BiSPL: Bidirectional Self-Paced Learning for Recognition from Web Data

Xiaoping Wu, Jianlong Chang, Yu-Kun Lai, Jufeng Yang, and Qi Tian, Fellow, IEEE

***Abstract*—Deep learning (DL) is inherently subject to the requirement of a large amount of well-labeled data, which is expensive and time-consuming to obtain manually. In order to broaden the reach of DL, leveraging free web data becomes an attractive strategy to alleviate the issue of data scarcity. However, directly utilizing collected web data to train a deep model is ineffective because of the mixed noisy data. To address such problems, we develop a novel bidirectional self-paced learning (BiSPL) framework which reduces the effect of noise by learning from web data in a meaningful order. Technically, the BiSPL framework consists of two essential steps. Relying on distances defined between web samples and labeled source samples, first, the web samples with short distances are sampled and combined to form a new training set. Second, based on the new training set, both easy and hard samples are initially employed to train deep models for higher stability, and hard samples are gradually dropped to reduce the noise as the training progresses. By iteratively alternating such steps, deep models converge to a better solution. We mainly focus on the fine-grained visual classification (FGVC) tasks because their corresponding datasets are generally small and therefore face a more significant data scarcity problem. Experiments conducted on six public FGVC tasks demonstrate that our proposed method outperforms the state-of-the-art approaches. Especially, BiSPL suffices to achieve the highest stable performance when the scale of the well-labeled training set decreases dramatically.**

***Index Terms*—Deep Learning, Image Recognition, Self-Paced Learning, Noisy Web Data.**

I. Introduction

DATA-driven deep convolutional neural networks (CNNs) are widely used in the vision community and achieve excellent performance on various tasks, *e.g.*, visual classification [1] and object detection [2]. Popular models such as ResNet-50 [1] have very deep network architectures and contain millions of parameters. In practice, they usually ensure sufficient model training by collecting a massive number of images with clean labels. It is expected that more well-labeled data results in higher performance and better robustness of the model. However, manual data annotation is expensive and time-consuming, especially for tasks that need expert knowledge.

To tackle this issue, a straightforward way is to pre-train CNNs on large-scale datasets (*e.g.*, ImageNet [3]) and then fine-tune them on new tasks such as visual classification of the dogs [4] and birds [5]. Later work by Cui *et al*. [6] proposes to transfer from a sub-set of ImageNet that is similar to the target domain-specific dataset. However, the learning of novel knowledge is still limited to the scale of well-labeled training data. More recently, several methods [7]–[9] have alternatively considered utilizing web data as auxiliary information to enhance the model performance on the source dataset. For example, Schroff *et al*. [10] utilize a multi-modal approach that combines the text, metadata, and visual features to obtain candidate images from web pages. Then, Krause *et al*. [11] additionally quantify the noise level of collected web images and apply active learning to filter the images with ambiguous category labels.

They propose that the web data can be freely collected from the Internet, yet it may contain a lot of noise due to the inherent characteristics of common search engines. Hence, the key to web data learning lies in the selection of confident data from noisy web data. In addition, some deep learning works [12]–[14] show that clean hard samples have a positive effect on model training. Their successful application proves that hard sample mining promotes better convergence and generalization of deep models. For example, focal loss [12] considers samples with large losses as hard samples and increases their contribution to the gradient by assigning larger weights. At the same time, some work such as GHM [14] also suggests that some very hard samples may be noisy and have a negative impact on model training. Therefore, in addition to selecting reliable web samples, how to distinguish informative hard samples from the data with noisy labels is also a challenge for the task of web data learning. As shown in Fig. 1, most algorithms [11], [15] usually rank the web data based on the classification scores and then choose relatively credible samples via a threshold. However, in this way, we may wrongly include noisy data (*e.g.*, the images surrounded by red boxes) if we want to add more hard but informative samples, since the noise may exist in both the procedures of web data sampling and model training.

Following the aforementioned observations, we propose a novel bidirectional self-paced learning (BiSPL) strategy for improving the effectiveness of learning using web data. First, an initial model is trained on the source dataset with clean labels. Second, in the procedure of web data sampling, inspired by the self-paced learning (SPL) paradigm which learns from the data with multiple paces, we infer the confidence of each web sample and rank them in the meaningful order, *i.e.*, from easy to hard. Contrary to the standard SPL, the cosine similarity is utilized to represent the relation between source data and web data in a robust feature space. The complexity

X. Wu and J. Yang are with the College of Computer Science, Nankai University, Tianjin 300350, China (e-mail: xpwu95@163.com, yangjufeng@nankai.edu.cn).

J. Chang and Q. Tian are with Huawei Cloud & AI, Shenzhen 518000, China (jianlong.chang@huawei.com and tian.qi1@huawei.com).

Y.-K. Lai is with the School of Computer Science and Informatics, Cardiff University, Wales, UK (e-mail: laiy4@cardiff.ac.uk).

Fig. 1. Examples from the Dog-120 dataset (the first column) and corresponding web dataset (the remaining columns). The images in each row belong to a category, where borders of different colors, *i.e.*, green, orange, and red, represent positive, ambiguous, and noisy data, respectively. The numbers in the boxes denote the cosine similarity between web samples and source samples.

of each web sample is measured by its cosine distance to each class center of the source set. The easier samples are preferentially selected to re-train the model and the harder ones will be learned at the next pace. Benefiting from it, web data is gradually and safely added to the training set in the order from easy to hard. And we can stop learning when we observe a significant drop in model performance to avoid introducing too much noise. Third, in the procedure of model training, opposite to the concept of SPL, we further propose to train the model from hard to easy to adaptively reduce the effect of possible noise mixed in the training set. Specifically, we learn all the data including hard samples in the early training stage, because the CNN model is robust to noisy data when the learning rate is large [16], [17]. As the model training iterates and learning rate decreases, we gradually drop the samples with large classification losses which are treated as noisy data. Finally, we repeat the processes of web data sampling and model training to mine useful knowledge from web data.

To sum up, the key contributions of this paper are:

1) Inspired by the human learning strategy that learns knowledge in a meaningful order, we develop a new bidirectional self-paced learning (BiSPL) framework to reduce the effect of noise by learning from web data in a meaningful order.
2) By sorting the learning complexity of web data with the cosine similarity, BiSPL suffices to endow deep models with the capabilities of steadily expanding training set capacity from easy to hard and learning from hard to easy as the learning progresses and the learning rate decreases.
3) Extensive experiments on six popular fine-grained visual classification (FGVC) datasets (*i.e.*, Indoor-67, Dog-120, Food-101, Food-101N, CUB-200, and Flower-102) demonstrate the superiority of BiSPL.

The remaining of the paper is organized as follows: In Section II, we briefly introduce the related research. Section III details our proposed BiSPL strategy. Then extensive experiments and analysis are conducted in Section IV and Section V provides the conclusion of this paper.

II. Related Work

In this section, we introduce the methods which are related to our paper, including image recognition from web data, relation learning, and self-paced learning.

A. Recognition from Web Data

Recently, deep learning has boosted the performance of many vision tasks, such as image classification [1] and object detection [2]. However, a large amount of well-labeled data is required to train deep CNNs with numerous parameters. To address this issue, recent research works have considered learning useful knowledge from free web data.

In the past few years, learning from web data has received widespread attention in the vision community [7], [15], [18]–[21] and led to great success in a variety of tasks including scene classification [19], clothing recognition [7], skin disease diagnosis [15], and action recognition [22]–[25], *etc*. Several works [10], [15], [26] gather web images from common web search engines for given query classes. The collection of web data minimizes human effort, but the resulting datasets contain noisy labels and cannot directly improve model performance [15], [18], [27]. Wong *et al*. [26] utilize rich semantic information such as image parametric dimensions and metadata to automatically annotate real-world web images. Cao *et al*. [28] also leverage a knowledge graph constructed from free DBpedia-Wikipedia and successfully employ the distillation technology. And Schroff *et al*. [10] alternatively harvest high-quality web data by combining the multi-modal features of text, metadata, and vision. In contrast, [29], [30] propose noise-robust methods to make the classifier robust to data with noisy labels. For example, Goldberger *et al*. [31] train a deep neural network with an additional noise adaptation layer to bridge the gap between correct labels and noisy ones. However, the noise-robust algorithm seems to be only suitable for the simple case of label noise [20]. Semi-supervised learning based methods instead propose to learn from web data along with an auxiliary dataset with clean labels. Recently, Yang *et al*. [15] initially train a CNN on a small source dataset and then use it to progressively filter web data with estimated pseudo-labels. Chen *et al*. [32] also utilize a semi-supervised learning method to jointly discover common-sense relationships and predict pseudo-labels at the instance level. They demonstrate the effectiveness of utilizing web data to boost image recognition performance on the source dataset. However, the classifier trained on the source dataset may easily be over-fitting and not sensitive to noisy data. Thus the predicted pseudo-labels based on it are unreliable and may lead to noise accumulation along with the iteration procedure.

In our work, the reliability of each web sample is measured based on its cosine distance from source data in the feature space. And the relation between web data and source data is also associated during the model training phase. Specifically, the web samples that have abnormal loss values to the average case will also be dropped.

It is worth noting that the goal of self-supervised learning [33]–[35] is also to learn from unlabeled or even noisy data. Specifically, self-supervised learning learns features in various pretext tasks by automatically generating pseudo-labels. For example, the well-known MoCo [33], SimCLR [34], and BYOL [35] perform basic data augmentation operations on unlabeled images to obtain anchors, positive examples and negative examples, and then minimize the

distances between positive pairs while maximizing those between negative pairs. In addition, some studies also design various pretext tasks such as image colorization [36], jigsaw puzzles [37], and image inpainting [38] for unlabeled image data and vehicle tracking [39], relative speed perception [40], background erasure [41], and sound generation [42] for unlabeled audio and video data. Based on models trained on pretext tasks, self-supervised learning can further transfer the learned features to downstream tasks. In this way, as a specific form of unsupervised learning, self-supervised learning can be applied to learn from unlabeled web data. However, different from self-supervised learning which assigns pseudo labels (*e.g*., rotation angles) corresponding to the pretext task to all data, the web data learning paradigm removes noisy data and alternatively chooses to directly assign each confident web data a label that belongs to the label system of the specific downstream task. This maximizes the utilization of web data. The indirect learning process of self-supervised learning can learn a certain degree of knowledge from web data, while it is hard to ensure sufficient learning of web data.

B. Relation Learning

Recently, many researchers [43]–[45] are interested in mining the latent relationships between the data. For example, Chang *et al*. [44] consider similar and dissimilar pairwise patterns to train a deep self-evolution clusterer. In contrast, graph-based methods [43], [46] tend to utilize a graph to reflect more complex structure. Zhan *et al*. [43] approximate the semantic relationships between a massive amount of unlabeled face data by constructing a bottom-up relational graph. More recently, graph convolutional networks (GCNs) [46], [47] show considerable performance improvement on graphical pattern modeling. For example, Kipf *et al*. [47] extend the convolutional neural networks to operate directly on graph-structured data and achieve impressive performance on the semi-supervised classification task. The relation learning strategy is also widely used in the field of one-/few-shot learning [48]–[50]. The idea of learning to compare, *i.e*., learning the distance metric between samples and inferring the few examples of new classes by comparing to the labeled query images, enables the image classification of new classes. For example, the architecture of Matching Networks [50] is like an end-to-end version of nearest neighbor classifier which encodes the features of each episode and then calculates the category with the highest similarity score to the test sample. Sung *et al*. [49] also propose a Relation Network architecture by automatically learning a deep distance metric between sample items and the query in each training episode. Inspired by the above-mentioned works, we propagate pseudo labels and measure the complexity of web data by considering its relation to source data.

C. Self-Paced Learning

Inspired by the learning pattern of humans, curriculum learning (CL) [7], [51], [52] gradually incorporates training samples into several ordered curriculums from easy to hard. The key of CL lies in the pre-definition of each curriculum and corresponding training orders. For example, Zhang *et al*. [52] start with a simple task that learns the strong idiosyncrasies (*e.g*., size and spatial relations) between urban scene images and then train a segmentation network and regularize predictions at the same time. However, it is time-consuming and difficult in the scenario of large-scale datasets. To address this critical issue, Kumar *et al*. [53] propose the self-paced learning (SPL) paradigm which assigns each sample with an importance parameter (*i.e*., loss value) to measure whether it is easy or hard and iteratively updates it. Subsequently, the SPL dynamically trains the model with the same goal as the CL rather than using heuristic knowledge. Meng *et al*. [54] further provide some theoretical analysis for SPL.

Recently, the SPL paradigm has been successfully applied in many tasks [55]–[57]. For example for multimedia search, Jiang *et al*. [55] propose a self-paced reranking approach for image and video search, which far exceeds the traditional reranking methods that primarily rely on heuristic weighting. To avoid poor local optimum and enhance the generalization of the model, ClusterGAN [58] trains the clusterer by gradually increasing the hardness of the included samples. Inspired by the SPL, we progressively select web data from easy to hard according to the similarity between the web and source data. During the training procedure, our method also detects outliers to efficiently avoid label noise from the selected web samples. And its implementation procedure is task-independent and can be easily generalized to varying scenarios.

III. Bidirectional Self-Paced Learning

In this section, we detail the BiSPL to mine useful knowledge from noisy web data for the fine-grained visual classification task. For this purpose, Section III-A models the pairwise relation learning to consider the relations in web data. By taking advantage of the learned relationships, the relation-based pseudo-labeling in Section III-B is employed to assign a pseudo-label with a confidence score to each web sample. The formulation of bidirectional self-paced learning during the procedures of data sampling and model training is presented in Section III-C according to the meaningful orders of web data. Finally, the model training process is described in Section III-D.

A. Learning Pairwise Relations

Given a standard source dataset $\mathcal{D}_s = \{(x_i^s, y_i^s)\}_{i=1}^{N^s}$ with N^s sample-label pairs (x_i^s, y_i^s), where $y_i^s \in \{1, 2, \cdots, C\}$ and C indicates the number of classes, our goal is to learn the relation between each image pair in the feature space. To achieve this goal, a common way is to train a CNN model (*e.g*., ResNet-50 [1]) on $\mathcal{D}_s$ via the *softmax* loss:

$$L_s = -\frac{1}{N^s}\sum_{i=1}^{N^s}\log\frac{e^{W_{y_i^s}^T x_i^s + b_{y_i^s}}}{\sum_{j=1}^{C} e^{W_j^T x_i^s + b_j}}, \tag{1}$$

where $W_j \in \mathbb{R}^d$ and $b_j \in \mathbb{R}$ denote the jth column of the model weight $W \in \mathbb{R}^{d\times C}$ and bias item, respectively. The feature dimension d is set according to the dimension of the last convolution block plus a global average pooling layer of the backbone network, such as $2,048$ for ResNet-50.

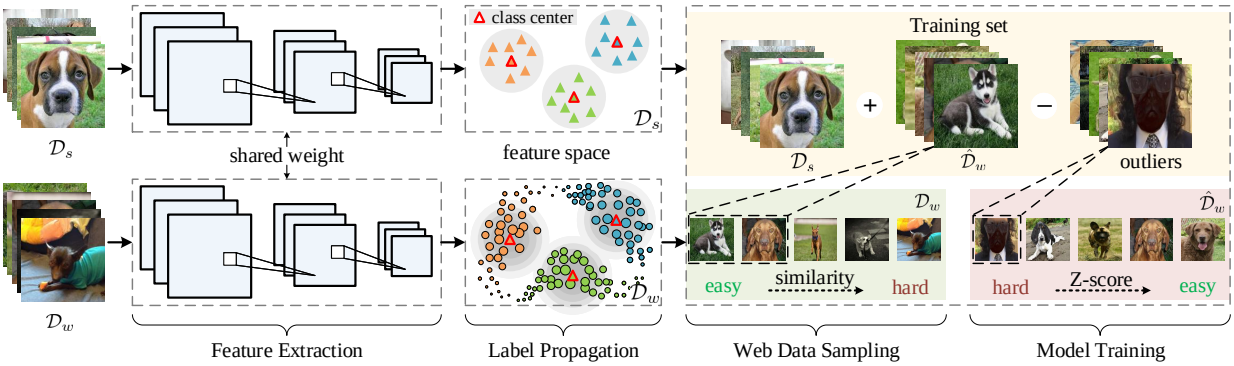

Fig. 2. Pipeline of our proposed framework. At each pace, 1) we train the ResNet-50 [1] as our backbone network on the source set and extract features of the source ($\mathcal{D}_s$, triangle) and web ($\mathcal{D}_w$, circular) sets, respectively. 2) In the pseudo-labeling step, we assign each web sample with a pseudo label and confidence score based on its similarity to the class center (red triangle). 3) Then we rank the web samples and add the easy ones $\hat{\mathcal{D}}_w$ into the training set. The harder samples will be added in the next pace. 4) During the model training phase, as the model approaches convergence, we drop more and more hard outliers with large Z-scores from the training set.

The *softmax* loss separates the features of different classes, although it is not sufficiently effective for minimizing the distances of the features belonging to the same class [59]. To learn a discriminative feature representation for a better measure of the cosine similarity between samples in subsequent steps of the proposed method, *i.e.*, enhancing the intra-class cosine similarity and weakening the inter-class one, we employ the commonly used *ArcFace* loss [59] which adds a geodesic distance margin penalty to the *softmax* loss. With the fixed bias $b_j = 0$, the logit can be transformed as $W_j^T x_i^s + b_j = \|W_j\| \|x_i^s\| \cos\theta_j$, where $\cos\theta_j$ is the cosine angle between the weight W_j and sample x_i^s. For simplicity, the l_2 normalization is used to fix $\|W_j\| = \|x_i^s\| = 1$, and then the item $W_j^T x_i^s + b_j = \cos\theta_j$. The *softmax* loss is simplified to:

$$L_s = -\frac{1}{N^s}\sum_{i=1}^{N^s}\log\frac{e^{\cos\theta_{y_i^s}}}{\sum_{j=1}^{C} e^{\cos\theta_j}}. \tag{2}$$

After that, the *ArcFace* loss can be formulated as:

$$L_a = -\frac{1}{N^s}\sum_{i=1}^{N^s}\log\frac{e^{\cos(\theta_{y_i^s}+m)}}{e^{\cos(\theta_{y_i^s}+m)} + \sum_{j=1,j\neq y_i^s}^{C} e^{\cos\theta_j}}, \tag{3}$$

where m denotes the geodesic distance margin penalty and is set to 0.5 following [59]. We can observe that the *ArcFace* loss is a modified version of *softmax* loss. To keep the classification ability of our model, we independently separate these two losses into different branches. And the feature dimension for the *ArcFace* loss is set to 512 to trade-off the model complexity and performance. Then the total loss for relation learning can be defined as:

$$L = (1-\lambda)L_s + \lambda L_a, \tag{4}$$

where λ is set to 0.1 to balance the contribution of *softmax* and *ArcFace* losses according to their loss values observed during the experiment.

B. Relation-Based Pseudo-Labeling

Given N^w unlabeled web samples $\mathcal{D}_w = \{x_i^w\}_{i=1}^{N^w}$, the goal is to assign each web sample with a confident class label and filter out noisy samples. As we know, during the collection procedure of web data, we utilize the name of each class as the keyword to query images from the Internet. Hence, we can simply treat them as the weak labels for retrieved web samples. Then we can obtain the labeled web dataset $\mathcal{D}_w = \{x_i^w, y_i^w\}_{i=1}^{N^w}$. However, in this way, $\mathcal{D}_w$ cannot be directly used to train the model because it is hard to discriminate between the clean samples and noisy data yet. Hence, we further assign each web sample with a confidence score via its relation (*i.e.*, cosine similarity) to the class center on the source dataset in the feature space:

$$R(x_i^w, c_{y_i^w}) = \frac{f(x_i^w)\cdot c_{y_i^w}}{\|f(x_i^w)\|\,\|c_{y_i^w}\|}, \tag{5}$$

where $f(x_i^w)$ and y_i^w denote the feature of web sample x_i^w and corresponding weak label. $c_{y_i^w}$ indicates the feature center (arithmetic mean) of the y_i^w-th class of the source dataset $\mathcal{D}_s$. Note that in each pace, only the data in the source dataset participates in the calculation of the class center, and the web data does not participate in either high-confidence or low-confidence. The range of R belongs to $[0, 1]$ and a larger value indicates more similarity. And the samples with the confidences less than a threshold can be regarded as outliers.

Note that the raw self-paced learning methods [54], [56], [57] use classification loss to determine whether a sample is easy or hard to learn, while we find that the cosine similarity in the feature space is more robust than the classification loss to measure the confidence of web data. And the experimental evaluation in Section IV-E also proves this. In addition, the core of the web data learning task lies in the detection of noise. Compared with the feature relationship, the optimization of the classifier may be more susceptible to noisy data. This will make it easier for the model to accumulate noise information.

C. Bidirectional Self-Paced Learning

In this section, we design the bidirectional self-paced learning, which contains two strategies of sampling web data from easy to hard and training model from hard to easy, to avoid noise information while learning more knowledge from the web data.

Sampling Web Data from Easy to Hard: To stably add reliable web data to the training set, we follow the SPL [51] paradigm and sample web data in several paces in the order from easy to hard. In this way, the model becomes more

Algorithm 1 Bidirectional Self-Paced Learning

Input: Source dataset $\mathcal{D}_s$ and web dataset $\mathcal{D}_w$
Output: Model parameters W

Train an initial model W on the source dataset $\mathcal{D}_s$;
Predetermine the fixed pace parameter γ and threshold σ;
repeat
  Extract the features of $\mathcal{D}_s$ and $\mathcal{D}_w$ using model W and calculate the feature center of each class on $\mathcal{D}_s$;
  Calculate the cosine distance of each web sample to its corresponding class center via Eq. 5;
  Select web samples from $\mathcal{D}_w$ to build $\hat{\mathcal{D}}_w$ via Eq. 7;
  for $N_e = \{1, \cdots, N_E\}$ **do**
    Compute Z-scores and detect outliers from $\hat{\mathcal{D}}_w$ via Eq. 9;
    Build outlier set $\hat{\mathcal{D}}_w^*$ by randomly selecting $\frac{N_e}{N_E}$ of outliers;
    Update model W on $\mathcal{D}_s \cup (\hat{\mathcal{D}}_w - \hat{\mathcal{D}}_w^*)$ via Eq. 8;
  end for
until Model gains no performance improvement
return W

and more robust and we can gradually add hard web samples with more confidence compared to sampling data at once. The training loss can be re-defined as:

$$L = \sum_{i=1}^{N^s} L(x_i^s, y_i^s) + \sum_{j=1}^{N^w} v_j L(x_j^w, y_j^w), \tag{6}$$

where $v_j \in \{0, 1\}$ controls the selection of web samples. Different from most SPL related works [54], [56], [57] which utilize the classification loss to measure the confidence of each sample and determine whether it is easy or hard, we consider the relationship (*i.e.*, cosine similarity) between web and source data in the feature space and the value of v_j is decided by:

$$v_j = \begin{cases} 1, \text{ if } R(x_j^w, c_{y_j^w}) \geqslant \gamma \\ 0, \text{ otherwise} \end{cases}. \tag{7}$$

The web sample x_i^w will be added into the training set if the cosine similarity of $R(x_j^w, c_{y_j^w})$ is greater than the fixed pace parameter of γ.

Note that the raw self-paced learning [54] gradually changes the pace parameter during the training process to select samples from easy to hard. However, following this setting, the model will have a great risk of introducing a lot of noise into the training set after experimental evaluation in Section IV-E. In our algorithm, we alternatively fix the pace parameter of γ. For each class in the feature space, before training, hard web samples (with confidence scores less than γ) are distributed around the class center of the source dataset. As the training converges on the source dataset and selected web samples, the *ArcFace* loss [59] maximizes the inter-class variance and minimizes the intra-class variance. As a result, the density of sample cluster of each class in the feature space will gradually increase and further narrow the distance between the hard samples and class center [59]. This is reflected in the experiment that with the γ parameter fixed, in each pace, there is still part of the hard samples (with confidence scores less than γ in the previous pace) whose similarity to the class center (*i.e.*, the confidence scores) in the current pace is greater than γ. In this way, we can safely and stably use web samples from easy to hard to expand the training set capacity.

Training Model from Hard to Easy: At each pace, the data we sample from the web set is still likely to contain noise. Inspired by the related works [16], [17] that deep networks are robust to noisy labels under a large learning rate, during the model training phase, we learn from the samples from hard to easy as the learning rate gradually declines, *i.e.*, at the beginning we learn from all the samples in the training set and then we gradually ignore the samples with large loss values. Then we re-formulate the training loss of the model as:

$$L = \sum_{i=1}^{N^s} L(x_i^s, y_i^s) + \sum_{j=1}^{N^w} u_j v_j L(x_j^w, y_j^w), \tag{8}$$

where the sample x_j is treated as noise if $u_j = 0$ or otherwise as clean data. And u_j is defined as:

$$u_j = \begin{cases} 1, \text{ if } \textit{Z-score}(j, \{L(x_i^w, y_i^w)\}_{i=1}^{N^b}) \leqslant \sigma \\ 0, \text{ otherwise} \end{cases}, \tag{9}$$

where N^b denotes the batch size and Z-score (also called standard score) [60] measures how many standard deviations an observed value is from the mean of a group of values. In this paper, it can be defined as a specific loss value $L(x_j^w, y_j^w)$ minus the average loss (denoted as $avg(\cdot)$) and then divided by the population standard deviation (denoted as $std(\cdot)$):

$$\textit{Z-score}(j, \{L(x_i^w, y_i^w)\}_{i=1}^{N^b}) = \frac{L(x_j^w, y_j^w) \text{ - } avg(\{L(x_i^w, y_i^w)\}_{i=1}^{N^b})}{std(\{L(x_i^w, y_i^w)\}_{i=1}^{N^b})}. \tag{10}$$

Here, σ varies from 1 to $\propto$ and the samples with their Z-scores larger than the threshold are treated as noisy data. Larger σ means the model contains more hard samples for training. In addition, to avoid losing a large amount of useful information at one time, we drop the noisy data with a probability of $\frac{N_e}{N_E}$ in each training batch, where the epoch number N_e gradually increases from 1 to the total epoch number N_E.

We utilize different metrics (*i.e.*, cosine similarity and Z-score) for determining whether the web data is easy or hard in the aforementioned two proposed strategies for two reasons. First, Z-score is not used in the web sample sampling stage because it measures how many standard deviations the observed sample point is from the mean of all samples. Unfortunately, the mean and standard deviation of unlabeled web data are unreliable. Second, if the metric of cosine similarity is used to drop web data, the similarity threshold will be difficult to determine. Due to the fact that the web data is selected based on its similarity with the source dataset (*i.e.*, greater than γ). If a threshold is set during model training to remove part of the training data that may be noise, the direct result is equivalent to increasing the γ parameter.

Note that the paradigm of online hard example mining [13] (OHEM) also sorts examples from hard to easy. The core of the OHEM algorithm is to select some hard examples with diversity and high loss as a training set to improve the parameter learning effect of the network. The purpose of

hard negative mining in [61] is also to make the model give more attention or learning weight to the hard examples. In contrast, the training strategy of BiSPL aims to avoid noisy web examples and progressively drop the hard examples at each training iteration. Specifically, during the model training procedure, we dynamically detect outlier (*i.e.*, noisy web data) in each batch via Z-score and keep all training examples in the early stages. As the model tends to be robust and convergent, we begin to consider outliers as noise. However, we may lose important information if we simply discard the top K hard examples like OHEM. And the ablation experiment in Section IV-E also demonstrates this.

Dynamic Threshold Hyper-Parameters: We can observe that the setting of the threshold hyper-parameters (*i.e.*, γ and σ) plays an important role in the proposed BiSPL method. The comprehensive experiment conducted in the experiment chapter also proves that we can adjust the values of these two thresholds to obtain state-of-the-art model performance. However, this manual process is time-consuming. So we further design dynamic versions for these two parameters as follows.

For the parameter γ, we independently set the safety threshold for each class by considering its positional relationship with the nearest neighbor class in the feature space:

$$\gamma^*_{y^w_j} = 1 - d_{inter}(y^w_j, \hat{y^w_j}) \frac{d_{intra}(y^w_j)}{d_{intra}(y^w_j) + d_{intra}(\hat{y^w_j})}, \quad (11)$$

where $y^w_j \in \{1, 2, \cdots, C\}$ denotes the weak label of web sample x^w_i. $\hat{y^w_j}$ is the nearest neighbor class of the class y^w_j and the inter-class distance $d_{inter}(\cdot, \cdot)$ is calculated by the cosine distance between the centers of the two classes on the source dataset. $d_{intra}(\cdot)$ represents the intra-class distance, defined as the maximum cosine distance between each sample and the center of the class it belongs to on the source dataset. In this way, we can safely select web samples for each class and avoid confusion with similar classes.

For the parameter σ, in each training batch, we observe that the Z-score values for most samples are around 0 (both positive or negative), and those samples are assumed to be clean samples. And the smaller the Z-score value of a sample is, the more likely it is a clean sample. Outliers tend to have larger Z-score values. Therefore, we simply use the absolute value of the Z-score of the cleanest sample (*i.e.*, with the smallest loss and Z-score value) as the threshold to dynamically distinguish the outliers:

$$\begin{aligned} \sigma^* &= |min(\{Z\text{-}score(j, \{L(x^w_i, y^w_i)\}_{i=1}^{N^b})\}_{j=1}^{N^b})| \\ &= \frac{avg(\{L(x^w_i, y^w_i)\}_{i=1}^{N^b}) \text{ - } min(\{L(x^w_i, y^w_i)\}_{i=1}^{N^b})}{std(\{L(x^w_i, y^w_i)\}_{i=1}^{N^b})}. \end{aligned} \quad (12)$$

Therefore, samples whose Z-score values are in the interval of $[-\sigma^*, \sigma^*]$ are regarded as clean samples, and a small number of remaining samples with large Z-score values are regarded as noise.

D. Model Iteration

Following the related SPL works [55]–[57], we train an initial model on the source dataset and split the training procedure into several paces. In each pace, as illustrated in Fig. 2 and Algorithm 1, we first extract features for both the web and source data. Second, we propagate the pseudo labels for web data and utilize their relation to source data in the feature space to assign each web sample with a confidence score. The relation is measured by the cosine similarity between the web sample and the feature center of the pseudo-label. Third, we sample the confident web data following the order from easy to hard. Then we re-train the model with a new training set that combines the source and labeled web data. During this training process, under the control of σ, hard samples with large losses are considered to be noisy and are gradually removed with the increase of the training epoch. Finally, we process the next pace and repeat the model iteration.

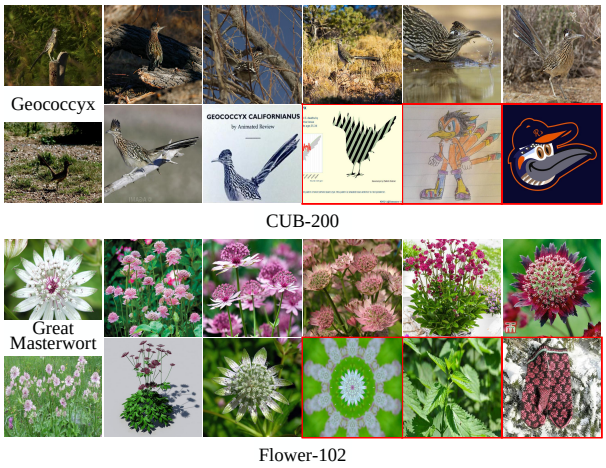

Fig. 3. Examples of collected web datasets for CUB-200 [5] and Flower-102 [65]. The images surrounded by red boxes indicate noisy data which does not help with the classification task.

IV. Experiments

In this section, we introduce the datasets and implementation details. Then we conduct comprehensive experiments for parameters, ablation studies, and performance evaluation of BiSPL. All the code and datasets will be released to the community.

A. Datasets

Following previous works on web data learning, we experiment on six fine-grained image recognition datasets with different characteristics, including Dog-120 [4], Indoor-67 [62], Food-101 [63], Food-101N [64], Flower-102 [65], and CUB-200 [5]. These datasets cover classification tasks of a diverse range of targets and are to some extent complementary. For example, the object classification tasks (*e.g.*, dog and bird) which recognize obvious foreground targets in the image are very different from the scene classification task. Besides, among these objects (*i.e.*, dog, food, bird, and flower), the recognition of food is different from others since food generally has no fixed shape, color, *etc*.

Existing Datasets. Table I shows the detailed statistics of the source and web datasets employed in this work, including image number, class number, and training/test split protocol. Specifically, we utilize each original benchmark dataset as the source set in our experiment and keep the same split setting of

TABLE I
STATISTICS OF SOURCE AND WEB DATASETS. 'CLASS' AND 'NUM.' DENOTE THE NUMBERS OF CLASSES AND IMAGES, RESPECTIVELY. '*' INDICATES THAT FOOD-101N EMPLOYS THE SAME TEST SET AS FOOD-101 [63].

Dataset	Source Set			Web Set
	Class	Num.	Training/Test	Num.
Dog-120	120	20,580	12,000/8,580	52,115
Indoor-67	67	15,620	14,280/1,340	76,907
Food-101	101	101,000	75,750/25,250	240,096
Food-101N	101	57,609	43,121/25,250*	252,400
Flower-102	102	8,189	2,040/6,149	213,404
CUB-200	200	11,788	5,994/5,794	307,348

training and test sets as previous works. For example, for the Dog-120 [4] dataset, $12,000$ samples are utilized to train the model and the rest are used as the test set. For the first four source datasets in Table I, we use their corresponding web sets collected by previous works [15], [64]. Note that the web sets are only used for model training, which is consistent with real-life applications. And each web sample has an unreliable pseudo label since the images obtained from Internet resources may be inconsistent with the objects we want.

Newly Collected Web Datasets. For the last two tasks in Table I, we further collect their web datasets following the procedure of previous works [15], [64]. For a given class name, we use it as the keyword to query from the image search engines (*e.g.*, Google, Bing). Only top-$3,000$ retrieved results are retained for each class. Then we simply filter the images which have the wrong format or are duplicated in the test set. Finally, as shown in Table I, we collect a total of $213,404$ and $307,348$ web images for Flower-102 [65] and CUB-200 [5] datasets, respectively. Fig. 3 illustrates some example web images. We can observe that the web data contains both informative images and noise.

Food-101N. Food-101N [64] is a web data learning dataset with 101 food categories such as Hamburger and Cheesecake and shares the same taxonomy as Food-101 [63] dataset. $310,009$ images are collected from Google, Bing, Yelp, and TripAdvisor, with foodspotting.com excluded to avoid duplication with the original Food-101 [63] dataset. The corresponding weak labels are annotated through query keywords and the related work [64] estimates that its noisy class label accuracy is 80%. The additional meta information includes $52,868$ verification labels for training and $4,741$ ones for validation which belong to $\{0, 1\}$ and denote whether each class label is correct. Among the verified training set, $43,121$ images with correct labels (*i.e.*, corresponding verification label equal to 1) are selected as source set in our BiSPL mechanism. Following previous works [64], [102], we utilize the test set of Food-101 [63] dataset to evaluate the proposed method.

B. Implementation Details

In our experiments, ResNet-50 [1] is employed as the backbone network and initialized with the parameters pre-trained on ImageNet [3]. The size of input images is set to 448×448 unless otherwise specified. In addition, common data augmentation strategies like random crop, dropout, and cutout [105] are utilized to avoid overfitting in our experiments. In total, we train the model for 50 epochs in each pace and optimize it using Stochastic Gradient Descent (SGD). The mini-batch, momentum, and weight decay are set to 40, 0.9, and $5e\text{-}4$, respectively. The learning rate of backbone and last fully connected layer are set to 0.001 and 0.01 respectively. Our method is implemented on the platform of PyTorch framework and an NVIDIA 1080Ti GPU with 11 GB on-board memory.

C. Comparison with the State-of-the-Arts

In this section, we evaluate the proposed BiSPL and compare with state-of-the-art methods on fine-grained classification tasks.

Comparison Protocols. For a fair comparison, we try our best to compare with related methods in the same experimental scenario. First, on the Dog-120 [4], Indoor-67 [62], and Food-101 [63] datasets, we mainly cite the results from recent works of ADNN [27] and PF [15]. Both methods exploit the same source and web sets in a similar mechanism to ours. On the Food-101N [64] dataset which originally comprises of labeled source set and noisy web set, we perform the experiment by directly following settings of previous methods. Second, on all the datasets especially the CUB-200 [5] and Flower-102 [65] datasets whose corresponding web data is first collected in this work, we find existing works based on these datasets from top journals and conferences and compare with the state-of-the-art methods. In Table II, we detail the experimental settings of backbone network and image input size of each method. Moreover, except on the Food-101N [64] dataset, specific statistics of the comparison methods which utilize additional data are also reported in Table III. For example, DPTL [6] utilizes a more powerful network architecture (*i.e.*, Inception-ResNet-v2 SE [66]), higher image resolution and transfer knowledge from the subset of large-scale well-labeled dataset (*i.e.*, iNaturalist [106]) that is similar to the target domain. In this case, it shows excellent classification results and even outperforms the web data learning methods (*e.g.*, PF [15] and ADNN [27]) on the Dog-120 [4] and Food-101 [63] datasets. Third, we also report our model results under different settings of input size (*e.g.*, 224×224 and 448×448) to compare methods in a controlled protocol.

Comparison on Existing Datasets. Specifically, on the Dog-120 [4] dataset, as shown in Table II, our BiSPL achieves the best classification performance of 88.66%. Compared with web data learning methods (*i.e.*, PF [15], ADNN [27], Goldfinch [11], and Hybrid [19]), our BiSPL outperforms them in accuracy by at least 1.30%. For the methods of PF [15] and ADNN [27] using the same web set as ours, we utilize the fair experimental settings (*i.e.*, backbone of ResNet-50 and input size of 224×224) and achieve the classification accuracy of 88.11%. PF [15] has a similar procedure that step-wisely learns from noisy web data and processes ambiguous samples which are mixed in the added web data. Different from it, we select confident samples from noisy web data based on the relation (*i.e.*, cosine similarity) on feature space which is more robust than classification scores. Besides, the process

TABLE II
COMPARISON WITH STATE-OF-THE-ART METHODS ON THE CLASSIFICATION TASKS OF DOG, INDOOR SCENE, FOOD, BIRD, AND FLOWER. WE COMPARE THE PROPOSED BISPL WITH SEVERAL METHODS INCLUDING WEB DATA LEARNING ALGORITHMS AND FINE-GRAINED VISUAL CLASSIFICATION FRAMEWORKS. '-' REPRESENTS THAT THE SPECIFIC DATA IS NOT REPORTED IN CORRESPONDING PAPER. '*' INDICATES THE USE OF ADDITIONAL DATA (*e.g.*, WEB IMAGES AND THE INATURALIST DATASET, SPECIFIC STATISTICS ARE IN TABLE III). 'INCRESV2SE' IS THE NETWORK OF INCEPTION-RESNET-V2 SE [66]. † AND ‡ DENOTE THE USE OF DYNAMIC THRESHOLD HYPER-PARAMETERS AND LARGER INPUT SIZE, RESPECTIVELY. BOLD VALUES DENOTE THE TOP 3 PERFORMANCE OF ACCURACY (ACC.).

Dog-120 [4]				Indoor-67 [62]				Food-101 [63]			
Method	Backbone	Input Size	Acc.	Method	Backbone	Input Size	Acc.	Method	Backbone	Input Size	Acc.
PBC [67]	GoogLeNet	224×224	78.30	HybirdCNN* [68]	Places-CNN	-	70.80	RF [63]	Random-Forest	-	50.76
SCDA [69]	VGG-16	224×224	78.86	SNN [70]	VGG-16	-	72.20	DCNN [63]	AlexNet	-	56.40
VSM [71]	VGG-16	-	79.50	SFV [72]	CaffeNet	-	72.86	Im2Cal [73]	GooLeNet	-	79.00
Weakly [74]	VGG-16	224×224	80.43	BCNNs [75]	VGG-16	multi-scale	79.00	KELM [76]	ResNet-50	300×300	82.60
PC [77]	DenseNet-161	-	83.75	Places* [78]	VGG-16	-	79.76	Grassmann [79]	VGG-16	224×224	85.70
Hybrid* [19]	VGG-16	-	85.16	MPP [80]	CaffeNet	multi-scale	80.78	CNet [7]	Inception-V2	-	87.30
Goldfinch* [11]	Inception-V3	-	85.90	DFHybrid* [81]	Places-CNN	multi-scale	80.97	Inception [82]	Inception-V3	299×299	88.28
ADNN* [27]	ResNet-50	224×224	87.07	LSDH [83]	VGG-11	-	83.75	ADNN* [27]	ResNet-50	224×224	89.35
PF* [15]	ResNet-50	224×224	87.36	ADNN* [27]	ResNet-50	224×224	84.59	PF* [15]	ResNet-50	224×224	89.77
DPTL* [6]	IncResV2SE	448×448	88.00	PF* [15]	ResNet-50	224×224	84.78	DPTL* [6]	IncResV2SE	448×448	**90.40**
BiSPL	ResNet-50	224×224	**88.11**	BiSPL	ResNet-50	224×224	**85.82**	BiSPL	ResNet-50	224×224	**90.39**
BiSPL†	ResNet-50	224×224	**88.52**	BiSPL†	ResNet-50	224×224	**85.60**	BiSPL†	ResNet-50	224×224	89.35
BiSPL‡	ResNet-50	448×448	**88.66**	BiSPL‡	ResNet-50	448×448	**87.01**	BiSPL‡	ResNet-50	448×448	**91.18**
CUB-200 [5]				Flower-102 [65]				Food-101N [64]			
Method	Backbone	Input Size	Acc.	Method	Backbone	Input Size	Acc.	Method	Backbone	Input Size	Acc.
Part-RCNN [84]	CaffeNet	227×227	76.40	TriCoS [85]	SVM	-	85.20	verified	ResNet-50	224×224	74.19
PSA-CNN [86]	VGG-19	-	82.80	LSVM [87]	SVM	-	87.14	clean	ResNet-50	224×224	78.57
MG-CNN [88]	VGG-19	-	83.00	N-Piars [89]	GoogLeNet	-	88.50	noisy [64]	ResNet-50	-	81.44
DLA [90]	DLA-102	448×448	85.10	RepMet [91]	Inception-V3	-	89.00	clean* [64]	ResNet-50	-	81.67
MA-CNN [92]	VGG-19	448×448	86.50	MsML [87]	-	224×224	89.45	Weakly [20]	VGG-16	-	83.43
NTS-Net [93]	ResNet-50	448×448	87.50	Magnet [94]	GoogLeNet	224×224	91.40	CleanNet(hard) [64]	ResNet-50	-	83.47
PA-CNN [95]	VGG-19	448×448	87.80	BOA* [96]	ResNet-152	-	92.50	CleanNet(soft) [64]	ResNet-50	-	83.95
DCL [97]	ResNet-50	448×448	87.80	VMF [98]	GoogLeNet	224×224	95.60	Guidance [99]	ResNet-50	224×224	84.20
iSQRT [100]	ResNet-101	448×448	88.70	DAT [101]	Inception-V3	-	97.70	MetaCleaner [102]	ResNet-50	-	85.05
DPTL* [6]	Inception-V3	448×448	89.60	EfficientNet [103]	EfficientNet-b7	600×600	98.80	DeepSelf [104]	ResNet-50	224×224	85.11
BiSPL	ResNet-50	224×224	**90.20**	BiSPL	ResNet-50	224×224	**99.09**	BiSPL	ResNet-50	224×224	**86.23**
BiSPL†	ResNet-50	224×224	**90.39**	BiSPL†	ResNet-50	224×224	**99.10**	BiSPL†	ResNet-50	224×224	**86.60**
BiSPL‡	ResNet-50	448×448	**91.11**	BiSPL‡	ResNet-50	448×448	**99.33**	BiSPL‡	ResNet-50	448×448	**87.22**

TABLE III
STATISTICS ON THE USE OF ADDITIONAL DATA FOR COMPARISON METHODS.

Dataset	Method	Use of Additional Data
Dog-120	Hybrid [19]	100 noisy web images per category
	Goldfinch [11]	342,632 noisy web images
	ADNN [27]	52,115 noisy web images
	PF [15]	52,115 noisy web images
	DPTL [6]	subset of iNaturalist [106]
Indoor-67	HybirdCNN [68]	Places 205 (2,448,873 images)
	Places [78]	Places 205 (2,448,873 images)
	DFHybrid [81]	Places 205 (2,448,873 images)
	ADNN [27]	76,907 noisy web images
	PF [15]	76,907 noisy web images
Food-101	ADNN [27]	240,096 noisy web images
	PF [15]	240,096 noisy web images
	DPTL [6]	subset of iNaturalist [106]
CUB-200	DPTL [6]	subset of iNaturalist [106]
Flower-102	BOA [96]	unknown amount of web data

of assigning each sample with multiple labels performs well when one image contains objects from multiple categories, but ambiguous web samples may be noisy and do not belong to any category. We choose to directly drop them instead of introducing extra label information for every training sample. Even compared to the method of Goldfinch [11] which utilizes a much larger scale of web dataset both in the levels of category (*i.e.*, 515 vs. 120) and image number (*i.e.*, $342,632$ vs. $52,115$), our method still performs better by at least 2%. Compared with the state-of-the-art methods (*e.g.*, PC [77] and Weakly [74]) in the fine-grained visual classification field, our method shows great advantages in classification results. This indicates that web data contains rich information that can effectively improve the model performance. For these methods (*i.e.*, DPTL [6] and PC [77]) that usually utilize deeper CNNs (*e.g.*, DenseNet-161), our method still outperforms them to a certain extent. For example, DPTL [6] which is based on the network of Inception-ResNet-v2 SE [66], input size of 448×448 and additional images from the subset of iNaturalist [106] achieves excellent accuracy performance of 88% and even outperforms the state-of-the-art web data based methods (*e.g.*, PF [15]).

Our BiSPL performs similarly on the datasets of Indoor-

67 [62] and Food-101 [63]. As shown in Table II and Table III, under the different experimental settings, our method outperforms both the web data learning methods and state-of-the-art fine-grained visual classification methods. Moreover, on the Indoor-67, the methods of HybirdCNN [68], Places [78], and DFHybrid [81] benefit from the initial model which is pre-trained on the Places 205 dataset [78]. This large-scale scene recognition dataset (*i.e.*, Places 205 [78]) comprises of 205 scene categories and about 2.5 million well labeled images. Having at least 5k images of each category guarantees that rich information can be transferred from the Places 205 [78] dataset, while the better performance of our method demonstrates that unlabeled noisy web data also contains a lot of valuable knowledge and our method can effectively learn from it.

Comparison on Newly Collected Web Datasets. On the CUB-200 [5] dataset, our method outperforms the related works by at least 0.60% in accuracy. The part-based methods such as PA-CNN [95], DCL [97], and MA-CNN [92] attend to generate attention maps or utilize local information, *e.g.*, bounding boxes and attributes, to assist the feature representation. They achieve good performance since part regions can guide the model to focus on the foreground area and ignore the noisy background information. However, the data diversity is still limited to the source data scale and the annotation of fine-grained labels is time-consuming. Benefiting from web data, as shown in Fig. 3, we can freely obtain extensive bird images that differ in scale, pose, and environment. This can help to decrease both the intra- and inter-class variation in the fine-grained visual classification task. In our BiSPL strategy, we use these data samples that contain rich information and effectively drop noisy data. Therefore, the model gains the improvement of robustness and generalization and achieves state-of-the-art results. On the Flower-102 [65] dataset, BOA [96] also augments training set from web data according to the similarity between the web sample and labeled source sample. However, it only samples the web data once and may introduce noisy information easily. In contrast, our method gradually adds reliable web data to the training set in the order from easy to hard. In addition, we also detect and remove potentially noisy samples during model training. Then, as shown in Table II, we observe that the pure deep network (*i.e.*, EfficientNet-b7 [103] with the input size of 600×600) can also achieve excellent performance (*i.e.*, accuracy of 98.8%) with an average of only 20 training samples per class. This may be because the task of flower classification is relatively simple and the diversity of flowers (*e.g.*, colors, shapes, *etc.*) in the corresponding test set may be relatively limited.

Comparison on Food-101N. The basic ResNet-50 achieves the accuracy of 74.19% on the 53k verified subset of Food-101N [64]. The verification labels denote whether the label of each image is correct. Hence we remove the noisy training images and gain performance improvement on the 43k clean set (utilized as source set in our method). As reported in [64], the model trained on a total of 310k noisy training images easily achieves the classification performance of 81.44%. This is very close to the performance of 81.67% on the clean Food-101 [63] dataset with 76k well-labeled training images. This indicates that the images collected from the Internet are relatively reliable for the task of food classification and we can simply obtain extensive knowledge from the web data without any human effort for annotation. For the other compared methods on the Food-101N [64] dataset, they can be grouped into two categories: unsupervised and semi-supervised methods. The unsupervised web data learning methods (*i.e.*, DeepSelf [104] and Weakly [20]) usually utilize weak labels (*e.g.*, the query keywords when collecting data from Internet) and correct them with the information such as losses and gradients during model training. For example, DeepSelf [104] generates multi-prototypes for each class by feature clustering and computes the similarity score to determine whether each sample label is correct. However, this method may perform poorly in some categories where the number of samples is very small or the corresponding web data contains many error samples. In contrast, the semi-supervised mechanisms (*i.e.*, MetaCleaner [102], Guidance [99], and CleanNet [64]) require a small well-labeled source dataset as auxiliary or prior information and generally achieve higher classification performance. For example, MetaCleaner [102] assigns each noisy web sample with a confidence score by calculating its relation to clean samples and gains the accuracy of 85.05% on the Food-101N [64] dataset. Compared with it, our method adds the noisy web data into training set through several steps instead of one time. This process reduces the risk of biasing to poor optimization to some extent and makes the model more robust. As shown in Table II, our proposed BiSPL outperforms the state-of-the-art methods under the fair experimental settings (*i.e.*, backbone of ResNet-50 and input size of 224×224).

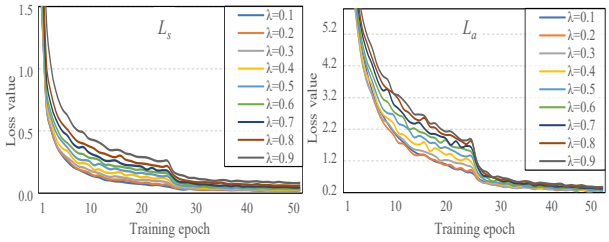

Fig. 4. Sensitivity of λ (denoted as different colors) on the Indoor-67 dataset. The losses of L_s and L_a vary from the training epochs under the different settings of λ.

D. Parameters

In this section, we evaluate the model performance with varying parameter settings of λ, γ, and σ.

Impact of the Weight between Losses. The weight parameter λ in Eq. 4 controls the loss contributions of L_s and L_a. On the Indoor-67 dataest, we train a ResNet-50 with the input size of 224×224 and 50 epochs and report the loss curves of L_s and L_a in Fig. 4. We can observe that both losses converge relatively faster as the λ parameter decreases. In addition, during the experiment, we find that the value of L_s is usually close to 10 times that of L_a. Therefore, we finally set the λ parameter to 0.1 in this paper.

Impact of the Pace Parameter. The pace parameter γ determines whether one sample is confident to be added to the training set in the next pace. Smaller γ may lead to the

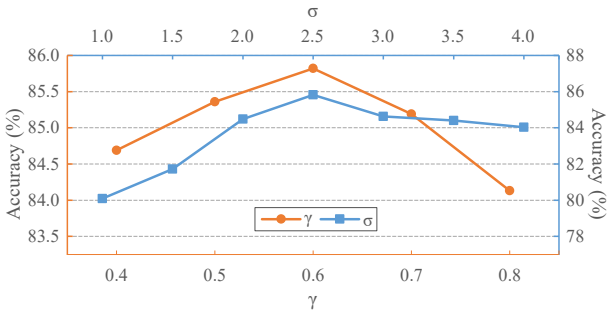

Fig. 5. Sensitivity of γ and σ (denoted as orange and blue, respectively).

TABLE IV
PARAMETER SETTINGS OF γ AND σ ON DIFFERENT DATASETS.

#	Dog-120	Indoor-67	Food-101	CUB-200	Flower-102	Food-101N
γ	0.6	0.6	0.7	0.7	0.7	0.7
σ	2.0	2.5	2.5	2.5	2.5	2.5

model ignoring those hard but informative web samples. In contrast, larger γ may easily introduce more noise and make the model accumulate more error. As illustrated in Fig. 5, with the fixed setting of $\sigma = 2.5$, the model performs increasingly better when γ varies from 0.4 to 0.6 on the Indoor-67 [62] dataset. Yet the accuracy performance drops dramatically in the interval of $[0.6, 0.8]$. As a result, we fix the γ to 0.6 for the experiments on the Indoor-67 dataset in this paper.

Impact of the Threshold of Z-Score. σ is the Z-score threshold that controls the number of outliers in a batch in the model training procedure. Larger σ means fewer outliers will be dropped and we set $\sigma = 2.5$, based on the observation from Fig. 5. In summary, we can find that the model performance changes regularly as the hyper-parameters vary on the Indoor-67 [62] dataset. And we can easily set the parameters of γ and σ to 0.6 and 2.5, respectively.

Sensitivity of Hyper-Parameters. Hyper-parameters have a certain influence on model performance. Because the noise in the web data has a direct and significant impact on the performance of the model. Too large σ and γ will increase the risk of introducing noise data (around 1.5% accuracy), and too small σ and γ will make hard samples with rich information that are beneficial to the model generalization to be filtered out too early. While for a σ that is extremely small, a large number of training samples will be discarded, resulting in a more dramatic drop in performance relative to the parameter γ. In addition, benefiting from the regular influence of parameters on model performance, in Table IV, We can easily find the hyper-parameters by simply adjusting the hyper-parameters near the ones of the Indoor-67 dataset. The specific hyper-parameter settings of the other evaluation datasets (*i.e.*, Dog-120, Food-101, CUB-200, Flower-102, and Food-101N) are shown in Table IV.

E. Ablation Studies

We conduct extensive ablation experiments on the Indoor-67 dataset to demonstrate the effectiveness of BiSPL.

Effect of Web Data. Specifically, as shown in Table V, we first evaluate the basic fine-tuning technology in the deep learning field which transfers knowledge through pre-training

TABLE V
ABLATION EXPERIMENTS ON THE INDOOR-67 DATASET. '224' AND '448' INDICATE THE DIFFERENT SIZES OF INPUT IMAGE. 'SRC' INDICATES SOURCE DATA IS USED FOR TRAINING, WHILE 'WEB' DENOTES THE USE OF ADDITIONAL WEB DATA. 'SPL^1' AND 'SPL^2' REPRESENT THE STRATEGIES OF SAMPLING DATA FROM EASY TO HARD AND TRAINING MODEL FROM HARD TO EASY, RESPECTIVELY. 'RL' IS PROPOSED RELATION LEARNING, 'OHEM' MEANS FIXEDLY REMOVING TOP K HARD SAMPLES IN EACH TRAINING BATCH. '†' AND '‡' DENOTE THE USE OF EUCLIDEAN DISTANCE. '‡' INDICATES THE *Center* LOSS IS FURHTER EMPLOYED.

	Method	Input Size	Indoor
Baseline	SRC	224 × 224	79.63
	SRC	448 × 448	83.13
	Web	224 × 224	67.54
	SRC+Web	224 × 224	78.10
	Web → SRC	224 × 224	80.37
	SRC+Web → SRC	224 × 224	81.04
	SRC+Web+Filter	224 × 224	81.34
	SRC+Web+RL	224 × 224	83.66
	SRC+Web+RL†	224 × 224	82.12
	SRC+Web+RL‡	224 × 224	83.05
Ablation	SRC+Web+RL+SPL^1	224 × 224	84.03
	SRC+Web+RL+SPL^2	224 × 224	84.25
	SRC+Web+RL+OHEM	224 × 224	83.51
	SRC+Web+RL+SPL^1+OHEM	224 × 224	84.26
Ours	SRC+Web+RL+BiSPL	224 × 224	**85.82**
	SRC+Web+RL+BiSPL	448 × 448	**87.01**

the model on a large dataset (*e.g.*, ImageNet) and then fine-tuning it on the source dataset (*e.g.*, Indoor-67). We train a basic CNN model (*i.e.*, ResNet-50) on the source dataset and achieve the accuracy of 79.63%. To evaluate the effectiveness of web data, in the 3rd row, we directly train on the noisy web data and the model gets worse on the Indoor-67 dataset. Then the source and noisy web datasets are combined in the 4th row and the model improves the classification result on the Indoor-67 dataset (*i.e.*, the accuracy of 78.10%) while it is still lower than the basic model trained on the source dataset. This case indicates that the categories of indoor scenes in the Indoor-67 dataset are abstract and this makes the collected web data difficult to use without filtering. To alleviate the influence of noisy data in a simple way, in the 5th-6th rows, we further fine-tune the model in the 3rd-4th rows on the source dataset. Compared with the classification result on the source dataset, this simple strategy easily obtains performance improvement and demonstrates the efficiency and importance of research to mine knowledge from free web data. In our method, we do not choose to fine-tune the final model on the source dataset, because the deep networks are learned with memorization [16], [17] and the fine-tuned model may forget valid information learned from the web set. Moreover, the better performance of the fine-tuned model may be due to the small domain shift between the clean source set and the test set, which may lead to lack of better generalization.

Effect of Noise Filtering. Afterward, we filter and add web data into the training set via the classification results from the basic CNN model and set different thresholds following [15]. And the model trained by the combination of source data

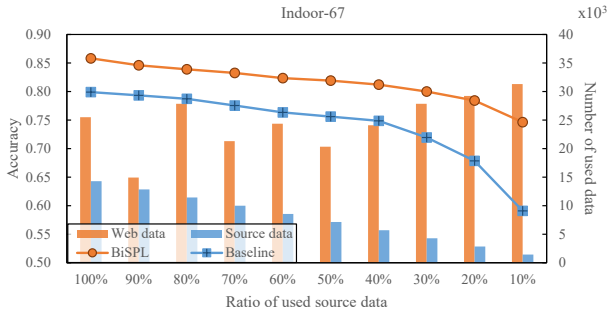

Fig. 6. Performance on the limited-data setting on the Indoor-67 dataset. The x-axis denotes the ratio of used source data for initial model training. The orange and blue histograms denote the amount of used web data and source data, respectively. The curves indicate the classification performance.

and filtered web data outperforms the basic model, achieving accuracy boost of 1.71%. Note that we set the input size of CNN to 224×224 for accelerating the experimental procedure and all the models are pre-trained on the ImageNet dataset.

Effect of Relation Learning. Different from most of the web data learning methods that measure the confidence of web data via a classifier trained on the source set (*i.e.*, the 7th row), in the 8th row of Table V, we process it in a more robust way and introduce the relation learning (RL) into the proposed framework, *i.e.*, utilizing the cosine similarity in the feature space to represent the relationship between source and web data. We can observe that the relation-based data sampling strategy outperforms the conventional classifier-based one. Furthermore, we also verify the impact of different similarity measures (*e.g.*, Euclidean distance) on our proposed BiSPL. Specifically, in the 9th row, we train a classification network on the source dataset as in row 7 and further use Euclidean distance to measure the similarity between the source data and the web data. And the result indicates that the similarity measure in the feature space is more robust than using classification loss directly. This may be because web data learning is an open-set problem. For noisy web samples not included in the label set of the source dataset, in the feature space, they will always be classified into a hyperplane of a certain category by the classifier. And the classification results normalized by *softmax* will always assign predicted probabilities to each category belonging to the source dataset and the sum is 1. This may make it more difficult to distinguish between hard samples and noisy samples. In the 10th row, we further employ the metric learning loss (*i.e.*, the commonly used *Center* loss) to embedding features and achieve the performance of 83.05%. Compared with *ArcFace* loss, the *Center* loss relies on the classifier to ensure that the features are separable while minimizing the distance within the class. As a result, its feature discriminability is related to classifier performance.

Effect of BiSPL. Then we introduce the proposed bidirectional self-paced learning pattern in the 11th-12th rows. First, inspired by the learning pattern of humans, the order *from easy to hard* confirms the web data is sampled through a meaningful fashion. Compared with the simple one-stage sampling in the 8th row, the self-paced learning strategy in the 11th row samples data step by step and boosts the model performance by 0.37%. It is worth noting that we fix the pace parameter γ as described in Eq. 7. When we gradually reduce this parameter following the pattern of the original self-paced learning [54], the model will only achieve the highest performance in the first pace which equals to the performance in the 8th row. In the later paces, the performance degradation may be caused by the introduction of too much noise. Then, to avoid the effects of noise contained in the data sampled from the web set, we insightfully propose to train the model *from hard to easy* in the 12th row. Along with the training process of the model, this mechanism dynamically drops more and more outliers with large Z-scores which are calculated by loss values. As shown in Table V, the model trained by reversed self-paced learning during model training is more robust and obtains performance improvement. In contrast, in the 13th-14th rows, the OHEM strategy which fixedly removes the top K hard examples in each training batch can easily lose critical information and performs worse than our training strategy. In this paper, we set K to $\{1, 5, 10, 20\}$ respectively on each dataset and report the best experimental performance. Finally, we combine both above-mentioned strategies into a unified framework, called bidirectional self-paced learning (BiSPL), in the 15th row. We can find that the BiSPL can perform stably both on the procedures of web data sampling and model training and achieves the best accuracy performance. In addition, in the last row of Table V, we resize the input image to higher resolution 448×448 and the model gains further performance improvement. This indicates that the larger size of the input image can help the model capture richer information.

F. Evaluation on the Limited-Data Setting

In practice, as the difficulty of labeling different types of data is different (*e.g.*, the classification of medical images versus general targets), it is hard to control the scale of well-labeled data. In this section, we evaluate the efficiency of our proposed method under the limited-data setting.

Specifically, as illustrated in Fig. 6, we gradually decrease the scale of source training set from 100% to 10% by random sampling. The sample size ratio of all categories remains unchanged relative to the original training set. Experiments are conducted with the backbone of ResNet-50 and the input size of 224×224. We first report the classification performance of ResNet-50 (denoted as 'Baseline') which is trained on the limited source set. Then the proposed method is evaluated on the combination of partial training set and web set. Note that the rest of training set is not used under each setting of training scale. This is to simulate real-world application scenarios that we can only obtain web data of unknown quality through the Internet and the degree of overlap between the web set and training set is difficult to control. As observed from Fig. 6, on the Indoor-67 [62] dataset, the performance of the Baseline method decreases dramatically as the amount of supervised data decreases. Due to the fact that the massive parameters of deep learning technology require a large amount of data. As the amount of data gradually decreases, the performance of the deep model will definitely decrease. Especially when the amount of data in the training set of the source dataset is too sparse (*i.e.*, less than 40% on the Indoor-67 dataset) and is not enough to fit the test set, the performance of the 'Baseline'

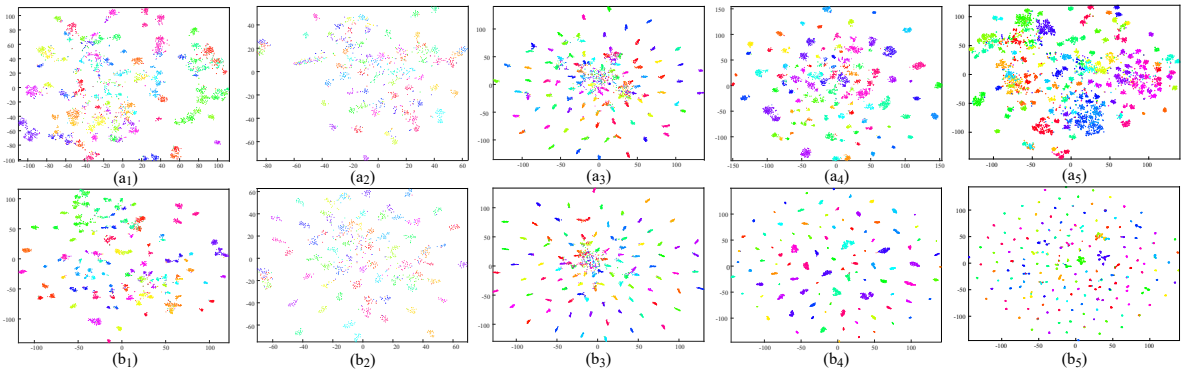

Fig. 7. 2D t-SNE [107] visualizations of the feature embeddings on different datasets, *i.e.*, (1) Dog-120, (2) Indoor-67, (3) Food-101, (4) Flower-102, and (5) CUB-200. Two rows indicate the features extracted from (a) ResNet-50 [1] combined with *ArcFace* loss and (b) our BiSPL model, respectively.

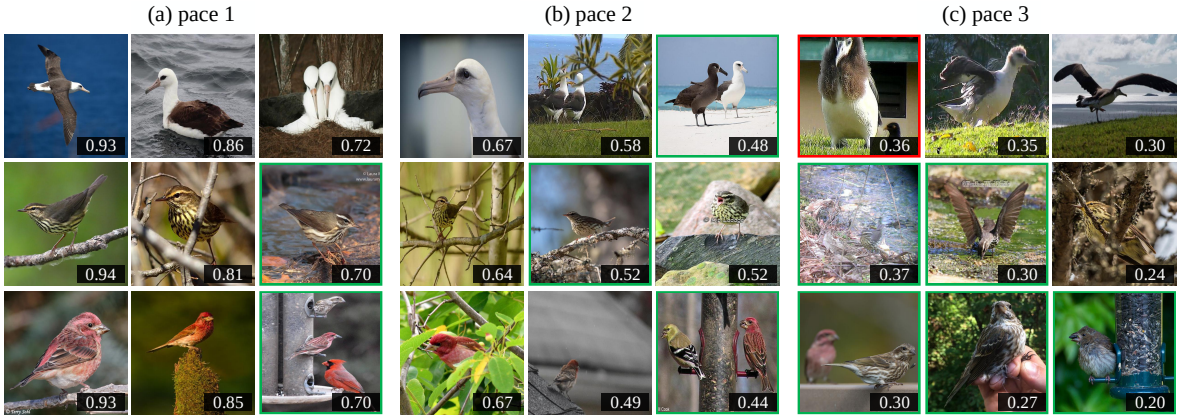

Fig. 8. Web images selected from easy (a) to hard (c) by the proposed BiSPL algorithm in different learning paces. Different rows indicate different categories (*i.e.*, Laysan Albatross, Northern Waterthrush, and Purple Finch). The numbers under the images represent corresponding cosine distances to the feature centers of the pseudo-labels. The images which are surrounded by green boxes denote the correctly identified noisy data during the model training phase, while the red boxes indicate the missed outliers.

method drops sharply. In contrast, it can be found that as long as the amount of data can ensure that the model can learn the basic general pattern of each category (*i.e.*, more than 20% on the Indoor-67 dataset), our work can safely learn knowledge conducive to rich feature diversity from web data pace by pace, thus contributing to the generalization performance of the model and achieves relatively stable performance. As a result, even if the reduction in the training set scale results in a bias from the test set, our model can still have a significant performance improvement on the test set relative to the 'Baseline' method. This demonstrates that the task of web data learning can indeed effectively alleviate the data-hungry problem. However, when well-labeled data is less than 20%, the performance of our method also begins to decrease significantly. This may be due to the fact that too little data is not enough to train a deep network such as ResNet-50 and obtain a good initial feature representation. In addition, our method tends to utilize more web data when the ratio of used source data is less than 40%, and at the same time the performance of the 'Baseline' method is greatly reduced. In this case, the sampled web data contains relatively more noise, and the BiSPL can still achieve a more stable classification performance than the 'Baseline' method, which demonstrates the effectiveness of our method.

G. Visualization Results

Fig. 7 shows the 2D t-SNE [107] feature embeddings. Compared with the baseline method (*i.e.*, ResNet-50), our BiSPL learns discriminative feature representations on all the datasets. Further, as illustrated in Fig. 8, our method samples the web data via several paces and follows the meaningful order from easy to hard. As the learning pace progresses, more and more outlier hard samples are successfully detected (surrounded by green boxes) and treated as noise by our BiSPL. At the same time, we can observe that there are still some informative hard samples that are retained, and they can often enrich the diversity of corresponding categories and enhance its generalization performance. For example, in row 2, column 3 of Fig. 8, the outlier which belongs to Louisiana Waterthrush is similar to Northern Waterthrush, yet it has bright white at the rear of eyebrow. In the third row, several web samples that contain multiple categories of birds are also dropped. However, the noise sample surrounded by an orange box in the first row fails to be detected as it has a similar representation as Laysan Albatross (*i.e.*, white underpart and dark gray-brown upper wings) and its true class is not contained in the CUB-200 [5] dataset. More specifically, Fig. 9 presents more examples including training images, web images selected by our algorithm, and classification results. The web data contains diverse information compared with the training set and helps improve the model generation. However, several noisy samples have extremely similar characteristics to the training set and do not belong to any category of the dataset, so they are easy to be wrongly selected, *e.g.*, the single computer and monitor (surrounded by red boxes) in the fourth

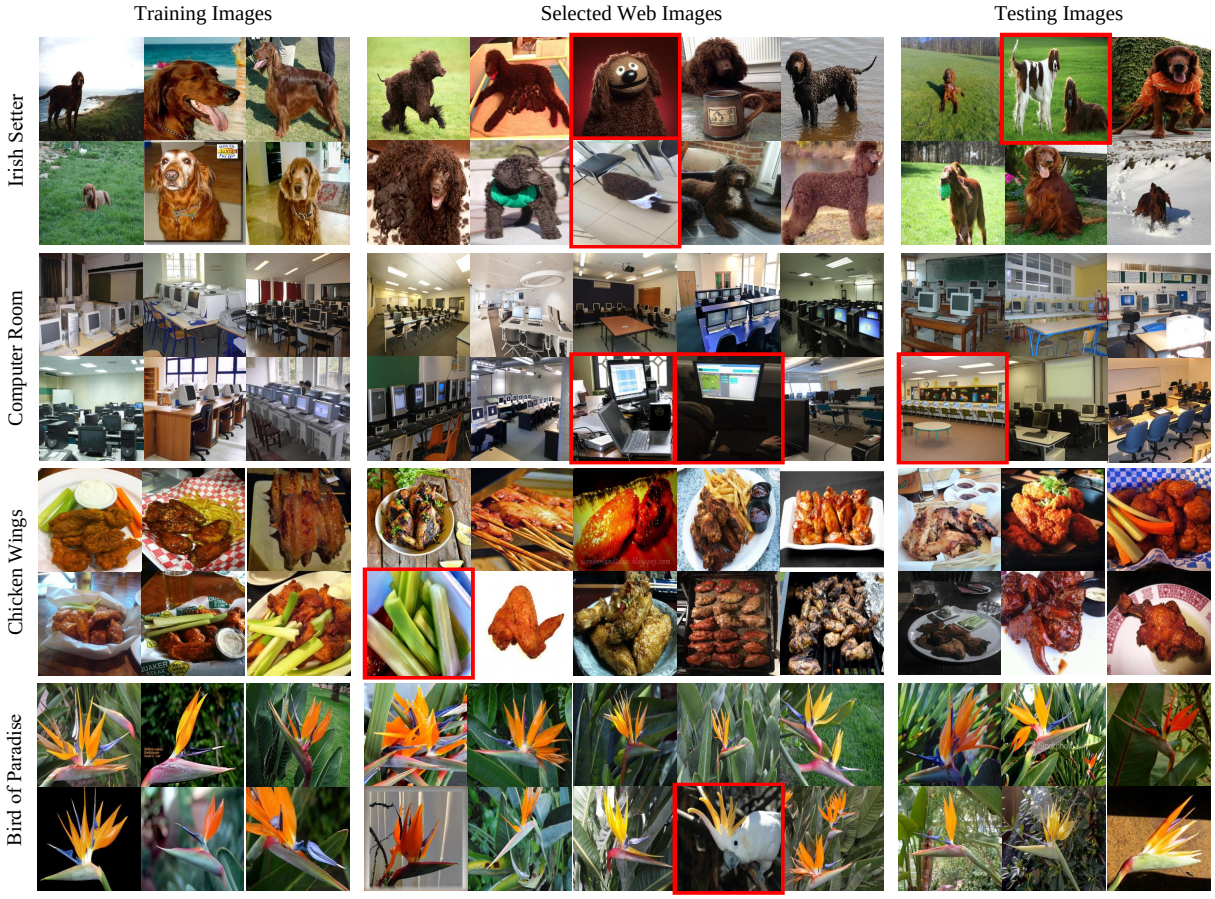

Fig. 9. Samples of the training images, selected web images, and test images on the Dog-120 (Irish Setter), Indoor-67 (Computer Room), Food-101 (Chicken Wings), and Flower-102 (Bird of Paradise) datasets. The images surrounded by red boxes indicate noisy data or wrong predictions.

row of Fig. 9 are easily misidentified as the 'Computer Room' category, since they appear frequently in the training images of this category. And the side dishes in the food (in the sixth row of Fig. 9) are also easily confused with the main dish category. Moreover, we occasionally fail to recognize images that are mislabeled. For example, in the first row of Fig. 9, multi-categories of dogs appear in the testing set of the 'Irish Setter' category in the Dog-120 dataset.

H. Further Analysis

In practice, sometimes it is challenging to collect large amounts of data for each category of some small-scale tasks (*e.g.*, 'laboratory wet' on Indoor-67 [62]) on the web. Besides, the scale of valuable samples for each category may also be imbalanced. These may cause the model to recognize well the common categories with large scale training samples while poorly the rare ones in contrast. In our proposed BiSPL, we observe that the data imbalance is a natural phenomenon, both when collecting web data and sampling training samples. For example, on the Dog-120 [4] dataset, it is more difficult to automatically collect web images for 'Brabancon Griffon' compared to 'Doberman'. During the sampling stage, the BiSPL samples a total of $30,532$ images from the noisy web set for 120 categories before achieving the best performance of 88.66% accuracy, which contains 623 images of 'Samoyed', while only 7 images of 'Papillon'. Our algorithm can automatically mine valuable information from imbalanced web data, while imbalanced training set may bias the model to these majority classes with a relatively large number of samples. Therefore, in the web data learning task, how to obtain a balanced set throughout the web data learning procedure is crucial and we will pay attention to this point in future work.

V. Conclusion

This paper focuses on the data scarcity problem of deep CNNs via learning from web data, where the web data is free to obtain from the Internet and does not need any extra manual annotation. For such purpose, we propose the BiSPL framework which alternatively iterates the procedures of web data sampling and model training. During the sampling phase, we rank the web data by cosine distance and sample confident ones in a meaningful order, *i.e.*, from easy to hard. The training phase optimizes the model from hard to easy via gradually dropping outliers with large losses which are regarded as noise. Extensive experiments on six fine-grained datasets demonstrate the superiority of BiSPL against state-of-the-art methods.

In the future work, we plan to explore more application scenarios for the proposed algorithm, such as experimental evaluation on more tasks (*e.g.*, multi-label classification, object detection, and semantic segmentation) and large-scale datasets. In addition, how to automatically eliminate interference from open set categories in the process of web data learning is also a challenging issue.

VI. Acknowledgement

This work was supported by the National Key Research and Development Program of China Grant (NO. 2018AAA0100403), NSFC (NO.61876094, U1933114), Natural Science Foundation of Tianjin, China (NO.20JCJQJC00020, 18JCYBJC15400, 18ZXZNGX00110).

References

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.
[2] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *NIPS*, 2015.
[3] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.
[4] A. Khosla, N. Jayadevaprakash, B. Yao, and F.-F. Li, "Novel dataset for fine-grained image categorization: Stanford dogs," in *CVPR Workshop*, 2011.
[5] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, "The Caltech-UCSD birds-200-2011 dataset," California Institute of Technology, Tech. Rep. CNS-TR-2011-001, 2011.
[6] Y. Cui, Y. Song, C. Sun, A. Howard, and S. Belongie, "Large scale fine-grained categorization and domain-specific transfer learning," in *CVPR*, 2018.
[7] S. Guo, W. Huang, H. Zhang, C. Zhuang, D. Dong, M. R. Scott, and D. Huang, "Curriculumnet: Weakly supervised learning from large-scale web images," in *ECCV*, 2018.
[8] Z. Wei, J. Zhang, Z. Lin, J.-Y. Lee, N. Balasubramanian, M. Hoai, and D. Samaras, "Learning visual emotion representations from web data," in *CVPR*, 2020.
[9] Y. Shen, R. Ji, Z. Chen, X. Hong, F. Zheng, J. Liu, M. Xu, and Q. Tian, "Noise-aware fully webly supervised object detection," in *CVPR*, 2020.
[10] F. Schroff, A. Criminisi, and A. Zisserman, "Harvesting image databases from the web," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 4, pp. 754–766, 2010.
[11] J. Krause, B. Sapp, A. Howard, H. Zhou, A. Toshev, T. Duerig, J. Philbin, and L. Fei-Fei, "The unreasonable effectiveness of noisy data for fine-grained recognition," in *ECCV*, 2016.
[12] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *ICCV*, 2017.
[13] A. Shrivastava, A. Gupta, and R. Girshick, "Training region-based object detectors with online hard example mining," in *CVPR*, 2016.
[14] B. Li, Y. Liu, and X. Wang, "Gradient harmonized single-stage detector," in *AAAI*, 2019.
[15] J. Yang, X. Sun, Y. Lai, L. Zheng, and M. Cheng, "Recognition from web data: A progressive filtering approach," *IEEE Transactions on Image Processing*, vol. 27, no. 11, pp. 5303–5315, 2018.
[16] D. Tanaka, D. Ikami, T. Yamasaki, and K. Aizawa, "Joint optimization framework for learning with noisy labels," in *CVPR*, 2018.
[17] D. Arpit, S. Jastrzkebski, N. Ballas, D. Krueger, E. Bengio, M. S. Kanwal, T. Maharaj, A. Fischer, A. C. Courville, Y. Bengio, and S. Lacoste-Julien, "A closer look at memorization in deep networks," in *ICML*, 2017.
[18] J. Li, Y. Song, J. Zhu, L. Cheng, Y. Su, L. Ye, P. Yuan, and S. Han, "Learning from large-scale noisy web data with ubiquitous reweighting for image classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.
[19] L. Niu, A. Veeraraghavan, and A. Sabharwal, "Webly supervised learning meets zero-shot learning: A hybrid approach for fine-grained classification," in *CVPR*, 2018.
[20] B. Zhuang, L. Liu, Y. Li, C. Shen, and I. Reid, "Attend in groups: A weakly-supervised deep learning framework for learning from web data," in *CVPR*, 2017.
[21] Y. Tu, L. Niu, J. Chen, D. Cheng, and L. Zhang, "Learning from web data with self-organizing memory module," in *CVPR*, 2020.
[22] C. Gan, T. Yao, K. Yang, Y. Yang, and T. Mei, "You lead, we exceed: Labor-free video concept learning by jointly exploiting web videos and images," in *CVPR*, 2016.
[23] C. Gan, C. Sun, and R. Nevatia, "Deck: Discovering event composition knowledge from web images for zero-shot event detection and recounting in videos," in *AAAI*, 2017.
[24] J. Li, Y. Wong, Q. Zhao, and M. S. Kankanhalli, "Attention transfer from web images for video recognition," in *ACM MM*, 2017.
[25] C. Gan, C. Sun, L. Duan, and B. Gong, "Webly-supervised video recognition by mutually voting for relevant web images and web video frames," in *ECCV*, 2016.
[26] R. C. Wong and C. H. Leung, "Automatic semantic annotation of real-world web images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 11, pp. 1933–1944, 2008.
[27] J. Yang, X. Sun, and L. Chen, "Learning from web data using adversarial discriminative neural networks for fine-grained classification," in *AAAI*, 2019.
[28] Y. Li, J. Yang, Y. Song, L. Cao, J. Luo, and L.-J. Li, "Learning from noisy labels with distillation," in *ICCV*, 2017.
[29] G. Patrini, A. Rozza, A. Krishna Menon, R. Nock, and L. Qu, "Making deep neural networks robust to label noise: A loss correction approach," in *CVPR*, 2017.
[30] K. Yi and J. Wu, "Probabilistic end-to-end noise correction for learning with noisy labels," in *CVPR*, 2019.
[31] J. Goldberger and E. Ben-Reuven, "Training deep neural-networks using a noise adaptation layer," in *ICLR*, 2017.
[32] X. Chen, A. Shrivastava, and A. Gupta, "Neil: Extracting visual knowledge from web data," in *ICCV*, 2013.
[33] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *CVPR*, 2020.
[34] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *ICML*, 2020.
[35] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. A. Pires, Z. D. Guo, M. G. Azar *et al.*, "Bootstrap your own latent: A new approach to self-supervised learning," in *NeurIPS*, 2020.
[36] R. Zhang, P. Isola, and A. A. Efros, "Colorful image colorization," in *ECCV*, 2016.
[37] M. Noroozi and P. Favaro, "Unsupervised learning of visual representations by solving jigsaw puzzles," in *ECCV*, 2016.
[38] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *CVPR*, 2016.
[39] C. Gan, H. Zhao, P. Chen, D. Cox, and A. Torralba, "Self-supervised moving vehicle tracking with stereo sound," in *ICCV*, 2019.
[40] P. Chen, D. Huang, D. He, X. Long, R. Zeng, S. Wen, M. Tan, and C. Gan, "Rspnet: Relative speed perception for unsupervised video representation learning," in *AAAI*, 2021.
[41] J. Wang, Y. Gao, K. Li, Y. Lin, A. J. Ma, H. Cheng, P. Peng, R. Ji, and X. Sun, "Removing the background by adding the background: Towards background robust self-supervised video representation learning," in *CVPR*, 2021.
[42] P. Chen, Y. Zhang, M. Tan, H. Xiao, D. Huang, and C. Gan, "Generating visually aligned sound from videos," *IEEE Transactions on Image Processing*, vol. 29, pp. 8292–8302, 2020.
[43] X. Zhan, Z. Liu, J. Yan, D. Lin, and C. C. Loy, "Consensus-driven propagation in massive unlabeled data for face recognition," in *ECCV*, 2018.
[44] J. Chang, G. Meng, L. Wang, S. Xiang, and C. Pan, "Deep self-evolution clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.
[45] J. Chang, L. Wang, G. Meng, Q. Zhang, S. Xiang, and C. Pan, "Local-aggregation graph networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.
[46] J. Kim, T. Kim, S. Kim, and C. D. Yoo, "Edge-labeling graph neural network for few-shot learning," in *CVPR*, 2019.
[47] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.
[48] J. Guan, Z. Lu, T. Xiang, A. Li, A. Zhao, and J.-R. Wen, "Zero and few shot learning with semantic feature synthesis and competitive learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.
[49] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales, "Learning to compare: Relation network for few-shot learning," in *CVPR*, 2018.
[50] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra *et al.*, "Matching networks for one shot learning," in *NIPS*, 2016.
[51] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *ICML*, 2009.
[52] Y. Zhang, P. David, H. Foroosh, and B. Gong, "A curriculum domain adaptation approach to the semantic segmentation of urban scenes," *IEEE transactions on pattern analysis and machine intelligence*, 2019.
[53] M. P. Kumar, B. Packer, and D. Koller, "Self-paced learning for latent variable models," in *NIPS*, 2010.
[54] D. Meng, Q. Zhao, and L. Jiang, "A theoretical understanding of self-paced learning," *Information Sciences*, vol. 414, pp. 319–328, 2017.

[55] L. Jiang, D. Meng, T. Mitamura, and A. G. Hauptmann, "Easy samples first: Self-paced reranking for zero-example multimedia search," in *ACM MM*, 2014.

[56] D. Zhang, D. Meng, and J. Han, "Co-saliency detection via a self-paced multiple-instance learning framework," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 5, pp. 865–878, 2017.

[57] J. Yang, X. Wu, J. Liang, X. Sun, M.-M. Cheng, P. L. Rosin, and L. Wang, "Self-paced balance learning for clinical skin disease recognition," *IEEE Transactions on Neural Networks and Learning Systems*, 2019.

[58] K. Ghasedi, X. Wang, C. Deng, and H. Huang, "Balanced self-paced learning for generative adversarial clustering network," in *CVPR*, 2019.

[59] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," in *CVPR*, 2019.

[60] E. Kreyszig, *Advanced engineering mathematics (Fourth Edition)*. John Wiley and Sons Ltd, 1979.

[61] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *ECCV*, 2016.

[62] A. Quattoni and A. Torralba, "Recognizing indoor scenes," in *CVPR*, 2009.

[63] L. Bossard, M. Guillaumin, and L. Van Gool, "Food-101 – Mining discriminative components with random forests," in *ECCV*, 2014.

[64] K.-H. Lee, X. He, L. Zhang, and L. Yang, "Cleannet: Transfer learning for scalable image classifier training with label noise," in *CVPR*, 2018.

[65] M.-E. Nilsback and A. Zisserman, "Automated flower classification over a large number of classes," in *ICVGIP*, 2008.

[66] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *CVPR*, 2018.

[67] C. Huang, H. Li, Y. Xie, Q. Wu, and B. Luo, "PBC: Polygon-based classifier for fine-grained categorization," *IEEE Transactions on Multimedia*, vol. 19, no. 4, pp. 673–684, 2016.

[68] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva, "Learning deep features for scene recognition using places database," in *NIPS*, 2014.

[69] X.-S. Wei, J.-H. Luo, J. Wu, and Z.-H. Zhou, "Selective convolutional descriptor aggregation for fine-grained image retrieval," *IEEE Transactions on Image Processing*, vol. 26, no. 6, pp. 2868–2881, 2017.

[70] M. Mohammadi and S. Das, "SNN: Stacked neural networks," *arXiv preprint arXiv:1605.08512*, 2016.

[71] K. Chen and Z. Zhang, "Learning to classify fine-grained categories with privileged visual-semantic misalignment," *IEEE Transactions on Big Data*, vol. 3, no. 1, pp. 37–43, 2016.

[72] M. Dixit, S. Chen, D. Gao, N. Rasiwasia, and N. Vasconcelos, "Scene classification with semantic fisher vectors," in *CVPR*, 2015.

[73] A. Meyers, N. Johnston, V. Rathod, A. Korattikara, A. Gorban, N. Silberman, S. Guadarrama, G. Papandreou, J. Huang, and K. P. Murphy, "Im2Calories: Towards an automated mobile vision food diary," in *ICCV*, 2015.

[74] Y. Zhang, X.-S. Wei, J. Wu, J. Cai, J. Lu, V.-A. Nguyen, and M. N. Do, "Weakly supervised fine-grained categorization with part-based image representation," *IEEE Transactions on Image Processing*, vol. 25, no. 4, pp. 1713–1725, 2016.

[75] T.-Y. Lin, A. RoyChowdhury, and S. Maji, "Bilinear convolutional neural networks for fine-grained visual recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 6, pp. 1309–1322, 2017.

[76] Z. Li, X. Zhu, L. Wang, and P. Guo, "Image classification using convolutional neural networks and kernel extreme learning machines," in *ICIP*, 2018.

[77] A. Dubey, O. Gupta, P. Guo, R. Raskar, R. Farrell, and N. Naik, "Pairwise confusion for fine-grained visual classification," in *ECCV*, 2018.

[78] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, "Places: A 10 million image database for scene recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 6, pp. 1452–1464, 2017.

[79] X. Wei, Y. Zhang, Y. Gong, J. Zhang, and N. Zheng, "Grassmann pooling as compact homogeneous bilinear pooling for fine-grained visual classification," in *ECCV*, 2018.

[80] D. Yoo, S. Park, J.-Y. Lee, and I. So Kweon, "Multi-scale pyramid pooling for deep convolutional representation," in *CVPR*, 2015.

[81] L. Herranz, S. Jiang, and X. Li, "Scene recognition with CNNs: Objects, scales and dataset bias," in *CVPR*, 2016.

[82] H. Hassannejad, G. Matrella, P. Ciampolini, I. De Munari, M. Mordonini, and S. Cagnoni, "Food image recognition using very deep convolutional networks," in *MADiMa Workshop*, 2016.

[83] S. Guo, W. Huang, L. Wang, and Y. Qiao, "Locally supervised deep hybrid model for scene recognition," *IEEE Transactions on Image Processing*, vol. 26, no. 2, pp. 808–820, 2016.

[84] N. Zhang, J. Donahue, R. Girshick, and T. Darrell, "Part-based R-CNNs for fine-grained category detection," in *ECCV*, 2014.

[85] Y. Chai, E. Rahtu, V. Lempitsky, L. Van Gool, and A. Zisserman, "Tricos: A tri-level class-discriminative co-segmentation method for image classification," in *ECCV*, 2012.

[86] J. Krause, H. Jin, J. Yang, and L. Fei-Fei, "Fine-grained recognition without part annotations," in *CVPR*, 2015.

[87] Q. Qian, R. Jin, S. Zhu, and Y. Lin, "Fine-grained visual categorization via multi-stage metric learning," in *CVPR*, 2015.

[88] D. Wang, Z. Shen, J. Shao, W. Zhang, X. Xue, and Z. Zhang, "Multiple granularity descriptors for fine-grained categorization," in *ICCV*, 2015.

[89] K. Sohn, "Improved deep metric learning with multi-class n-pair loss objective," in *NIPS*, 2016.

[90] F. Yu, D. Wang, E. Shelhamer, and T. Darrell, "Deep layer aggregation," in *CVPR*, 2018.

[91] L. Karlinsky, J. Shtok, S. Harary, E. Schwartz, A. Aides, R. Feris, R. Giryes, and A. M. Bronstein, "RepMet: Representative-based metric learning for classification and few-shot object detection," in *CVPR*, 2019.

[92] H. Zheng, J. Fu, T. Mei, and J. Luo, "Learning multi-attention convolutional neural network for fine-grained image recognition," in *ICCV*, 2017.

[93] Z. Yang, T. Luo, D. Wang, Z. Hu, J. Gao, and L. Wang, "Learning to navigate for fine-grained classification," in *ECCV*, 2018.

[94] O. Rippel, M. Paluri, P. Dollar, and L. Bourdev, "Metric learning with adaptive density discrimination," in *ICLR*, 2016.

[95] H. Zheng, J. Fu, Z.-J. Zha, J. Luo, and T. Mei, "Learning rich part hierarchies with progressive attention networks for fine-grained image recognition," *IEEE Transactions on Image Processing*, 2019.

[96] D. Han, Q. Liu, and W. Fan, "A new image classification method using CNN transfer learning and web data augmentation," *Expert Systems with Applications*, vol. 95, pp. 43–56, 2018.

[97] Y. Chen, Y. Bai, W. Zhang, and T. Mei, "Destruction and construction learning for fine-grained image recognition," in *CVPR*, 2019.

[98] X. Zhe, S. Chen, and H. Yan, "Directional statistics-based deep metric learning for image classification and retrieval," *Pattern Recognition*, vol. 93, pp. 113–123, 2019.

[99] Q. Li, X. Peng, L. Cao, W. Du, H. Xing, Y. Qiao, and Q. Peng, "Product image recognition with guidance learning and noisy supervision," *Computer Vision and Image Understanding*, 2019.

[100] P. Li, J. Xie, Q. Wang, and Z. Gao, "Towards faster training of global covariance pooling networks by iterative matrix square root normalization," in *CVPR*, 2018.

[101] J. Ngiam, D. Peng, V. Vasudevan, S. Kornblith, Q. V. Le, and R. Pang, "Domain adaptive transfer learning with specialist models," *arXiv preprint arXiv:1811.07056*, 2018.

[102] W. Zhang, Y. Wang, and Y. Qiao, "Metacleaner: Learning to hallucinate clean representations for noisy-labeled visual recognition," in *CVPR*, 2019.

[103] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *ICML*, 2019.

[104] J. Han, P. Luo, and X. Wang, "Deep self-learning from noisy labels," in *ICCV*, 2019.

[105] T. DeVries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," *arXiv preprint arXiv:1708.04552*, 2017.

[106] G. Van Horn, O. Mac Aodha, Y. Song, Y. Cui, C. Sun, A. Shepard, H. Adam, P. Perona, and S. Belongie, "The inaturalist species classification and detection dataset," in *CVPR*, 2018.

[107] L. v. d. Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.