Lightweight Filtering of Noisy Web Data: Augmenting Fine-grained Datasets with Selected Internet Images

Julia Böhlke[1], Dimitri Korsch[1], Paul Bodesheim[1], and Joachim Denzler[1,2,3]
[1]*Computer Vision Group, Friedrich Schiller University Jena, Ernst-Abbe-Platz 2, Jena, Germany*
[2]*Deutsches Zentrum für Luft- und Raumfahrt e.V. (DLR), Institute for Data Science (IDW), Mälzerstraße 3, Jena, Germany*
[3]*Michael Stifel Center Jena for Data-Driven and Simulation Science, Ernst-Abbe-Platz 2, Jena, Germany*

Keywords: Noisy Web Data, Label Noise Filtering, Fine-grained Categorization, Duplicate Detection.

Abstract: Despite the availability of huge annotated benchmark datasets and the potential of transfer learning, i.e., fine-tuning a pre-trained neural network to a specific task, deep learning struggles in applications where no labeled datasets of sufficient size exist. This issue affects fine-grained recognition tasks the most since correct image data annotations are expensive and require expert knowledge. Nevertheless, the Internet offers a lot of weakly annotated images. In contrast to existing work, we suggest a new lightweight filtering strategy to exploit this source of information without supervision and minimal additional costs. Our main contributions are specific filter operations that allow the selection of downloaded images to augment a training set. We filter test duplicates to avoid a biased evaluation of the methods, and two types of label noise: cross-domain noise, i.e., images outside any class in the dataset, and cross-class noise, a form of label-swapping noise. We evaluate our suggested filter operations in a controlled environment and demonstrate our methods' effectiveness with two small annotated seed datasets for moth species recognition. While noisy web images consistently improve classification accuracies, our filtering methods retain a fraction of the data such that high accuracies are achieved with a significantly smaller training dataset.

1 INTRODUCTION

The field of computer vision utilizes huge, publicly available datasets to develop and compare methods. Famous datasets, such as ImageNet (Russakovsky et al., 2015), aim at the classification of objects from our daily life, like pedestrians, desks, cats, and dogs, while datasets such as CUB-200-2011 (Wah et al., 2011) and iNaturalist (Van Horn et al., 2018) pose fine-grained classification problems. The performance in terms of classification accuracy on these benchmark datasets has continuously improved through innovations in deep neural network architectures and transfer learning. Achieving further boosts in classification performance through even larger datasets stands in no relation to the overhead of acquiring a larger dataset. However, application-oriented, specific classification problems often lack sufficient training data and would benefit from additional images.

One prominent example of this situation is found in the field of biodiversity research. The classification of moth species, a fine-grained recognition problem, is necessary to monitor the population changes. Unfortunately, constructing a robust classifier with a convolutional neural network (CNN) cannot be done out of the box for this use case. The number of training samples is crucial for good recognition performances due to the high number of parameters in a CNN model. Since only experts can reliably distinguish very similar looking species, training datasets for such a highly specific task often consist of very few images. Generally, the generation of high-quality data in the field of fine-grained classification is an expensive and challenging task.

However, in many cases, there is a rich source of information available in the form of weakly labeled images on the Internet, accessible through image search engines. In such a situation, images from the Internet might enrich a small seed training dataset collected by experts. As demonstrated by (Krause et al., 2016), noisy data from the Internet can significantly improve the classification performance for fine-grained datasets. As the authors mention in their work, one should clean the noisy data before using it to train a classification model. They use addi-

Böhlke, J., Korsch, D., Bodesheim, P. and Denzler, J.
Lightweight Filtering of Noisy Web Data: Augmenting Fine-grained Datasets with Selected Internet Images.
DOI: 10.5220/0010244704660477
In *Proceedings of the 16th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2021) - Volume 5: VISAPP*, pages 466-477
ISBN: 978-989-758-488-6

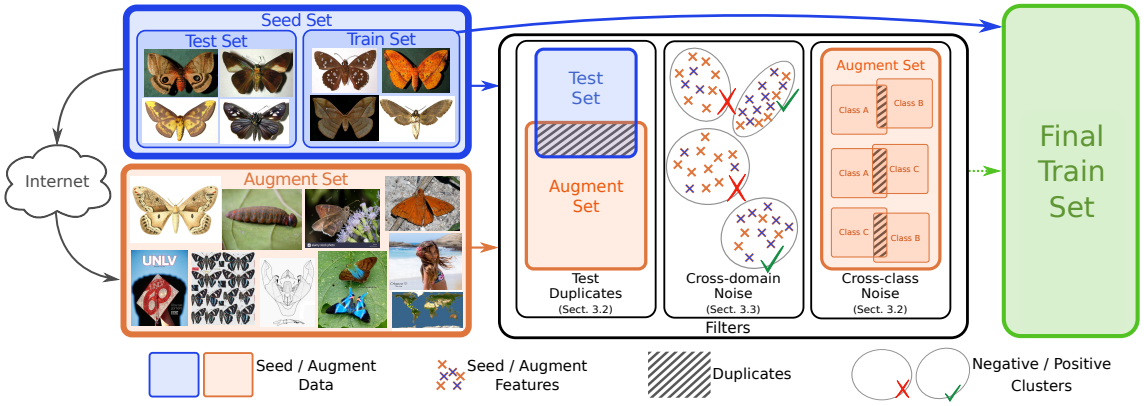


Figure 1: Overview of our approach. The results returned by image search engines when searching for the class names of the seed dataset are downloaded and constitute the augmentation dataset. Subsequently, filtering methods take the seed dataset as source of information to decide which augment images are added to the final training dataset. Our duplicate detection method is used to handle test duplicates and cross-class noise. A clustering-based approach is used to filter out cross-domain noise.

tional models and sophisticated training procedures to achieve their results.

In contrast, we suggest a new strategy to exploit images from the Internet without supervision and minimal additional costs. We call this *lightweight filtering*, and the process is summarized in Figure 1. Given a small, precisely annotated seed dataset, we use the species names as queries for an image search engine. Then, we use the search results as an augmentation dataset, also referred to as augment set, to extend the seed dataset. Likely, some of the downloaded images are already part of the test set, especially if the seed dataset is publicly available. These images would distort the evaluation results and yield a worse generalization behavior as the model is promoted to memorize the test duplicates in the training set. Therefore, the downloaded images need to be cleaned from test duplicates to ensure a fair model evaluation. Unlike (Krause et al., 2016), who use a sophisticated deep ranking model (Wang et al., 2014), we filter test duplicates without training additional models. In Sect. 3.2, we present an unsupervised ranking-based duplicate detection method. Besides *exact* duplicates, we also identify *near-duplicates*, i.e., image pairs that originated from the same camera shot but with slight transformation, contrast, or resolution changes.

Furthermore, the image acquisition from the Internet introduces label noise in the augmentation dataset. We differentiate between two types of label noise: cross-domain noise and cross-class noise. A domain of a dataset is a broader category to which the classes of a dataset belong. In the case of the CUB-200-2011 dataset (Wah et al., 2011), the domain is *birds*. As a result, all images depicting not a bird are cross-domain noise. Images within the dataset domain but with a wrong class label are cross-class noise, i.e., an American Crow image downloaded for the Common Raven class. Note that cross-class noise is not restricted to the original classes from the seed dataset but by the domain it defines. For CUB-200-2011, an image of any bird downloaded in the wrong class would be considered as cross-class noise.

We detect cross-class noise with a ranking-based duplicate detection method described in Sect. 3.2. In Sect. 3.3, we propose a novel clustering-based approach to identify out-of-domain images and to reduce cross-domain noise. To the best of our knowledge, cross-domain noise in fine-grained datasets has not been addressed so far. Hence, our contribution is a set of methods for handling the entire range of problems that arise when using weakly annotated data from the Internet for dataset augmentation. Finally, we evaluate our experiments on various datasets (Sect. 4). Using a real-world biodiversity task, our experiments show that the proposed filtering methods retrain the classification performance even though they reduce the amount of training data by about 58%.

2 RELATED WORK

In this section, we review four topics related to our work: fine-grained classification (Sect. 2.1), handling different types of label noise (Sect. 2.2), data set augmentation using images from the Internet (Sect. 2.3), and identifying duplicate images (Sect. 2.4).

2.1 Fine-grained Classification

Fine-grained classification refers to distinguishing classes with small inter-class variance, i.e., classes

differ only in few distinctive features. Biodiversity research is one of the areas applying fine-grained classifiers to distinguish different animal species. There are two main approaches for tackling this task: part- or attention-based methods using global features extracted from the whole image.

Part-based approaches employ the idea of extracting relevant local regions of an image that are often interpreted as parts of the object and perform the classification based on features from these regions (Ge et al., 2019; Korsch et al., 2019; Zheng et al., 2017).

On the other hand, global approaches aim to classify instances without extracting any parts but instead use images entirely. These methods use either a sophisticated pre-training (Cui et al., 2018) or a specific feature pooling technique (Lin et al., 2015; Simon et al., 2017). We stick to the global approach to avoid the overhead introduced by part-based approaches and thus keep our model comparably lightweight in terms of required computational costs.

2.2 Label Noise Handling

Existing work dealing with label noise can be divided into methods robust against label noise and cleansing methods.

Robust methods such as (Cortes and Vapnik, 1995; Xiao et al., 2015; Zhuang et al., 2017) aim to learn directly from noisy data using a noise-tolerant learning algorithm. Often these approaches rely on complex models, relatively low levels of label noise, or require some prior knowledge of either the noise distribution or the noise-inducing process. (Rolnick et al., 2017) investigated robust CNN architectures and hyperparameters. They showed empirically that deep neural networks are surprisingly robust to high levels of label noise when the total number of clean labels is high. However, this rarely holds when considering only small seed datasets or in the case of many falsely labeled images. While robust methods might be a cheap option when dealing with noise, reducing label noise would likely lead to better performances.

With cleansing methods, the data is pre-processed and mislabeled data is removed or relabeled. The challenge in identifying noisy instances is the separation of correctly labeled instances that are hard to classify (because they deviate from the norm) from actually mislabeled ones. These hard-to-classify instances are valuable in a training set because they capture a wider diversity of appearance for the corresponding class.

An overview of label noise handling methods is given by (Frénay and Verleysen, 2013). Further approaches include ensemble methods (Garcia et al., 2016), semi-supervised verification of support vectors by (Fefilatyev et al., 2012), probabilistic approaches and anomaly detection techniques (Eskin, 2000; Akcay et al., 2018; Zhang and Tan, 2019), our outlier detection methods, e.g., utilizing class-wise autoencoders (Zhang and Tan, 2019).

In contrast, an unsupervised approach was proposed by (Nicholson et al., 2015). They used *k-Means* clustering, an algorithm entirely independent of labels and, therefore, of label noise, to cluster the weakly labeled data and identify noise using cluster statistics. We use a simplified version of this clustering-based idea described in Sect 3.3.

2.3 Images from the Internet

Utilizing the vast visual information available on the Internet requires dealing with label noise. Several approaches (Li and Fei-Fei, 2010; Zhang et al., 2020; Xu et al., 2015; Chen et al., 2013; Berg and Forsyth, 2006; Schroff et al., 2010) are built on the idea of incrementally constructing a training set by choosing useful images from weakly labeled data with a model trained on a small, precisely labeled dataset. These strategies often rely on a representative, diverse seed dataset and involve high computational costs when retraining a model with added data. In contrast, we aim for computationally lightweight solutions for filtering label noise.

(Krause et al., 2016) showed the potential of noisy data from the Internet for several popular fine-grained recognition datasets. They gathered the results of several image search engines when using category names as search query keywords. Thus, they added more fine-grained categories of the corresponding domain for each dataset and downloaded images for these new categories in the same way. Furthermore, they identified test duplicates with a method by (Wang et al., 2014) described in Sect. 2.4. They argued that cross-domain noise was less detrimental and only handled cross-class noise by removing ambiguous images downloaded multiple times for different classes. We expand on this approach. In summary, our work differs from the approach of (Krause et al., 2016) in three crucial aspects. First, in the case of the cross-class noise, we additionally filter near-duplicates while (Krause et al., 2016) only consider exact duplicates. Second, our approach with lightweight filtering methods do not require additional and computationally expensive pre-training of a neural network model with the downloaded images. Third, we additionally propose a method for handling cross-domain noise, which has been ignored by (Krause et al., 2016).

As an alternative to dataset augmentation, also few-shot learning approaches could be considered. Recent work in this area focuses on metric learning (Sung et al., 2018; Snell et al., 2017) or complex LSTM models (Ravi and Larochelle, 2016). Although impressive accuracies can be achieved on small datasets using these approaches, we follow the strategy of exploiting additional image data that is available via the Internet.

2.4 Duplicate Detection

Duplicate detection is essential to identify test duplicates in the augment set and handle cross-class noise. The task is not trivial if, besides exact duplicates, also near-duplicates need to be detected. Furthermore, efficiency plays a crucial role since the number of comparisons grows with the number of samples in a dataset.

Traditionally, efficient duplicate detection was done by comparing hand-crafted features extracted from the images (Ke et al., 2004; Luo and Nascimento, 2003; Wang et al., 2006). Feature representations of images learned by a CNN offer an alternative to the hand-crafted features, as shown by (Barz and Denzler, 2020). They explored the presence of duplicates between training and test set in the well-known and widely-used CIFAR-10 and CIFAR-100 datasets (Krizhevsky, 2009). L^2-normalized feature representations were extracted for all images from a CNN pre-trained on the respective training set. The test images were then ranked based on the nearest neighbor in the training set concerning the Euclidean distance in this feature space. Because simply thresholding this distance resulted in a high false-positive rate, they proposed a tool for manually identifying duplicates, which utilizes the ranking to reduce the number of comparisons. With this approach, they found that 3.25 % of the CIFAR-10 test images and 10 % of the CIFAR-100 test images had a duplicate either in the training set or within the test set.

Another way of duplicate identification is metric learning. The most prominent work is the deep ranking method by (Wang et al., 2014), which was also used by (Krause et al., 2016) to identify test duplicates. The aim is to learn an embedding of the images in a lower-dimensional space, where similar images are located close to each other, and dissimilar ones are far apart. Since metric learning involves training a CNN and an appropriate dataset, we employ the lightweight variant of the idea proposed by (Barz and Denzler, 2020) and utilize a CNN pre-trained on ImageNet for feature extraction.

3 METHODS

In the following section, we formally define generic noise filters. We then introduce our filtering methods in detail: we use a cluster-based approach for cross-domain noise filtering and duplicate detection to detect test duplicates and cross-class noise. Finally, we describe a method for a dataset generation that can be used to evaluate any cross-domain filtering method.

3.1 Problem Definition

In this work, we consider two different types of datasets. The first one is a labeled *seed dataset* $T_{seed} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ with images $X_{seed} = \{x_1, \ldots, x_N\}$ and class labels $Y_{seed} = \{y_1, \ldots, y_N\}$. Based on the class names associated with the labels, we construct a second dataset called *augment dataset*. As mentioned in Sect. 1, we create this one using an image search engine, i.e., Google Image Search. The resulting dataset $T_{aug} = \{(\hat{x}_1, \hat{y}_1), \ldots, (\hat{x}_M, \hat{y}_M)\}$ consists of web images $X_{aug} = \{\hat{x}_i, \ldots, \hat{x}_M\}$ and noisy labels $Y_{aug} = \{\hat{y}_1, \ldots, \hat{y}_M\}$.

In the following sections, we will describe several functions belonging to a set of binary filter functions $F = \{f_1, \ldots, f_L\}$, which decide for each image in X_{aug} whether it is added to the final training set depending on the desired filtering behavior. An image in X_{aug} is only added to the final training dataset if all of the functions in F select the image:

$$T_{final} = T_{seed} \cup \{(\hat{x}_i, \hat{y}_i) \mid \hat{x}_i \in X_{aug} \wedge \forall f \in F : f(\hat{x}_i) = 1\}. \quad (1)$$

The images in T_{final} are then used to train or fine-tune a classification model.

We create the final training set based on the decision functions F, which may vary for different experimental setups. For example, we might test how the classification model performs when only test duplicate filtering is applied.

3.2 Filters for Test Duplicates and Cross-class Noise

As mentioned before, we utilize duplicate detection for both identifying test duplicates and filtering cross-class noise. Test duplicates, i.e., images in the augment set that have a duplicate in the test set, are filtered out class-wise. Training on images in the augment set that have a duplicate with the same class label in the test set would lead to an unfair overestimation of the recognition accuracies and distort the evaluation of the classification. However, training with duplicates in different classes is less critical, since this

does not affect the results positively but instead leads to an underestimation of accuracies. Hence, we do not consider test duplicates in different classes.

Dealing with cross-class label noise is problematic because, in a fine-grained domain, only experts can identify wrongly labeled instances. (Krause et al., 2016) handled this problem by filtering out all images that had an exact duplicate in a different class of the augment set. Strictly speaking, this does not filter cross-class noise but instead mitigates the issue by removing ambiguous images. We expand on this idea and further exploit its potential by filtering images that also have *near-duplicates* in a different class. Thus, we apply near-duplicate detection for cross-class noise filtering.

We identify duplicates with two similarity measures. First, we utilize the structural similarity index (*SSIM*) proposed by (Wang et al., 2004), a pixel-based image comparison method that takes luminance, contrast, and structural distortions via a sliding-window approach into account. Second, we use the cosine similarity computed by dot product (*Dot*) of L^2-normalized feature representations extracted from a pre-trained CNN.

For each image $\hat{x}_i$ in the augment set, we compute four similarity scores either to images in a subset of T_{test} for test duplicate filtering, or to images in a subset of T_{aug} for cross-class filtering. For the sake of simplicity, we call this subset $T^{\hat{x}_i}$ for both cases. In the case of *test duplicate filtering*, we are only interested in class-wise duplicates such that $T^{\hat{x}_i}$ consists of all images in T_{test} from the same class $\hat{y}_i$ as $\hat{x}_i$. For filtering cross-class noise, $T^{\hat{x}_i}$ consists of all images in T_{aug} that have a different class label $\hat{y} \neq \hat{y}_i$ than $\hat{x}_i$. The first two scores are calculated as follows:

$$maxDot(\hat{x}_i) = \max_{x_j \in T^{\hat{x}_i}} Dot(\hat{x}_i, x_j) \tag{2}$$

$$maxSSIM(\hat{x}_i) = \max_{x_j \in T^{\hat{x}_i}} SSIM(\hat{x}_i, x_j) \tag{3}$$

where $Dot(\cdot,\cdot)$ computes the dot product of L^2-normalized features of the input images and $SSIM(\cdot,\cdot)$ computes the structural similarity index (Wang et al., 2004) of two images. For the sake of completeness, we also compute the other score for each pair of images with maximum value for one score. This leads to the third and fourth score:

$$SSIM_{maxDot}(\hat{x}_i) = SSIM(\hat{x}_i, \operatorname{argmax}_{x_j \in T^{\hat{x}_i}} Dot(\hat{x}_i, x_j)) \tag{4}$$

$$Dot_{maxSSIM}(\hat{x}_i) = Dot(\hat{x}_i, \operatorname{argmax}_{x_j \in T^{\hat{x}_i}} SSIM(\hat{x}_i, x_j)) \tag{5}$$

An image x_j in $maxDot(\hat{x}_i)$ and $maxSSIM(\hat{x}_i)$ might denote two different images for the same $\hat{x}_i$, hence, we extract more information than if we only computed the maximum scores.

Finally, each score defines an ordered list of the images in the augment dataset by sorting them in descending order concerning the obtained values. All four lists are used to estimate duplicates from the augment dataset as follows. Starting with $D = 1$ and later gradually incrementing D by 1, we consider an augment image a duplicate if present in the first D images of all four lists. We repeat this incrementation until the required *portion* of images is identified as duplicates and removed from the augment set. This *portion* is a hyperparameter that also depends on the type of data that needs to be filtered. It can be specified by either using apriori knowledge about the level of duplicates or assuming a fraction of duplicates that is reasonable to expect for a particular application. In case of test duplicate filtering (TD), the filter function f_{TD} selects images from the augment set T_{aug} for the final training set that do not fall into the *portion* of T_{aug} identified as having a duplicate in the test set.

For *cross-class noise filtering*, the subset $T^{\hat{x}_i}$ for each image $\hat{x}_i$ is considerably larger than for test duplicate filtering. Therefore, we utilized two tricks to speed up computations. First, we approximate the $maxSSIM(\hat{x}_i)$ score for each image $\hat{x}_i$ by searching only among the ten images in the set $T^{\hat{x}_i}$ that have the highest dot product. Second, we identify exact duplicates by comparing MD5 hashes[1] of the images and skip the computations of the scores for those images.

Furthermore, for cross-class noise filtering, the parameter *portion* is set such that it depends on how many exact duplicates are detected using MD5 hash values. For this purpose, we introduce another parameter called *relative_portion*. Formally, if M_{MD5} is the total number of images in the augment set that have an exact duplicate, then the final *portion* of the augment set removed by cross-class noise filtering is $(1 + relative\_portion) \cdot M_{MD5}$. With this method, we specify the number of images that are filtered additionally to the exact duplicates as a percentage (*relative_portion*) of total exact duplicates and ensure that exact duplicates are always filtered. Our cross-class noise filtering function f_{CC} selects images from the augment set T_{aug} to be added to the final training set if they neither have an exact duplicate nor fall into the set of images defined by *relative_portion* for having a near-duplicate in T_{aug}.

3.3 Cross-domain Noise Filter

This filter aims to identify images that do not depict the domain of the seed dataset T_{seed}. To achieve this,

[1] https://tools.ietf.org/html/rfc1321

we use CNN features of the images and compare them using a clustering approach. Intuitively, features of those images in T_{aug} that belong to the domain of T_{seed} have smaller distances to features of images from the seed dataset. After jointly clustering images of the augment and the seed training set, the clusters that contain a certain amount of the seed dataset indicate clusters of images belonging to the domain. Thus, images from T_{aug} in these clusters can then be identified as images within the domain, while clusters with few seed training images most likely contain the out-of-domain images from T_{aug} called cross-domain noise.

We call a cluster a *strong positive* cluster if it contains more than $\frac{N}{k}$ samples of the seed data with N being the number of training samples in the seed dataset and k being the number of clusters. When clustering with a considerable value for k, the seed images are likely spread across more clusters. The adaptive threshold that depends on k accounts for this and ensures that strong positive clusters are identified.

If the seed dataset is visually homogeneous, all seed images are likely assigned to only a small fraction of the clusters. Even though the augment dataset contains images of the same domain, these images would not be assigned to strong positive clusters, but to nearby ones. We mitigate this effect by also identifying *weak positive* clusters as those with small Euclidean distance to strong positive clusters. More precisely, a cluster is *weak positive* if its center is closer to one of the strong positive cluster centers than the average pairwise distance between all cluster centers.

We end up with two functions for cross-domain noise filtering: $f_{CD_{strong}}$ selects images for the final training dataset from T_{aug} that are assigned to a strong positive cluster and $f_{CD_{weak}}$ retains images assigned to either a strong positive or a weak positive cluster.

3.4 Generating Datasets with Controlled Cross-domain Noise

To evaluate cross-domain noise filtering, we propose a method for generating datasets that contain cross-domain noise with different levels of noise. The aim is to create a controlled setting with a seed dataset and a noisy augment dataset, where a controlled amount of cross-domain noise occurs.

To construct a dataset with a pre-defined data-to-noise ratio, we require two datasets. The first dataset is a fine-grained dataset, which defines the domain of the classification task. A small subset of the training images from this first dataset is used as a seed dataset. The rest of the training images are added to the augment set as positive, non-noisy samples, which should be retained by the cross-domain noise filter.

The negative samples that should be filtered out are gathered from a second dataset. We have chosen ImageNet (Russakovsky et al., 2015) as a coarse-grained dataset for image classification. We aim to select samples that are somehow related to the seed dataset but outside the domain, i.e., caterpillars for moths, as well as unrelated samples that a search engine might return. We rank each class of the second dataset according to its similarity to the entire first dataset. We compute the similarity by following the approach of (Cui et al., 2018) that utilizes the earth mover's distance (EMD). After obtaining a ranking of the classes from the second dataset, we discard the *Top-X* classes (in the case of ImageNet, we discard 100 classes). As a result, we do not consider classes belonging to the first dataset's domain as cross-domain inducing classes. Afterward, we rank the remaining classes according to the similarity to a single class form the seed dataset. Finally, we use these rankings to construct for each class in the seed dataset its own cross-domain noise. Thus, we sampled images uniformly from the $S_1 = 10$ most related and $S_2 = 10$ least related classes that are determined for each class of the seed dataset. The number of images sampled from these classes depends on the specified data-to-noise ratio that the final augmentation dataset should have.

4 EXPERIMENTS

Our experiments have three main objectives: (i) evaluating the cluster-based cross-domain noise filter, (ii) applying the duplicate detection method for identifying test duplicate and filtering cross-class noise, and (iii) testing the filter methods in a real-world application, where a small fine-grained dataset is augmented using web images that are filtered successively.

4.1 Datasets

We use two seed datasets, namely Costa Rica Moths and European Moths, for our application of moth species recognition. We augment these datasets with images obtained from the Internet, and we call these augmentation datasets Web Costa Rica Moths and Web European Moths, respectively. We also use the CIFAR-10 and CIFAR-100 datasets (Krizhevsky, 2009) together with the annotations from the corresponding ciFAIR counterparts (Barz and Denzler, 2020) and extensions of the CUB-200-2011 dataset (Wah et al., 2011) called Noisy CUB-200-2011 for testing individual filtering methods. All these datasets are described in the following.

Costa Rica Moths: A small fine-grained seed dataset, initially introduced by (Rodner et al., 2015), depicts pinned moths from 331 species found in a conservation area in northwest Costa Rica. The wings of the moths are spread out artificially to show the features of the hind wings. With only 990 training images (and 1,320 for testing), this is a rather small dataset. The moths were photographed from two perspectives, top and bottom, making the dataset more challenging as a species' appearance differs remarkably between both sides. This intensifies the problem of few training images per class.

Web Costa Rica Moths: We augment the Costa Rica Moths dataset with images downloaded with the Google Image Search engine. We used species names as keywords and saved up to 30 images per class. After initial cleaning of unsupported image types, this dataset consists of 10,124 (990 seed and 9,134 augmentation) images. Furthermore, to evaluate our proposed duplicate filter method, we checked manually for duplicates in the test set of the Costa Rica Moths dataset. To identify the duplicates, we used the tool proposed by (Barz and Denzler, 2020). We found that 32 downloaded images have either an exact or a near-duplicate in the test set.

European Moths: This is another small fine-grained seed dataset of 100 moth species found in Europe, which were photographed using a light trap. Each class has three images in the training set and eight test images. This dataset is not publicly available.

Web European Moths: We augment the European Moths dataset with images obtained in the same way as for Web Costa Rica Moths. After removing unsupported image types, each class had between 90 and 99 downloaded images left, leading to a dataset of 9,691 (300 seed and 9,391 augmentation) images in total. Since the original dataset of European Moths is not publicly available, the Web European Moths dataset does not contain test duplicates.

CIFAR / ciFAIR: In their work, (Barz and Denzler, 2020) probed the widely known CIFAR-10 and CIFAR-100 datasets for duplicates. We use their findings to evaluate our duplicate detection method presented in Sect. 4.3

Noisy CUB-200-2011: We have generated noisy datasets with a varying amount of cross-domain noise following our descriptions in Sect. 3.4. We have chosen the CUB-200-2011 dataset (Wah et al., 2011) for the first dataset, which defines the domain and originally consists of roughly 12,000 images from 200 bird species. The dataset is balanced, and the images are split evenly in a training and test set, resulting in about 30 training and 30 test images per class. We have picked 5 images per class from the training dataset to produce the seed dataset. The rest of the training data from the original dataset was used in the augmentation set, defining the positive, non-noisy images. The induced cross-domain noise comes from classes of ImageNet (Russakovsky et al., 2015) as described in Sect. 3.4, where the Top-100 classes are discarded in the first ranking. The $S_1 = 10$ most related and $S_2 = 10$ least related classes according to the second rankings are used for sampling noisy images uniformly. The number of sampled noisy images depends on the desired data-to-noise ratio in the augmentation set.

4.2 Evaluating Cross-Domain Noise Filtering

To evaluate the cross-domain noise filter, we utilized the *Noisy CUB-200-2011* dataset. We checked how many positive samples were retained and how many negative samples were rejected in this controlled environment. Therefore, we created four datasets with different data-to-noise ratios: 2:1, 1:1, 1:2, and 1:10. Afterward, we applied our proposed filtering method (Sect. 3.3) and observed the retention and rejection rates. As can be seen in Figure 2, our filtering method is robust across different noise levels. Furthermore, the retention and rejection rates stabilize with an increasing number of clusters.

Additionally, we perform classification experiments on the *Noisy CUB-200-2011* dataset. As a first baseline, we train on the seed dataset consisting of 5 images per class randomly chosen among the positive samples. For a second baseline, we train on the merged dataset that is the class-wise union of the seed dataset and the unfiltered augment dataset. Figure 3 shows the classification accuracies for different data-to-noise ratios in the augment sets. The augmentation of the seed dataset consistently improves the classification from roughly 50 % to over 75 %. As expected, a more significant amount of out-of-domain images reduces the performance significantly. When applying our proposed cross-domain filtering to the augment sets, the classification performance remains stable, even at high data-to-noise ratios.

4.3 Evaluating Our Duplicate Detection Method

We evaluated the general approach of our ranking-based duplicate detection method on three datasets: *CIFAR-10*, *CIFAR-100*, and *Web Costa Rica Moths*. Therefore, we used the annotations provided by (Barz and Denzler, 2020) for duplicates of the CIFAR datasets. In the case of the Web Costa Rica Moths, we used our manual annotations.

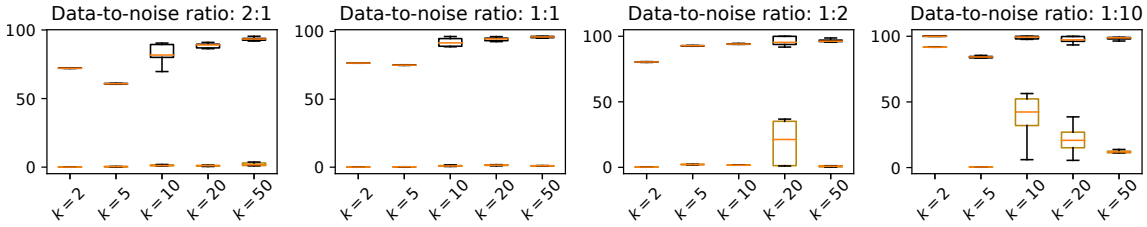


Figure 2: Percentage of data and noise retained after applying our proposed cross-domain filtering method with a varying number of clusters k. The top box plot at each setup (which has shrunk to a lines due to low variance in some cases) indicated the percent of positive data retained while the bottom box plot shows the percent of noise retained. At $k = 50$ our method performs consistently well across all data-to-noise ratios.

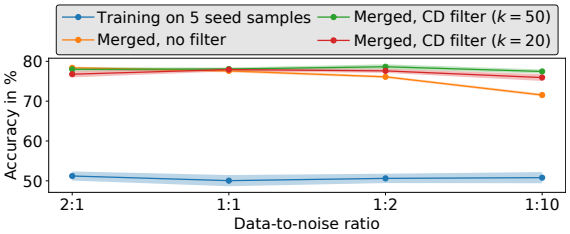


Figure 3: Performance development with different data-to-noise ratios. Here, baseline refers to training with 5 random positive samples from the CUB200-2011 training images. The merged dataset is this seed dataset joined with the different augment datasets of different data-to-noise ratios. Our filter method was tested with two different values for k. Then, the filtered datasets together with the seed dataset were used to train a classification model.

Our baseline methods for these experiments are the random baseline and a method closely related to the approach used by (Barz and Denzler, 2020). They compute the Euclidean distance of L^2-normalized CNN features and use this distance metric in a graphical interface to manually identify the duplicates. We transform the distance metric to a fully automatic single-value ranking very similar to the one we proposed in Sect. 3.2. More specifically, the *maxDot* values from Eq. 3 are inversely related to the metric proposed by (Barz and Denzler, 2020) and we use them for this single-value ranking baseline. As Figure 4 shows, our duplicate detection approach consistently outperforms the single-value ranking, since it additionally takes the SSIM values and several rankings into account.

Furthermore, we performed the experiment with CNN features of different architectures (ResNet50, ResNet101 (He et al., 2016), InceptionV3 (Szegedy et al., 2016), and Xception (Chollet, 2017)) pre-trained on the ImageNet dataset. The improvement of our approach can be seen across all these architectures. Note that there are far fewer test duplicates in the Web Costa Rica Moths dataset, which causes the different shapes of the precision-recall curves compared to the CIFAR datasets.

Table 1: Recall of the duplicate filter on the Costa Rica Moths dataset. We found 32 test duplicates among 9,134 web images. In each cell we compare the baseline with our proposed approach (*baseline* / *our approach*).

CNN	Portion 0.02	0.05	0.1
InceptionV3	0.84 / 0.97	0.88 / 1.00	1.00 / 1.00
ResNet50	0.94 / 0.97	1.00 / 1.00	1.00 / 1.00
ResNet101	0.97 / 1.00	1.00 / 1.00	1.00 / 1.00
Xception	0.81 / 0.94	0.91 / 1.00	0.97 / 1.00

Table 1 provides recall rates for the Costa Rica Moths dataset at different values for the parameter *portion*, which specifies the percent of images removed at the top of the rankings. Here, we compare the single-value baseline with our approach. In test duplicate filtering, recall, as the percentage of actual test duplicates detected, is crucial when evaluating different *portion* values. Since the aim is to maintain the evaluation's validity when using the web images for training, we want to achieve high recall and give less priority to the precision. With *portion* set to 0.02, we found a reasonable balance between high recall and good precision. Therefore, this value is used for the experiments presented in the next section, where the two moth datasets are augmented, and the downloaded images are filtered. In addition, Tables 2 and 3 contain recall rates for CIFAR-10 and CIFAR-100.

4.4 Training with Filtered Web Images

Finally, we evaluate the impact of all our methods on the classification performance for the European and Costa Rica Moths. We use the InceptionV3 (Szegedy et al., 2016) architecture pre-trained on ImageNet. Each setup is executed five times with 30 epochs for each run, and we report the mean and the standard deviation of the accuracies.

As baselines, we report the accuracies when training a CNN on the original seed datasets. Additionally, for the Costa Rica Moths, we also report the results of (Rodner et al., 2015) using CNN features and a linear

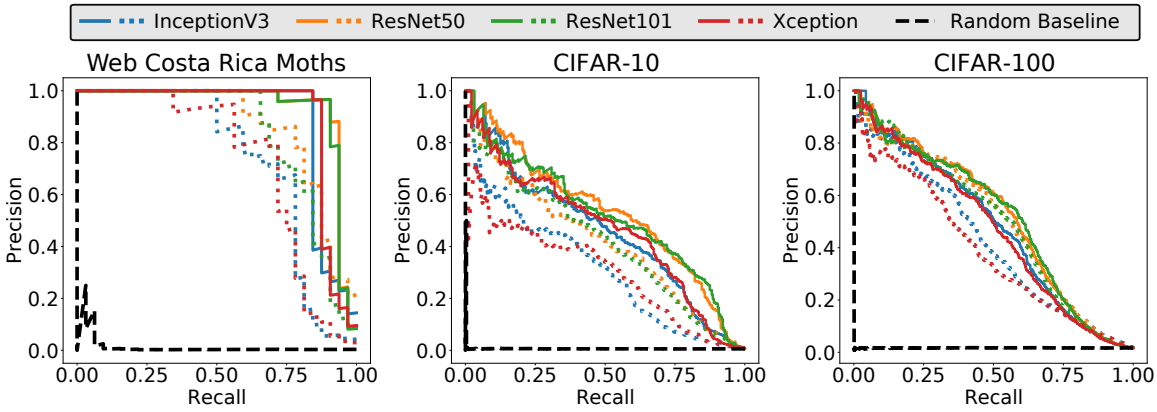


Figure 4: Precision-recall-curves for Web Costa Rica Moths duplicates as well as for CIFAR-10 and CIFAR-100 duplicates by varying the values of the parameter *portion*, which specifies the percentage of images from the augment set that are filtered out, i.e., discarded at the top of the ranking. We evaluate four different CNN architectures and compare our approach (*colored solid lines*) to the random baseline (*black dashed line*) and the single-value ranking (*colored dotted lines*).

Table 2: Recall of the duplicate filter on the CIFAR-10 dataset (286 test duplicates in 50,000 training images). In each cell we compare the baseline with our proposed approach (*baseline* / *our approach*).

CNN	Portion 0.02	0.05	0.1
InceptionV3	0.62 / 0.78	0.74 / 0.85	0.82 / 0.93
ResNet50	0.74 / 0.85	0.85 / 0.92	0.91 / 0.94
ResNet101	0.75 / 0.85	0.85 / 0.91	0.92 / 0.94
Xception	0.66 / 0.78	0.78 / 0.86	0.84 / 0.89

Table 3: Recall of the duplicate filter on the CIFAR-100 dataset (891 test duplicates in 50,000 training images). In each cell we compare the baseline with our proposed approach (*baseline* / *our approach*).

CNN	Portion 0.02	0.05	0.1
InceptionV3	0.48 / 0.55	0.65 / 0.70	0.79 / 0.78
ResNet50	0.58 / 0.60	0.71 / 0.73	0.79 / 0.80
ResNet101	0.57 / 0.59	0.73 / 0.72	0.81 / 0.80
Xception	0.45 / 0.59	0.65 / 0.69	0.77 / 0.78

SVM classifier. The last baseline is the classification accuracy on the merged datasets (the combination of the seed dataset and its corresponding unfiltered augment dataset). In the case of the Costa Rica Moths, this merged dataset includes test duplicates and the corresponding baseline is therefore not directly comparable to other baselines. Hence, we filter the test duplicates (TD) with our method (Sect. 3.2) and the parameter *portion* set to 0.02. The accuracy of the resulting subset represents a more valid baseline. Some example images are shown in Fig. 5.

For cross-class (CC) noise filtering, the final *portion* was set using the parameter *relative_portion* as a percentage of the number of the exact duplicates to be filtered out additionally among the near-duplicates. However, identifying or quantifying cross-class noise requires costly expert knowledge. Therefore, we do not have any reference for how much cross-class noise can be expected when downloading images from the web, which would indicate a suitable value for *relative_portion*. Instead, we tested three values for the parameter *relative_portion* corresponding to three hypotheses: (i) 0.1, assuming only a few near-duplicates compared to exact duplicates, (ii) 0.5, assuming half as many near-duplicates as exact duplicates, and (iii) 1.0, assuming as many near-duplicates as there are exact ones. We found that the classifier trained on the augmented set filtered with $relative\_portion = 0.1$ performed best. Example images for cross-class noise are shown in Fig. 6.

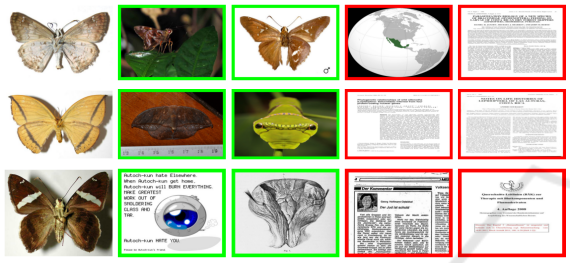
Cross-domain (CD) noise filtering was evaluated with three values for the number of clusters $k \in \{5, 10, 50\}$. Qualitative results of this filter are shown in Fig. 7. To estimate the effect of every single filter, we evaluated different filter combinations.

Tables 4 and 5 show results for the Costa Rica Moths and European Moths, respectively. The tables show that the augmentation of the training data from the Internet results in considerable improvements of the classifier ($\sim$11 % for Costa Rica Moths and $\sim$23 % for European Moths). Furthermore, we observe that although our filters reduce the amount of data by up to 58 %, the classification performances remain stable. This indicates that our filtering methods remove training samples that do not contribute to the correct classification. Finally, since the classification performance remains the same even with noisy

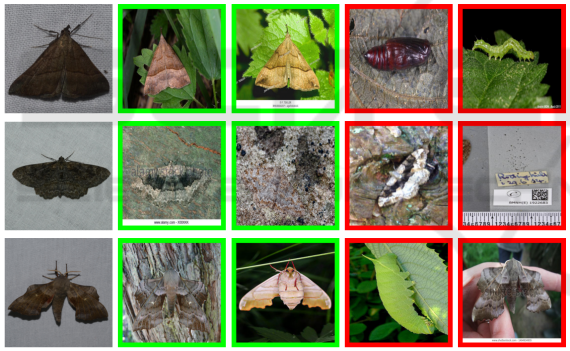
Figure 5: Some examples of detected test duplicates in the Web Costa Rica Moths dataset. Each pair contains an image from the test set and a near-duplicate in the augmentation dataset.

(a) Costa Rica Moths (b) European Moths

Figure 6: Some examples of detected cross-class noise in the Costa Rica and European Moths datasets. Each pair contains a reference image and its detected near-duplicate downloaded in another class.



(a) Costa Rica Moths



(b) European Moths

Figure 7: Detecting cross-domain noise in the augmented moth datasets. In each row five images are displayed: a training image from the seed dataset, two augment images from positive clusters (green frames, retained), and two images from negative clusters (red frames, filtered out).

data, these results show that CNNs can handle a certain amount of noisy data. This insight confirms the investigations of (Rolnick et al., 2017) in their work.

5 CONCLUSIONS

In this paper, we proposed a set of lightweight filtering methods for different kinds of label noise that occur when acquiring data from the Internet. We presented a duplicate detection method that combines features from a pre-trained CNN and a pixel-based similarity measure (SSIM). With this method, we removed test set duplicates, i.e., exact and near-duplicates between the downloaded augmentation set and the test set, and filtered cross-class noise by identifying ambiguous samples across the classes in the augmentation set. Additionally, we used the CNN features in combination with a clustering approach for identifying cross-domain noise, a problem that has poorly been studied so far.

We extensively evaluated the duplicate detection approach on various datasets with existing duplicate annotations. Furthermore, we proposed a technique to generate datasets with cross-domain noise and an arbitrary data-to-noise ratio due to the lack of datasets

Table 4: Results achieved for the Costa Rica Moths dataset. The baseline of training on the seed dataset only is compared to training with a merged training set (seed dataset joined with the entire augment set) and to training with those subsets of the merged dataset that our filter methods retained. Test duplicate filtering (TD), cross-domain noise filtering (CD) with different values for the number of clusters k, and cross-class noise filtering (CC) have been applied in different combinations (✻ = training dataset contains test duplicates, **bold** = best mean accuracy).

METHOD		ACCURACY ± STD (# TRAINING IMAGES)	
CNN features + SVM (Rodner et al., 2015)		79.20 %	(990)
Baseline InceptionV3 (only seed data)		75.24 % ±0.76	(990)
Merged, no filter ✻		86.17 % ±0.84	(10,124)
Merged + TD filter		86.11 % ±0.61	(9,941)
Merged + TD + CC filter		86.06 % ±0.69	(7,736)
Merged + TD + CD filter	$k = 5$	86.03 % ±0.70	(8,842)
	$k = 10$	85.26 % ±0.53	(8,820)
	$k = 50$	85.56 % ±0.92	(9,199)
Merged + TD + CC + CD filter	$k = 5$	86.33 % ±0.65	(7,348)
	$k = 10$	**86.73%** ±1.34	(7,347)
	$k = 50$	86.17 % ±0.65	(7,437)

Table 5: Results achieved for the European Moths dataset. The baseline of training on the seed dataset only is compared to training with a merged training set (seed dataset joined with the entire augment set) and to training with those subsets of the merged dataset that our filter methods retained. Test duplicate filtering (TD), cross-domain noise filtering (CD) with different values for the number of clusters k, and cross-class noise filtering (CC) have been applied in different combinations (**bold** = best mean accuracy).

METHOD		ACCURACY ± STD (# TRAINING IMAGES)	
Baseline InceptionV3 (only seed data)		72.75 % ±1.46	(300)
Merged, no filter		95.43 % ±0.57	(9,691)
Merged + CC filter		**95.93%** ±0.45	(9,424)
Merged + CD filter	$k = 5$	95.02 % ±0.47	(4,162)
	$k = 10$	95.62 % ±0.45	(6,458)
	$k = 50$	95.75 % ±0.34	(6,938)
Merged + CC + CD filter	$k = 5$	95.53 % ±0.22	(4,069)
	$k = 10$	95.65 % ±0.74	(6,276)
	$k = 50$	95.65 % ±0.18	(6,776)

with annotated cross-domain noise. We used these datasets to validate our cross-domain noise filter. In all these cases, we were able to demonstrate the effectiveness of our proposed methods.

Finally, we applied all our filtering methods for the moth species classification problem testing two different seed datasets. The downloaded images improved the classification accuracies. Furthermore, even though the filters reduce the amount of the training data by up to 58 %, the classification performance is not affected significantly compared to the utilization of all downloaded images. This indicates that the eliminated samples did not contribute to the classifier's decisions and were correctly filtered out.

The focus of this work was to develop and evaluate the filtering methods. Hence, we used a simple global classification approach. We assume that the deployment of a part- or attention-based method would also benefit from the label noise reduction offered by our filtering methods.

REFERENCES

Akcay, S., Atapour-Abarghouei, A., and Breckon, T. P. (2018). Ganomaly: Semi-supervised anomaly detection via adversarial training. In *Asian conference on computer vision*, pages 622–637. Springer.

Barz, B. and Denzler, J. (2020). Do we train on test data? purging cifar of near-duplicates. *Journal of Imaging*, 6(6):41.

Berg, T. L. and Forsyth, D. A. (2006). Animals on the web. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 1463–1470.

Chen, X., Shrivastava, A., and Gupta, A. (2013). Neil: Extracting visual knowledge from web data. In *IEEE International Conference on Computer Vision*, pages 1409–1416.

Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1251–1258.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3):273–297.

Cui, Y., Song, Y., Sun, C., Howard, A., and Belongie, S. (2018). Large scale fine-grained categorization and domain-specific transfer learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4109–4118.

Eskin, E. (2000). Detecting errors within a corpus using anomaly detection. In *1st Meeting of the North American Chapter of the Association for Computational Linguistics*.

Fefilatyev, S., Shreve, M., Kramer, K., Hall, L., Goldgof, D., Kasturi, R., Daly, K., Remsen, A., and Bunke, H. (2012). Label-noise reduction with support vector machines. In *21st International Conference on Pattern Recognition*, pages 3504–3508.

Frénay, B. and Verleysen, M. (2013). Classification in the presence of label noise: a survey. *IEEE Transactions on Neural Networks and Learning Systems*, 25(5):845–869.

Garcia, L. P., Lorena, A. C., Matwin, S., and de Carvalho, A. C. (2016). Ensembles of label noise filters: a ranking approach. *Data Mining and Knowledge Discovery*, 30(5):1192–1216.

Ge, W., Lin, X., and Yu, Y. (2019). Weakly supervised complementary parts models for fine-grained image classification from the bottom up. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3034–3043.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.

Ke, Y., Sukthankar, R., Huston, L., Ke, Y., and Sukthankar, R. (2004). Efficient near-duplicate detection and sub-image retrieval. In *ACM Multimedia*, volume 4, page 5.

Korsch, D., Bodesheim, P., and Denzler, J. (2019). Classification-specific parts for improving fine-grained visual categorization. In *German Conference on Pattern Recognition*, pages 62–75.

Krause, J., Sapp, B., Howard, A., Zhou, H., Toshev, A., Duerig, T., Philbin, J., and Fei-Fei, L. (2016). The unreasonable effectiveness of noisy data for fine-grained recognition. In *European Conference on Computer Vision*, pages 301–320.

Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. Technical report, University of Toronto.

Li, L.-J. and Fei-Fei, L. (2010). Optimol: automatic online picture collection via incremental model learning. *International Journal of Computer Vision*, 88(2):147–168.

Lin, T.-Y., RoyChowdhury, A., and Maji, S. (2015). Bilinear cnn models for fine-grained visual recognition. In *IEEE International Conference on Computer Vision*, pages 1449–1457.

Luo, J. and Nascimento, M. A. (2003). Content based sub-image retrieval via hierarchical tree matching. In *1st ACM International Workshop on Multimedia Databases*, pages 63–69.

Nicholson, B., Zhang, J., Sheng, V. S., and Wang, Z. (2015). Label noise correction methods. In *IEEE International Conference on Data Science and Advanced Analytics*, pages 1–9.

Ravi, S. and Larochelle, H. (2016). Optimization as a model for few-shot learning.

Rodner, E., Simon, M., Brehm, G., Pietsch, S., Wägele, J. W., and Denzler, J. (2015). Fine-grained recognition datasets for biodiversity analysis. In *CVPR Workshop on Fine-grained Visual Classification*.

Rolnick, D., Veit, A., Belongie, S., and Shavit, N. (2017). Deep learning is robust to massive label noise. *arXiv preprint arXiv:1705.10694*.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252.

Schroff, F., Criminisi, A., and Zisserman, A. (2010). Harvesting image databases from the web. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(4):754–766.

Simon, M., Gao, Y., Darrell, T., Denzler, J., and Rodner, E. (2017). Generalized orderless pooling performs implicit salient matching. In *IEEE International Conference on Computer Vision*, pages 4970–4979.

Snell, J., Swersky, K., and Zemel, R. (2017). Prototypical networks for few-shot learning. In *Advances in neural information processing systems*, pages 4077–4087.

Sung, F., Yang, Y., Zhang, L., Xiang, T., Torr, P. H., and Hospedales, T. M. (2018). Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1199–1208.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826.

Van Horn, G., Mac Aodha, O., Song, Y., Cui, Y., Sun, C., Shepard, A., Adam, H., Perona, P., and Belongie, S. (2018). The inaturalist species classification and detection dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8769–8778.

Wah, C., Branson, S., Welinder, P., Perona, P., and Belongie, S. (2011). The caltech-ucsd birds-200-2011 dataset. Technical Report CNS-TR-2011-001, California Institute of Technology.

Wang, B., Li, Z., Li, M., and Ma, W.-Y. (2006). Large-scale duplicate detection for web image search. In *IEEE International Conference on Multimedia and Expo*, pages 353–356.

Wang, J., Song, Y., Leung, T., Rosenberg, C., Wang, J., Philbin, J., Chen, B., and Wu, Y. (2014). Learning fine-grained image similarity with deep ranking. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1386–1393.

Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612.

Xiao, T., Xia, T., Yang, Y., Huang, C., and Wang, X. (2015). Learning from massive noisy labeled data for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2691–2699.

Xu, Z., Huang, S., Zhang, Y., and Tao, D. (2015). Augmenting strong supervision using web data for fine-grained categorization. In *IEEE International Conference on Computer Vision*.

Zhang, C., Yao, Y., Zhang, J., Chen, J., Huang, P., Zhang, J., and Tang, Z. (2020). Web-supervised network for fine-grained visual classification. In *IEEE International Conference on Multimedia and Expo*, pages 1–6.

Zhang, W. and Tan, X. (2019). Combining outlier detection and reconstruction error minimization for label noise reduction. In *IEEE International Conference on Big Data and Smart Computing*, pages 1–4.

Zheng, H., Fu, J., Mei, T., and Luo, J. (2017). Learning multi-attention convolutional neural network for fine-grained image recognition. In *IEEE International Conference on Computer Vision*, pages 5209–5217.

Zhuang, B., Liu, L., Li, Y., Shen, C., and Reid, I. (2017). Attend in groups: a weakly-supervised deep learning framework for learning from web data. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1878–1887.