Learning from Web Data with Self-Organizing Memory Module

Yi Tu, Li Niu*, Junjie Chen, Dawei Cheng, and Liqing Zhang*
MoE Key Lab of Artificial Intelligence, Department of Computer Science and Engineering
Shanghai Jiao Tong University, Shanghai, China
{tuyi1991,ustcnewly,chen.bys,dawei.cheng}@sjtu.edu.cn,zhang-lq@cs.sjtu.edu.cn

Abstract

Learning from web data has attracted lots of research interest in recent years. However, crawled web images usually have two types of noises, label noise and background noise, which induce extra difficulties in utilizing them effectively. Most existing methods either rely on human supervision or ignore the background noise. In this paper, we propose a novel method, which is capable of handling these two types of noises together, without the supervision of clean images in the training stage. Particularly, we formulate our method under the framework of multi-instance learning by grouping ROIs (i.e., images and their region proposals) from the same category into bags. ROIs in each bag are assigned with different weights based on the representative/discriminative scores of their nearest clusters, in which the clusters and their scores are obtained via our designed memory module. Our memory module could be naturally integrated with the classification module, leading to an end-to-end trainable system. Extensive experiments on four benchmark datasets demonstrate the effectiveness of our method.

1. Introduction

Deep learning is a data-hungry method that demands large numbers of well-labeled training samples, but acquiring massive images with clean labels is expensive, time-consuming, and labor-intensive. Considering that there are abundant freely available web data online, learning from web images could be promising. However, web data have two severe flaws: label noise and background noise. Label noise means the incorrectly labeled images. Since web images are usually retrieved by using the category name as the keyword when searching from public websites, unrelated images might appear in the searching results. Different from label noise, background noise is caused by the cluttered and diverse contents of web images compared with standard datasets. Specifically, in manually labeled datasets like Cifar-10, the target objects of each category usually appear at the center and occupy relatively large areas, yielding little background noise. However, in web images, background or irrelevant objects may occupy the majority of the whole image. One example is provided in Figure 1, in which two images are crawled with the keyword "dog". The left image belongs to label noise since it has dog food, which is indirectly related to "dog". Meanwhile, the right image belongs to background noise because the grassland occupies the majority of the whole image, and a kid also takes a salient position.

Label noise **Background noise**

Figure 1. Two web images crawled with keyword "dog". **Left:** Dog food; **Right:** A kid and a dog on the grassland.

There are already many studies [33, 23, 36, 46, 52, 16, 31, 32, 34] on using web images to learn classifiers. However, most of them [53, 24, 13, 33, 23, 28, 19] only focused on label noise. In contrast, some recent works began to consider the background noise. In particular, Zhuang *et al*. [60] used attention maps to suppress background noise, but this method did not fully exploit the relationship among different regions, which might limit its ability to remove noisy regions. Sun *et al*. [46] utilized weakly supervised region proposal network to distill clean region proposals from web images, but this approach requires extra clean images in the training stage.

In this work, we propose a novel method to address the label noise and background noise simultaneously without using human annotation. We first use an unsupervised proposal extraction method [61] to capture image regions which are likely to contain meaningful objects. In the re-

*Corresponding author.

mainder of this paper, we use "ROI" (Region Of Interest) to denote both images and their region proposals. Following the idea of multi-instance learning, ROIs from the same category are grouped into bags and the ROIs in each bag are called instances. Based on the assumption that there is at least a proportion of clean ROIs in each bag, we tend to learn different weights for different ROIs with lower weights indicating noisy ROIs, through which label/background noise can be mitigated. With ROI weights, we can use the weighted average of ROI-level features within each bag as bag-level features, which are cleaner than ROI-level features and thus more suitable for training a robust classifier.

Instead of learning weights via self-attention like [17, 60], in order to fully exploit the relationship among different ROIs, we tend to learn ROI weights by comparing them with prototypes, which are obtained via clustering bag-level features. Each cluster center (*i.e.*, prototype) has a representative (*resp.*, discriminative) score for each category, which means how this cluster center is representative (*resp.*, discriminative) for each category. Then, the weight of each ROI can be calculated based on its nearest cluster center for the corresponding category. Although the idea of the prototype has been studied in many areas such as semi-supervised learning [6] and few-shot learning [43], they usually cluster the samples within each category, while we cluster bag-level features from all categories to capture the cross-category relationship.

Traditional clustering methods like K-means could be used to cluster bag-level features. However, we use recently proposed key-value memory module [29] to achieve this goal, which is more powerful and flexible. The memory module could be integrated with the classification module, yielding an end-to-end trainable system. Moreover, it can online store and update the category-specific representative/discriminative scores of cluster centers at the same time. As a minor contribution, we adopt the idea of Self-Organizing Map [48] to improve the existing memory module to stabilize the training process.

Our contributions can be summarized as follows: 1) The major contribution is handling the label/background noise of web data under the multi-instance learning framework with the memory module; 2) The minor contribution is proposing the self-organizing memory module to stabilize the training process and results; 3) The experiments on several benchmark datasets demonstrate the effectiveness of our method in learning classifiers with web images.

2. Related Work

2.1. Webly Supervised Learning

For learning from web data, previous works focus on handling the label noise in three directions, removing label noise [42, 7, 8, 30, 53, 24, 33, 23, 28, 12, 39, 14], building noise-robust model [4, 10, 5, 36, 44, 3, 38, 49, 47, 21], and curriculum learning [13, 19]. The above approaches focused on label noise. However, web data also have background noise, as mentioned in [46]. To address this issue, Zhuang *et al*. [60] utilized the attention mechanism [50] to reduce the attention on background regions while Sun *et al*. [46] used a weakly unsupervised object localization method to reduce the background noise.

Most previous works utilize extra information like a small clean dataset or only consider the label noise issue. In contrast, our method can solve both label noise and background noise by only using noisy web images in the training stage.

2.2. Memory Networks

Memory networks were recently introduced to solve the question answering task [18, 45, 29]. Memory network was first proposed in [18] and extended to be end-to-end trainable in [45]. Miller *et al*. [29] added the key and value module for directly reading documents, rendering the memory network more flexible and powerful. More recently, memory networks have been employed for one-shot learning [41, 20], few-shot learning [55, 58], and semi-supervised learning [6].

Although memory networks have been studied in many tasks, our work is the first to utilize the memory network to handle the label/background noise of web data.

2.3. Multi-Instance Learning

In multi-instance learning (MIL), multiple instances are grouped into a bag, with at least one positive instance triggering the bag-level label. The main goal of MIL is to learn a robust classifier with unknown instance labels. Some early methods based on SVM [2, 27] treat one bag as an entirety or infer instance labels within each bag. In the deep learning era, various pooling operations have been studied like mean pooling and max pooling [37, 59, 11]. Different from these non-trainable pooling operators, some works [35, 17, 22, 51] proposed trainable operators to learn different weights for different instances. By utilizing the attention mechanism, Pappas and PopescuBelis [35] proposed an attention-based MIL with attention weights trained in an auxiliary linear regression model. AD-MIL [17] took a further step and designed permutation-invariant aggregation operator with the gated attention mechanism.

Under the MIL framework, we utilize a memory module to learn weights for the instances in each bag, which has not been explored before.

3. Methodology

In this paper, we denote a matrix/vector by using a uppercase/lowercase letter in boldface (*e.g.*, $\mathbf{A}$ denotes a matrix

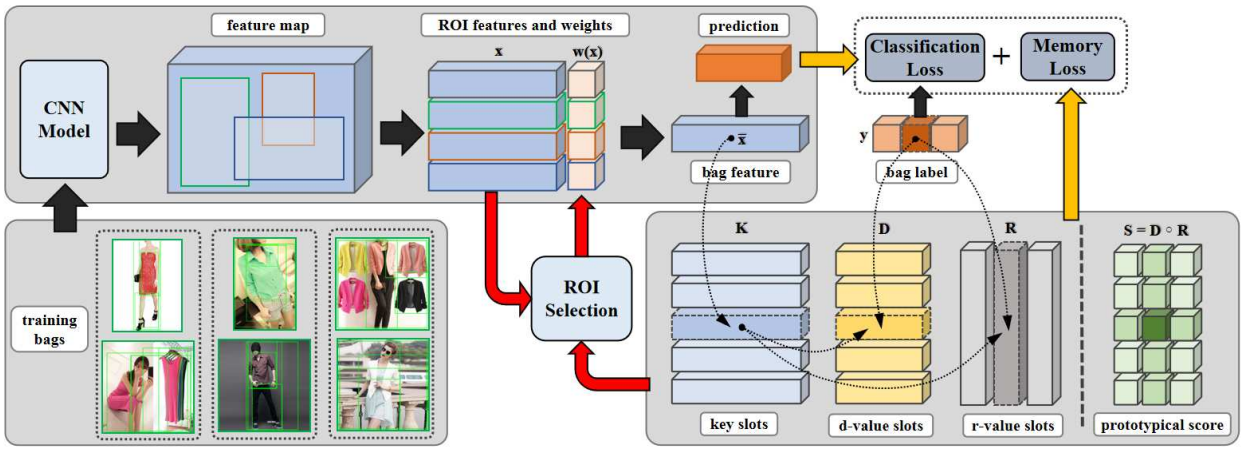

Figure 2. Illustration of our method. **Black Arrow:** Update the CNN model with bag-level features and bag labels. **Dashed Arrow:** Update the memory module with bag-level features and bag labels. **Red Arrow:** Update the weights of ROIs based on the memory module. The whole training algorithm is listed in Algorithm 1.

and $\mathbf{a}$ denotes a vector). $\mathbf{a}_i$ denotes a particular row or column of $\mathbf{A}$ indexed by subscript i. $a_{i,j}$ denotes an element of $\mathbf{A}$ at the i-th row and j-th column. Moreover, we use $\mathbf{a}^T$ to denote the transpose of $\mathbf{a}$, and $\mathbf{A} \circ \mathbf{B}$ to denote the element-wise product between $\mathbf{A}$ and $\mathbf{B}$.

3.1. Overview of Our Method

The flowchart of our method is illustrated in Figure 2. We first extract region proposals for each image using the unsupervised EdgeBox method [61]. By tuning the hyper-parameters in EdgeBox, we expect the extracted proposals to cover most objects to avoid missing important information (see details in Section 4.2). We group ROIs (*i.e.*, images and their proposals) from the same category into training bags, in which ROIs within each bag are treated as instances. To assign different weights to different ROIs in each bag, we compare each ROI with its nearest key in the memory module. Then we take the weighted average of ROI-level features as bag-level features, which are used to train the classifier and update the memory module.

3.2. Multi-Instance Learning Framework

We build our method under the multi-instance learning framework. Particularly, we group several images of the same category and their region proposals into one bag, so that each bag has multiple instances (*i.e.*, ROIs). We aim to assign higher weights to clean ROIs and use the weighted average of ROI-level features within each bag as bag-level features, which are supposed to be cleaner than ROI-level features.

Formally, we use $\mathcal{S}$ to denote the training set of multiple bags, and $\mathcal{B} \in \mathcal{S}$ denotes a single training bag. Note that we use the same number n_g of images in each bag and generate the same number n_p of region proposals for each image, leading to the same number n_b of ROIs in each bag with $n_b = n_g(n_p + 1)$. Specifically, $\mathcal{B} = \{\mathbf{x}_i | i = 1, 2, ..., n_b\}$ is a bag of n_b ROIs, in which $\mathbf{x}_i \in \mathcal{R}^d$ is the d-dim feature vector of i-th ROI. We use $\mathtt{w}(\mathbf{x}_i)$ to denote the weight of $\mathbf{x}_i$ with $\sum_{\mathbf{x}_i \in \mathcal{B}} \mathtt{w}(\mathbf{x}_i) = 1$. As shown in Figure 2, the features of ROIs are pooled from the corresponding regions on the feature map of the last convolutional layer in CNN model, similar to [40].

Given a bag with category label $y \in [1, 2, ..., C]$ with C being the number of total categories, we can also represent its bag label as a C-dim one-hot vector $\mathbf{y}$ with only the y-th element being one. After assigning weight $\mathtt{w}(\mathbf{x}_i)$ to each $\mathbf{x}_i$, we use weighted average of ROI features in each bag as the bag-level feature: $\bar{\mathbf{x}} = \sum_{\mathbf{x}_i \in \mathcal{B}} \mathtt{w}(\mathbf{x}_i) \cdot \mathbf{x}_i \in \mathcal{R}^d$. Our classification module is based on bag-level features with the cross-entropy loss:

$$\mathcal{L}_{\text{cls}} = -\sum_{\mathcal{B} \in \mathcal{S}} \mathbf{y}^T \log \left(f\Big(\sum_{\mathbf{x}_i \in \mathcal{B}} \mathtt{w}(\mathbf{x}_i) \cdot \mathbf{x}_i \Big) \right), \qquad (1)$$

in which $f(\cdot)$ is a softmax classification layer. At the initialization step, the weights of region proposals are all set as zero while the images are assigned with uniform weights in each bag. We use $\bar{\mathtt{w}}(\mathbf{x}_i)$ to denote such initialized ROI weights. After initializing the CNN model, we tend to learn different weights for ROIs by virtue of the memory module. Next, we first introduce our memory module and then describe how to assign weights to ROIs based on our memory module.

3.3. Self-Organizing Memory Module

The basic function of our memory module is clustering bag-level features and each cluster center can be treated as a prototype [42, 7, 8, 6, 43]. Although traditional clustering methods like K-means can realize a similar function, the

memory module is more flexible and powerful. Specifically, the memory module can be easily integrated with the classification module, leading to an end-to-end trainable system. Besides, the memory module can simultaneously store and update additional useful information, *i.e.*, category-specific representative/discriminative scores of each cluster center.

3.3.1 Memory Module Architecture:

Our memory module consists of key slots and values slots. The key slots contain cluster centers, while the value slots contain their corresponding category-specific representative/discriminative scores.

We use $\mathbf{K} \in \mathcal{R}^{d\times L}$ to denote key slots, in which L is the number of key slots and the l-th column $\mathbf{k}_l$ is the l-th key slot representing the l-th cluster center.

To capture the relationship between clusters and categories, we design two value slots. We investigate two types of cluster-category relationships, *i.e.*, how discriminative and how representative a learned cluster is to a category. Correspondingly, we have two types of value slots: the "discriminative" value slots (d-value slots) and the "representative" value slots (r-value slots). Each pair of d-value slot and r-value slot corresponds to one key slot. Formally, we use $\mathbf{D} \in \mathcal{R}^{C\times L}$ (*resp.*, $\mathbf{R} \in \mathcal{R}^{C\times L}$) to denote d-value (*resp.*, r-value) slots, where $d_{y,l}$ (*resp.*, $r_{y,l}$) is the discriminative (*resp.*, representative) score of $\mathbf{k}_l$ for the y-th category.

To better explain discriminative/representative score, we assume that L clusters are obtained based on all training bags and $n_{y,l}$ is the number of training bags from the y-th category in the l-th cluster. Then, we can calculate $\tilde{d}_{y,l} = \frac{n_{y,l}}{\sum_{y=1}^{C} n_{y,l}}$ and $\tilde{r}_{y,l} = \frac{n_{y,l}}{\sum_{l=1}^{L} n_{y,l}}$. Intuitively, $\tilde{d}_{y,l}$ means the percentage of the bags from the y-th category among the bags in the l-th cluster. The larger $\tilde{d}_{y,l}$ is, the more discriminative the l-th cluster is to the y-th category. So we expect $d_{y,l}$ in d-value slots $\mathbf{D}$ to approximate $\tilde{d}_{y,l}$. Similarly, $\tilde{r}_{y,l}$ means the percentage of the bags in the l-th cluster among the bags from the y-th category. The larger $\tilde{r}_{y,l}$ is, the more representative the l-th cluster is to the y-th category. So we expect $r_{y,l}$ in r-value slots $\mathbf{R}$ to approximate $\tilde{r}_{y,l}$.

3.3.2 Memory Module Updating:

With all key and value slots randomly initialized, they are updated based on the bag-level feature $\bar{\mathbf{x}}$ of training bag $\mathcal{B}$ from c-th category and its one-hot label vector $\mathbf{y}$.

First, we seek for the cluster that $\bar{\mathbf{x}}$ belongs to, which is also referred to as the "winner key slot" of $\bar{\mathbf{x}}$. Precisely, we calculate the cosine similarity between $\bar{\mathbf{x}}$ and all keys as $\cos(\bar{\mathbf{x}}, \mathbf{k}_l) = \frac{\mathbf{k}_l^T \bar{\mathbf{x}}}{\|\mathbf{k}_l\|_2 \|\bar{\mathbf{x}}\|_2}$ for $l = 1, 2, ..., L$, and find the winner key slot $\mathbf{k}_z$ of $\bar{\mathbf{x}}$ where:

$$z = \arg\max_{l} \cos(\bar{\mathbf{x}}, \mathbf{k}_l) \tag{2}$$

After determining that $\bar{\mathbf{x}}$ belongs to the z-th cluster, the cluster center $\mathbf{k}_z$ needs to be updated based on $\bar{\mathbf{x}}$. Unlike the previous approach [58] which updates cluster centers with computed gradients, we employ a loss function which is more elegant and functionally similar:

$$\mathcal{L}_{\text{key}} = -\sum_{\mathcal{B}\in\mathcal{S}} \cos(\bar{\mathbf{x}}, \mathbf{k}_z), \tag{3}$$

which can push the winner cluster center $\mathbf{k}_z$ closer to $\bar{\mathbf{x}}$.

With similar loss functions, we also update d-value slots and r-value slots accordingly. For d-value slots $\mathbf{D}$, recall that we expect $d_{y,i}$ to approximate $\tilde{d}_{y,i}$ which means the percentage of the bags from the y-th category among the bags in the i-th cluster. Then, the z-th column $\mathbf{d}_z$ of $\mathbf{D}$ could represent the category distribution in the z-th cluster, so we need to update $\mathbf{d}_z$ with the label vector $\mathbf{y}$ of $\bar{\mathbf{x}}$ as follows,

$$\mathcal{L}_{\text{d-value}} = -\sum_{\mathcal{B}\in\mathcal{S}} \cos(\mathbf{y}, \mathbf{d}_z), \tag{4}$$

$$\text{s.t. } \|\mathbf{d}_z\|_1 = 1,\ \mathbf{d}_z \geq \mathbf{0}. \tag{5}$$

$\mathcal{L}_{\text{d-value}}$ can push $\mathbf{d}_z$ towards $\mathbf{y}$ while maintaining $\mathbf{d}_z$ as a valid distribution with (5), so $d_{y,z}$ will approximate $\tilde{d}_{y,z}$ eventually.

For r-value slots $\mathbf{R}$, recall that we expect $r_{y,i}$ to approximate $\tilde{r}_{y,i}$ which means the percentage of the bags in the i-th cluster among the bags from the y-th category. Then, $\mathbf{r}_y$, the y-th row of $\mathbf{R}$, could represent the distribution of all bags from the y-th category over all clusters, so we need to update $\mathbf{r}_y$ with the one-hot cluster indicator vector $\mathbf{z}$ of $\bar{\mathbf{x}}$ (only the z-th element is 1) as follows,

$$\mathcal{L}_{\text{r-value}} = -\sum_{\mathcal{B}\in\mathcal{S}} \cos(\mathbf{z}, \mathbf{r}_y), \tag{6}$$

$$\text{s.t. } \|\mathbf{r}_y\|_1 = 1,\ \mathbf{r}_y \geq \mathbf{0}. \tag{7}$$

Similar to $\mathbf{d}_z$, $\mathcal{L}_{\text{r-value}}$ can push $\mathbf{r}_y$ towards $\mathbf{z}$ while keeping it a valid distribution with (7), so $r_{y,z}$ will approximate $\tilde{r}_{y,z}$ eventually. The theoretical proof and more details for $\mathcal{L}_{\text{d-value}}$ and $\mathcal{L}_{\text{r-value}}$ can be found in Supplementary.

3.3.3 Self-Organizing Map (SOM) Extension:

A good clustering algorithm should be insensitive to initialization and produce balanced clustering results. Inspired by Self-Organizing Map [48], we design a neighborhood constraint on the key slots to achieve this goal, leading to our self-organizing memory module.

In particular, we arrange the key slots on a square grid. When updating the winner key slot $\mathbf{k}_z$, we also update its spatial neighbors. The neighborhood of $\mathbf{k}_z$ is defined as $\mathcal{N}(\mathbf{k}_z, \delta) = \{\mathbf{k}_i | \text{geo}(\mathbf{k}_z, \mathbf{k}_i) \leq \delta\}$, in which $\text{geo}(\cdot, \cdot)$ is the geodesic distance of two key slots on the square grid and δ is a hyper-parameter that controls the neighborhood size.

Then, the key loss $\mathcal{L}_{\text{key}}$ in (3) can be replaced by

$$\mathcal{L}_{\text{SOM-key}} = -\sum_{\mathcal{B}\in\mathcal{S}} \sum_{\mathbf{k}_i \in \mathcal{N}(\mathbf{k}_z, \delta)} \eta(\mathbf{k}_z, \mathbf{k}_i) \cdot \cos(\bar{\mathbf{x}}, \mathbf{k}_i), \quad (8)$$

in which $\eta(\mathbf{k}_z, \mathbf{k}_i) = (1 + \text{geo}(\mathbf{k}_z, \mathbf{k}_i))^{-1}$ is the weight assigned to $\mathbf{k}_i$ and negatively correlated with the geodesic distance (see more technical details in the Supplementary). In summary, the total loss of updating our self-organizing memory module can be written as:

$$\mathcal{L}_{\text{memory}} = \mathcal{L}_{\text{SOM-key}} + \mathcal{L}_{\text{d-value}} + \mathcal{L}_{\text{r-value}}. \quad (9)$$

3.4. ROI Selection Based on Memory Module

Based on the memory module, we can assign different weights to different ROIs in each bag. Specifically, given a ROI $\mathbf{x}$ with its bag label y, we first seek for its winner key slot $\mathbf{k}_z$, and obtain the discriminative (*resp.*, representative) score of $\mathbf{k}_z$ for the y-th category, that is, $d_{y,z}$ (*resp.*, $r_{y,z}$). For a clean ROI, we expect its winner key to be both discriminative and representative for its category. For ease of description, we define $\mathbf{S} = \mathbf{D} \circ \mathbf{R}$ with $s_{y,z} = d_{y,z} \cdot r_{y,z}$. We refer to $s_{y,z}$ as the prototypical score of $\mathbf{k}_z$ for the y-th category. Therefore, ROIs with higher prototypical scores are more prone to be clean ROIs.

Besides the prototypical score, we propose another discount factor by considering ROI areas. Intuitively, we conjecture that smaller ROIs are less likely to have meaningful content and thus should be penalized. Thus, we use area score (a-score) $\sigma(\cdot)$ to describe the relative size of each ROI. Recall that there are two types of ROIs in each bag: image and region proposal. For original images, we set $\sigma(\mathbf{x}) = 1$. For region proposals, we calculate $\sigma(\mathbf{x})$ as the ratio between the area of region proposal $\mathbf{x}$ and the maximum area of all region proposals (excluding the full image) from the same image. To this end, we use a-score $\sigma(\mathbf{x})$ to discount $s_{y,z}$, resulting in a new weight for $\mathbf{x}$:

$$\mathtt{w}(\mathbf{x}) = s_{y,z} \cdot \sigma(\mathbf{x}). \quad (10)$$

After calculating the ROI weights based on (10), we only keep the top p (*e.g.*, 10%) weights of ROIs in each bag while the other weights are set to be zero. The ROI weights in each bag are then normalized so that they sum to one.

3.5. Training Algorithm

For better representation, we use $\boldsymbol{\theta}_{\text{cnn}}$ to denote the model parameters of CNN and $\boldsymbol{\theta}_{\text{mem}}$ to denote $\{\mathbf{K}, \mathbf{D}, \mathbf{R}\}$ in memory module.

Algorithm 1 : The Training Process of Our Network

Require: Bags of ROIs $\mathcal{B}$ and bag label $\mathbf{y}$. Initialize p =10%. Initialize ROI weights as $\bar{\mathtt{w}}(\mathbf{x}_i)$ in Section 3.2.

Ensure: Model parameters $\{\boldsymbol{\theta}_{\text{cnn}}, \boldsymbol{\theta}_{\text{mem}}\}$.

1: Initialize $\boldsymbol{\theta}_{\text{cnn}}$ based on (1) with $\bar{\mathtt{w}}(\mathbf{x}_i)$.
2: Initialize $\boldsymbol{\theta}_{\text{mem}}$ based on (9) with $\bar{\mathtt{w}}(\mathbf{x}_i)$.
3: **Repeat**:
4: Update $\boldsymbol{\theta}_{\text{cnn}}$ and $\boldsymbol{\theta}_{\text{mem}}$ based on (11) while $\mathtt{w}(\mathbf{x}_i)$ are updated based on (10) accordingly.
5: $p \leftarrow p + 5\%$.
6: Break if $p > 40\%$.
7: **End Repeat.**

At first, we utilize initial ROI weights $\bar{\mathtt{w}}(\mathbf{x}_i)$ mentioned in Section 3.2 to obtain weighted average of ROI-level features as bag-level features, which are used to train the CNN model $\boldsymbol{\theta}_{\text{cnn}}$ and the memory module $\boldsymbol{\theta}_{\text{mem}}$. Then, we train the whole system in an end-to-end manner. Specifically, we train the CNN model $\boldsymbol{\theta}_{\text{cnn}}$ and the memory module $\boldsymbol{\theta}_{\text{mem}}$ with the bag-level features $\bar{\mathbf{x}} = \sum_{\mathbf{x}_i \in \mathcal{B}} \mathtt{w}(\mathbf{x}_i) \cdot \mathbf{x}_i$, while the weights of ROIs $\mathtt{w}(\mathbf{x}_i)$ are updated accordingly based on updated $\boldsymbol{\theta}_{\text{cnn}}$ and $\boldsymbol{\theta}_{\text{mem}}$. In this way, cleaner bag-level features can help learn better key slots and value slots in the memory module, while the enhanced memory module can assign more reliable ROI weights and contribute to cleaner bag-level features in turn. The total loss of the whole system can be written as:

$$\mathcal{L}_{\text{all}} = \mathcal{L}_{\text{cls}} + \mathcal{L}_{\text{memory}}. \quad (11)$$

For better performance, we leverage the idea of curriculum learning [56]. It suggests that when training a model, we should start with clean or simple training samples to have a good initialization, and then add noisy or difficulty training samples gradually to improve the generalization ability of the model. After calculating the ROI weights, the top-score ROIs in each bag should be cleaner than the low-score ones. So p is used as a threshold parameter to filter out the noisy ROIs in each bag. Following the idea of curriculum learning, p is set to be relatively small at first so that the selected ROIs are very discriminative and representative. Then we increase p gradually to enhance the generalization ability of the trained model. The total training process can be seen in Algorithm 1.

For evaluation, we directly use the well-trained CNN model to classify test images based on image-level features without extracting region proposals. The memory module is only used to denoise web data in the training stage, but not used in the testing stage.

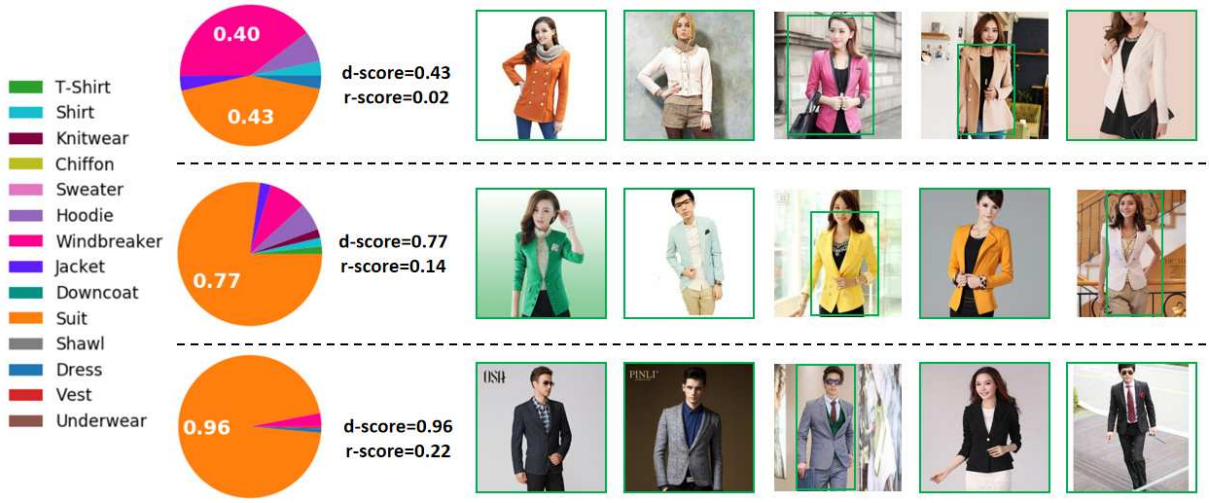

Figure 3. Visualization of our memory module on Clothing1m dataset. We exhibit three key slots with "suit" as the category of interest, with each row standing for one key slot. The left pie chart shows the category distribution in each key slot, and the right images are representative ROIs belonging to each key slot. The d-score and r-score for "suit" category of each key slot are also reported.

4. Experiments

In this section, we introduce the experimental settings and demonstrate the performance of our proposed method.

4.1. Datasets

Clothing1M: Clothing1M [54] is a large-scale fashion dataset designed for webly supervised learning. It contains about one million clothing images crawled from the Internet, and the images are categorized into 14 categories. Most images are associated with noisy labels extracted from their surrounding texts and used as the training set. A few images with human-annotated clean labels are used as the clean dataset for evaluation.

Food-101 & Food-101N: Food-101 dataset [1] is a large food image dataset collected from *foodspotting.com*. It has 101 categories and 1k images for each category with human-annotated labels. Food-101N is a web dataset provided by [24]. It has 310k images crawled with the same taxonomy in Food101 from several websites (excluding *foodspotting.com*). In our experiments, we use Food-101N for training and Food-101 for evaluation.

Webvision & ILSVRC: The WebVision dataset [26] is composed of training, validation, and test set. The training set is crawled from Flickr and Google by using the same 1000 semantic concepts as in the ILSVRC-2012 [9] dataset. It has 2.4 million images with noisy labels. The validation and test set are manually annotated. In our experiments, we only use the WebVision training set for training but perform the evaluation on both WebVision validation set (50k) and ILSVRC-2012 validation set (50k).

4.2. Implementation Details

We adopt ResNet50 [15] as the CNN model and use the output of its last convolutional layer as the feature map to extract ROI features. For Clothing1M and Food101N, we use ResNet50 pretrained on ImageNet following previous works [24, 13]. For WebVision and ImageNet, ResNet50 is trained from scratch with the web training images in WebVision.

For the proposal extractor (*i.e.*, Edge Boxes), there are two important parameters `MaxBoxes` and `MinBoxArea`, in which `MaxBoxes` controls the maximal number of returned region proposals and `MinBoxArea` determines the minimal area. In our experiments, we use `MaxBoxes` $= 20$ (*i.e.*, $n_p = 20$) and `MinBoxArea` $= 5000$. By default, we use two images in each training bag (*i.e.*, $n_g = 2$), so the number of ROIs in each bag is $n_b = n_g(n_p + 1) = 2 \times (20 + 1) = 42$.

4.3. Qualitative Analyses

In this section, we provide in-depth qualitative analyses to elaborate on how our method works. We first explore the memory module and then explore training bags.

Memory Module: By taking Clothing1M dataset as an example and "suit" as category of interest, we choose three key slots for illustration in Figure 3, in which each row stands for one key slot with its corresponding d-score and r-score. To visualize each key slot, we select five ROIs with highest cosine similarity to this key slot, *i.e.*, $\cos(\mathbf{x}_i, \mathbf{k}_l)$.

The first key slot almost clusters an equal number of bags from both "Suit" and "Windbreaker" as the pie chart shows, so it has the lowest d-score. In the meantime, its total number of bags from "Suit" is smaller than those of the other

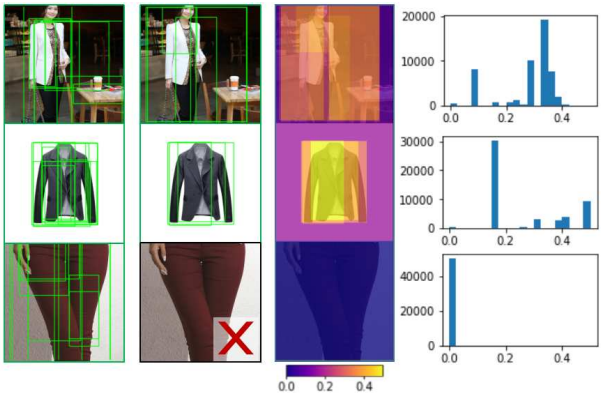

Figure 4. **Column-1:** A training bag for "Suit" with $n_g = 3$. **Column-2:** Top 30% ROIs selected by our network. The bottom image together with all its region proposals are removed. **Column-3&4**: The heat map obtained by summing over ROI weights and the histogram of the heat map pixels.

two slots, so its r-score is also the lowest. Hence, such a key slot is neither discriminative nor representative to "Suit".

The other two key slots are very discriminative for the category "Suit" and have a very high d-score. However, the third key slot is more representative for "Suit" than the second one, resulting in a higher r-score. The explanation is that the total number of bags with colorful suits (second key slot) is smaller than those with black/grey suits (third key slot), so we claim that the third key slot is more representative to "Suit". Combining the r-score and d-score, we would claim that the third key slot is the most prototypical to "Suit" (see the visualization of all $L = 144$ key slots in Supplementary).

Training Bags: Based on the memory module, different ROIs within each bag are assigned with different weights, according to their areas and prototypical scores of their nearest key slots. In Figure 4, we show a training bag with three images (*i.e.*, $n_g = 3$). By comparing the first and the second column, we can observe that the noisy images and noisy region proposals have been removed based on the learned ROI weights. By summing over the ROI weights, we can obtain the attention heat map with brighter colors indicating higher weights. The heat maps and their corresponding histograms are shown in the third and fourth columns, respectively. It can be seen that the background region has lower weights than the main objects. Therefore, the results in Figure 4 demonstrate the ability of our network to address the label noise and background noise.

4.4. Ablation Study

We first compare the results of our method with different n_g and L in Table 1, by taking the Clothing1M dataset as an example.

The bag size: As Table 1 shows, our method with $n_g = 2$ achieves the best performance, so we use it as the default parameter in the rest experiments. Furthermore, it can be seen that the performance with $n_g = 1$ is worse than those with $n_g > 1$, because our method with only one image in a bag will be unable to reduce the label noise when it is a noisy image.

Parameter n_g	1	**2**	3	4	5
Accuracy	72.9	**82.1**	81.1	78.5	75.9
Parameter L	4^2	8^2	$\mathbf{12^2}$	16^2	20^2
Accuracy	78.7	81.7	**82.1**	81.9	81.6

Table 1. Accuracies (%) of our method with different n_g and L on the Clothing1M. The best result is denoted in boldface.

The number of key slots: As in Table 1, the performance of our method is quite robust when the number of key slots is big enough ($L \geq 8 \times 8$), while a too-small number ($L = 4 \times 4$) will lead to a performance drop. Notice that the best performance is achieved with $L = 12 \times 12$ while the Clothing1M dataset has 14 categories, it suggests that when each category can occupy about $12 \times 12 \div 14 \approx 10$ clustering centers in the memory module, our method will generally achieve satisfactory performance. Following this observation, we have $L = 12 \times 12$ for Clothing1M, $L = 32 \times 32$ for Food101, and $L = 100 \times 100$ for WebVision and ImageNet in the rest experiments.

Secondly, we study the contribution of each component in our method in Table 2. ResNet50 and ResNet50+ROI are two naive baselines, and SOMNet denotes our method. ResNet50+ROI uses both images and region proposals as input, but it only achieves slight improvement over ResNet50, which shows that simply using proposals as data augmentation is not very effective.

Three types of scores: Recall that we use three types of scores to weight ROIs: r-score, d-score, and a-score. To investigate their benefit, we report the performance of our method by ablating each type of score, denoted by SOMNet(w/o d-score), SOMNet(w/o r-score), and SOMNet(w/o a-score) in Table 2. We observe that the performance will decrease with the absence of each type of score, which indicates the effectiveness of our designed scores.

Curriculum learning: Notice we utilize the idea of curriculum learning by gradually increasing p from 10% to 40% during training. In Table 2, SOMNet ($p = 40\%$) denotes the result of directly using $p = 40\%$ from the start of training, which has a performance drop of 0.8%. It proves the effectiveness of using curriculum learning.

Background noise removal: We claimed that web data have both label noise and background noise. To study the influence of background noise, we only handle label noise by not using region proposals, and the result is denoted by SOMNet (w/o ROI) in Table 2. To make a fair comparison, we find out that $n_g = 5$ and $p = 60\%$ are the opti-

ResNet50	ResNet50+ROI	SOMNet (w/o d-score)	SOMNet (w/o r-score)	SOMNet (w/o a-score)
68.6	69.1	76.8	73.5	74.8
SOMNet ($p = 40\%$)	SOMNet (w/o ROI)	SOMNet+K-means	SOMNet (w/o SOM)	**SOMNet**
81.3	74.1	79.5	78.2	**82.1**

Table 2. Accuracies (%) of our method and special cases on the Clothing1M. The best result is denoted in boldface.

Training set	**Clothing1M**	**Food101N**	**WebVision**			
Test set	**Clothing1M**	**Food101**	**WebVision**		**ImageNet**	
Evaluation metric	Top-1	Top-1	Top-1	Top-5	Top-1	Top-5
ResNet50	68.6	77.4	66.4	83.4	57.7	78.4
2014 Sukhbaatar *et al*. [44]	71.9	80.8	67.1	84.2	58.4	79.5
2015 DRAE [53]	73.0	81.1	67.5	85.1	58.6	79.4
2017 Zhuang *et al*. [60]	74.3	82.5	68.7	85.4	60.4	80.3
2018 AD-MIL [17]	71.1	79.2	66.9	84.0	58.0	78.9
2018 Tanaka *et al*. [47]	72.2*	81.5	67.4	84.7	59.5	80.0
2018 CurriculumNet [13]	79.8	84.5	70.7	88.6	62.7	83.4
2019 SL [52]	71.0*	80.9	66.2	82.3	58.7	78.8
2019 MLNT [25]	73.5*	82.5	68.3	85.0	60.2	80.1
2019 PENCIL [57]	73.5*	83.1	68.9	85.7	60.8	81.1
SOMNet	**82.1**	**87.5**	**72.2**	**89.5**	**65.0**	**85.1**

Table 3. The accuracies (%) of different methods on Clothing1M, Food101, Webvision, and ImageNet datasets. The results directly copied from corresponding papers are marked with "*". The best results are denoted in boldface.

mal parameters in this setting. However, the best result is only 74.1%, which is much worse than using region proposals. This result demonstrates the necessity of handling background noise.

The self-organizing memory module: Since traditional clustering methods like K-means can realize a similar function to the memory module, we replace our self-organizing memory module with the K-means method and refer to this baseline as SOMNet+K-means (see implementation details in Supplementary). The performance drop in this setting proves the effectiveness of joint optimization in an end-to-end manner with our memory module. Moreover, to demonstrate the effectiveness of using SOM as an extension, we set neighborhood size as 1, which is equivalent to removing SOM. The comparison between SOMNet (w/o SOM) and SOMNet indicates that it is beneficial to use SOM in our memory module.

4.5. Comparison with the State-of-the-Art

We compare our method with the state-of-the-art webly or weakly supervised learning methods in Table 3 on four benchmark datasets: Clothing1M, Food101, WebVision, and ImageNet. The baseline methods include Sukhbaatar *et al*. [44], DRAE [53], Zhuang *et al*. [60], AD-MIL [17], Tanaka *et al*. [47], CurriculumNet [13], SL [52], MLNT [25], and PENCIL [57]. Some methods did not report their results on the above four datasets. Even with evaluation on the above datasets, different methods conduct experiments in different settings (*e.g.*, backbone network, training set), so we re-run their released code in the same setting as our method for fair comparison. For those methods which already report results in exactly the same setting as ours, we directly copy their reported results (marked with "*").

From Table 3, we can observe that our method achieves significant improvement over the backbone ResNet50. The average relative improvement (Top-1) on all four datasets is 9.18%. It also outperforms all the baselines, demonstrating the effectiveness of our method for handling label noise and background noise using the memory module.

5. Conclusion

In this paper, we have proposed a novel method, which can address the label noise and background noise of web data at the same time. Specifically, we have designed a novel memory module to remove noisy images and noisy region proposals under the multi-instance learning framework. Comprehensive experiments on four benchmark datasets have verified the effectiveness of our method.

Acknowledgement

The work is supported by the National Key R&D Program of China (2018AAA0100704) and is partially sponsored by National Natural Science Foundation of China (Grant No.61902247) and Shanghai Sailing Program (19YF1424400).

References

[1] Lukas Bossard, Matthieu Guillaumin, and Luc J. Van Gool. Food-101 - mining discriminative components with random forests. In *ECCV*, 2014. 6

[2] Razvan C. Bunescu and Raymond J. Mooney. Multiple instance learning for sparse positive bags. In *ICML*, 2007. 2

[3] Xinlei Chen and Abhinav Gupta. Webly supervised learning of convolutional networks. In *ICCV*, 2015. 2

[4] Xinlei Chen, Abhinav Shrivastava, and Abhinav Gupta. Neil: Extracting visual knowledge from web data. In *ICCV*, 2013. 2

[5] Xinlei Chen, Abhinav Shrivastava, and Abhinav Gupta. Enriching visual knowledge bases via object discovery and segmentation. In *CVPR*, 2014. 2

[6] Yanbei Chen, Xiatian Zhu, and Shaogang Gong. Semi-supervised deep learning with memory. In *ECCV*, 2018. 2, 3

[7] Dengxin Dai and Luc Van Gool. Ensemble projection for semi-supervised image classification. In *ICCV*, 2013. 2, 3

[8] Dengxin Dai and Luc Van Gool. Unsupervised high-level feature learning by ensemble projection for semi-supervised image classification and image clustering. *arXiv preprint arXiv:1602.00955*, 2016. 2, 3

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 6

[10] Santosh K Divvala, Ali Farhadi, and Carlos Guestrin. Learning everything about anything: Webly-supervised visual concept learning. In *CVPR*, 2014. 2

[11] Ji Feng and Zhi-Hua Zhou. Deep MIML network. In *AAAI*, 2017. 2

[12] Benoît Frénay and Michel Verleysen. Classification in the presence of label noise: A survey. *IEEE Trans. Neural Netw. Learning Syst.*, 25(5):845–869, 2014. 2

[13] Sheng Guo, Weilin Huang, Haozhi Zhang, Chenfan Zhuang, Dengke Dong, Matthew R. Scott, and Dinglong Huang. Curriculumnet: Weakly supervised learning from large-scale web images. In *ECCV*, 2018. 1, 2, 6, 8

[14] Jiangfan Han, Ping Luo, and Xiaogang Wang. Deep self-learning from noisy labels. In *ICCV*, pages 5138–5147, 2019. 2

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 6

[16] Jinchi Huang, Lie Qu, Rongfei Jia, and Binqiang Zhao. O2u-net: A simple noisy label detection approach for deep neural networks. In *ICCV*, pages 3326–3334, 2019. 1

[17] Maximilian Ilse, Jakub Tomczak, and Max Welling. Attention-based deep multiple instance learning. In *ICML*, pages 2132–2141, 2018. 2, 8

[18] Weston Jason, Sumit Chopra, and Antoine Bordes. Memory networks. *CoRR*, abs/1410.3916, 2014. 2

[19] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. Mentornet: Regularizing very deep neural networks on corrupted labels. *CoRR*, abs/1712.05055, 2017. 1, 2

[20] Lukasz Kaiser, Ofir Nachum, Aurko Roy, and Samy Bengio. Learning to remember rare events. *CoRR*, abs/1703.03129, 2017. 2

[21] Youngdong Kim, Junho Yim, Juseung Yun, and Junmo Kim. Nlnl: Negative learning for noisy labels. In *ICCV*, pages 101–110, 2019. 2

[22] Oren Z. Kraus, Lei Jimmy Ba, and Brendan J. Frey. Classifying and segmenting microscopy images with deep multiple instance learning. *Bioinformatics*, 32(12):52–59, 2016. 2

[23] Jonathan Krause, Benjamin Sapp, Andrew Howard, Howard Zhou, Alexander Toshev, Tom Duerig, James Philbin, and Li Fei-Fei. The unreasonable effectiveness of noisy data for fine-grained recognition. In *ECCV*, 2016. 1, 2

[24] Kuang-Huei Lee, Xiaodong He, Lei Zhang, and Linjun Yang. Cleannet: Transfer learning for scalable image classifier training with label noise. In *CVPR*, 2017. 1, 2, 6

[25] Junnan Li, Yongkang Wong, Qi Zhao, and Mohan S. Kankanhalli. Learning to learn from noisy labeled data. In *CVPR*, June 2019. 8

[26] Wen Li, Limin Wang, Wei Li, Eirikur Agustsson, and Luc Van Gool. Webvision database: Visual learning and understanding from web data. *CoRR*, abs/1708.02862, 2017. 6

[27] Yu-Feng Li, James T. Kwok, Ivor W. Tsang, and Zhi-Hua Zhou. A convex method for locating regions of interest with multi-instance learning. In *ECML-PKDD*, 2009. 2

[28] Tongliang Liu and Dacheng Tao. Classification with noisy labels by importance reweighting. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38(3):447–461, 2016. 1, 2

[29] Alexander H. Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. Key-value memory networks for directly reading documents. In *EMNLP*, 2016. 2

[30] Ishan Misra, C Lawrence Zitnick, Margaret Mitchell, and Ross Girshick. Seeing through the human reporting bias: Visual classifiers from noisy human-centric labels. In *CVPR*, 2016. 2

[31] Li Niu, Wen Li, and Dong Xu. Visual recognition by learning from web data: A weakly supervised domain generalization approach. In *CVPR*, 2015. 1

[32] Li Niu, Wen Li, and Dong Xu. Exploiting privileged information from web data for action and event recognition. *International Journal of Computer Vision*, 118(2):130–150, 2016. 1

[33] Li Niu, Qingtao Tang, Ashok Veeraraghavan, and Ashutosh Sabharwal. Learning from noisy web data with category-level supervision. In *CVPR*, 2018. 1, 2

[34] Li Niu, Ashok Veeraraghavan, and Ashutosh Sabharwal. Webly supervised learning meets zero-shot learning: A hybrid approach for fine-grained classification. In *CVPR*. 1

[35] Nikolaos Pappas and Andrei Popescu-Belis. Explaining the stars: Weighted multiple-instance learning for aspect-based sentiment analysis. In *EMNLP*, 2014. 2

[36] Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. Making deep neural networks robust to label noise: A loss correction approach. In *CVPR*, 2017. 1, 2

[37] Pedro H. O. Pinheiro and Ronan Collobert. From image-level to pixel-level labeling with convolutional networks. In *CVPR*, 2015. 2

[38] Scott E. Reed, Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan, and Andrew Rabinovich. Training deep neural networks on noisy labels with bootstrapping. *CoRR*, abs/1412.6596, 2014. 2

[39] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In *ICML*, 2018. 2

[40] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. In *NIPS*, 2015. 3

[41] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy P. Lillicrap. Meta-learning with memory-augmented neural networks. In *ICML*, 2016. 2

[42] Saurabh Singh, Abhinav Gupta, and Alexei A Efros. Unsupervised discovery of mid-level discriminative patches. In *ECCV*. Springer, 2012. 2, 3

[43] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *NIPS*, 2017. 2, 3

[44] S Sukhbaatar, J Bruna, and M Paluri. Training convolutional networks with noisy labels. *CoRR*, abs/1406.2080, 2014. 2, 8

[45] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. In *NIPS*, 2015. 2

[46] Xiaoxiao Sun, Liang Zheng, Yu-Kun Lai, and Jufeng Yang. Learning from web data: the benefit of unsupervised object localization. *CoRR*, abs/1812.09232, 2018. 1, 2

[47] Daiki Tanaka, Daiki Ikami, Toshihiko Yamasaki, and Kiyoharu Aizawa. Joint optimization framework for learning with noisy labels. In *CVPR*, pages 5552–5560, 2018. 2, 8

[48] Kohonen Teuvo. The self-organizing map. *Proceedings of the IEEE*, 1464-1480, 1990. 2, 4

[49] Arash Vahdat. Toward robustness against label noise in training deep discriminative neural networks. In *NIPS*, pages 5596–5605, 2017. 2

[50] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017. 2

[51] Fang Wan, Chang Liu, Wei Ke, Xiangyang Ji, Jianbin Jiao, and Qixiang Ye. C-mil: Continuation multiple instance learning for weakly supervised object detection. In *CVPR*, 2019. 2

[52] Yisen Wang, Xingjun Ma, Zaiyi Chen, Yuan Luo, Jinfeng Yi, and James Bailey. Symmetric cross entropy for robust learning with noisy labels. In *ICCV*, pages 322–330, 2019. 1, 8

[53] Yan Xia, Xudong Cao, Fang Wen, Gang Hua, and Jian Sun. Learning discriminative reconstructions for unsupervised outlier removal. In *ICCV*, 2015. 1, 2, 8

[54] Tong Xiao, Tian Xia, Yi Yang, Chang Huang, and Xiaogang Wang. Learning from massive noisy labeled data for image classification. In *CVPR*, 2015. 6

[55] Zhongwen Xu, Linchao Zhu, and Yi Yang. Few-shot object recognition from machine-labeled web images. In *CVPR*, 2017. 2

[56] Bengio Y, Louradour J, and Collobert R. Curriculum learning. *ACM*, 2009. 5

[57] Kun Yi and Jianxin Wu. Probabilistic end-to-end noise correction for learning with noisy labels. In *CVPR*, 2019. 8

[58] Linchao Zhu and Yi Yang. Compound memory networks for few-shot video classification. In *ECCV*, 2018. 2, 4

[59] Wentao Zhu, Qi Lou, Yeeleng Scott Vang, and Xiaohui Xie. Deep multi-instance networks with sparse label assignment for whole mammogram classification. In *MICCAI*, 2017. 2

[60] Bohan Zhuang, Lingqiao Liu, Yao Li, Chunhua Shen, and Ian D. Reid. Attend in groups: A weakly-supervised deep learning framework for learning from web data. In *CVPR*, 2017. 1, 2, 8

[61] C. Lawrence Zitnick and Piotr Dollár. Edge boxes: Locating object proposals from edges. In *ECCV*, 2014. 1, 3