±8.6
AsyCLR [28]	46.74±11.6	42.33±11.3	38.61±10.8	34.51±10.0	31.20±9.1	28.46±8.5	26.51±8.2	25.10±7.8	23.42±7.4	22.54±7.2
FFCIL	76.78±0.7	74.50±1.1	72.40±2.8	69.68±4.6	68.48±5.1	66.86±6.2	65.60±7.0	64.64±7.5	63.34±8.3	63.04±8.4

while maintaining or enhancing performance on previously learned classes.

To further examine FFCIL's decision-making process, we present representative heatmaps for eight examples across three classes (Fig. 10). Subfigures 10a and 10b show correctly classified instances of class 1, where the model clearly attends to class-relevant regions, demonstrating focused and semantically meaningful feature extraction. In contrast, subfigures 10c and 10d depict false positives for class 1, exhibiting more diffuse attention and highlighting regions that may have contributed to misclassification. Subfigures 10e and 10f illustrate true positives from class 2, while 10g and 10h correspond to true positives from class 8. Despite the high visual similarity among these last four samples and their slight resemblance to class 1, the model successfully differentiates the classes and concentrates on the discriminative features specific to each sample. These qualitative results complement quantitative metrics, providing clear evidence that FFCIL not only achieves high accuracy but also relies on meaningful, class-specific features, even in challenging few-shot incremental scenarios.

B. ABLATION TEST

The first ablation test removes one or more components and observes their impact on the accuracy of the first and

TABLE 7. ACC_{add} comparison of five baseline methods and the proposed FFCIL.

t	1	2	3	4	5	6	7	8	9	10
Dataset: A19										
CEC [13]	25.99±5.4	18.41±3.3	16.31±2.9	16.31±3.6	–	–	–	–	–	–
FACT [10]	39.24±4.3	19.94±2.3	19.60±3.2	13.28±3.1	–	–	–	–	–	–
BiDist [26]	33.05±9.1	10.85±6.8	20.98±9.2	9.30±5.0	–	–	–	–	–	–
TEEN [27]	25.43±6.4	18.21±3.5	14.34±3.5	7.31±3.2	–	–	–	–	–	–
AsyCLR [28]	31.24±12.7	11.86±6.1	16.23±5.4	8.70±5.6	–	–	–	–	–	–
FFCIL	47.16±7.5	30.17±3.6	28.36±3.9	15.91±4.2	–	–	–	–	–	–
Dataset: Stanford Dogs [30]										
CEC [13]	37.62±4.9	40.18±3.3	32.65±3.9	38.77±5.9	34.92±3.2	26.66±1.6	–	–	–	–
FACT [10]	55.76±6.5	57.55±0.6	50.84±1.2	64.43±2.2	54.79±3.7	38.39±4.8	–	–	–	–
BiDist [26]	21.01±5.5	24.13±5.0	18.96±6.5	22.97±5.8	20.14±6.0	12.42±4.2	–	–	–	–
TEEN [27]	51.06±5.0	47.03±8.4	39.27±8.7	49.59±6.4	41.10±6.2	31.45±6.3	–	–	–	–
AsyCLR [28]	17.11±5.5	15.20±4.5	13.84±4.2	13.41±3.7	9.09±3.9	7.33±2.7	–	–	–	–
FFCIL	62.54±3.8	61.90±2.6	52.29±3.7	70.37±3.3	63.02±4.1	44.96±2.1	–	–	–	–
Dataset: CUB-200 [29]										
CEC [13]	50.89±2.5	36.11±1.5	27.89±5.7	38.22±5.2	29.44±2.1	36.89±1.0	42.56±2.5	27.56±3.6	50.45±3.0	31.00±0.8
FACT [10]	49.22±4.6	43.33±2.6	33.44±0.2	41.56±3.2	33.44±1.1	46.44±3.5	48.44±4.0	32.00±3.0	58.33±3.6	46.33±1.8
BiDist [26]	25.78±8.0	22.95±7.2	20.05±6.1	0.12±0.1	0.06±0.1	23.22±8.5	31.02±9.4	18.19±6.9	33.31±7.4	22.87±8.9
TEEN [27]	46.09±7.3	37.40±9.9	30.84±7.2	0.00±0.0	38.16±12.1	43.30±6.5	38.63±10.0	29.18±5.2	47.25±9.3	33.12±9.2
AsyCLR [28]	19.50±6.8	10.85±6.6	6.51±4.3	0.67±0.7	0.91±1.1	9.70±4.3	10.90±5.0	6.34±4.7	12.26±4.8	4.00±3.0
FFCIL	50.39±2.2	46.65±4.9	31.64±2.2	50.31±3.1	39.98±1.7	44.51±2.6	49.71±3.2	36.65±2.9	59.92±1.0	45.84±0.6

TABLE 8. Comparison of performance dropping rate (PD) on all three datasets.

	A19	Stanford Dogs [30]	CUB-200 [29]
CEC [13]	35.77%	26.17%	24.37%
FACT [10]	34.81%	18.11%	19.75%
BiDist [26]	36.67%	28.49%	28.36%
TEEN [27]	39.76%	27.76%	27.68%
AsyCLR [28]	37.36%	30.29%	29.25%
FFCIL	33.16%	15.16%	20.64%

TABLE 9. Ablation study on removing individual components.

DCL [17]	MM [18]	triplet loss [19]	$ACC_4^{d_0}$	$ACC_4^{d_4}$
-	-	-	83.73%	15.26%
v	-	-	84.29%	19.90%
-	v	-	87.15%	23.44%
v	v	-	86.83%	27.81%
v	v	v	88.88%	28.55%

last image group (d_0 and d_4) at the last task ($t = 4$) on the A19 dataset. As shown in Table 9, removing all components results in the lowest accuracies (83.73% and 15.26%), indicating that none of the individual techniques alone is sufficient to address the FSCIL challenges. Adding DCL or MM individually increases $ACC_4^{d_0}$ by 0.56% and 3.42%, respectively, demonstrating their limited but complementary effects when applied in isolation. Notably, their combination boosts $ACC_4^{d_0}$ by 3.1%, and $ACC_4^{d_4}$ by 12.55%, an improvement that cannot be achieved by applying either component alone. This indicates that DCL and MM positively influence both previously learned classes and the newly introduced ones.

Furthermore, incorporating triplet loss yields additional gains of 5.15% and 13.29% for $ACC_4^{d_0}$ and $ACC_4^{d_4}$, respectively. These results confirm that the performance gains arise not from any single technique, but from the coordinated combination of DCL, MM, and triplet loss within the CEC framework, validating the effectiveness of the proposed FFCIL method. More importantly, the consistent performance gains observed across different component combinations indicate that the multiple applied losses improve generalization rather than causing overfitting, as no performance degradation or instability is observed when losses are combined.

TABLE 10. Ablation study on weight settings.

α	β	θ	Val C	Val S	Val R
1	1	1	56.40%	78.78%	79.03%
1	1	0.01	43.07%	45.31%	76.55%
1	0.01	1	43.64%	78.87%	94.42%
1	0.01	0.01	02.41%	44.06%	73.49%
0.01	1	1	12.56%	56.17%	71.23%
0.01	1	0.01	1.94%	44.30%	78.42%
0.01	0.01	1	02.09%	44.06%	63.97%
0.01	0.01	0.01	00.01%	43.62%	73.01%

The second ablation study investigates the effect of alternating the loss weights (9) between the recommended DCL values: 0.01 and 1. For each configuration, we report the validation accuracies of the classification (val C), discrimination (val S), and region alignment networks (val R), as the loss weights directly affect their respective optimization objectives. Therefore, it allows a closer analysis of weight sensitivity without requiring evaluation of the entire training pipeline.

As shown in Table 10, the configuration with all weights set to 1 yields the most balanced validation accuracies across all networks. Notably, this setting also yields the highest validation accuracy for the classification network.

An exception is observed when β is set to 0.01 while the other weights remain at 1. In this case, the region alignment subnetwork achieves a higher validation accuracy of 94.42%; however, the classification network accuracy drops to 43.64%, representing a decrease of 12.76% compared to the all-ones configuration. Since the classification network is directly used during the subsequent training stage, maintaining its performance is critical. Consequently, we select the configuration with all weights set to 1, as it provides the most stable and reliable performance across three DCL networks.

VI. CONCLUSION

This study introduces FFCIL, a few-shot class incremental learning method for fine-grained defect classification in thin-film transistor liquid crystal display. FFCIL demonstrates strong performance in addressing the challenge of incremental fine-grained datasets, achieving competitive or superior results against existing methods in our evaluations. This method employs destruction and construction learning to capture fine-grained features during pretraining, applies manifold-mixup as a placeholder strategy for seamless new classes integration with minimal samples, and utilizes triplet loss in the pseudo-incremental learning to refine classification. The experimental findings on the proprietary A19 dataset against TEEN show a consistent improvement, with a 16.36% (ACC_{all}), 18.86% (ACC_{prm}), and 14.08% (ACC_{add}). On the Stanford Dogs and CUB-200 public datasets, FFCIL achieves substantial performance gains, improving overall accuracy by up to 39.58% and 31.81%; increasing previous knowledge retention by up to 40.46% and 34.48%; and enhancing new knowledge acquisition by up to 46.52% and 25.89%, respectively. Furthermore, FFCIL exhibits the lowest performance dropping rate (PD), reducing it to as low as 15.16% on the Stanford Dogs dataset. These findings underscore FFCIL's adaptability across domains and its effectiveness in addressing data limitations in fine-grained defect classification systems.

REFERENCES

[1] Y.-P. Huang and K. Bhalla, "Automatic generation of laser cutting paths in defective TFT-LCD panel images by using neutrosophic Canny segmentation," *IEEE Trans. Instrum. Meas.*, vol. 71, pp. 1–16, 2022.

[2] Y. Liu, W.-T. Lee, H.-P. Lu, and H.-W. Chen, "A novel multi-modal learning approach for cross-process defect classification in TFT-LCD array manufacturing," *IEEE Trans. Semicond. Manuf.*, vol. 37, no. 4, pp. 527–534, Nov. 2024.

[3] C.-Y. Chang, A. Khanum, S.-G. Su, M. Lebaku Moses, and K.-X. Liu, "Vertical-line Mura defect detection for TFT-LCDs," *IEEE Access*, vol. 12, pp. 158927–158938, 2024.

[4] S. Luo, H. Chen, and B. Liu, "DACA-Net: Detail-aware network with contrast attention for locating liquid crystal display defects," *Displays*, vol. 87, Apr. 2025, Art. no. 102913.

[5] M. R. Islam, M. Z. H. Zamil, M. E. Rayed, M. M. Kabir, M. F. Mridha, S. Nishimura, and J. Shin, "Deep learning and computer vision techniques for enhanced quality control in manufacturing processes," *IEEE Access*, vol. 12, pp. 121449–121479, 2024.

[6] K. Masita, A. N. Hasan, T. Shongwe, and H. A. Hilal, "Deep learning in defect detection of wind turbine blades: A review," *IEEE Access*, vol. 13, pp. 98399–98425, 2025.

[7] Q. Ling and N. A. M. Isa, "Printed circuit board defect detection methods based on image processing, machine learning and deep learning: A survey," *IEEE Access*, vol. 11, pp. 15921–15944, 2023.

[8] P. Shourie, V. Anand, and S. Gupta, "Classifying mammographic images for predicting breast cancer using CNN," in *Proc. Int. Conf. Res. Methodologies Knowl. Manage., Artif. Intell. Telecommun. Eng. (RMKMATE)*, Nov. 2023, pp. 1–5.

[9] T. Ahmad, A. R. Dhamija, S. Cruz, R. Rabinowitz, C. Li, M. Jafarzadeh, and T. E. Boult, "Few-shot class incremental learning leveraging self-supervised features," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2022, pp. 3899–3909.

[10] D.-W. Zhou, F.-Y. Wang, H.-J. Ye, L. Ma, S. Pu, and D.-C. Zhan, "Forward compatible few-shot class-incremental learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 9036–9046.

[11] Y. Han, L. Wang, J. Liu, L. Yuan, H. Liu, B. Ma, and Z. Geng, "Novel intelligent defects detection of boiler water walls in thermal power plants based on ofm," *Displays*, vol. 85, Sep. 2024, Art. no. 102847.

[12] A. Suryarasmi, C.-C. Chang, R. Akhmalia, M. Marshallia, W.-J. Wang, and D. Liang, "FN-Net: A lightweight CNN-based architecture for fabric defect detection with adaptive threshold-based class determination," *Displays*, vol. 73, Jul. 2022, Art. no. 102241.

[13] C. Zhang, N. Song, G. Lin, Y. Zheng, P. Pan, and Y. Xu, "Few-shot incremental learning with continually evolved classifiers," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 12450–12459.

[14] X. Tao, X. Hong, X. Chang, S. Dong, X. Wei, and Y. Gong, "Few-shot class-incremental learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 12180–12189.

[15] P. Mazumder, P. Singh, and P. Rai, "Few-shot lifelong learning," in *Proc. AAAI Conf. Artif. Intell.*, 2021, pp. 2337–2345.

[16] S. Dong, X. Hong, X. Tao, X. Chang, X. Wei, and Y. Gong, "Few-shot class-incremental learning via relation knowledge distillation," in *Proc. AAAI Conf. Artif. Intell.*, vol. 35, 2021, pp. 1255–1263.

[17] Y. Chen, Y. Bai, W. Zhang, and T. Mei, "Destruction and construction learning for fine-grained image recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 5152–5161.

[18] V. Verma, A. Lamb, C. Beckham, A. Najafi, I. Mitliagkas, D. López-Paz, Y. Bengio, and B. Yoshua, "Manifold mixup: Better representations by interpolating hidden states," in *Proc. 36th Int. Conf. Mach. Learn. ICML*, 2018, pp. 6438–6447.

[19] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 815–823.

[20] D.-W. Zhou, Z.-W. Cai, H.-J. Ye, L. Zhang, and D.-C. Zhan, "Dual consolidation for pre-trained model-based domain-incremental learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2025, pp. 20547–20557.

[21] D. Zhou, K. Li, J. H. Ning, H.-J. Ye, L. Zhang, and D. Zhan, "External knowledge injection for CLIP-based class-incremental learning," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2025, pp. 3314–3325.

[22] Y. Wang, D.-W. Zhou, and H.-J. Ye, "Integrating task-specific and universal adapters for pre-trained model-based class-incremental learning," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2025, pp. 806–816.

[23] X. Sun, H. Xv, J. Dong, H. Zhou, C. Chen, and Q. Li, "Few-shot learning for domain-specific fine-grained image classification," *IEEE Trans. Ind. Electron.*, vol. 68, no. 4, pp. 3588–3598, Apr. 2021.

[24] K. Zhu, Y. Cao, W. Zhai, J. Cheng, and Z.-J. Zha, "Self-promoted prototype refinement for few-shot class-incremental learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 6797–6806.

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[26] L. Zhao, J. Lu, Y. Xu, Z. Cheng, D. Guo, Y. Niu, and X. Fang, "Few-shot class-incremental learning via class-aware bilateral distillation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2023, pp. 11838–11847.

[27] Q. Wang, D.-W. Zhou, Y. Zhang, D. Zhan, and H.-J. Ye, "Few-shot class-incremental learning via training-free prototype calibration," in *Proc. NeurIPS*, 2023, pp. 15060–15076.

[28] D. Liu, L. Zhao, Z. Zhang, F. Lyu, X. Fang, and L. Wang, "Few-shot class-incremental learning via asymmetric supervised contrastive learning," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 35, no. 10, pp. 9650–9664, Oct. 2025.
[29] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. (2010). *Caltech-UCSD Birds 200*. [Online]. Available: https://www.vision.caltech.edu/datasets/cub2002011
[30] A. Khosla, N. Jayadevaprakash, B. Yao, and F.-F. Li, "Novel dataset for fine-grained image categorization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2011, pp. 806–813. [Online]. Available: http://vision.stanford.edu/aditya86/ImageNetDogs/

ANINDITA SURYARASMI received the bachelor's and master's degrees in computer science from Universitas Gadjah Mada, Indonesia. She is currently pursuing the Ph.D. degree with the Department of Computer Science and Information Engineering, National Central University, Taiwan. She is also a Lecturer with the Department of Software Engineering Technology, Universitas Gadjah Mada. Her recent research interests include artificial intelligence, deep learning, and software engineering.

CHIA-YU LIN (Member, IEEE) received the B.S. and M.S. degrees in computer science from National Chiao Tung University (NCTU), Taiwan, in 2010 and 2012, respectively, and the Ph.D. degree from the Institute of Communications Engineering, NCTU, in 2019. She is currently an Associate Professor with National Central University. She has cooperated with many manufacturing companies to increase production efficiency, improve defect detection accuracy, and save machine maintenance costs with AI techniques. Her research interests include AI for smart manufacturing, real-time AI model updating techniques, and recommendation systems.

LE-CHIEN CHU received the master's degree in computer science from Yuan Ze University. His recent research interests include artificial intelligence and convolutional neural networks.

WEI-JEN WANG (Member, IEEE) received the Ph.D. degree in computer science from Rensselaer Polytechnic Institute (RPI), USA, in 2006. He is currently a Professor with the Department of Computer Science and Information Engineering, National Central University, Taiwan. His research interests include cloud computing, edge computing, smart manufacturing, distributed programming, high availability, and fault tolerance.

DERON LIANG (Member, IEEE) received the Ph.D. degree in computer science from the University of Maryland at College Park, USA, in 1992. He is currently a Professor and the Director of the Software Research Center, National Central University, Taiwan. His research interests include intelligent systems software, smart manufacturing, software engineering, data analysis, and information system security.