Confusion Graph: Detecting Confusion Communities in Large Scale Image Classification

Ruochun Jin, Yong Dou, Yueqing Wang and Xin Niu
National Laboratory for Parallel and Distributed Processing,
National University of Defense Technology,
Changsha, Hunan, 410073, China
{jinruochun,yongdou,xinniu}@nudt.edu.cn, yqwang2013@163.com

Abstract

For deep CNN-based image classification models, we observe that confusions between classes with high visual similarity are much stronger than those where classes are visually dissimilar. With these unbalanced confusions, classes can be organized in communities, which is similar to cliques of people in the social network. Based on this, we propose a graph-based tool named "confusion graph" to quantify these confusions and further reveal the community structure inside the database. With this community structure, we can diagnose the model's weaknesses and improve the classification accuracy using specialized expert sub-nets, which is comparable to other state-of-the-art techniques. Utilizing this community information, we can also employ pre-trained models to automatically identify mislabeled images in the large scale database. With our method, researchers just need to manually check approximate 3% of the ILSVRC2012 classification database to locate almost all mislabeled samples.

1 Introduction

In the last few years, researchers have witnessed the great leap of image classification [Russakovsky *et al.*, 2015], especially after the significant success of deep convolutional neural networks [Krizhevsky *et al.*, 2012; Simonyan and Zisserman, 2014] and the emergence of large scale image datasets such as ImageNet [Deng *et al.*, 2009]. Though the classification accuracy of the state-of-the-art model is surpassing that of human beings, there still remain two critical challenges.

Firstly, it is extremely difficult to improve existing deep CNN-based models. Although practical optimization methods have been put forward [Yan *et al.*, 2015; Ahmed *et al.*, 2016; Kontschieder *et al.*, 2015], few well-developed theories have been proposed to guide the model design or optimization. In view of this situation, most efforts tend to employ experimental methods to improve existing models and an important part of this approach is how to diagnose and understand the weaknesses of the model. In order to address this issue, focusing on isolated samples that are most responsible for the model's errors, several diagnostic methods [Kabra *et al.*, 2015; Breiman and Wald Lecture, 2002] and visualization tools [Zeiler and Fergus, 2014; Vondrick *et al.*, 2013] have been developed. However, based on our observation, actually, it is the confusions between classes with high visual similarity that result in most mispredictions, which can hardly be discovered by those sample-scale diagnostic methods. For example, by previous methods, specific images of "hen" or "cock" may be found to be responsible for the failures of the model. But only with this sample-scale analysis, one can hardly notice that it is actually the confusion between the class "hen" and the class "cock" that results in most mispredictions. Contrarily, if confusions between classes are revealed, we will then efficiently locate the specific samples related to the errors. Thus, in order to detect the model's weaknesses, which further supports the improvement of the model, quantifying and understanding confusions between different classes are of vital importance.

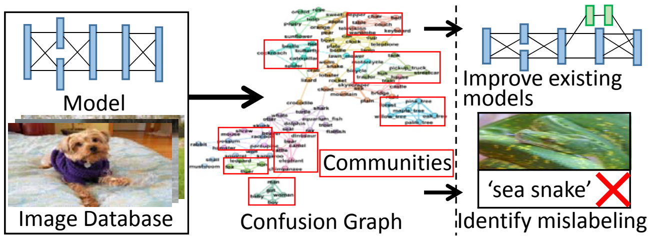


Figure 1: We use the "confusion graph" to quantify confusions between classes and reveal the community structure inside. Then we utilize the community information to improve existing models and automatically identify mislabeling in the image database.

The second challenge is the inevitable mislabeling in the large scale image database constructed by crowdsourcing, which can result in severe negative effects on supervised classifiers [Russakovsky *et al.*, 2015; Frénay and Verleysen, 2014]. This label noise is unavoidable due to two main reasons. Firstly, with more fine-grained classes added, expertise in certain fields, such as ornithology, is required to correctly label the images [Van Horn *et al.*, 2015; Nilsback and Zisserman, 2008]. However, most people involved in crowdsourcing have limited professional knowledge, which increases the probability of mislabeling. Secondly, as the database scale increases rapidly [Deng *et al.*, 2009], it is extremely laborious to identify all mislabeled samples by manual checking, which makes it almost impossible to eliminate all label noise.

Though researchers have noticed the severity of this issue, unfortunately, few methods have been proposed to automatically detect mislabeled images in the large scale database.

By analyzing the outputs of deep CNN models, we find that confusions between classes that have similar visual characteristics, such as shape, color, texture and background, is much stronger than those between classes with low visual similarity. This phenomenon is analogous to people's relationships in the social network, where relations between friends are closer than those between strangers. Based on this analogy, we propose a graph-based tool, named "confusion graph", to quantify confusions between different classes by accumulating the top predictions of each test image. Applying the community detection algorithm, we then reveal the expected community structure inside the graph, where classes within the same community have high visual similarity while those from different communities are visually dissimilar.

There are at least two applications of the confusion graph. Firstly, it can be used as a diagnostic tool to detect weaknesses of a given model. Each community in the graph is a weakness and the overall performance of the model can be improved if these weaknesses are overcome. For illustration purpose, we select ten 3-class communities and design specialized layers to overcome each weakness. For AlexNet-based and VGG-verydeep-16-based models, the mean decreases of the top-1 error rate are 1.49% and 3.45% respectively, which are comparable to other state-of-the-art methods [Yan *et al.*, 2015; Ahmed *et al.*, 2016]. Secondly, we employ pre-trained models along with the community information to automatically identify mislabeled samples in the image database. Evaluated with the randomly polluted Oxford-102 flowers dataset where 15% of the images are mislabeled, our method can detect approximate 89% of all wrong labels with the precision of 72%. When detecting mislabeling in the ILSVRC2012 classification validation set, with our method, researchers just need to manually check approximate 3% of the whole database to locate almost all wrongly labeled samples, which significantly reduce the labour work. To the best of our knowledge, there has been few similar work reported.

There are two main contributions in this paper as follows.

- We observe that most errors of deep CNN models occur due to the confusions between classes which have high visual similarity and image classes can be divided into communities based on their visual confusions.
- We develop a graph-based tool, named "confusion graph", to quantify confusions between classes. We further utilize the community structure inside the graph to diagnose weaknesses of the model and automatically identify mislabeled images in large scale datasets.

2 Related Work

As far as we know, few diagnostic methods have been proposed to understand errors of the image classification model. The most related work is [Kabra *et al.*, 2015], which locates specific samples that is most responsible for the errors by examining influential neighbors. However, their method can not figure out which classes are responsible for the model's mispredictions, which we believe is the more fundamental reason for failures. In addition, various visualization methods have been proposed to visualize feature representations [Zeiler and Fergus, 2014; Vondrick *et al.*, 2013], which are also helpful in terms of understanding failures of the model.

In order to improve the model's robustness to label noise, several approaches have been proposed by automatically identifying and down weighting mislabeled samples [Sánchez *et al.*, 2003; Freund, 2009; Brodley and Friedl, 1999]. However, few methods have been applied to the identification of label noise in image databases. [Stokes *et al.*, 2016] verifies their method in some simple image databases such as MNIST, which contains only 10 classes of digits. The performance of mislabeled image identification in large scale database, such as ImageNet, remains unknown.

3 Confusion Graph and Communities Inside

3.1 Definition of the Confusion Graph

Definition 1 *Given a N-class classification with a model M and a dataset T, the confusion graph $G = (V, E)$ of the classification consists of a set of vertexes $V = \{v_1, \ldots, v_N\}$ and undirected edges $E \subseteq V \times V$ without self loop.*

Each vertex $v \in V$ represents one class in the classification. The edge $e_{i,j} \in E$ indicates that M may confuse class i with class j. The weight $w_{i,j}$ (detailed in Section 3.2) of $e_{i,j}$ quantifies the likelihood that M may mistake class i for class j or mistake class j for class i. The larger the weight of an edge is, the higher the likelihood is.

3.2 Establish a Confusion Graph

Given a model M, a testing dataset T that includes N classes with n single-label samples in each class and an integer parameter τ, Algorithm 1 establishes the corresponding confusion graph G by mapping the top-τ predictions of each test sample to an undirected graph. The main idea of the algorithm is firstly normalizing the top-τ classification scores, where the confusion information is hidden, of each test sample and then accumulating each normalized score to the weight of the edge that connects the labeled class and the predicted class. For example, assume that we feed an image of "cat" to the model and obtain the class "cat" with the score of 0.5 as the top prediction, "dog" with the score of 0.2 as the second and "deer" with the score of 0.1 as the third. Then the algorithm will normalize the three scores as 0.625, 0.25 and 0.125 respectively. Eventually, ignoring self loops, the algorithm will accumulate 0.25 to the weight of the edge between "cat" and "dog" and add 0.125 to that connecting "cat" and "deer".

Specifically, in Algorithm 1, the function "TestOneSample" feeds one image t to model M and the output includes the predicted classes and scores which are saved in $R.c$ and $R.s$ respectively. The function "ScoreNormalization" takes an array of scores as input and normalize each value by the softmax-like Equation 1. By normalizing the top-τ scores, each test sample has the same contribution in terms of constructing the weights of edges in graph G.

$$topR[i].s = \frac{e^{topR[i].s}}{\sum_{j=1}^{\tau} e^{topR[j].s}} \tag{1}$$

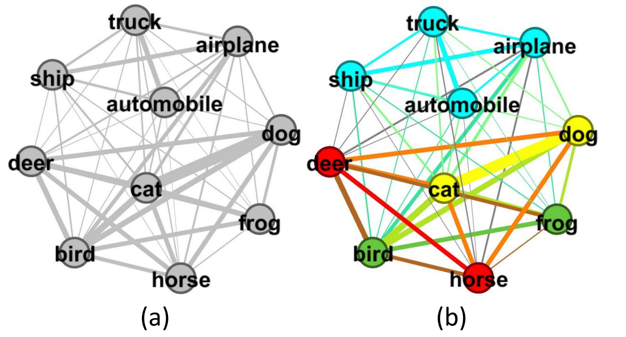


Figure 2: Confusion graph of LeNet-CIFAR10 (a) and communities inside (b).

Using Algorithm 1 with τ set as 5, for illustration purpose, we evaluate the pre-trained LeNet [LeCun *et al.*, 1995] with the CIFAR10 [Krizhevsky and Hinton, 2009] validation set and obtain the confusion graph named LeNet-CIFAR10. All experiments in this paper are completed using Matlab2014a with MatConvNet [Vedaldi and Lenc, 2015] and the pre-trained models are downloaded from the MatConvNet website. As is shown in Figure 2 (a), each vertex represents one class in the dataset and the weight of each edge quantifies the confusion between the two classes. The strongest edge connecting "dog" and "cat" in the graph, for instance, indicates that the model is likely to confuse dogs with cats. Contrarily, the tiny link between "deer" and "truck" means that the model seldom confuses deer with trucks.

Algorithm 1: Establish the confusion graph of a N-class classification

```
   Input: A $N$-class classification model $M$; a dataset $T$;
   Top concern number $\tau(\tau \leq N)$;
   Output: The confusion graph $G = (V, E)$;
1  $V \Leftarrow \{v_1, \dots, v_N\}$, $E \Leftarrow \phi$;
2  for each $t \in T$ ( $t$'s label is $l$ ) do
3      $R_c$,$R_s \Leftarrow$ TestOneSample($M, t$);
4      $topR.s \Leftarrow R_s[1:\tau]$; $topR.c \Leftarrow R_c[1:\tau]$;
5      $topR.s \Leftarrow$ ScoreNormalization($topR.s$);
6      for $i$ from 1 to $\tau$ do
7          if $topR[i].c \neq l$ then
8              if $e_{l,topR[i].c} \notin E$ then
9                  $E \Leftarrow E \cup e_{l,topR[i].c}$;
10                 $w_{l,topR[i].c} \Leftarrow topR[i].s$;
11             else
12                 $w_{l,topR[i].c} \Leftarrow w_{l,topR[i].c} + topR[i].s$;
13 return $G$;
```

3.3 Detect Communities in a Confusion Graph

The confusions between different classes in the large scale image database, such as ImageNet, are so complex that we can never simply use isolated edges to analyze the many-to-many relationships inside. Thus, inspired by the community structure in the social network, which reveals the organization of people, we apply the community detection algorithm to explore communities inside the confusion graph and utilize the modularity of each community as a metric to quantify each community's compactness. In a confusion graph, the meaning of a community is twofold. On the one hand, from the model's perspective, most of the mistakes result from the confusions within the communities, which indicates that each community can be viewed as a weakness of the model. On the other hand, from the database's perspective, classes belonging to the same community are hard problems for the model because most errors occur within these communities.

We utilize the first iteration of the fast community detection algorithm [Blondel *et al.*, 2008] to find out fine-grained communities inside the confusion graph. Due to the extremely unbalanced weight distribution of edges in the confusion graph, in order to highlight main confusions, we delete most tiny edges before applying the algorithm. Specifically, we firstly sort the edges based on their weights. Then we use the p-th percentile (0< p <100) as the cutting point to filter out the tiny edges. Any edge whose weight is less than the p-th percentile is deleted. We also introduce the concept of modularity from [Blondel *et al.*, 2008] to measure the density of links inside communities as compared to links among communities. This value enables us to compare the compactness of different communities (used in Section 4.1). Given a certain community partition, we can compute the modularity Q_k of the kth community by Equation 2

$$Q_k = \frac{1}{2m} \sum_{i,j} (w_{i,j} - \frac{s_i s_j}{2m}) \delta(c_i, c_j) \theta(c_i, c_j, k) \qquad (2)$$

where $s_i = \sum_j w_{i,j}$ is the sum of the weights of edges attached to vertex i, c_i is the community to which vertex i belongs, the δ-function $\delta(u, v)$ equals 1 if $u = v$ and 0 otherwise, the θ-function $\theta(c_i, c_j, k)$ is 1 if c_i or c_j is the kth community and 0 otherwise and $m = \frac{1}{2} \sum_{ij} w_{ij}$.

Setting p as 50, we obtain the community partition of the LeNet-CIFAR10 (Figure 2 (b)) where classes from the same community are colored the same. "Truck", "automobile", "ship" and "airplane" belong to the same community because they are all man-made transportation carriers with shells, which separates them from the rest. Similarly, classes of animals can further be separated into three communities based on their visual characteristics.

In order to explain why certain classes gather together, we select CIFAR100 [Krizhevsky and Hinton, 2009] as an object of study because its complexity is between those of CIFAR10 and ImageNet. This moderate complexity provides diverse communities for investigation and the manual analysis workload of the community structure is bearable. In CIFAR100, There are 100 fine-grained classes which further come in 20 coarse-grained superclasses of 5 subclasses each. This class structure design is based on the idea that classes within the same superclass are similar and thus harder to distinguish than classes belonging to different superclasses. By training a LeNet-based model whose top-5 error rate is 22.5%, setting τ as 5, we obtain its confusion graph with the validation set.

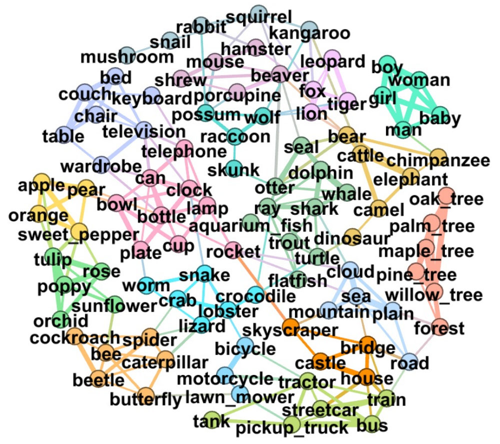


Figure 3: Community structure inside CIFAR100 dataset.

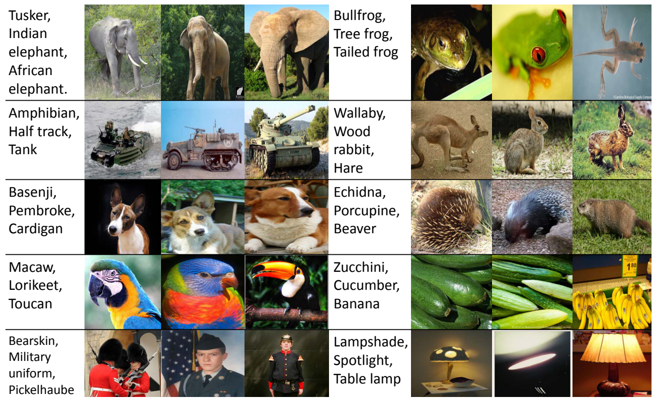

Tusker, Indian elephant, African elephant.
Bullfrog, Tree frog, Tailed frog
Amphibian, Half track, Tank
Wallaby, Wood rabbit, Hare
Basenji, Pembroke, Cardigan
Echidna, Porcupine, Beaver
Macaw, Lorikeet, Toucan
Zucchini, Cucumber, Banana
Bearskin, Military uniform, Pickelhaube
Lampshade, Spotlight, Table lamp

Figure 4: Examples of 3-class communities in ILSVRC2012. Classes within each community are visually similar.

Concealing tiny edges for illustration purpose, with p set as 95, we further reveal the communities inside (Figure 3).

By meticulous comparison, we find significant difference between the community structure inside the confusion graph and the original class structure of CIFAR100. The main reasons that some classes from different superclasses in CIFAR100 assemble as a community in the confusion graph are summarized as follows.

Similar shape For example, although “snake” belongs to superclass “reptile” and “worm” is a member of “non-insect invertebrates”, these two classes are in the same confusion community because both of them have long bodies.

Similar background or environment A good example for this is the biggest community consisting of “otter”, “seal”, “whale”, “turtle”, “dolphin”, “aquarium fish”, “ray”, “shark”, “trout” and “flatfish”. Though these creatures come from different superclasses, they all live in or near water. These similar water backgrounds combine them as a community.

Similar texture or color “Forest”, “willow tree”, “pine tree”, “maple tree”, “oak tree” and “palm tree”, for instance, constitute a compact community because the color and texture of foliage are quite similar.

Cooccurence Though “bed”, “table”, “chair”, “television”, “couch”, “wardrobe” and “keyboard” have little visual similarity, these classes form a community because these furniture and electronic devices always appear together in a picture of a living room or a bedroom.

Setting τ as 5, we obtain the confusion graph of AlexNet evaluated with the ILSVRC2012 validation set, named AN-ILSVRC2012. Then we perform similar analysis on the AN-ILSVRC2012 and the phenomena, that classes with high visual similarity gather as a community, can also be observed. Setting p as 92, we obtain a community list L recording 143 communities inside AN-ILSVRC2012. The sizes of the communities vary from 1 to 24 (detailed in Figure 6) and 10 communities with 3 classes in each are shown in Figure 4 as examples. From these examples, we can see that reasons summarized based on CIFAR100 are compatible with the ImageNet database. This further indicates that most classification errors of the state-of-the-art CNN models result from classes that have minor difference in visual characteristics.

4 Applications of the Confusion Graph

4.1 Detect Class-scale Weaknesses of the Model

The first application of the confusion graph is diagnosing weaknesses of the the corresponding classification model because most errors occur within each community while few confusions exist between different communities. Each community can be viewed as a weakness and the prediction accuracy can be improved if these weaknesses are overcome. In addition, based on our experiments, we find that communities with higher modularity values are more promising in terms of improving the model’s classification performance.

In order to prove the advantage of the graph-based diagnosis and the effect of the modularity value, we select five 3-class communities with the highest modularity values and another five with the lowest modularity values from the community list L obtained in Section 3.3. For each selected community, we train an AlexNet-based expert sub-net (ES) which is shown in Figure 5. Each ES contains three full connection layers and the forward prediction process can be divided into 2 stages. First, the image is classified by the original AlexNet via path ①. If none of the top-3 predictions belongs to the ES’s community, the whole process ends. Otherwise, secondly, the feature extracted by the CNN part will be directly sent to the corresponding ES via path ② and the output of the ES will replace the top-3 predictions of stage one. By cascading the randomly initialized ES to the pre-trained convolutional layers of AlexNet, we train each ES with images of the corresponding 3 classes from the ILSVRC2012 training set.

We tested each refined model using images from the ILSVRC2012 validation set. Each test utilized 150 images from 3 classes in which the ES specialized. The top-1 error rate was employed as the performance metric and the results are shown in Table 1. With the same parameter setting, we have also performed similar experiments based on VGG-verydeep-16, where we construct the confusion graph of VGG-verydeep-16, detect communities inside and eventually employ the ES to overcome each weakness.

As is shown in Table 1, in AlexNet-based experiments, all error rates decline and the mean decrease is approximate 1.49%. Additionally, the mean decrease in top-1 error rate of the communities with top 5 modularity is larger than that

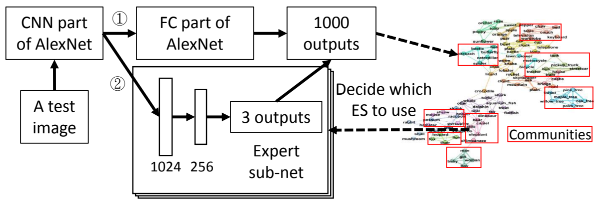


Figure 5: Structure of the AlexNet-based expert sub-net.

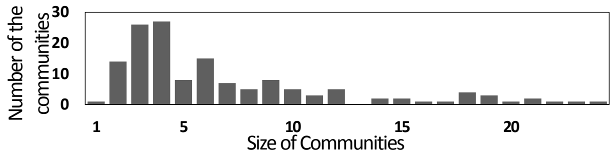


Figure 6: The number of communities of different sizes in AN-ILSVRC2012.

of the communities with the lowest 5 modularity, which are 2.15% and 0.84% respectively. Similar results can also be observed in the VGG-based experiments, which are 5.52% and 1.38% respectively. This indicates that with the same optimization method, more improvements can be obtained from communities with higher compactness. Thus, by detecting communities in the confusion graph, overcoming the weakness represented by each community and focusing on communities with high modularity, we can reduce the model's overall error rate effectively.

Comparable to ours, similar optimization results have been reported in [Yan *et al.*, 2015] which decreases the top-1 errors by 1.11%. Specifically, they use spectral clustering based on the confusion matrix to cluster fine-grained classes into coarse classes. However, their method can not clearly show which coarse class has more potential in terms of accuracy improvement and spectral clustering is sensitive to parameter selection [Zelnik-Manor and Perona, 2004], which is less robust than our graph-based method [Blondel *et al.*, 2008].

4.2 Identify Mislabeled Images in the Database

Mislabeled images are defined as images with totally irrelevant labels. If any object in an image is correctly labeled, the image is not mislabeled. This definition is in accord with that in [Deng *et al.*, 2009]. With Algorithm 2, we can employ a pre-trained model to automatically detect mislabeled images.

Algorithm 2: Detect mislabeled images in a dataset

```
Input: The classification model $M$;
  A subset $S$ of the whole dataset, $S = \{s_1, \ldots, s_n\}$
  where all images are labeled $\alpha$;
  Community list $L$ and an integer parameter $\mu$ ($\mu > 0$);
Output: A dataset $W$ containing wrongly labeled
        images;
1 $W \Leftarrow \phi$;
2 for each $s_i \in S$, do
3     $R_c$,$R_s \Leftarrow$ TestOneSample($M, s_i$);
4     If more than half of top-$\mu$ classes in $R_c$ do not share
      the same community with $\alpha$, the Root-Mean-Square
      of top-$\mu$ scores in $R_s$ is higher than average and $\alpha$
      does not exist in top-$\mu$ classes of $R_c$, add $s_i$ to $W$.
5 return $W$;
```

Our method utilizes two criteria to filter out the suspiciously mislabeled samples. Firstly, if most top-μ predicted classes do not stay in the same community with the labeled class, this image may probably be mislabeled. Secondly, if the RMS of the top-μ predicted scores is higher than average, this prediction is credible. Based on these two criteria, most mislabeling can be identified because a mislabeled image usually has a credible prediction result but most of its top predicted classes do not stay in the same community with the labeled class. The community information in Algorithm 2 is of vital importance because only with the community list can we take advantage of the human-level top-5 error rate of state-of-the-art models to identify the labeling mistakes. The parameter μ controls the trade-off between the precision and the recall of the detection, where a high μ leads to high precision with low recall and a low μ leads to low precision with high recall. In practice, we iteratively correct the mislabeled images by manually checking the output of Algorithm 2 in each iteration. The parameter μ decreases from 5 to 2 successively during iterations of auto-detection and the iterative process ends until no image in the output is true mislabeled.

We design two experiments to verify our method. Firstly, in order to demonstrate the high precision and recall of our method, we apply Algorithm 2 to cleaning the randomly polluted Oxford 102 flowers dataset. The original dataset [Nilsback and Zisserman, 2008] contains 102 categories of fine-grained flowers and all images are correctly labeled by experts. We firstly train the classification model with the clean dataset and obtain four models whose top-1 prediction error rates are 40%, 30%, 20% and 10% respectively. Then we use the validation set, containing 1020 images, i.e. 10 image for each class, to construct the confusion graph and obtain the community list of each model. In order to simulate the situation where some images are mislabeled, we randomly select 3%, 5% and 10% of the images in the validation set and mislabel each of them with a random class. With our iterative method, we utilize the four models to identify mislabeled samples in the polluted validation sets and the results are shown in Table 2, where "PM" means the percentage of mislabeling, "ER" means the error rate of the model, "NM" denotes the number of mislabeled samples, "NMD" means the number of mislabeled samples that are detected, "NTMD" indicates the number of true mislabeled samples in the detection result. The "precision" is the ratio of "NTMD" to "NMD" and the "recall" is the ratio of "NTMD" to "NM". As is shown in Table 2, our method is able to identify approximate 92% of all mislabeled images and the corresponding precision is around 80%. In addition, the results suggest that the "precision" and the "recall" of our method are negatively correlated with the model's error rate, which means that we can more accurately find more mislabeled samples if we employ a model with lower classification error rate.

In order to investigate the performance when processing large scale database, in our second experiment, we employ the pre-trained VGG-verydeep-16 to detect wrong labels in

Table 1: Accuracy improvement (error rate in %) in the experiments based on two different existing models.

AlexNet	**Communities with top 5 modularity**						**Communities with lowest 5 modularity**					
Category No.	102 386 387	352 353 354	56 60 65	231 232 233	409 587 848	Mean	607 787 883	651 829 856	589 791 792	571 692 797	463 793 841	Mean
Top1 Error	51.67	29.33	54.00	44.66	46.20	45.17	48.96	65.71	51.67	52.48	51.38	54.04
ESTop1 Error	45.63	28.00	54.00	42.66	44.82	43.02	48.96	65.71	51.00	50.35	50.00	53.20
Top1Decrease	6.04	1.33	0.00	2.00	1.38	**2.15**	0.00	0.00	0.67	2.13	1.38	**0.84**
VGG-VD-16	**Communities with top 5 modularity**						**Communities with lowest 5 modularity**					
Category No.	36 37 38	278 279 281	357 359 360	172 173 177	156 201 205	Mean	31 32 33	548 566 706	367 368 370	11 14 16	188 194 202	Mean
Top1 Error	42.95	44.00	56.75	33.10	44.59	44.28	39.33	22.38	28.57	12.66	43.33	29.25
ESTop1 Error	40.26	38.66	52.70	25.67	36.48	38.75	38.00	20.15	28.57	10.66	42.00	27.88
Top1Decrease	2.69	5.34	4.05	7.43	8.11	**5.52**	1.33	2.23	0.00	2.00	1.33	**1.38**

Table 2: Mislabeled image detection results in the polluted Oxford 102 flowers dataset.

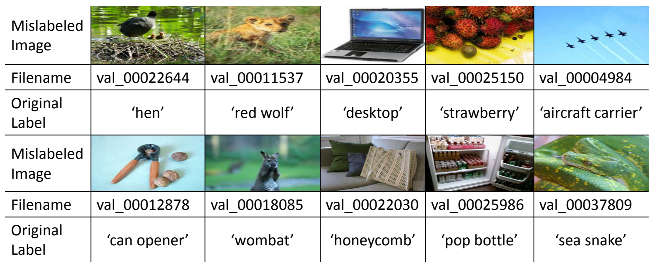
PM	top-1 ER	NM	NMD	NTMD	precision	recall
3%	40%	30	110	25	20.00%	83.33%
	30%	30	58	22	37.93%	73.33%
	20%	30	37	25	67.57%	83.33%
	10%	30	39	28	**71.79%**	**93.33%**
5%	40%	51	164	34	20.73%	66.67%
	30%	51	76	41	54.79%	80.39%
	20%	51	50	38	76.00%	74.51%
	10%	51	71	48	**80.34%**	**90.20%**
10%	40%	102	241	71	29.46%	69.61%
	30%	102	116	82	70.69%	80.39%
	20%	102	124	82	66.13%	80.39%
	10%	102	117	94	**80.34%**	**92.16%**

the classification validation set of ILSVRC2012. The corresponding confusion graph and the community list are obtained in Section 4.1. In our experiments, we firstly use Algorithm 2 to detect the suspicious samples. Then we manually check each auto-detected sample and confirm the number of true wrong labels. The parameter μ varies from 0 to 5 and the results are shown in Table 3, where "TMP" is the true mislabeled percentage and "DMP" is the detected mislabeled percentage, which are the ratios of "NTMD" and "NMD" to the number of all images in the dataset (50,000 in this experiment) respectively. Figure 7 illustrates some of our auto-detection results as examples.

Table 3: Mislabeled image detection results in the ILSVRC2012 validation set using pre-trained VGG-verydeep-16.

μ	NMD	NTMD	precision	TMP	DMP
5	1431	137	9.57%	0.27%	**2.86%**
4	1669	159	9.53%	0.32%	3.34%
3	1976	190	9.62%	0.38%	3.95%
2	2586	224	8.66%	0.45%	5.17%
1	3993	301	7.54%	0.60%	7.99%
0	6395	**383**	5.99%	**0.77%**	12.79%

Based on our second experiment, we can draw three main conclusions. Firstly, according to [Deng *et al.*, 2009], the labeling accuracy of the ImageNet is 99.7%. However, by manually checking the suspicious samples detected by VGG-verydeep-16, we confirm 383 true wrong labels in 50K images, which indicates that the previously reported accuracy of the ImageNet may be higher than reality. Secondly, if the accuracy of the ImageNet is truly 99.7% as reported, with our method, researchers just need to manually check approximate 3% of the ImageNet database to find out almost all wrongly labeled samples, which significantly reduces the labour work. Thirdly, we observe that most annotation errors occur in peculiar flora and fauna, such as "whiptail" or "langur", which proves that expertise is necessary when labeling a large scale image database with multifarious categories inside.



Mislabeled Image					
Filename	val_00022644	val_00011537	val_00020355	val_00025150	val_00004984
Original Label	'hen'	'red wolf'	'desktop'	'strawberry'	'aircraft carrier'
Mislabeled Image					
Filename	val_00012878	val_00018085	val_00022030	val_00025986	val_00037809
Original Label	'can opener'	'wombat'	'honeycomb'	'pop bottle'	'sea snake'

Figure 7: Examples of mislabeling that are automatically detected in the ILSVRC2012 classification validation set.

5 Conclusion

We propose the confusion graph to quantify confusions between different classes in image classification. Applying the community detection algorithm, we further reveal the communities inside the confusion graph, which are similar to cliques of people in the social network. Classes within the same community have high visual similarity and those from different communities are visually dissimilar. The confusion graph can be used to detect weaknesses of a classification model. By designing specialized layers for each weakness, we achieved state-of-the-art improvements of existing models. The community information can also be employed to identify mislabeled images in the database. With our method, researchers only need to manually check approximate 3% of the ImageNet database to locate almost all mislabeled images. To the best of our knowledge, this is the first reported work to automatically detect mislabeled images in ImageNet.

Acknowledgements

This work was supported by the Natural Science Foundation of China under the grant No. U1435219, No. 61402507 and No. 61303070.

References

[Ahmed *et al.*, 2016] Karim Ahmed, Mohammad Haris Baig, and Lorenzo Torresani. Network of experts for large-scale image categorization. *arXiv preprint arXiv:1604.06119*, 2016.

[Blondel *et al.*, 2008] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008, 2008.

[Breiman and Wald Lecture, 2002] Leo Breiman and I-I Wald Lecture. Looking inside the black box. *Wald Lecture II, Department of Statistics, California University*, 2002.

[Brodley and Friedl, 1999] Carla E. Brodley and Mark A. Friedl. Identifying mislabeled training data. *Journal of Artificial Intelligence Research*, 11:131–167, 1999.

[Deng *et al.*, 2009] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[Frénay and Verleysen, 2014] Benoît Frénay and Michel Verleysen. Classification in the presence of label noise: a survey. *IEEE transactions on neural networks and learning systems*, 25(5):845–869, 2014.

[Freund, 2009] Yoav Freund. A more robust boosting algorithm. *arXiv preprint arXiv:0905.2138*, 2009.

[Kabra *et al.*, 2015] Mayank Kabra, Alice Robie, and Kristin Branson. Understanding classifier errors by examining influential neighbors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3917–3925, 2015.

[Kontschieder *et al.*, 2015] Peter Kontschieder, Madalina Fiterau, Antonio Criminisi, and Samuel Rota Bulo. Deep neural decision forests. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1467–1475, 2015.

[Krizhevsky and Hinton, 2009] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

[Krizhevsky *et al.*, 2012] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[LeCun *et al.*, 1995] Yann LeCun, LD Jackel, Leon Bottou, A Brunot, Corinna Cortes, JS Denker, Harris Drucker, I Guyon, UA Muller, Eduard Sackinger, et al. Comparison of learning algorithms for handwritten digit recognition. In *International conference on artificial neural networks*, volume 60, pages 53–60. Perth, Australia, 1995.

[Nilsback and Zisserman, 2008] M-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008.

[Russakovsky *et al.*, 2015] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[Sánchez *et al.*, 2003] José Salvador Sánchez, Ricardo Barandela, Ana I Marqués, Roberto Alejo, and Jorge Badenas. Analysis of new techniques to obtain quality training sets. *Pattern Recognition Letters*, 24(7):1015–1022, 2003.

[Simonyan and Zisserman, 2014] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[Stokes *et al.*, 2016] Jack W Stokes, Ashish Kapoor, and Debajyoti Ray. Asking for a second opinion: Re-querying of noisy multi-class labels. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2329–2333. IEEE, 2016.

[Van Horn *et al.*, 2015] Grant Van Horn, Steve Branson, Ryan Farrell, Scott Haber, Jessie Barry, Panos Ipeirotis, Pietro Perona, and Serge Belongie. Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 595–604, 2015.

[Vedaldi and Lenc, 2015] Andrea Vedaldi and Karel Lenc. Matconvnet: Convolutional neural networks for matlab. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 689–692. ACM, 2015.

[Vondrick *et al.*, 2013] Carl Vondrick, Aditya Khosla, Tomasz Malisiewicz, and Antonio Torralba. Hoggles: Visualizing object detection features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1–8, 2013.

[Yan *et al.*, 2015] Zhicheng Yan, Hao Zhang, Robinson Piramuthu, Vignesh Jagadeesh, Dennis DeCoste, Wei Di, and Yizhou Yu. Hd-cnn: hierarchical deep convolutional neural networks for large scale visual recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2740–2748, 2015.

[Zeiler and Fergus, 2014] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, pages 818–833. Springer, 2014.

[Zelnik-Manor and Perona, 2004] Lihi Zelnik-Manor and Pietro Perona. Self-tuning spectral clustering. In *NIPS*, volume 17, page 16, 2004.