Classification of natural landmarks and human footprints of Amazon using satellite data

Kaifeng Chen
Department of Applied Physics
kfchen@stanford.edu

Yao Zhou
Department of Material Science and Engineering
yaozhou@stanford.edu

Abstract

Amazon satellite image data, provided by Kaggle, contains image classification results with in total 17 *classes. Different from simple multi-class classification problems, here one image contains complicated features and thus can belong to multiple classes. In this project, we train the classifier using three different algorithms: (1) using multi-class Support Vector Machine (SVM) as the baseline, (2) deep learning using Convolutional Neural Network (CNN), and (3) transfer learning using VGG-16. We explain these algorithms in detail, and show the accuracies as well as* F_2 *scores obtained from the three algorithms. We further highlight a validation* F_2 *score of* 0.9177 *with VGG-16 and parameter fine tuning.*

1. Introduction

The understanding of natural landmarks and human footprints of Amazon is of great importance for the preservation of the forest and habitat of Amazon. It will help us to find the source of deforestation and further provide us with information to make response. Such understanding requires not only the knowledge of the nature itself, but also the allocation of natural resources and the effects of human activities.

Previous research typically uses coarse resolution images to study changes in rainforest, which is not enough for small-scale deforestation. Now we are provided with a high resolution dataset of the whole Amazon landform from the satellite thanks to *Planet* [1], who collects Amazon land surface imagery with the world's largest constellation of Earth-imaging satellites. It is useful to classify the different regions of Amazon with correct labels.

This is a subset of the general classification problem, which has been massively discussed in the context of not only image processing [4], but also machine learning and even deep learning [6, 8]. Methods either suitable for general classification problems such as Support Vector Machine (SVM), or specifically designed for image related problems such as Convolutional Neural Network (CNN), can be employed directly. However, different from the traditional binary classification problem, here one image from the satellite can belong to multiple classes. Therefore, to correctly classify each image becomes challenging.

In our project, we implement three algorithms to tackle this problem: (1) a simple SVM classifier as the baseline, (2) using a commonly used CNN structure and (3) transfer learning combining pretrained VGG-16 [7] and CNN. The report is thus organized as a follows. We present some related work in Section. 2. In Section. 3, we discuss the dataset provided by Kaggle. In Section. 4, the three algorithms are discussed in detail. We show the results using the three algorithms in Section. 5. Section. 6 summarizes the results and points to future improvements.

2. Related Work

Multi-label classification problem is important in many research fields and has been discussed widely in the literature. For example in the field of bioinformatics, various state-of-the-art classification methods have been used on gene expression datasets in order to correctly classify tissues [5]. In addition to the field of geography, multi-class SVMs has been adopted to classify the airborne sensor data, in conjunction with a few other methods such as discriminant analysis, decision tree, and multilayer perceptron neural network [3].

In addition to traditional multi-label classification methods, recent advances in deep learning give rise to the prevailing application of deep neural networks. For example, people have built flexible deep CNN infrastructure, called Hypotheses-CNN-Pooling (HCP) to obtain much better improved classification accuracy [10]. Moreover, more algorithms such as combining SVM with deep learning [9] and deep learning for large-scale data [2] were also proposed in the past few years.

3. Dataset

The dataset provided by Kaggle [1] is of in total 17 classes, and these classes address different aspects of the image content, for example landscape, weather and habitation. In Fig. 1 we show a few examples of the images in the dataset. Each image is of size $256 \times 256 \times 3$, and is labeled with one or many classes. In order to feed the data to SVM, we first binarize the classes.

Figure 1. A few examples of the data. Each image is labeled with one or many classes.

In the dataset, the labels are not evenly distributed. We show the distribution of labels in Fig. 2. For example, the classes "blow_down" and "conventional_mine" have much lower data compared to the classes "primary" and "clear". The imbalance of the data results in further difficulties in the classification. Ideally, in dealing with classes where the data are not evenly distributed, it would be better to introduce weights for different classes in training a classifier. Here in our report, we did not implement this extension and will mention it in the future work.

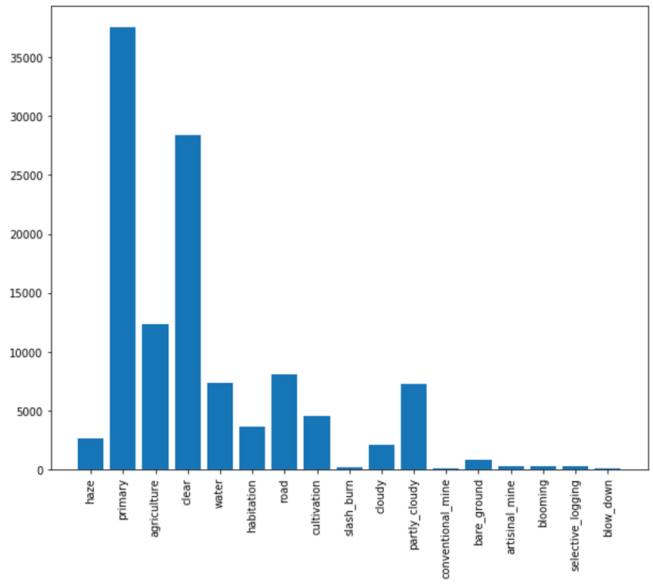

Figure 2. Distribution of labels.

We split the total 40479 images into training dataset (80%) and validation dataset (20%), and report the accuracies and F_2 scores in the validation dataset.

4. Algorithms

In this section, we discuss the details of the three algorithms implemented for the classification problem: support vector machine, convolutional neural network and transfer learning with VGG-16.

1. Support Vector Machine

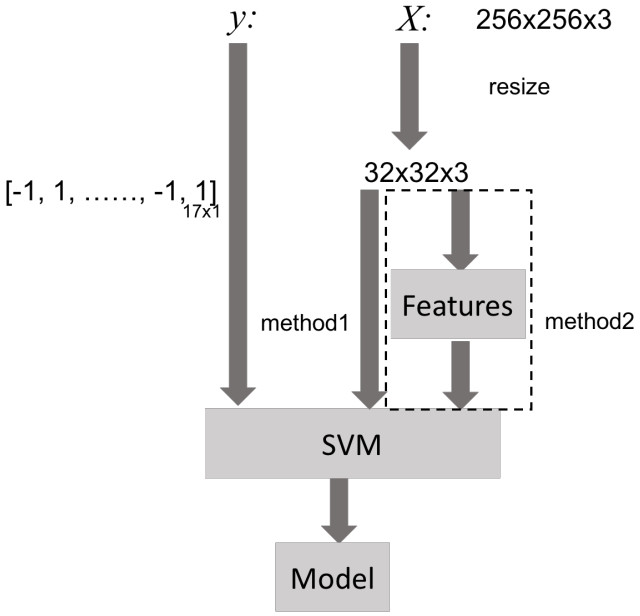

Figure 3. Pipline of the data reprocessing and model training. Each image is resized from its original size $256 \times 256 \times 3$ to $32 \times 32 \times 3$, and vectorized to a vector with length 3027. In the naive SVM implementation (method 1), the resulting vector is the input X. In the SVM with features (method 2), the color histogram and hog features are concatenated to be X. The label of an image is vectorized to a 17×1 vector, with -1 and 1 indicators indicating the class that the image belongs to.

Fig. 3 shows the pipeline of SVM. To reduce the computational cost, we first resize each image to size 32×32 and subtract the mean image. We also implemented two methods: method 1 is to use the RGB values directly as the input X, and method 2 is to use the color histogram and hog features as X. For the class labels, we binarize it to a vector with dimension 17×1. For example, since the supported classes are **haze, primary, agriculture, clear, water, habitation, road, cultivation, slash_burn, cloudy, partly_cloudy, conventional_mine, bare_ground, artisinal_mine, blooming, selective_logging, blow_down**. The label first image in Fig. 1 thus can be expressed by a vector with the first and second value being 1 and rest being -1.

$$L = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C}\max(0, 1-y_{ij}x_i w_j)+\lambda||W||^2, y_{ij} \in \{-1, 1\} \tag{1}$$

To obtain the model, one can either train the SVM on the 17 classes independently, or train all the classes simultaneously. Here we implemented both ways. For the first case, we directly employ the SVM code from assignment1 and adjust the loss function to our case as in Eq.1, and for the second method we use the multi-class SVM in package `sklearn`. In principle, these two methods should give the same results since both of them assume no correlation between different classes.

2. Convolutional Neural Network

We implemented a convolutional neural network with smaller capacity than the VGG-16 that will be discussed in the next section. We used eleven convolutional or fully connected layers in total. For faster computation, we first resized our input images to $32 \times 32 \times 3$. We also subtracted the mean image to ensure zero center. In Fig. 4, we show a flowchart of our architecture in sequence. In detail, as shown in Fig. 5, we have a $32 \times 32 \times 3$ input image, two CONV3-32 layers, a MAX POOL2 layer, two CONV3-64 layers, a MAX POOL2 layer, two CONV3-128 layers, a MAX POOL2 layer, a CONV3-256 layer, a MAX POOL2 layer, a CONV3-512 layer, a MAX POOL2 layer, a FC512 layer, a FC256 layer and a FC17 layer for final classification. We used ReLu activation function for each layer. To improve gradient flow, we used batch normalization. For regularization, we used dropout with rate 0.5. We used Adam optimizer for gradient update.

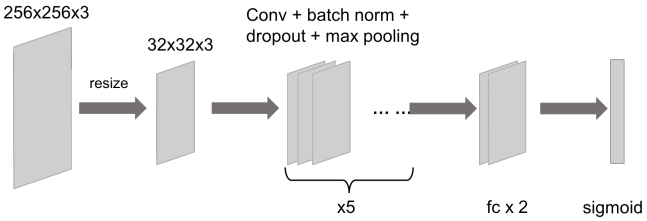

Figure 4. Convolutional neural network. Each image is resized from its original size $256 \times 256 \times 3$ to $32 \times 32 \times 3$. Then multiple commonly used CNN layers are stacked to extract features from the resized image. Finally the model has two fully connected layers and a sigmoid layer to output class labels.

In the end, we used a sigmoid cross entropy loss function, as given by:

Figure 5. Convolutional neural network architecture. Eleven convolutional or fully connected layers are used in total. We use ReLu activation function, batch normalization and dropout for each convolutional or fully connected layers.

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij}\log\sigma(s_{ij}) + (1-y_{ij})\log(1-\sigma(s_{ij})) \tag{2}$$

where $\sigma(s_{ij})$ is the sigmoid function, N is the number of images and C is the number of classes.

3. Transfer Learning with VGG-16

Transfer learning is a method in machine learning that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem. Here to fully utilize the power of existing well-known models, we implemented transfer learning with VGG-16 with its architecture shown in Fig. 7. VGG-16 has larger capacity and more parameters than the convolutional neural network implemented in the algorithm discussed above. Its architecture is shown in Fig. 6. Here to fit our needs, we modify the last

fully-connected layer to be dimension $1 \times 1 \times 17$.

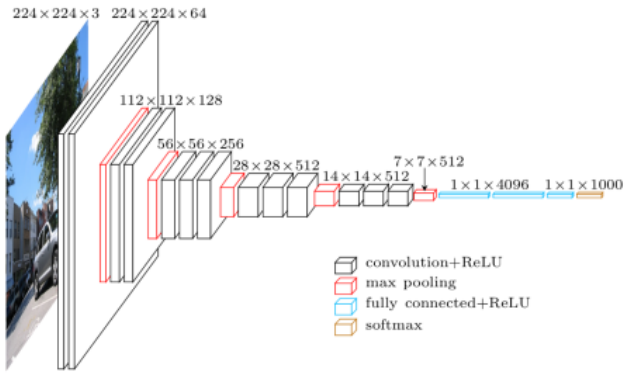

Figure 6. The architecture of VGG-16.

Because of the complexity of the model, the training process takes longer time but could result in higher accuracy. Also, the input images of VGG-16 need to be the size of $224 \times 224 \times 3$ to accommodate the input layer of VGG-16.

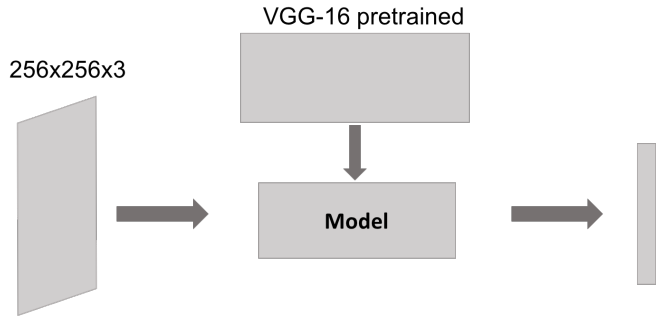

Figure 7. Transfer learning with VGG-16. The images are resized to $224 \times 224 \times 3$ for the purpose of using pretrained VGG-16 model. Then we directly use the weights from VGG-16 and compute the label for our images.

First, we preprocessed the input images. During training, we cropped the input images to $224 \times 224 \times 3$ randomly. We also randomly flipped images left to right. Then to center our images, each image subtracted the mean image of VGG-16. During validation, we centrally cropped the input images to $224 \times 224 \times 3$ and also subtracted the mean image of VGG-16.

Next, we restored only the layers up to FC7 with pretrained weights of VGG-16. We initialized operation from scratch for the new FC8 layer with 17 output classes. We used a sigmoid cross entropy loss function, as explicitly given in Eq. 2. Finally, we trained only the reinitialized last layer FC8 for five epochs, and then we fine-tuned the entire VGG-16 net for another five epochs. Note that we could continue to run for more epochs to further improve our model training and validation accuracy and F_2 score. But due to the slow training process, we ran for five plus five epochs here.

5. Results

In this section, we discuss the results obtained using the above three algorithms. In addition to accuracy, another important feature to characterize effectiveness of the classifier is the F_2 score. In general, the F score measures accuracy using the precision p and recall r, defined as

$$(1+\beta^2)\frac{pr}{\beta^2 p + r} \tag{3}$$

where p is ratio of true positives (tp) to all predicted positives ($tp + fp$), r is the ratio of tp to all actual positives ($tp + fn$):

$$p = \frac{tp}{tp + fp} \tag{4}$$

$$r = \frac{tp}{tp + fn} \tag{5}$$

For the F_2 score, we have $\beta = 2$. Therefore, the mean F_2 score is then the average of individual F_2 scores for each row in the dataset. Note that the F_2 score weighs recall higher than precision.

5.1. Baseline: SVM

In Table 1, we show the validation accuracy as well as the average F_2 score. The validation accuracy increases slightly with extracted features.

	validation	F_2 score
binary SVM	0.3365	0.6465
binary SVM (features)	0.3491	0.6770

Table 1. Validation accuracy with all labels being classified correctly, and the average F_2 score for the validation dataset.

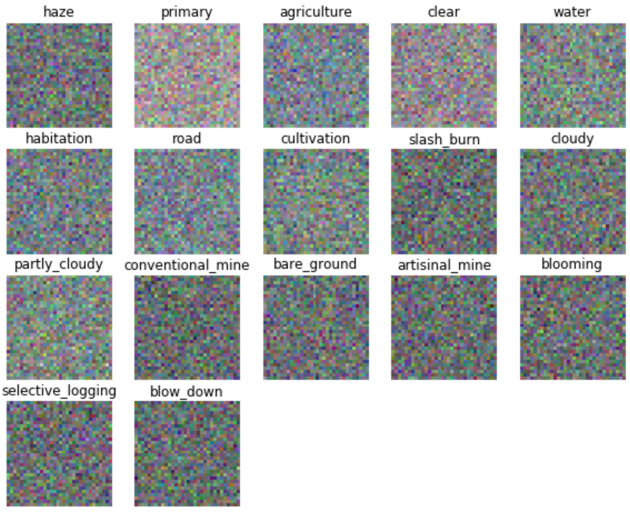

Figure 8. Visualization of learned weights for each class label.

In Fig. 8, we visualize the weights for each class in the obtained classifier. One thing to notice is that the weights are mostly filled with noises due to the relatively low accuracy. However, the color of the weights indicate some aspect of the class. For example "cloudy" is slightly darker than "partly_cloudy".

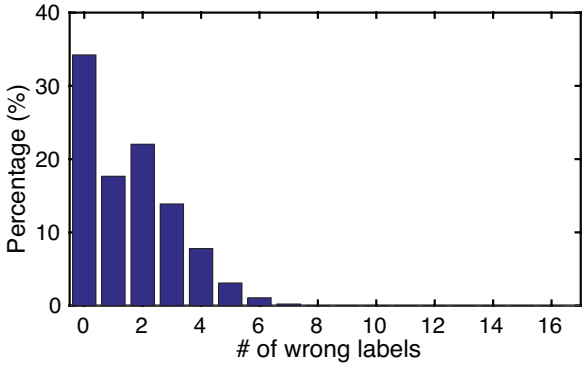

Figure 9. Distribution of number of wrong labels for the validation dataset with binary SVM.

In Fig. 9, we show the distribution of number of wrong labels for the validation dataset predicted by the binary SVM. We find that the number of wrong labels among all 17 labels is mostly smaller than 6, indicating an already good performance of the classifier.

5.2. Convolutional neural network

Using the "light-weight" convolutional neural network results in the following validation accuracies and F_2 score

	validation	F_2 score
CNN	0.4134	0.8520

Table 2. Validation accuracy with all labels being classified correctly, and the average F_2 score for the validation dataset.

Note that we are using the F_2 score to evaluate our model here. And the F_2 score is known to weigh recall higher than precision. Therefore, in order to improve the F_2 score, we lowered the classification threshold after the sigmoid function. We tuned the threshold value in the range of 0 to 0.5 to find the optimal threshold value 0.2. Here, we used the same threshold 0.2 across all classes.

Comparing to the SVM model, we directly observe the increase of both accuracy and the F_2 score. In fact, these are due to the complications of the CNN model, where the boundaries between each class becomes significantly nonlinear. To better assess the model, we start with a noisy image and let the image evolve by setting the target class as one of the 17 classes, we obtain the following visualization of class labels.

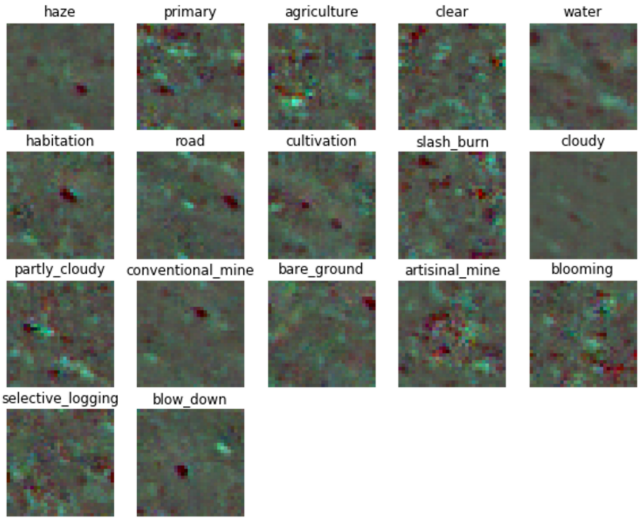

Figure 10. Class label visualization using CNN.

Comparing to the weights obtained using SVM, here we can clearly see some features that are unique to some classes. For example, "cloudy" is more uniform in color compared to "partly_cloudy", and one can directly see the shape of rivers in the class of "water". We note that using saliency map will achieve similar results, and these extracted features are the reasons for a higher accuracy and F_2 score.

5.3. Transfer learning with VGG-16

In combination with VGG-16, we obtain the accuracy and the mean F_2 score for the validation dataset as the following.

	validation	F_2 score
CNN with VGG-16	0.5488	0.9177

Table 3. Validation accuracy with all labels being classified correctly, and the average F_2 score for the validation dataset.

Here we see a further increase in both validation accuracy and the F_2 score, and this F_2 score is within the top 80 scores in Kaggle leaderboard. In order to examine the reason of such increase, we show the accuracies for each individual class in Fig. 11.

For those rare classes, the classification accuracies are high enough even using SVM. However, for classes that are prevailing in the dataset, their accuracies start with low values in SVM and thus will be the bottleneck for higher accuracy and F_2 score. By switching the model to CNN, we directly observe the increase of accuracy in these major classes, and finally using transfer learning, these accuracies are further boosted without any significant change in accuracies for the rare classes. Therefore, using a compli-

	SVM	SVM (features)	CNN	CNN (VGG16)
haze	0.9350	0.9350	0.9321	0.9526
primary	0.9307	0.9316	0.9527	0.9667
agriculture	0.6948	0.7294	0.8125	0.8829
clear	0.7020	0.7227	0.8714	0.9405
water	0.8221	0.8223	0.6950	0.8943
habitation	0.9119	0.9119	0.8489	0.9377
road	0.8005	0.8109	0.7720	0.8961
cultivation	0.8871	0.8871	0.8531	0.8697
slash_burn	0.9942	0.9942	0.9938	0.9941
cloudy	0.9520	0.9527	0.9704	0.9718
partly_cloudy	0.8150	0.8179	0.9171	0.9597
conventional_mine	0.9979	0.9979	0.9979	0.9981
bare_ground	0.9758	0.9758	0.9562	0.9717
artisinal_mine	0.9915	0.9915	0.9894	0.9970
blooming	0.9918	0.9918	0.9918	0.9907
selective_logging	0.9911	0.9911	0.9911	0.9902
blow_down	0.9975	0.9975	0.9975	0.9975

Figure 11. The validation accuracies by class for the three algorithms discussed above.

cated model really can help better classifying the classes with large amount of data, and hence improve the overall validation accuracy and the F_2 score.

For better comparison purpose, we summarize our main results in Fig. 12. We want to highlight the increase of the total accuracy and the F_2 score as the model gets more sophisticated.

	SVM	SVM (features)	CNN	CNN (VGG16)
Accuracy (all labels correct)	0.3365	0.3491	0.4134	0.5488
F_2 score	0.6465	0.6770	0.8520	0.9177

Figure 12. A summary of the total validation accuracies and F_2 scores for the three algorithms discussed above.

6. Conclusion and future work

In this project, we target at a multi-label classification problem where Amazon images obtained from satellite belong to in total 17 classes, and each image can have one or many labels. We tried three methods to train a classifier: using a standard SVM and adjusting it for multi-classification, and using simple CNN architecture for label prediction and using a more complicated transfer learning model with pretrained VGG-16. We present in detail the results from the three algorithms. Results from SVM, as the baseline of the project, can already achieve high classification accuracy and F_2 score. With deep learning approach from CNN to transfer learning, we further observe significantly increasing validation accuracy and the F_2 score. We analyze the reason for such increase, and highlight the best F_2 score with a value of 0.9177 using transfer learning with VGG-16.

For future work, a few improvements can be made by transfer learning with more sophisticated models, such as ResNet and InceptionNet. Moreover, one can do ensemble algorithm by taking advantages of existing pretrained models to obtain even higher F_2 scores. However, the method is essentially the same as introduced in our report. Another possible direction for future improvement is to solve the problem of data imbalance. This can be either done by augmenting the classes with few occurrences in the overall dataset, or introducing weights in the loss function defined in Eq. 2. In addition, we can also tune the threshold for classification after the sigmoid function for each class individually to further improve the F_2 score, since the F_2 score weighs recall higher than precision.

We would like to thank the TA for useful discussions of our project.

References

[1] Planet: Understanding the amazon from space.

[2] Z. Akata, F. Perronnin, Z. Harchaoui, and C. Schmid. Good practice in large-scale learning for image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(3):507–520, March 2014.

[3] G. M. Foody and A. Mathur. A relative evaluation of multiclass image classification by support vector machines. *IEEE Transactions on Geoscience and Remote Sensing*, 42(6):1335–1343, June 2004.

[4] R. M. Haralick, K. Shanmugam, and I. Dinstein. Textural features for image classification. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-3(6):610–621, Nov 1973.

[5] T. Li, C. Zhang, and M. Ogihara. A comparative study of feature selection and multiclass classification methods for tissue classification based on gene expression. *Bioinformatics*, 20(15):2429, 2004.

[6] A. Mathur and G. M. Foody. Multiclass and binary svm classification: Implications for training and classification users. *IEEE Geoscience and Remote Sensing Letters*, 5(2):241–245, April 2008.

[7] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. April 2014.

[8] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. The german traffic sign recognition benchmark: A multi-class classification competition. In *The 2011 International Joint Conference on Neural Networks*, pages 1453–1460, July 2011.

[9] Y. Tang. Deep learning using linear support vector machines. Feb. 2015.

[10] J. H. B. N. J. D. Y. Z. Yunchao Wei, Wei Xia and S. Yan. Cnn: Single-label to multi-label. July 2014.