2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)
Banff Center, Banff, Canada, October 5-8, 2017

Finding Label Noise Examples in Large Scale Datasets

Rajmadhan Ekambaram
Department of Computer Science and Engineering
University of South Florida
Tampa, Florida 33620
Email: rajmadhan@mail.usf.edu

Dmitry B. Goldgof
Department of Computer Science and Engineering
University of South Florida
Tampa, Florida 33620
Email: goldgof@mail.usf.edu

Lawrence O. Hall
Department of Computer Science and Engineering
University of South Florida
Tampa, Florida 33620
Email: lohall@mail.usf.edu

Mislabeled examples are difficult to avoid while building large scale datasets. In this paper we discuss an efficient approach for finding those mislabeled examples. Our approach involves selecting a small number of potentially mislabeled examples for review by an expert. We demonstrate the utility of our method by finding some mislabeled examples in one large scale dataset (ImageNet). We found 92 errors by automatically selecting 3607 examples to review out of 22951 images from 18 classes. This requires reviewing 9 times fewer examples than the random sampling method to find an equivalent number of mislabels.

I. Introduction

The ground truth for large scale datasets, by necessity, is usually done by non-experts, and may therefore be prone to error. Heuristic approaches are usually designed to minimize errors. For example, in creating the ImageNet image classification dataset a voting scheme coupled with a confidence score was used [1]. A confidence score was determined for each synset [2] based on an initial subset of images. For the remaining images in each synset, voting from non-experts was gathered from Amazon Mechanical Turk until the predetermined confidence score was reached. It is difficult to avoid label noise even after following such a stringent label collection process. It is reported that the ImageNet dataset has only 0.3% label noise errors across all synsets [2]. We use the term "mislabeled examples" and "label noise" in this paper interchangeably.

Our goal is to show an approach to finding mislabels that is both applicable to very large labeling efforts and requires as little human intervention as possible. In particular, the label noise finding approach is applied to ImageNet to see if it helps in discovering unknown mislabels.

The reported label noise error of 0.3% in the ImageNet dataset is based on the manual verification of 80 synsets. The same amount of label noise error was found in their recent evaluation [1]. There were five mislabeled examples discovered in manual verification of 1500 randomly sampled examples in the ILSVRC2012-2014 image classification test set images [1]. Though it is possible that the reported noise level is approximately correct, we believe that reaching the conclusion based on the evaluation on such a small number of random examples (80 synsets in [2] and 1500 in [1]) might not convey the correct information. Instead of randomly sampling the examples, in this paper, we would like to systematically find the mislabeled examples. In particular, we follow up with the previous work on the label noise problem [3] [4] and demonstrate the usefulness of our approach by testing and uncovering previously unknown mislabeled examples in the ImageNet dataset. One of the mislabeled examples that is found by our method is shown in Figure 1.

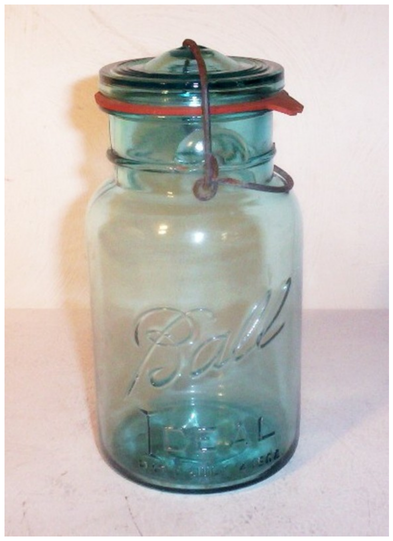

Fig. 1. The above image is mislabeled as hatchet in the ImageNet dataset. Image ID: n03498962_14162

The performance of supervised learning algorithms depends on the quality of the training data [5] [6]. Presence of label noise in the dataset might result in the deterioration of the classifier performance, increase in model complexity, increase in the need for training data size, difficulty in identifying the relevant features, etc. Nevertheless there are some reports to the contrary. Results in [7] and [8] shows that label noise can be used or added intentionally to learn better classifiers. We are not dealing with these learning methods in this paper. It can also be the case that for some (small) classes it is particularly important to have labels correct to improve prediction accuracy for them. This work can be particularly useful in that case.

In [9] an excellent and comprehensive survey about the label noise problem was presented. The taxonomy in [9] classifies the label noise techniques into three different categories: 1. The learning algorithm is robust to label noise examples, 2. The label noise examples are filtered (either removed or relabeled) before training, 3. Explicitly consider the label noise examples in the learning algorithm model. Though our approach is similar to the filtering category, techniques that involve manual review of the training examples are not considered in [9]. The reason is that manual review is usually expensive and time consuming for large datasets. We particularly try to address these issues in our research.

Filtering based approaches suffer from the chicken-and-egg problem as discussed in [10]. It is due to the two constraints: 1) good classifiers are required to find the mislabeled examples and 2) good examples are needed for training a good classifier. Our assumption that the majority of the label noise examples are captured by support vectors of a support vector machine finds a trade-off between these two constraints and overcome the chicken-and-egg problem to some extent.

Several filtering based techniques to handle label noise problem were proposed in the literature [10], [11] and [12]. Performance comparison of some filtering techniques is reported in [13]. Filtering approaches can be classified into three types. The first type of filtering approach creates an ensemble of classifiers and finds the mislabeled examples either based on majority voting or using probabilistic voting [10]. The second type of approach computes a score or confidence measure to rank the examples for their chance of being mislabeled [11]. The third type of approach finds the mislabeled examples based on their geometric neighbors [12]. The important difference between our method and other filtering based techniques is in the clear separation of the subset of examples which capture a majority of the label noise examples. A recent work [14] shows that this approach is useful to find malwares among android applications.

We describe our general method for finding mislabeled images by using the ImageNet dataset as an example in Section II. The initial experimental results for the ImageNet dataset are described in Section III. We also performed experiments on the UCI character recognition and MNIST digit recognition datasets. Since our previous work [3] and [4] involved datasets with higher noise levels (above 10%), experiments described in this paper are performed with noise levels comparable to those estimated for the ImageNet dataset. The results of these experiments show that it is possible to find mislabels in very large data sets such as ImageNet.

II. Algorithm

Support Vector Machines (SVM) is a machine learning algorithm based on the maximum margin principle. An SVM classifier maximizes the distance (margin) between the examples from the two classes. The examples which lie on the boundary separating the two classes are called support vectors (SVs). The decision boundary can be described as a linear combination of the support vector examples. The examples that violate the margin constraints are penalized in the soft margin SVM optimization process to improve the generalization performance and to apply the maximum margin method for non-separable classes. The soft margin SVM [15] for the two class problem is defined as

$$\begin{aligned} \min_{\mathbf{w}\in R^d, b, \xi_i \in R^+} \quad & \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{N}\xi_i \\ \text{s.t.} \quad & y_i(\mathbf{w}^T\mathbf{x}_i - b) \geq 1 - \xi_i, \forall i \\ & \xi_i \geq 0, \forall i \end{aligned} \tag{1}$$

where $\mathbf{w}$ is the normal to the hyperplane separating the two classes, N is the number of training examples, the ξ_i is the slackness for the examples that violates the margin constraint, the $y_i \in [-1, 1]$ are the class labels, $\mathbf{x}_i$ is a d–dimensional example, b is the bias.

Our method is based on the hypothesis that an SVM classifier has the property of selecting a majority of the uniform random label noise examples as its SVs [3], [4]. This hypothesis is based on the intuition that SVs are the important examples as they are involved in the decision boundary and they are the most confusing examples in the feature space as they lie closer to the examples from the opposite class. In a recent work [16] it was shown that majority of the uniform random label noise examples will get selected as SVs. This property suggests that it is sufficient to review the support vector examples to find the mislabeled examples in a dataset. The method was further refined to reduce the number of examples that need to be reviewed in [4]. The experimental results shown in [3], [4] supports this claim. The steps involved in the method is shown in Figure 2 and is described below:

1) Train a two-class SVM classifier for any two potentially confusing classes in the dataset and obtain the support vector (noisy set) and non-support vector (relatively noise free set) examples.
2) Create another two-class classifier (SVM, Random Forest, etc) with the examples in the relatively noise free set.
3) Predict the labels of the examples in the noisy set with the classifier created in step 2.
4) Manually verify the examples whose prediction in step 3 differ from its ground truth (i.e. misclassified in step 3). For the confirmed mislabeled examples either change the label of the examples or remove the examples from the dataset as required.
5) Repeat the above steps until all the examples are correctly classified in step 4.

In the ALNR approach described in [4] an SVM classifier is learned with the non-SV examples and is used to predict the SV examples. In this paper we extend this method to use classifier to learn from the non-SV examples, i.e., in Step 2. Our important finding is in separating the dataset into two subsets, where a majority of the label noise examples are captured by one of the subsets, i.e., the support vector

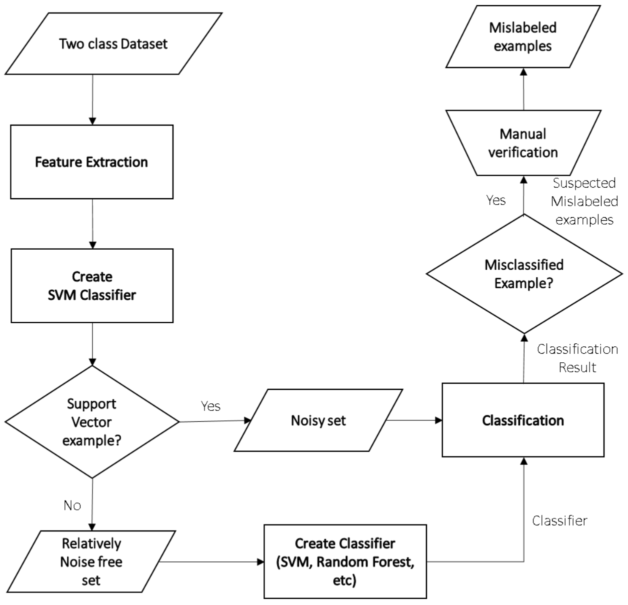

Fig. 2. Steps to find the mislabeled examples in the dataset.

examples subset. Any method can then be applied to efficiently target the mislabeled examples from this subset.

For feature extraction we explored the state of the art methods and selected the method in [17]. Recent results [1] show that deep neural network based methods perform well for image classification tasks, so we have used the ImageNet pretrained *GoogLeNet* convolutional neural network [17] model for feature extraction. The 1024 features extracted from the *GoogLeNet* were ranked using symmetric uncertainty [18] and only the top 200 features were used for subsequent processing. The selected features were scaled between -1 and 1 before training with the SVM classifier. The flowchart for the steps involved in finding the mislabeled examples in the ImageNet dataset with our method is shown in Figure 2.

III. Experiments

We conducted experiments with three datasets: ImageNet, UCI character recognition and MNIST digit recognition. First we discuss the results for the ImageNet dataset and then the results for the UCI character and MNIST digit datasets are discussed. All the SVM experiments were performed using the LIBSVM library [19] which implements the SMO-type optimization algorithm for SVM classification. The random forest experiments were perfomed using scikit-learn python machine learning library [20]. We used the pre-trained GPU implementation of the *GoogLeNet* model obtained from [23] for feature extraction. A 1024 dimensional feature vector was extracted from the average pooling layer "cls3_pool" of the *GoogLeNet* model and feature selection using symmetric uncertainty measure was done with Weka [22].

We did not experiment with different features for this paper, since the goal of this paper is only to show the potential of this method in finding label noise in large scale datasets. Experimentation with different features and algorithms is beyond the scope of this paper. The experiments were performed with linear kernel SVM and the cost parameter "C" is set to 1. It is well known that the number of support vectors changes significantly as the "C" and "γ" (for RBF kernel) values are changed. But the experimental results reported in [4] shows that change in "C" and "γ" does not affect the performance of finding label noise examples significantly for a broad range of values.

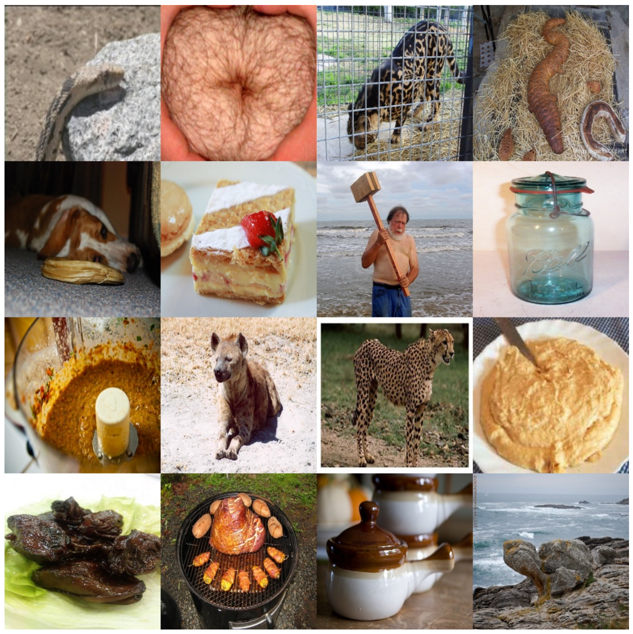

Fig. 3. Some of the found mislabeled images in the ImageNet dataset. The classes of the images from left to right and from top to bottom: alligator, bagel, cheetah, crocodile, donut, french bread, hammer, hatchet, italian bread, jackal, jaguar, ladle, oyster, plate, soup bowl, tortoise. All the mislabeled images and their image IDs can be obtained from https://drive.google.com/file/d/0B172WZL9tlsDejJRQmVTNFNZMW8

A. Imagenet Dataset

In our previous work we tested with the most confusing classes in the datasets. There is no such confusion matrix for the ImageNet database to our knowledge. So we selected some potentially confusing classes based on our knowledge from the literature and our intuition. First we found some of the hard classes from [1], then based on the intuition that objects that appear together and similar might confuse the feature extraction process, we selected the competing classes. For the initial experiment we first selected three hard classes: hatchet, ladle and oyster. Then using the same intuition we selected the respective competing classes: hammer, soup bowl and plate. Though oyster was not mentioned as one of the hard classes in [1], it was selected due to our initial observation (or confusion) that one of the images shown in [1] with the oyster label might be a mislabeled example. Using the intuition that the objects that appear similar might be confusing we selected the following class pairs: alligator vs crocodile, donut vs bagel, cheetah vs jaguar, french bread vs italian bread, turtle vs tortoise and wolf vs jackal.

There were 22951 examples in the 18 image classes mentioned above. From these examples our method with SVM and Random Forest classifiers in Step 2 selected 2690 and 3037 suspected examples respectively. Manually reviewing these examples results in the selection of 72 and 77 mislabeled examples respectively. Combining the suspected examples of both the SVM and Random Forest classifiers results in finding 92 mislabeled examples by reviewing 3607 examples. The details of the results are presented in Table I. Some of the found mislabeled images are shown in Figure 3.

The level of label noise found by our method is comparable (slightly more) to the reported value of 0.3%. The method with SVM classifier found 0.313% and random forest found 0.335%. Union or combination of the two methods found 0.4%. Compared with the manual verification of randomly sampled examples our method requires reviewing 9 times fewer examples. It should be noted that the result shown here is an underestimate of the total noise found in the ImageNet dataset. We have only targeted the examples which appear to be obviously wrong to us. For example: a snake image labeled as crocodile. We did count the examples which cannot be labeled correctly. For example: the image containing only the tail portion of an alligator or a crocodile. We acknowledge that there is a subjectivity here in selecting the mislabeled examples. But in contrast we would like to highlight that the intention of developing this method is to find such examples and present them to an expert, who will be able to correctly label them.

We also tested some non-confusing synsets: cheetah vs bagel, french bread vs jackal, alligator vs tortoise and jaguar vs wolf. Out of the four class pairs tested only "alligator vs tortoise" pair had two examples selected for review and both the examples were correctly labeled. It shows that the class pairs need to be confusing for our method to find label noise examples. To find the mislabeled examples in a dataset with n classes we need to create between n/2 and n class pairs.

B. Character Recognition Datasets

In our previous work [3], [4] we have only considered a minimum noise level of about 10%. Since the amount of noise in the ImageNet dataset is very low projected to be about 0.3%, it will raise doubts if no noise is found in any of the classes, for example in the "Turtle" and "Wolf" classes. In order to verify whether our method works at low noise levels, experiments were conducted with the MNIST digit recognition and UCI character recognition datasets at the 0.3% noise level similar to the projected noise level for the ImageNet dataset. The results from this experiment can provide some confidence about the amount of noise found in the ImageNet dataset.

The most confusing digits 4, 7 and 9 from the MNIST digit recognition dataset and the characters B, H and R from the UCI character recognition dataset were used in the experiment. Six experiments were conducted for both the datasets and each experiment was repeated 30 times with different random examples. The experiments in the MNIST dataset were between the digits 4 and 7, 4 and 9, and, 7 and 9. The experiments in the UCI dataset were between the characters B and H, B and R, and, H and R. One thousand examples from each class were randomly sampled from the MNIST dataset and labels of three random examples were flipped. Five hundered examples from each class were randomly sampled from the UCI character dataset and labels of the two random examples were flipped.

The MNIST digits were represented by a 784 dimensional feature vector obtained from the pixel values of the digit images. The UCI character examples were represented by a 16 dimensional feature vector provided with the dataset. We experimented with linear and RBF kernels. The RBF kernel parameter "γ" is set to 1/(number of features). For both the kernels the SVM cost parameter "C" is set to 1. The results of this experiment is presented in Table II.

From the results in Table II, it can be observed that our method is able to remove almost all the mislabeled examples from the MNIST digit and UCI character recognition datasets. Out of the total 90 experiments for the MNIST digit dataset all the mislabeled examples were removed in 66 and 62 experiments with linear and RBF kernels respectively. For the UCI character dataset all the mislabeled examples were removed in 63 experiments with both the linear and RBF kernels. These results suggest the possibility of our method finding most (if not all) of the mislabeled examples in the ImageNet dataset for the tested classes.

IV. Conclusion

A method to find the mislabeled examples in very large data sets was discussed. We showed that on the ImageNet dataset 92 mislabeled examples, that were previously unknown, were found in 18 of the image classes. The proposed method requires review of up to nine times fewer examples to find the same number of mislabeled examples compared to randomly selecting examples for review as done during the study conducted while building the dataset. The results show that the proposed method is a focused method for finding mislabeled examples in large datasets.

References

[1] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M. and Berg, A.C.: Imagenet large scale visual recognition challenge. International Journal of Computer Vision, 115(3), pp.211-252 (2015)

[2] Deng, J., Dong, W., Socher, R., Li, L.J., Li, K. and Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In IEEE Conference on Computer Vision and Pattern Recognition, pp. 248-255 (2009)

[3] Fefilatyev, S., Shreve, M., Kramer, K., Hall, L., Goldgof, D., Kasturi, R., Daly, K., Remsen, A. and Bunke, H.: Label-noise reduction with support vector machines. In IEEE International Conference on Pattern Recognition. pp. 3504-3508. (2012)

[4] Ekambaram, R., Fefilatyev, S., Shreve, M., Kramer, K., Hall, L.O., Goldgof, D.B. and Kasturi, R.: Active cleaning of label noise. Pattern Recognition, 51, pp.463-480. (2016)

[5] Brodley, C.E. and Friedl, M.A.: Identifying mislabeled training data. Journal of Artificial Intelligence Research, 11, pp.131-167. (1999)

[6] Zhu, X. and Wu, X.: Class noise vs. attribute noise: A quantitative study. Artificial Intelligence Review, 22(3), pp.177-210 (2004)

[7] Breiman, L.: Randomizing outputs to increase prediction accuracy. Machine Learning, 40(3), pp.229-242. (2000)

[8] Martnez-Muoz, G., Snchez-Martnez, A., Hernndez-Lobato, D. and Surez, A.: Class-switching neural network ensembles. Neurocomputing, 71(13), pp.2521-2528. (2008)

TABLE I
LABEL NOISE EXPERIMENT RESULTS ON IMAGENET DATASET

Class	Total # examples	# SV	SVM		Random Forest		Combined	
			# examples reviewed	# mislabeled examples found	# examples reviewed	# mislabeled examples found	# examples reviewed	# mislabeled examples found
Hatchet	845	204	144	3	199	3	204	3
Hammer	1382	83	71	2	43	2	83	3
Plate	1236	45	31	2	31	1	45	2
Oyster	827	80	53	1	69	1	80	1
Soup bowl	1371	71	50	4	60	2	71	4
Ladle	1810	116	81	1	106	1	116	1
Alligator	1346	389	309	6	328	8	389	10
Crocodile	1322	391	310	10	337	14	391	18
Donut	1314	324	217	4	267	5	324	5
Bagel	1277	316	219	5	255	4	316	5
Cheetah	1424	50	47	10	39	9	50	10
Jaguar	1512	59	40	12	54	12	59	12
French bread	1279	335	282	5	223	3	335	5
Italian bread	967	575	410	0	537	4	575	4
Turtle	1209	219	145	0	200	0	219	0
Tortoise	1221	236	203	3	189	3	236	3
Wolf	1390	63	36	0	59	0	63	0
Jackal	1219	51	42	4	41	5	51	6
Cumulative	**22951**	**3607**	**2690**	**72**	**3037**	**77**	**3607**	**92**

TABLE II
LABEL NOISE EXPERIMENT RESULTS ON MNIST AND UCI DATASETS

Kernel	Dataset	Total # examples	# mislabeled examples	Average # examples reviewed	Average # of mislabeled examples found
Linear	MNIST	2000	6	65.3	5.7
Linear	UCI	1000	4	84.6	3.7
RBF	MNIST	2000	6	73.1	5.1
RBF	UCI	1000	4	98	3.7

[9] Frnay, B. and Verleysen, M.: Classification in the presence of label noise: a survey. IEEE transactions on neural networks and learning systems, 25(5), pp.845-869. (2014)

[10] Angelova, A., Abu-Mostafam, Y. and Perona, P.: Pruning training sets for learning of object categories. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Vol. 1, pp. 494-501). (2005)

[11] Thongkam, J., Xu, G., Zhang, Y. and Huang, F.: Support vector machine for outlier detection in breast cancer survivability prediction. In Asia-Pacific Web Conference (pp. 99-109). Springer Berlin Heidelberg. (2008)

[12] Segata, N., Blanzieri, E., Delany, S.J. and Cunningham, P.: Noise reduction for instance-based learning with a local maximal margin approach. Journal of Intelligent Information Systems, 35(2), pp.301-331 (2010)

[13] Verbaeten, S.: Identifying mislabeled training examples in ILP classification problems. In Proceedings of twelfth Belgian-Dutch conference on machine learning, pp. 1-8 (2002)

[14] Li, Y., Wei, F., Ekambaram, R., Xinming, Ou, Hall, L.: Zero-Day Android Malware Detection Through Label Noise Identication. Submitted for publication.

[15] Cortes, C. and Vapnik, V.: Support-vector networks. Machine learning, 20(3), pp.273-297 (1995)

[16] Ekambaram, R., Goldgof B.D., O., Hall, L.: Finding Random Label Noise with SVMs. Submitted for publication.

[17] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. and Rabinovich, A.: Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 1-9) (2015)

[18] Hall, M., Witten, I. and Frank, E.: Data mining: Practical machine learning tools and techniques. Kaufmann, Burlington. (2011)

[19] Chang, C.C. and Lin, C.J.: LIBSVM: a library for support vector machines. ACM Transactions on Intelligent Systems and Technology (TIST), 2(3), p.27. (2011)

[20] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V. and Vanderplas, J.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12(Oct), pp.2825-2830 (2011)

[21] Vedaldi, A. and Lenc, K.: Matconvnet: Convolutional neural networks for matlab. In Proceedings of the 23rd ACM international conference on Multimedia pp. 689-692 (2015)

[22] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P. and Witten, I.H.: The WEKA data mining software: an update. ACM SIGKDD explorations newsletter, 11(1), pp.10-18. (2009)

[23] Pretrained GoogLeNet Model, http://www.vlfeat.org/matconvnet/pretrained