npj | heritage science

Article

https://doi.org/10.1038/s40494-025-01544-x

DFS: Dual-branch forward-looking simulation network for incremental learning of ancient Chinese characters

Check for updates

Qiuyue Ruan[1], Shanxiong Chen[1] ✉, Hailing Xiong[2] ✉, Yuqi Ma[1], Mingyue Yang[1], Wenjun Zhen[1], Jiang Yuan[1], Xiaoliang Li[3], Xun Pu[1] & Jian Qin[1]

Traditional Chinese ancient character recognition methods often fail to recognize new characters when working with continuously updated archeological materials. To cope with this ever-changing data stream, continual Learning becomes the key. Retraining the model using both new and old categories is a straightforward concept, but it is constrained by memory and data privacy issues. Currently, many existing approaches use incremental freezing technology, but due to the high similarity and few samples of ancient Chinese character datasets, typical incremental learning methods face tremendous challenges in this case. To this end, this paper proposes a forward-looking simulation network to pre-simulate unknown new categories through virtual sample generation technology. Specifically, we decouple the network into feature extractors and classifiers and expand the feature extractor into a dual-branch structure. During the basic training phase, techniques like Mixup are used to create deep virtual features and virtual datasets, which successfully enhances the basic model's capacity to represent new categories. Moreover, L-Selective Loss is proposed to further optimize the boundaries between categories. And enhance the extraction of identifiable high-level features between new categories and original categories. Experimental results show that the proposed method can effectively recognize all existing categories on oracle bone script, Yi script, and Dongba script without saving old category samples. Compared with the traditional incremental frozen framework FACT, the forgetting rate is improved by 2.021%, 0.949%, and 2.552%, respectively.

Open-set adaptive learning[1] is a popular issue for current study. Its key difficulty is how to successfully cope with continually changing and newly emerging category data. You can break the procedure into three essential steps, as shown in Fig. 1: (1) *Unknown rejection*: The model needs to reliably identify samples of known categories and detect unknown categories based on uncertainty estimation. (2) *New category discovery*: The unknown samples found are labeled with new categories. The labeling process can be performed manually or automatically. (3) *Incremental expansion*: As new category samples accumulate, the model expands the classifier through incremental learning and continuously adjusts to new data.

In the realm of ancient Chinese character recognition, open-set adaptive learning offers substantial research significance. There are various sorts of ancient Chinese characters, and existing databases make it difficult to cover all categories. When dealing with diverse ancient documents, typical closed-set recognition algorithms generally fail to distinguish unknown characters, resulting in frequent misrecognition. These unknown characters include rare, variant, and similar characters. After being gathered and identified by experts, new glyphs need to be added to the model for updating and learning. However, upgrading the model with only new class data will lead to forgetting the old classes[2,3]. Although teaching existing classes simultaneously with new classes is an appealing option, this strategy has apparent drawbacks in terms of memory and privacy. To tackle this problem, Fei Zhu et al.[4] proposed employing local softmax for classifier calibration, by learning richer, more universal, and transferable features and separating the training to calibrate the feature extractor. Kang et al.[5] prevented forgetting by limiting the updating of critical elements while allowing the less important aspects to change.

[1]College of Computer and Information Science, Southwest University, 400715 Chongqing, China. [2]College of Electronic and Information Engineering, Southwest University, 400715 Chongqing, China. [3]Chinese philology institute, Southwest University, 400715 Chongqing, China. ✉e-mail: csxpml@163.com; xionghl@swu.edu.cn

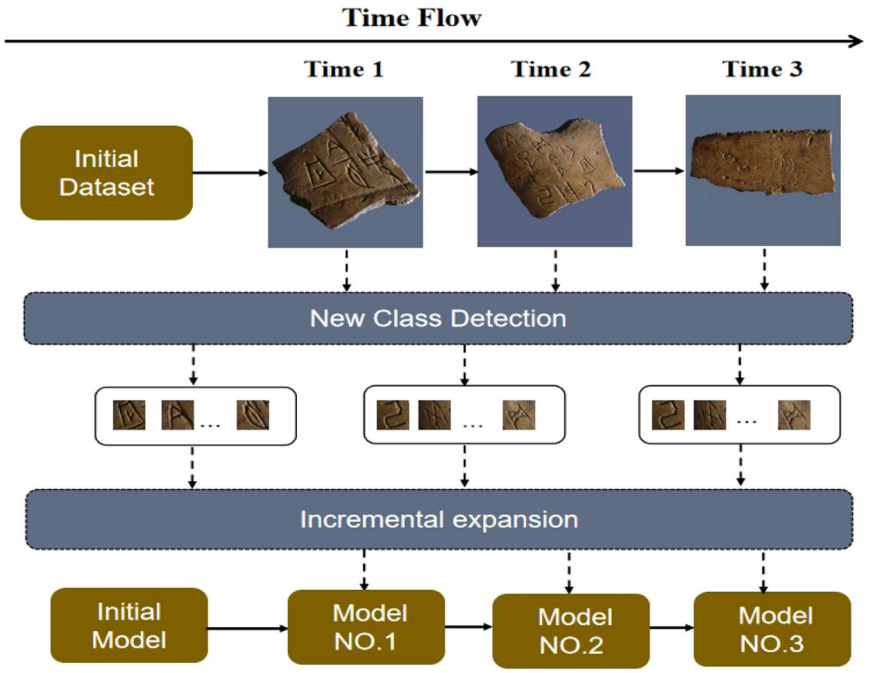

Fig. 1 | Continual learning process diagram.

Although there are many established methods, there are still significant challenges in the recognition of ancient Chinese characters. First, the training samples of new ancient Chinese characters are frequently restricted, and a small number of new class samples can readily lead to overfitting. Second, the ancient Chinese character dataset has a lot of similar characters, making it hard to tell them apart and causing the embedding space to overlap. Furthermore, there are not many studies on the continuous recognition of ancient Chinese characters, and the majority of incremental expansion techniques are mostly used on feature-rich traditional datasets. Ancient characters contain rich information, and typical feature extraction methods make it difficult to properly capture their complex features, which affects the recognition effect.

In recent years, the incremental freezing framework[6,7] has become the key to overcoming the problem of incremental recognition with few samples. Inspired by this, we propose a dual-branch forward simulation network (DFS). In view of the features of simple characters producing complicated glyphs in ancient Chinese characters, there is a considerable feature connection between the known ancient character dataset and the unknown ancient character dataset. In the first training stage of DFS, we combine the virtual dataset and deep virtual features created from known datasets into model training through a dual-branch structure, thereby keeping the prediction and accommodation capabilities of unknown characters in the embedding space. Further, we designed the L-Selective Loss to promote the network to accomplish mutual assistance optimization while extracting complicated features of virtual samples and original samples, and boost the recognition capacity of the model. During the incremental learning phase, when the training dataset only contains new class data, we freeze the feature extractor parameters and just replace the new class prototypes in the classifier for classification. The primary contributions of this study can be summarized as follows:

(1) For the first time, the incremental freezing architecture is used to continuously learn ancient Chinese characters, effectively addressing the challenges of open-set text recognition.
(2) A dual-branch simulation network is suggested to enable the initial model to forecast unknown categories. And in the incremental step, just a restricted number of new class samples are used to update the classifier, while yet maintaining good classification results.
(3) Designed the L-Selective Loss to assist in the fitting of virtual features and encourage the extraction of detailed features.
(4) The results of experiments on oracle bone script, Yi script and Dongba script datasets shown notable increases in accuracy and forgetting rates.

Related work

Incremental learning

Incremental learning(IL) aims to update the model when data comes in the form of streaming, without requiring retraining using the entire previous dataset. Research points out that in IL, a fatal problem occurs when updating the model with new classes, namely catastrophic forgetting[8,9]. To address the issue, current research is roughly divided into three categories: parameter regularization, knowledge distillation, and data playback. Parameter regularization[10–13] emphasizes constraints on important parameters. James Kirkpatrick et al.[10] propose to reduce forgetting by restricting important parameter variations that favor the learning of old tasks. Knowledge distillation[14–17] prevents forgetting by ensuring that the outputs of old and new models are consistent on the same data. And this method has become the basic module of many incremental learning methods because of its effectiveness. Data replay[18–20] focuses on preserving a small portion of old class data, and using it with new class data for model updates. For example, after completing each task, the iCaRL[18] method saves a small number of samples for each category, which are used for subsequent model training.

In the above IL method, new class data is used to adjust model parameters in the incremental phase. When new class data is limited, overfitting problems are prone to occur, leading to a significant decrease in the recognition performance of the model. As a result, research in recent years has turned to focus on Few-shot class-incremental learning (FSCIL), which aims to solve IL tasks in situations of insufficient data. Currently, FSCIL learning methods can be roughly divided into two categories: The first category is based on data augmentation[21–23], which alleviates the few-sample problem by increasing the diversity of existing data. The other category is based on metric learning[7,24–26], which classifies objects in the embedding space by calculating the similarity or distance between samples in the support set and the query set. In terms of applications, FSCIL has achieved remarkable results in various fields of computer vision, such as image

classification and target detection. However, ancient text recognition, as a typical few-shot recognition task, has not been studied in IL.

Open-set text recognition techniques

Open-set text recognition technology is designed to address the challenges of closed-set text recognition, focusing on processing new classes and new data[27]. Research in this field mainly focuses on the following two issues: (1) New class detection: The model is required not only to accurately classify known classes but also to effectively detect unknown classes. (2) New class recognition: After detecting new classes, the model should be quickly adjusted to recognize these new character classes.

In recent years researchers have mainly focused on the second problem, while most of the research work has concentrated on zero-shot text recognition. These methods can be divided into two categories: One is the classifier-based approach[10,28–31], which assigns the input data to known and unknown categories by learning generic classifiers in the training phase. To generalize to the unknown category, this type of approach requires shared features or attributes between the known and unknown categories. The other is the instance-based approach[32–36], which performs classification by comparing the similarity between test samples and the categories, both known and unknown. However, the real world is dynamic, many applications receive data in the form of manifolds, and new unknown categories will gradually emerge. This makes traditional zero-shot text recognition methods difficult to adapt to. In addition, the current IL framework has not been effectively applied in the field of open-set text recognition.

Methods

Problem formulation

In incremental learning, a model faces the sequence of consecutive tasks T_0, T_1, …, T_t. where T_0 represents the base stage, in this stage, there is a training dataset D^0_{train} and a test dataset D^0_{test}. The training dataset contains n training samples from the base stage, each sample represented by an image x_i and its corresponding label y_i. For T_1, T_2, …, T_t, representing incremental stage tasks, the training dataset for the t th stage is denoted as D^t_{train}, containing n_t training samples. Each sample's image is denoted as x_j, with the corresponding label y_j. It is important to note that the label space of the training dataset between different stages has no intersection. Specifically, the label space C^t_{train} of the training set for T_t does not include the label space of the training dataset from other stages. However, the label space of the test dataset between different stages is inclusive. More precisely, the test dataset for the subsequent stage should include the test datasets from all previous stages, i.e., the test label space for T_t is $C^t_{\text{test}} = C^t_{\text{test}} \cup C^{t-1}_{\text{test}} \cup \ldots \cup C^0_{\text{test}}$.

Throughout the learning process, aim to retain knowledge from previous tasks as each task is learned. In particular, the number of samples in ancient Chinese character recognition is limited, and the basic stage usually has sufficient training samples, while the subsequent incremental stages can only utilize a limited number of samples (i.e., few-shot incremental learning). Therefore, the datasets for each incremental stage are organized in an *N-way*, *K-shot* format, where each task comprises N categories, and each category contains K training images.

System overview

To address the challenge of the restricted dataset in the ancient Chinese character incremental recognition, some visionary work[7,24]recommends an incremental freezing framework. This framework adopts the prototype network[37] that decouples the classifier from the feature extractor. During the base training stage, the method reserves embedding space for potential new data in the future by introducing virtual prototypes. Subsequently, the model is optimized by minimizing the cross-entropy loss. Assuming the model f_n consists of the classifier g_e and the feature extractor f_e, the model parameters are continuously optimized through the following formula:

$$\mathrm{L}_{ce} = F_{ce}\left(g_e(f_e(Q,\theta), W^{\mathrm{T}}), y_q\right) \tag{1}$$

where F_{ce} represents the cross-entropy loss function[38], y_q denotes the true label corresponding to the input image, Q represents a training sample from the base training dataset D^0_{train}, θ represents the parameters of the feature extractor, and W^{T} represents the weights of the classifier.

After the completion of the base stage, the backbone network remains unchanged, and each incremental stage only extends the classifier. Specifically, the classifier weights are parameterized by the average embedding (i.e., prototype) for each class. Assuming the weights of the classifier g_e are represented as follows:

$$\mathrm{Weight}_{g_n} = [\mathrm{w}_0, \mathrm{w}_1, \mathrm{w}_3 \ldots\ldots \mathrm{w}_n, \mathrm{w}_{n+1} \ldots\ldots rmw_t] \tag{2}$$

where, w_0 to w_n represents the prototype representation of the classes in the base training samples, and w_n to w_t represents the prototype representation of the newly added classes after the incremental stage. The prototype representation is calculated as follows:

$$p^t_{\mathrm{c}} = \frac{1}{n^t_{\mathrm{c}}} \sum_{i=1}^{n^t_{\mathrm{c}}} f_{\mathrm{e}}(Q^i_{\mathrm{c}}, \theta) \tag{3}$$

where p^t_c represents the prototype representation, f_e represents the feature extractor, Q_c represents the current input sample Q belongs to category c, and θ represents the parameter weight

At each task stage, we measure the similarity between the input samples and the prototype representations of each category, and select the most similar category as the prediction result with the formula expressed as follows:

$$Q_{\mathrm{c}} = \arg\max\left(g_e\left(r^t_{\mathrm{c}} \cdot Weight_{g_n}\right)\right); \tag{4}$$

$$g_e = \frac{r^t_c \cdot Weight_{g_n}}{(\| r^t_c \| \cdot \| Weight_{g_n} \|)} \tag{5}$$

where g_n represents the cosine classifier, and r^t_{c} represents the feature representation of the current input sample obtained through the feature extractor.

However, there are several challenges with incremental recognizing ancient characters simply relying on the prototype network frozen in the incremental learning stage. First, the phenomenon of similar characters in the ancient character dataset is relatively serious, resulting in a decrease in model recognition. Also, in incremental learning, the frozen network parameters and the approach of replacing the virtual prototype with the new class prototype may lead the basic feature extractor to be unable to adequately represent the new class features, consequently affecting the classification performance. To overcome these issues, we propose an improved solution: expand the feature extractor to a dual-branch structure, as shown in Figure 2. The left branch creates virtual features and calculates the loss through the FFM module, while the right branch generates virtual data for training based on the CAM module. Although the two branches have different objectives, they share the same parameters and optimize the model together. We present L-Selective Loss as the goal function to further improve the distribution in the embedding space. During the incremental training stage, the feature extractor parameters remain frozen and the back-propagation process is no longer performed. Instead, the right branch that has not passed through the CAM module is directly used to extract new class features and generate class prototypes to replace the virtual prototypes in the classifier. Finally, classification is performed based on the cosine similarity with the class prototype. The specific method will be described in detail in sections "Dual-branch forward-looking simulation network" and "L-Selective Loss: combining L-Softmax and selective triplet loss".

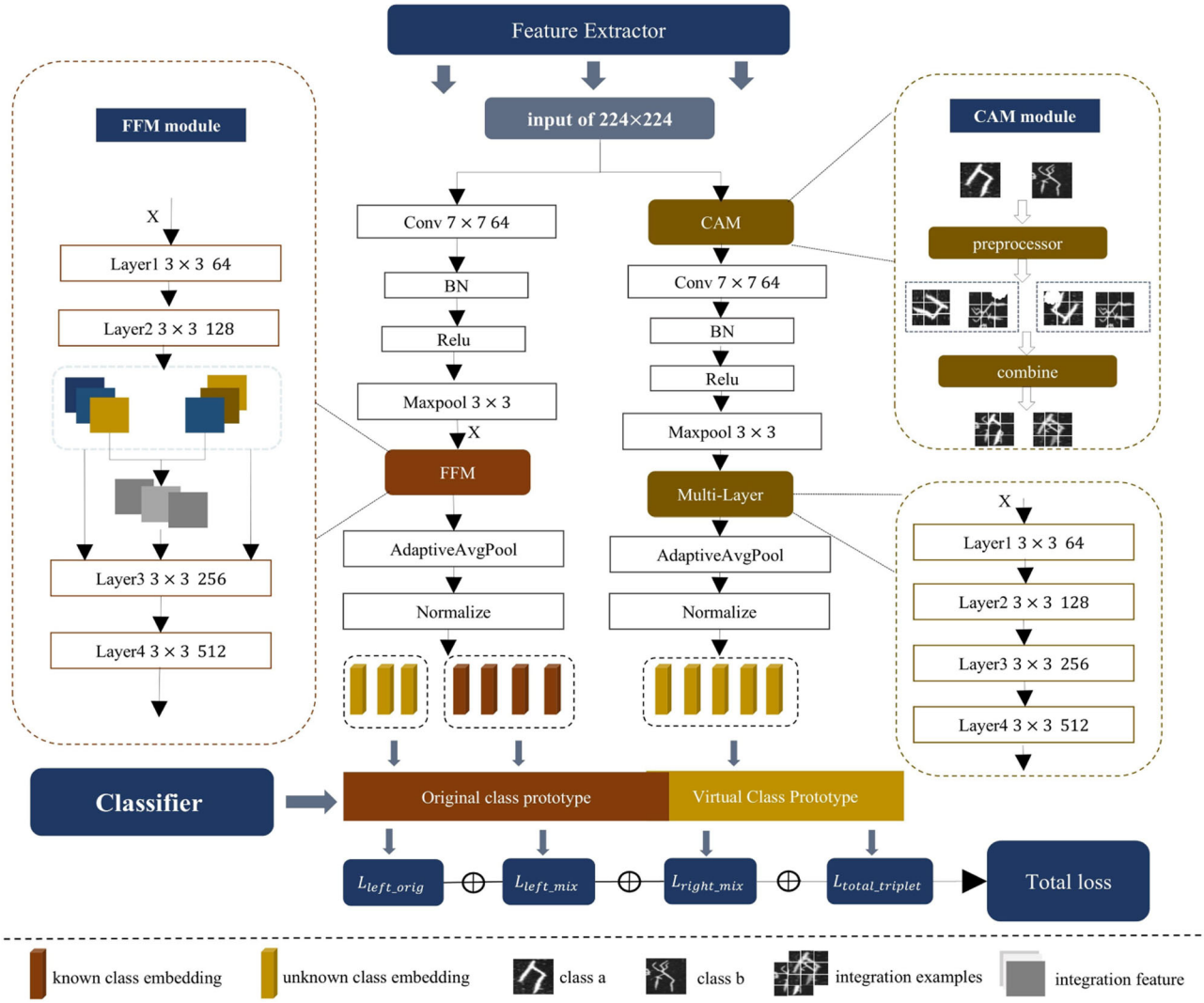

Fig. 2 | The overall framework structure of the model.

Dual-branch forward-looking simulation network

In incremental learning tasks, the model must continuously be exposed to new categories while maintaining a consistent recognition capacity for previously learned categories. To this purpose, we propose an innovative dual-branch feature extractor design to enhance the model's prediction ability on unknown categories. This design is based on the Manifold Mixup[39] and Mixup[40] methods, which use virtual prototypes for label assignment to push the model to optimize towards the feature space of new categories.

The traditional Manifold Mixup or Mixup method is trained by mixing samples and averaging weighted labels. It seeks to enhance the model's adaptability to data from old categories (such as deformed fonts, noise, etc.), thereby improving the model's generalization capacity. However, this label weighting mechanism has limitations in incremental learning tasks. Although weighted labels can expand the feature space of old categories, they fail to provide sufficient embedding space or feature update direction for new categories. As a result, the feature learning of new categories is restricted and "suppressed" in the feature space of old categories.

In contrast, our method is innovative and based on traditional methods. We propose a dual-branch feature extractor structure. The left branch generates deep virtual features based on the Manifold Mixup method, while the right branch uses a Mixup-like method to generate the original virtual dataset. By calculating the cosine similarity with the virtual prototype ensures that the mixed features can be more biased towards the learning of new categories, thereby providing sufficient feature embedding space for new categories, an algorithmic process as Algorithm 1.

Algorithm 1. Basic training phase algorithm flow

Require: Basic training stage data D^0_{train}
Ensure: Optimized feature extractor f_e and classifier g_e
1: **Input:** Training samples (x_i, y_i) and (x_j, y_j)
2: Use the shared feature extractor f_e to extract feature representations h_i and h_j
3: **Left Branch:**
4: Use Eq. (6) to get the mixed virtual features h_{mix} of h_i and h_j
5: Input h_{mix} into the subsequent network to generate the virtual feature $logit_{\text{left\_mix}}$
6: Use Eq. (6) to get the virtual label $y_{\text{left\_mix}}$
7: Input h_i and h_j into the network to generate the original features $logit_{\text{left\_orig}}$
8: Use Eq. (6) to get the virtual label y_{orig}
9: **Right Branch:**
10: Use Eq. (10) to get the reconstructed image x_{new}
11: Use Eq. (11) to get the virtual data x_{mix}
12: Extract feature representation of virtual data using shared feature extractor $logit_{\text{right\_mix}}$
13: Use Eq. (13) to get the virtual label $y_{\text{right\_mix}}$
14: **Loss calculation:**

15: Using Eq. (26), calculate the total loss function L_{total}
16: Update the parameters of the feature extractor f_e and the classifier g_e through backpropagation
return Optimized feature extractor f_e and classifier g_e

As shown in Fig. 2, the model adopts a dual-branch feature extractor structure with shared parameters. The left branch and the right branch share the same set of feature extractor parameters. However, each branch adopts a different data-mixing strategy to generate different feature representations.

Specifically, the left branch mixes the hidden features through the Manifold Mixup method to generate deep virtual features. Given two input samples (x_1, y_1) and (x_2, y_2). First, the feature extractor is used to obtain their feature representations h_1 and h_2 on Layer$_2$ hidden layer. Then, the mixing ratio of the two features is controlled by the hyperparameter λ sampled from the Beta distribution, thereby obtaining the mixed feature h_{mix}:

$$h_{mix} = \lambda h_1 + (1-\lambda)h_2 \tag{6}$$

The feature representation $\text{logit}_{\text{mix}}$ obtained by the subsequent network layer of the mixed feature h_{mix} will be calculated by cosine similarity with the virtual prototype p_i in the classifier, so as to assign the mixed feature to the most similar category label:

$$\text{cosine similarity}\,(logit_{\text{mix}}, p_i) = \frac{logit_{\text{mix}} \cdot p_i}{\| logit_{\text{mix}} \| \| p_i \|} \tag{7}$$

$$y_{\text{left\_mix}} = \arg\max_{i}\left(\text{cosine\_similarity}(\text{logit}_{\text{mix}}, p_i)\right) \tag{8}$$

Simultaneously, we will include the old class characteristics in the ensuing network layers for loss computation to preserve the previous class recognition. Two components make up the left branch's loss function: the loss determined by the original feature h_i and the label y_i, and the loss determined by the mixed feature h_{mix} and the label y_{mix}. Finally, the total loss of the left branch is the weighted sum of these two parts. The formula is expressed as

$$L_{\text{left}} = L_{\text{left\_orig}} + L_{\text{left\_mix}} \tag{9}$$

The right branch uses a Mixup-like method to mix data. Given input images x_1 and x_2, first divide them into N uniform blocks (for example, 9 blocks):

$$x_1 = \left[x_1^1, x_1^2, \ldots, x_1^N\right], x_2 = \left[x_2^1, x_2^2, \ldots, x_2^N\right] \tag{10}$$

Then, these blocks are shuffled and recombined to generate a new image x_{new_i}, which enhances the model's robustness to changes in image structure. After the images are reassembled, the standard Mixup operation is continued on these images, that is, weighted mixing. The ratio of the two images is controlled by the hyperparameter α sampled from the Beta distribution, and the final mixed image x_{mix} is obtained:

$$x_{\text{mix}} = \lambda x_{\text{new}_1} + (1-\lambda)x_{\text{new}_2} \tag{11}$$

This process helps avoid disrupting the coherence of text writing when reorganizing the blocks, ensuring that the structure of the image is maintained. Subsequently, the mixed image x_{mix} is passed to the shared feature extractor $f()$ to obtain its feature representation $logit_{right\_mix}$. Next, the cosine similarity between the mixed feature $logit_{right\_mix}$ and the virtual prototype p_i in the classifier is calculated, and the mixed image is assigned to the most similar category label:

$$\text{cosine similarity}\,(logit_{\text{right\_mix}}, p_i) = \frac{logit_{\text{right\_mix}} \cdot p_i}{\left\|logit_{\text{right\_mix}}\right\| \left\|p_i\right\|} \tag{12}$$

$$y_{\text{right\_mix}} = \arg\max_{i}\left(\text{cosine\_similarity}(\text{logit}_{\text{right\_mix}}, p_i)\right) \tag{13}$$

The loss function of the right branch is calculated from the characteristics of the mixed data, and the final loss is expressed as L_{right}. The total loss function of the model is a combination of the left branch loss and the right branch loss, where the left branch contains the loss of the original data and the mixed features, and the right branch only contains the loss of the mixed data set. The final total loss function is

$$L_{total} = L_{left\_orig} + L_{left\_mix} + L_{right\_mix} \tag{14}$$

During training, the left and right branches share the same set of parameters. Therefore, we do not need to update the parameters of the left and right branches separately. By calculating the total loss function, backpropagation will update the parameters of the shared feature extractor and classifier according to the total loss function, thereby minimizing the loss and optimizing the model performance in each round of training.

L-Selective Loss: combining L-Softmax and selective triplet loss

In this section, we will describe the Ltotal algorithm in detail. Here, we do not use the traditional cross-entropy loss function, but propose L-Selective Loss:

Virtual features are derived from original features, so they have certain similarities in the embedding space and also contain key features that are discriminative. Learning new classes is essentially a process of extracting information from these discriminative features. In the incremental learning stage, L-Selective Loss is suggested to enhance the model's capacity to capture the discriminative characteristics of both new and old classes. This loss combines Large-Margin Softmax Loss(L-Softmax)[41] and selective triplet loss[42] to optimize the category discrimination ability of the basic model from both global and local levels. L-Selective Loss first uses L-Softmax to strengthen the boundaries between categories at the global level, ensuring that each category in the embedding space has a compact intra-class distance and a large inter-class distance, thereby effectively avoiding the overlap of features between categories. Then, at the local level, the selective triplet loss focuses on sample pairs that are at the boundary of categories and are prone to misidentification (including virtual classes and virtual classes, virtual classes and original classes, and original classes and original classes), and further improves the model's clarity of similar feature boundaries through soft correction.

Large-Margin Softmax loss. The main principle of L-Softmax is to force similar samples to be closer by increasing the margin between categories, while the distance between heterogeneous samples is significantly increased. To gain better discriminative ability between categories. Different from the typical Softmax loss, L-Softmax adds a margin to explicitly optimize the borders between categories by modifying the calculation method of category logits.

Assume z_i is the feature vector of the ith sample, y_i is the target category of the sample, and the standard Softmax loss function is calculated as

$$L_{\text{CE}} = -\sum_{i=1}^{N} \log\left(\frac{\exp(z_{i,y_i})}{\sum_j \exp(z_{i,j})}\right) \tag{15}$$

where z_{i,y_i} is the *logits* value corresponding to category i. L-Softmax introduces margin by modifying the *logits*, making the logits of the target category larger and the logits of other categories smaller. Specifically, for the target category i, the formula of L-Softmax can be expressed as

$$\hat{z}_{i,y_i} = z_{i,y_i} + (m-1) \tag{16}$$

For non-target categories $j \neq y_i$, keep the original value:

$$\hat{z}_{i,j} = z_{i,j} \tag{17}$$

Among them, m is a margin hyper-parameter, which is usually set to a constant >1. By increasing the *logits* of the target category and keeping the logits of the non-target category unchanged, L-Softmax can force the model to enhance the distance between categories during training.

Next, the modified logits are normalized by softmax to get the predicted probability of the category:

$$L_{\mathrm{CE}} = -\sum_{i=1}^{N} \log\left(\frac{\exp(\hat{z}_{i,y_i})}{\sum_j \exp(\hat{z}_{i,j})}\right) \tag{18}$$

This modification can effectively increase the gap between the target category and other categories. The left and right branches in DFS are calculated using this loss function, and this loss function acts on both the original dataset and all virtual datasets. Therefore, L_{left_orgin},L_{left_mix}, and L_{right_mix} are:

$$L_{\mathrm{left\_origin}} = -\sum_{i=1}^{N_{\mathrm{origin}}} \log\left(\frac{\exp(\hat{z}_{i,y_{\mathrm{origin}}})}{\sum_j \exp(\hat{z}_{i,j})}\right) \tag{19}$$

Where N_{origin} is the number of samples in the original dataset, $\hat{z}_{i,y_{\mathrm{celgin}}}$ is the predicted value in the original dataset, and y_{origin} is the label of the original dataset.

$$L_{\mathrm{left\_mix}} = -\sum_{i=1}^{N_{\mathrm{left\_mix}}} \log\left(\frac{\exp(\hat{z}_{i,y_{\mathrm{left\_mix}}})}{\sum_j \exp(\hat{z}_{i,j})}\right) \tag{20}$$

Where $N_{\mathrm{left\_mix}}$ is the number of samples in the mixed dataset, $\hat{z}_{i,y_{\mathrm{left\_mix}}}$ is the predicted value of the mixed dataset, and $y_{\mathrm{left\_mix}}$ is the label of the mixed dataset, which includes the original class and the virtual class.

$$L_{\mathrm{right\_mix}} = -\sum_{i=1}^{N_{\mathrm{right\_mix}}} \log\left(\frac{\exp(\hat{z}_{i,y_{\mathrm{right\_mix}}})}{\sum_j \exp(\hat{z}_{i,j})}\right) \tag{21}$$

Where $N_{\mathrm{right\_mix}}$ is a virtual number set, $\hat{z}_{i,y_{\mathrm{right\_mix}}}$ is the predicted value of the virtual data of the right branch, and $y_{\mathrm{right\_mix}}$ is the total label of the virtual class.

Adding these parts together, we get the final total L-softmax loss as

$$L_{\mathrm{CE\_total}} = L_{\mathrm{left\_origin}} + L_{\mathrm{left\_mix}} + L_{\mathrm{right\_mix}} \tag{22}$$

Selective Triplet Loss. Selective Triplet Loss selectively guides positive and negative samples, allowing the model to fine-tune the learning of sample pairs that are at the boundary of categories and prone to misidentification. Different from the traditional triplet loss, the selective triplet loss introduces two types of samples: original data and virtual data. It also optimizes the embedding space through aiming point selection and negative sample selection.

Specifically, for the selection of positive samples, the selective triplet loss treats all samples as positive samples in each training round. According to the category of the positive sample (original data or virtual data), we further refine the selection process into the following two cases:

1. Original data is a positive sample:

Anchor point selection: Select the class prototype of the original data x_i as the anchor point C_{y_i}, which is the aggregate representation of all samples of this class.

Negative sample selection: The current data includes two negative samples. Negative sample1 is the original sample a of other categories that is most similar to the C_{y_i} prototype; negative sample2 is the virtual sample b that is most similar to the C_{y_i} prototype.

For this case, the triplet loss is calculated as

$$\begin{aligned} L_{\mathrm{triplet\_orig}} = {} & \max(0, d(c_{y_i}, x_i) - d(c_{y_i}, a) + m) \\ & + \max(0, d(c_{y_i}, x_i) - d(c_{y_i}, b) + m) \end{aligned} \tag{23}$$

Where m is a constant representing the minimum spacing between positive and negative samples.

2. Virtual data is a positive sample:

Anchor point selection: The aiming point of the virtual data x_j is the virtual prototype C_{vir} that has the greatest similarity with the virtual data.

Negative sample selection: The current data includes two negative samples. Negative sample1 is the negative sample c selected from the original data with the greatest similarity to the virtual prototype C_{vir}. Negative sample2 is the negative sample d selected from the virtual data with the greatest similarity to the virtual prototype C_{vir}.

For this case, the triplet loss is calculated as

$$\begin{aligned} L_{\mathrm{triplet\_vir}} = {} & \max(0, d(c_{\mathrm{vir}}, x_j) - d(c_{\mathrm{vir}}, c) + m) \\ & + \max(0, d(c_{\mathrm{vir}}, x_j) - d(c_{\mathrm{vir}}, d) + m) \end{aligned} \tag{24}$$

The final total triplet loss is the sum of the two triplet losses:

$$L_{\mathrm{triplet\_total}} = L_{\mathrm{triplet\_orig}} + L_{\mathrm{triplet\_vir}} \tag{25}$$

Algorithm 2. **Incremental learning phase algorithm flow**

Require: Incremental phase new class data D^t_{train} $(t \geq 1)$, optimized feature extractor f_e and classifier g_e

Ensure: Updated classifier g_e

1: **Input:** New class data x_t in the incremental phase
2: Freeze the parameters of the feature extractor f_e
3: Use the shared feature extractor f_e to extract the feature representation r^t_c of the new class data
4: Calculate the true class prototype p^t_c of the new class
5: Replace the corresponding virtual prototype in the classifier with the class prototype p^t_c of the new class

return Updated classifier g_e

Total loss function. L-Selective Loss is the total objective function L_{total} of DFS, which consists of global loss (L-Softmax) and local loss (selective triplet loss), expressed as

$$L_{\mathrm{total}} = L_{\mathrm{Selective\_loss}} = L_{\mathrm{CE\_total}} + L_{\mathrm{triplet\_total}} \tag{26}$$

In the basic training stage (Task 0), L-Selective Loss acts on the left and right branches of the dual-branch structure at the same time and back-propagates through the parameters of the shared feature extractor to optimize the model's prediction ability for unknown classes. After entering the incremental learning stage (Task 1, Task 2, etc.), the parameters of the feature extractor are frozen and no longer updated, an algorithmic process as Algorithm 2. At this time, the basic feature extractor is only used to extract feature representations of new class data without back-propagation. For each incremental task, we obtain the feature representation of the new class through the basic feature extractor and calculate the true class prototype of the class to replace the position of the virtual class in the classifier. The class prototypes of these new classes will be used for classification prediction of the new classes.

Discussion: There are limitations to using L-Softmax or selective triplet loss by itself. By increasing the border distance across categories, L-Softmax may increase global category discrimination; however, it is ineffective at optimizing for sample pairs with fuzzy boundaries or high similarity, which makes it easy for these samples to be confused. Although it improves the processing of comparable sample pairs and concentrates on local

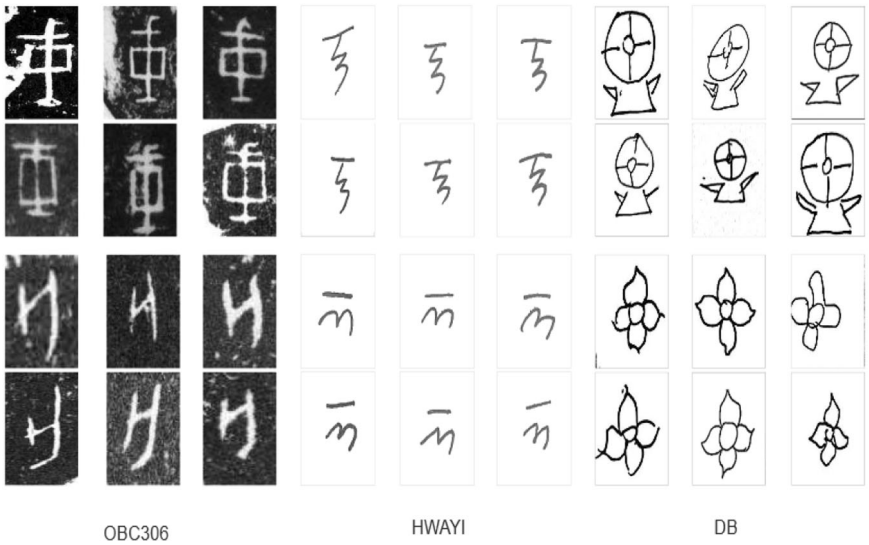

Fig. 3 | Datasets.

optimization, the selective triplet loss does not have a strong enough spatial expansion impact between global categories. Thus, L-Selective Loss combines L-Softmax with selective triplet loss, which increases the processing capacity of highly comparable data by local optimization in addition to increasing the global distance across categories. Simultaneously, L-Selective Loss may enhance the proportion of virtual features in the embedding space, which would enhance the capacity to capture novel features during the learning phase.

Experiment

Datasets

The paper conducted experiments on three datasets: HWAYI scripts, OBC306 scripts, and DB scripts, as shown in Fig. 3. The HWAYI scripts dataset contains a total of 112,031 handwritten Ancient Yi characters in 1764 categories (27–96 images per category), and each character image has been normalized to a size of 64 × 64. OBC30643 is a dataset of rubbing oracle bone characters. It contains a total of 309,551 images covering 306 different categories. The smallest category has only one image and the largest category has 25,898 images. The DB scripts dataset contains 2306 categories of Dongba characters, each category contains 1–50 images. The paper randomly selects 100 categories in the HWAYI scripts, OBC306 scripts, and DB scripts datasets as experimental data, denoted as HWAYI100, OBC 100, and DB100 (Due to limitations of experimental conditions, only 100 categories were selected for experiments.), each category contains 27–96 images. To make the model more generalizable, this paper did some preprocessing operations on the HWAYI dataset. The original black and white Yi text pictures were added with a background color, and some of the pictures inside were randomly rotated and cropped.

Dataset split

The three datasets, HWAYI100, OBC100, and DB100, were divided into 60 base classes and 40 new classes, respectively. The new classes were categorized into three task formats: eight 5-*way*, 5-*shot* incremental tasks, ten 4-*way*, 5-*shot* incremental tasks, and twenty 2-*way*, 5-*shot* incremental tasks. Among them, the eight 5-*way*, 5-*shot* incremental tasks represent 8 incremental stages, each incremental stage joins 5 new classes, and each new class contains 5 training samples(way represents the number of categories added to the new category in each incremental stage, and shot represents the number of sample images used for training for each new category). The rest of the tasks form meanings and so on. In the base training phase of the model, the focus is on these 60 base classes for training to establish the base performance. For a fair comparison, the same training segmentation strategy (both base and incremental phases) is used for each comparison method.

Training details

All models were implemented using PyTorch[43]. For all compared methods, the same network backbone was used. For the HWAYI100, OBC100, and DB100 datasets, ResNet 20[44] was employed with a batch size of 64 for 50 training epochs, optimizing with SGD with momentum. The learning rate started at 0.1 and gradually decreased.

Evaluation metrics

We represented the *Top*-1[45] accuracy after the *i*th session as A_i. We also quantitatively measured the forgetting phenomenon using the Performance Drop (*PD*), where $PD = A_0 - A_B$, with A_0 denoting accuracy after the baseline session and A_B denoting accuracy after the last session.

The lower the *PD*, the lower the forgetting phenomenon of the model, and the better the recognition effect of new and old classes. ΔPD is the difference in PD value between the method proposed in this article and other methods, expressed as a formula is $\Delta PD = PD_{ours} - PD_i$, i denotes other methods.

Benchmark comparison

Our method is compared with the classic few-class incremental learning methods in recent years on the HWAYI100, OBC100, and DB100 datasets, including CLOM[46], CEC[25], SAVC[24], FACT[7], and S3C[26]. Additionally, a baseline called "Fine-tuning" was established, which directly fine-tuned the decoupled prototype network using limited data. Performance curves are presented in Fig. 4, and detailed numerical values for the OBC100, HWAYI100, and DB100 datasets are provided in Tables 1, 2, and 3.

As can be seen from Tables 1–3, the paper-proposed method consistently outperforms the FACT method by 1–3% on all datasets. In the table, 0 represents the basic training stage, 1–8 represents the incremental stage, and each incremental stage adds 5 new categories of data. The performance difference in the incremental stage suggests that DFS strengthens the network's fit to unknown classes. Secondly, SACV generates virtual classes through predefined transformations. It can be seen from the table that it does not work well for the incremental recognition task of ancient characters. This is because each virtual class in SACV is equivalent to a "fine-grained" class derived from the original class. For ancient Chinese characters, such virtual classes are highly similar to the original classes, which can easily lead to prediction errors in the model. This paper proposes to generate virtual information from deep feature fitting and structural fitting, which is more in line with the glyph features of the ancient Chinese dataset. And combined with L-Selective Loss to refine the distinction between virtual classes and original classes. In summary, this method has strong applicability in the

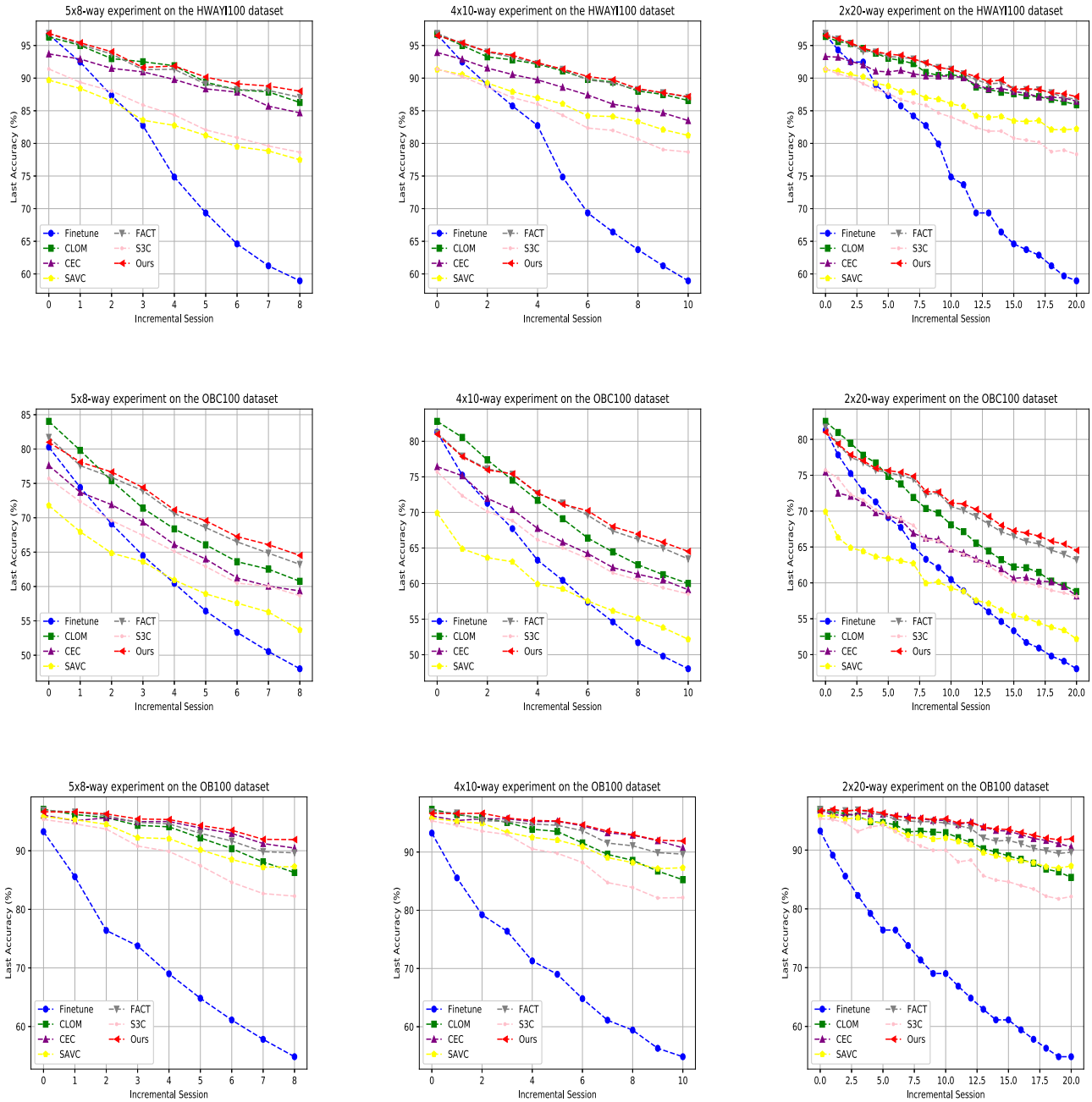

Fig. 4 | Comparative experiments of *k*-way on HWAYI100, OBC100, and DB100 datasets.

Table 1 | Detailed accuracy for each incremental session on the OBC100 dataset

Method	Accuracy in each session↑										
	0	1	2	3	4	5	6	7	8	PD↓	ΔPD
Finetune	80.283	74.416	69.101	64.494	60.463	56.432	53.322	50.536	48.027	32.256	+**15.785**
CEC[25]	77.605	73.712	71.908	69.389	66.097	64.0	61.244	60.053	59.354	18.251	+**1.78**
CLOM[46]	84.03	79.81	75.39	71.39	68.38	66.07	63.61	62.52	60.76	23.27	+**6.799**
S3C[26]	75.709	72.355	69.59	67.43	65.106	62.909	60.489	60.049	58.667	17.042	+**0.571**
FACT[7]	81.704	77.593	75.888	73.924	70.639	68.571	66.492	64.855	63.221	18.483	+**2.012**
SAVC[24]	71.79	67.966	64.863	63.615	60.929	58.935	57.579	56.284	53.679	18.111	+**1.67**
Ours	80.994	**78.102**	**76.67**	**74.405**	**71.15**	**69.571**	**67.259**	**66.089**	**64.523**	**16.471**	

The bold values in the last row represent the recognition accuracy of our proposed method in the incremental stage, and the bold values in the last column represent the improved accuracy of our proposed method in the forgetting rate indicator compared with other methods.

Table 2 | Detailed accuracy for each incremental session on the HWAYI100 dataset

Method	Accuracy in each session↑										
	0	1	2	3	4	5	6	7	8	PD↓	ΔPD
Finetune	96.677	92.463	87.326	82.73	74.851	69.347	64.598	61.242	58.945	37.732	+**28.887**
CEC[25]	93.71	92.921	91.501	90.959	89.787	88.356	87.853	85.70	84.684	9.026	+**0.181**
CLOM[46]	96.32	95.02	92.99	92.51	91.90	89.37	88.21	87.89	86.27	10.05	+**1.025**
S3C[26]	91.397	89.363	88.026	85.882	84.366	82.068	80.883	79.608	78.641	12.756	+**3.911**
FACT[7]	96.851	95.14	93.692	91.306	91.353	89.08	88.243	88.114	87.057	9.794	+**0.949**
SAVC[24]	89.686	88.413	86.495	83.55	82.753	81.227	79.504	78.852	77.484	12.202	+**3.357**
Ours	96.851	**95.417**	**94.047**	**91.631**	**91.864**	**90.159**	**89.142**	**88.787**	**88.006**	**8.845**	

The bold values in the last row represent the recognition accuracy of our proposed method in the incremental stage, and the bold values in the last column represent the improved accuracy of our proposed method in the forgetting rate indicator compared with other methods.

Table 3 | Detailed accuracy for each incremental session on the DB100 dataset

Method	Accuracy in each session↑										
	0	1	2	3	4	5	6	7	8	PD↓	ΔPD
Finetune	93.256	85.562	76.395	73.761	69.002	64.82	61.116	57.812	54.848	38.408	+**33.698**
CEC[25]	96.067	95.118	95.611	94.949	94.985	93.885	92.957	91.228	90.447	5.62	+**0.91**
CLOM[46]	97.07	96.18	95.66	94.33	94.08	92.18	90.36	88.08	86.28	10.79	+**6.08**
S3C[26]	95.341	94.585	93.697	90.795	89.885	87.441	84.608	82.689	82.255	13.086	+**8.376**
FACT[7]	96.906	96.602	95.981	94.844	94.59	92.973	91.652	89.84	89.644	7.262	+**2.552**
SAVC[24]	95.804	95.296	94.506	92.229	92.06	90.153	88.50	87.176	87.299	8.505	+**3.795**
Ours	96.639	**96.612**	**96.284**	**95.425**	**95.325**	**94.311**	**93.50**	**91.946**	**91.899**	**4.74**	

The bold values in the last row represent the recognition accuracy of our proposed method in the incremental stage, and the bold values in the last column represent the improved accuracy of our proposed method in the forgetting rate indicator compared with other methods.

Table 4 | Ablation experiments on the OBC100 dataset

L_{CE_total}	$L_{triplet\_total}$	CAM	FFM	Accuracy in each session↑										
				0	1	2	3	4	5	6	7	8	PD↓	ΔPD
×	×	×	×	80.827	77.525	75.024	73.024	69.651	68.00	65.87	64.141	62.487	18.34	
✓	×	×	×	80.667	77.881	75.707	73.942	70.522	68.584	66.431	65.023	63.478	**17.189**	+1.151
×	✓	×	×	80.364	77.153	74.85	73.27	69.918	68.0	66.102	64.8	63.022	**17.342**	+0.998
×	×	✓	×	80.525	76.932	74.077	73.017	69.793	67.935	65.87	64.224	62.922	**17.603**	+0.771
×	×	×	✓	80.37	77.153	74.632	73.086	69.827	68.338	66.114	64.262	62.801	**17.569**	+0.737
×	×	✓	✓	81.284	78.305	75.516	74.118	70.657	68.831	66.626	65.261	63.544	**17.74**	+0.6
×	✓	✓	✓	81.123	78.661	75.61	74.43	71.349	69.468	67.211	65.921	64.079	**17.044**	+1.296
✓	✓	✓	✓	80.994	78.102	76.67	74.405	71.15	69.571	67.259	66.089	64.523	**16.471**	+1.869

The bold values in the last row represent the recognition accuracy of our proposed method in the incremental stage, and the bold values in the last column represent the improved accuracy of our proposed method in the forgetting rate indicator compared with other methods.

incremental recognition task of ancient characters and performs well compared with other methods.

Furthermore, Fig. 4 presents the comparison results for different *k*-way experiments on the HWAYI100, OBC100, and DB100 datasets. Observing the results, DFS performance still does not drop significantly as the number of new categories increases. And in incremental experiments, whether adding 2 or 5 new classes per session, our proposed method still outperforms other incremental learning methods in terms of the forgetting rate metric.

Note: In Tables 1 and 2, the recognition accuracy of the basic stage (0 session) is slightly lower than that of the CEC and FACT models. This is because the use of known classes to pre-simulate unknown classes in the basic training stage affects the recognition ability of the model in the basic stage. However, this effect is negligible compared to the improvement of the model effect in the incremental stage. Therefore, we believe that this method is applicable and effective.

Ablation study

To assess the impact of each component, an ablation study was conducted, and the experimental results on the OBC100 dataset are summarized in the table. We use the incremental model without using CAM, FFM, and L-Selective Loss as a baseline comparison. The following is a detailed analysis of the experimental results: First, we examined the performance of each component separately. From Table 4, we can see that each component improves the model's forgetting rate and incremental stage accuracy to a certain extent. Second, we examine the impact of the CAM and FFM. The

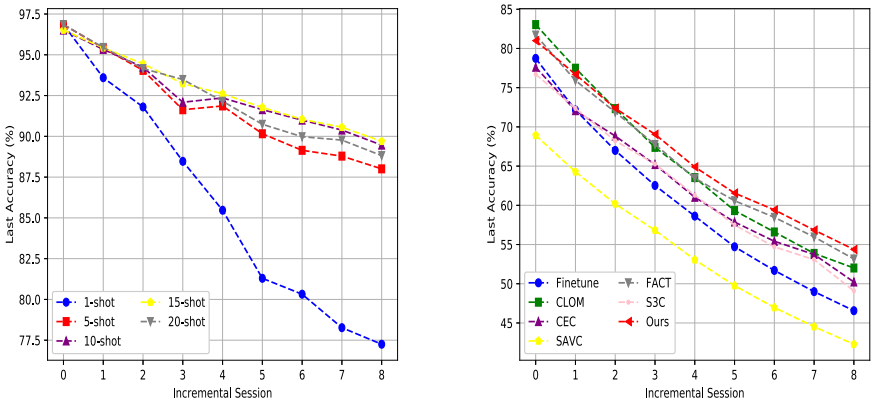

Fig. 5 | The left picture shows a *k*-shot comparison experiment on the HWAYI100 dataset, and the right picture shows a 1-shot comparison experiment on the OBC100 dataset.

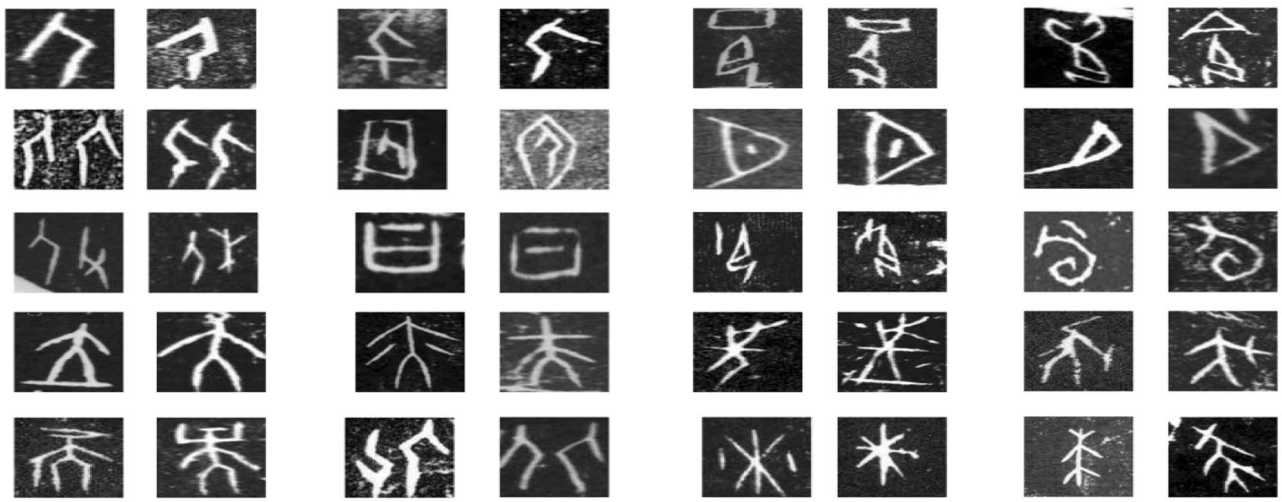

Fig. 6 | Sample similar dataset.

two modules effectively fill the embedding space of unknown classes from the aspects of feature and structure fitting respectively. The results in Table 4 show that the virtual information fitting method significantly improves the accuracy of the baseline model. The forgetting rate indicator increased by 0.60%. Improves the network's prediction ability for unknown categories. One point that needs clarification is that adding only the FFM component is slightly better than using both FFM and CAM components together on the forgetting rate metric. However, the classification accuracy indicators in the basic stage and incremental stage are better than just adding FFM. Therefore, we believe that the combined use of FFM and CAM has a promotional effect.

Finally, we propose to use the L-Selective Loss to further improve model performance. This method aims to maximize the separation of similar instances, reduce the mixing of virtual instances and original instances, and provide more spatial locations for the embedding of unknown classes. As shown in the results in Table 4, the method further increased the model's forgetting rate indicator by 1.869%. This shows that the two improved modules proposed in this paper complement each other, enhance the model's adaptability to unknown classes, and show better performance on the OBC100 dataset. It is worth noting that similar results were also observed on the HWAYI100 and DB datasets, further verifying the effectiveness and versatility of our method.

Additional experiments

Incremental experiments. To further verify the performance of this model in few-shot incremental learning tasks. As shown in the left picture of Fig. 5. The experimental results on the HWAYI100 dataset for *k*-shot experiments are presented, where *k* takes values of 1, 5, 10, 15, and 20. Among them, 1-shot means that only one sample of each unknown class is used for experiments in each incremental stage. Similarly, 5-shot, 10-shot, 15-shot, and 20-shot correspondingly denote experiments involving five, ten, fifteen, and twenty samples for each unknown class, respectively.

Experimental results show that in the incremental phase, the recognition accuracy of the model in the 1-shot experiment drops significantly. When each new class contains greater than or equal to 5 samples, the model's recognition accuracy remains between 87.5% and 97.5%. To further illustrate DFS's performance in Few-Shot Class-incremental learning tasks, conducted 1-shot experiments for each classic model on the OBC100 dataset, as shown in the right picture in Fig. 5. Note that the comparison model used here is consistent with the model used in the section "Datasets". It is observed that for the 1-shot experiment, the accuracy of each model decreases in the incremental stage, but our method still has advantages over other methods.

Similar character separation experiment. To verify the effectiveness of the L-Selective Loss, 20 groups of similar characters were selected from the OBC-100 dataset for experiments, as shown in Fig. 6. This group of similar-shaped characters is used as a training dataset, as shown in Fig. 7, which shows the t-SNE[47] embedding space visualization results of three methods: SACV, FACT, and DFS(Ours). During the experiment, only the Shape-Similar Character Separation Optimization Loss was used for model optimization. By observing the experimental results, it was found that SAVC performed poorly in the recognition task of similar-shaped characters, while FACT and DFS(Ours) had better separation effects on them. In contrast, DFS(Ours) further reduces the intra-class distance of similar characters and enlarges their inter-class distance.

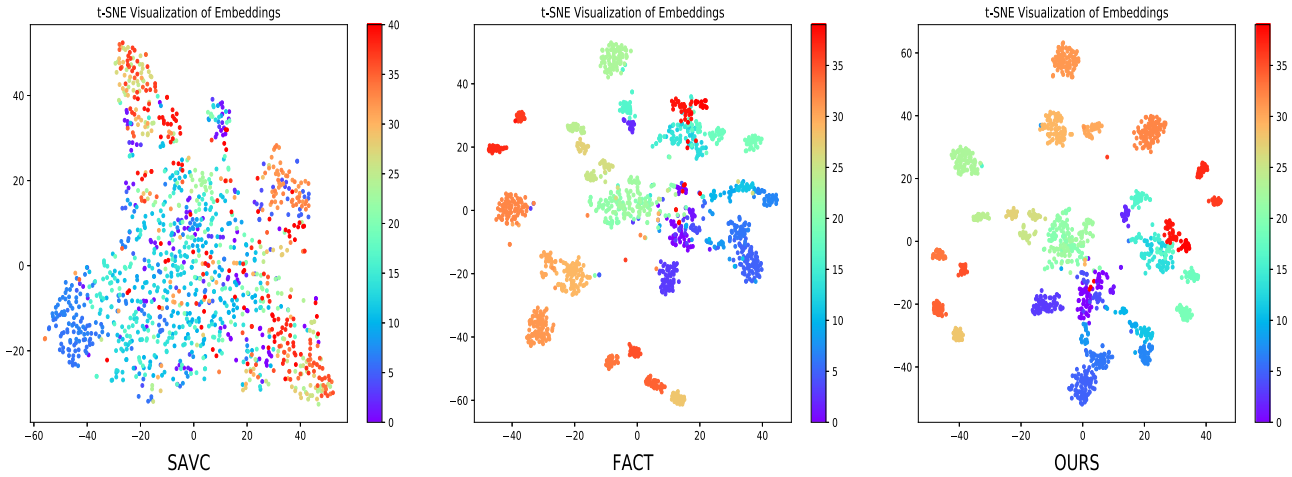

Fig. 7 | t-SNE embedding visualization results of similar datasets.

Table 5 | Comparison results of different fusion strategies on the Dongba script dataset

Method	Accuracy in each session↑									
	0	1	2	3	4	5	6	7	8	PD↓
Cutmix	96.251	96.553	96.556	95.421	95.325	94.214	93.109	91.84	90.976	5.275
Random Erasing	96.078	95.734	95.889	94.947	94.955	93.400	92.565	90.965	90.896	5.182
PatchMix	96.804	96.737	96.314	95.368	95.29	94.36	93.152	91.779	91.802	5.002
Augmix	96.11	96.043	95.839	95.211	95.155	93.782	92.913	91.215	90.284	5.826
Mixup	96.639	96.612	96.284	95.425	95.325	94.311	93.50	91.946	91.899	4.74

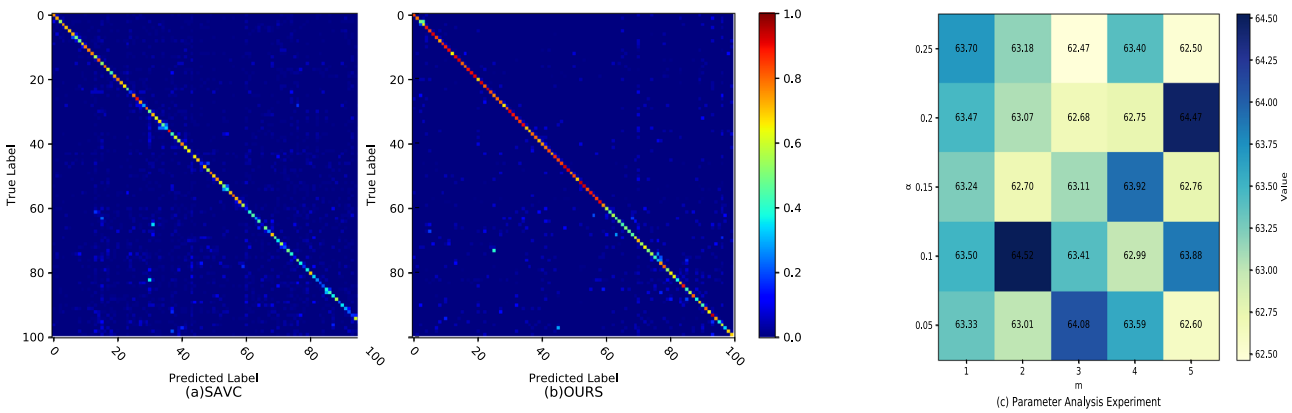

Fig. 8 | Confusion matrix visualization experiment and parameter analysis experiment. a, **b** Confusion matrix images of SAVC and DFS (ours) on the OBC100 dataset. **c** Experimental parameter analysis.

Comparison of results of different fusion strategies. To verify the fitting effect of Mixup method on unknown classes in the generation of virtual features of ancient characters. We compared a variety of classic image fusion methods on the Dongba script dataset, including CutMix[48], Random Erasing[49], PatchMix[50] and AugMix[51], and the experimental results are shown in Table 5. In the experiment, in order to maintain consistency, we only replaced the fusion method on the left and replaced the feature fusion strategy with the above image fusion method for comparison. The experimental results are shown in Table 5. Among all the fusion strategies, Mixup method achieved the best results. This shows that Mixup is more suitable for incremental learning tasks in the process of generating virtual features of ancient characters, and can provide more sufficient embedding space and higher fitting degree for new class features. Also, experimental results also show that DFS's dual-branch strategy and virtual prototype introduction mechanism can significantly improve model performance under different fusion strategies, further verifying the rationality and adaptability of our method.

Confusion matrix. Figure 8 in parts (a), and (b) illustrates the comparison of the confusion matrices between SAVC and DFS(Ours) on the OBC100 dataset. The confusion matrix is presented with a bright diagonal and a darker background, where the diagonal reflects a high level of classification accuracy. As mentioned in Section dataset split, the 60 classes were divided into known classes, and the remaining 40 classes were treated as new classes. Through observation, it is found that DFS(Ours) eliminates the misidentification of old classes to a certain extent compared with the SAVC method. And it performs well in predicting both known and unknown classes.

Parameter analysis. In prototype-based triplet loss, there are two key hyper-parameters: the coefficient α and the boundary control parameter m. These two hyper-parameters play crucial roles in the loss function, where α controls the relative weight between the class triplet loss and the maximum margin loss. And m controls the boundary distance value between similar characters. In the study, we report experimental results for multiple hyper-parameter settings on the OBC100 dataset, as shown in part c of Fig. 8. After analysis, the optimal performance of the model is achieved when $\alpha = 0.1$ and $m = 2$ This indicates that, under this specific combination of hyper-parameters, proximity-based triplet loss performs best on the OBC100 dataset.

Conclusions and future works

In this study, we applied the incremental frozen framework to the continuous learning of ancient Chinese characters for the first time to solve the problems of few samples and forgetting of old classes in the incremental recognition of ancient characters. To this end, a dual-branch forward simulation network (DFS) was proposed. Specifically, DFS constructs deep virtual features and virtual original data, respectively, through a dual-branch structure to train the feature extractor in the basic stage, and combines L-Selective Loss to improve the model's sensitivity to unknown features, thereby enhancing the feature expression of new class data in the incremental stage. Experimental results show that DFS exhibits excellent recognition performance on the ancient Chinese character dataset, especially in the incremental learning task of new classes, and can achieve the ability to continuously learn new classes without retraining the entire network. We hope that this study will provide new research perspectives and methods for the field of text recognition, especially the incremental learning of ancient characters, and promote the further development and application of related technologies.

We believe that future research will focus more on combining the detection of unknown classes with model extension. We look forward to future research exploring this issue more deeply and providing innovative methods and solutions for new class detection to advance the field further.

Data availability

The datasets used or analyzed during the current study are available from the corresponding author on reasonable request.

Received: 6 October 2024; Accepted: 14 January 2025;
Published online: 21 February 2025

References

1. Ma, B., Cong, Y. & Ren, Y. Iosl: Incremental open set learning. *IEEE Trans. Circuits Syst. Video Technol.* **34**, 2235–2248 (2024).
2. Wang, T. et al. Decoupled attention network for text recognition. In *Proc. AAAI Conference on Artificial Intelligence,* Vol. 34, 12216–12224 (ELSEVIER, 2020).
3. French, R. M. Catastrophic forgetting in connectionist networks. *Trends Cogn. Sci.* **3**, 128–135 (1999).
4. Zhu, F., Zhang, X.-Y. & Liu, C.-L. Calibration for non-exemplar based class-incremental learning. In *2021 IEEE International Conference on Multimedia and Expo (ICME)* 1–6 (IEEE, 2021).
5. Zhou, D.-W., Ye, H.-J. & Zhan, D.-C. Co-transport for class-incremental learning. In *Proc. 29th ACM International Conference on Multimedia, Association for Computing Machinery,* 1645–1654 (2021).
6. Zhou, D.-W. et al. Few-shot class-incremental learning by sampling multi-phase tasks. *IEEE Trans. Pattern Anal. Mach. Intell.* (2022).
7. Zhou, D.-W. et al. Forward compatible few-shot class-incremental learning. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE,* 9046–9056 (2022).
8. Zhang, J. et al. Class-incremental learning via deep model consolidation. In *Proc. IEEE/CVF Winter Conference on Applications of Computer Vision, IEEE,* 1131–1140 (2020).
9. Hou, S., Pan, X., Loy, C. C., Wang, Z. & Lin, D. Learning a unified classifier incrementally via rebalancing. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE,* 831–839 (2019).
10. Li, Y., Jia, Z., Zhang, J., Huang, K. & Tan, T. Deep semantic structural constraints for zero-shot learning. In *Proc. AAAI Conference on Artificial Intelligence, AAAI Technical Track: Vision,* Vol. 32 (2018).
11. Kirkpatrick, J. et al. Overcoming catastrophic forgetting in neural networks. *Proc. Natl Acad. Sci. USA* **114**, 3521–3526 (2017).
12. Zenke, F., Poole, B. & Ganguli, S. Continual learning through synaptic intelligence. In *International Conference on Machine Learning* 3987–3995 (PMLR, 2017).
13. Aljundi, R., Babiloni, F., Elhoseiny, M., Rohrbach, M. & Tuytelaars, T. Memory aware synapses: learning what (not) to forget. In *Proc. European Conference on Computer Vision (ECCV)*, 139–154 (2018).
14. Li, Z. & Hoiem, D. Learning without forgetting. *IEEE Trans. Pattern Anal. Mach. Intell.* **40**, 2935–2947 (2017).
15. Dhar, P., Singh, R. V., Peng, K.-C., Wu, Z. & Chellappa, R. Learning without memorizing. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE,* 5138–5146 (2019).
16. Tao, X., Chang, X., Hong, X., Wei, X. & Gong, Y. Topology-preserving class-incremental learning. In *Computer Vision—ECCV 2020: 16th European Conference,* Glasgow, UK, August 23–28, 2020*, Proc., Part XIX* 16, 254–270 (Springer, 2020).
17. Liu, Y. et al. Model behavior preserving for class-incremental learning. *IEEE Trans. Neural Netw. Learn. Syst.* **34**, 7529–7540 (2022).
18. Rebuffi, S.-A., Kolesnikov, A., Sperl, G. & Lampert, C. H. iCaRL: Incremental classifier and representation learning. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition, IEEE,* 2001–2010 (2017).
19. Wu, Y. et al. Large scale incremental learning. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE,* 374–382 (2019).
20. Zhao, B., Xiao, X., Gan, G., Zhang, B. & Xia, S.-T. Maintaining discrimination and fairness in class incremental learning. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE,* 13208–13217 (2020).
21. Wang, Y., Yao, Q., Kwok, J. T. & Ni, L. M. Generalizing from a few examples: a survey on few-shot learning. *ACM Comput. Surv.* **53**, 1–34 (2020).
22. Kong, C., Kim, J., Han, D. & Kwak, N. Few-shot image generation with mixup-based distance learning. In *European Conference on Computer Vision* 563–580 (Springer, 2022).
23. Zhao, H. et al. Mgsvf: Multi-grained slow vs. fast framework for few-shot class-incremental learning. *IEEE Trans. Pattern Anal. Mach. Intell.* **46**, 1576–1588 (2021).
24. Song, Z. et al. Learning with fantasy: semantic-aware virtual contrastive constraint for few-shot class-incremental learning. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE,* 24183–24192 (2023).
25. Zhang, C. et al. Few-shot incremental learning with continually evolved classifiers. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition* 12455–12464 (2021).
26. Kalla, J. & Biswas, S. S3c: Self-supervised stochastic classifiers for few-shot class-incremental learning. In *European Conference on Computer Vision* 432–448 (Springer, 2022).
27. Liu, C., Yang, C. & Yin, X.-C. Open-set text recognition via character-context decoupling. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE,* 4523–4532 (2022).
28. Verma, V. K. & Rai, P. A simple exponential family framework for zero-shot learning. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2017,* Skopje, Macedonia, September 18–22, 2017, *Proc., Part II 10*, 792–808 (Springer, 2017).

29. Li, Y., Wang, D., Hu, H., Lin, Y. & Zhuang, Y. Zero-shot recognition using dual visual-semantic mapping paths. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition, IEEE,* 3279–3287 (2017).
30. Wang, W., Miao, C. & Hao, S. Zero-shot human activity recognition via nonlinear compatibility based method. In *Proc. International Conference on Web Intelligence, ACM,* 322–330 (2017).
31. Yazdani, M. & Henderson, J. A model of zero-shot learning of spoken language understanding. In *Proc. 2015 Conference on Empirical Methods in Natural Language Processing* 244–249 (2015).
32. Palatucci, M., Pomerleau, D., Hinton, G. E. & Mitchell, T. M. Zero-shot learning with semantic output codes. *Adv. Neural Inf. Process. Syst.* **22**, 1410–1418 (2009).
33. Norouzi, M. et al. Zero-shot learning by convex combination of semantic embeddings. arXiv preprint arXiv:1312.5650 (2013).
34. Dinu, G., Lazaridou, A. & Baroni, M. Improving zero-shot learning by mitigating the hubness problem. arXiv preprint arXiv:1412.6568 (2014).
35. Kodirov, E., Xiang, T., Fu, Z. & Gong, S. Unsupervised domain adaptation for zero-shot learning. In *Proc. IEEE International Conference on Computer Vision, IEEE,* 2452–2460 (2015).
36. Liu, C. et al. Towards open-set text recognition via label-to-prototype learning. *Pattern Recognit.* **134**, 109109 (2023).
37. Snell, J., Swersky, K. & Zemel, R. Prototypical networks for few-shot learning. *Adv. Neural Inf. Process. Syst.* **30** (2017).
38. Hinton, G., Vinyals, O. & Dean, J. Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015).
39. Verma, V. et al. Manifold mixup: better representations by interpolating hidden states. In *International Conference on Machine Learning* 6438–6447 (PMLR, 2019).
40. Zhang, L., Deng, Z., Kawaguchi, K., Ghorbani, A. & Zou, J. How does mixup help with robustness and generalization? arXiv preprint arXiv:2010.04819 (2020).
41. Liu, W., Wen, Y., Yu, Z. & Yang, M. Large-margin softmax loss for convolutional neural networks. arXiv preprint arXiv:1612.02295 (2016).
42. Ge, W. Deep metric learning with hierarchical triplet loss. In *Proc. European Conference on Computer Vision (ECCV)*, 269–285 (2018).
43. Paszke, A. et al. Pytorch: an imperative style, high-performance deep learning library. *Adv. Neural Inf. Process. Syst.* **32**, (2019).
44. He, K., Zhang, X., Ren, S. & Sun, J. Deep residual learning for image recognition. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 770–778 (2016).
45. Tao, X. et al. Few-shot class-incremental learning. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition* 12183–12192 (2020).
46. Zou, Y., Zhang, S., Li, Y. & Li, R. Margin-based few-shot class-incremental learning with class-level overfitting mitigation. *Adv. Neural Inf. Process. Syst.* **35**, 27267–27279 (2022).
47. Van der Maaten, L. & Hinton, G. Visualizing data using t-SNE. *J. Mach. Learn. Res.* **9** (2008).
48. Yun, S. et al. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proc. IEEE/CVF International Conference on Computer Vision, IEEE,* 6023–6032 (2019).
49. Zhong, Z., Zheng, L., Kang, G., Li, S. & Yang, Y. Random erasing data augmentation. In *Proc. AAAI Conference on Artificial Intelligence* Vol. 34, 13001–13008 (2020).
50. Hong, Y. & Chen, Y. Patchmix: patch-level mixup for data augmentation in convolutional neural networks. *Knowl. Inf. Syst.* **66**, 1–27 (2024).
51. Hendrycks, D. et al. Augmix: a simple data processing method to improve robustness and uncertainty. arXiv preprint arXiv:1912.02781 (2019).

Acknowledgements

This paper is supported by the Ministry of Education Key Laboratory for Intelligent Analysis and Security Governance of Ethnic Languages and DENG FEI, Chinese department, College of Humanities, Hainan University. Key Project of National Social Science Fund 'Research on the Grammatical System Based on the Large-Scale Tagged Corpus of Chinese in the Shang Dynasty' (Project No. 24AYY005).

Author contributions

Q.R. participated in designing the study, developing the experimental plan, selecting the test model, and writing the main manuscript text. S.C. and H.X. provided overall guidance and supervision of the study and proposed an optimized protocol; Y.M. and M.Y. provided experimental-related datasets and assisted in article calibration; W.Z. and J.Y. assisted in the query and sorting of the literature; X.L. helped in the proofreading of the article; X.P and J.Q. assisted in the calibration of this article. All authors reviewed the manuscript.

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to Shanxiong Chen or Hailing Xiong.