XGBoost and CNN Applications on Multi-label Image Classification

Ke Chen
University of Waterloo
Electronic and Computer Engineering (MEng)
ID: 20456026
Email: kchen@uwaterloo.ca

***Abstract*—The techniques of single-label image classification have experienced a rapid development following with the florescence of traditional classifier combining feature extractions and emergence of Convolutional Neural Network. Both of these two techniques have achieved a great success on image recognition. However, a more generic task referred as multi-label-multi-class image classification is more practical and versatile than single-label image classification in real world situation. This paper investigates a particular traditional classifier based technique XGBoost and a deep learning technique CNN in terms of handling a multi-label image classification task. Based on a targeted experiment, I compare the difference and evaluate the performance of these two techniques. In addition, utilizations of SIFT, RGB channels and Sobel feature extraction techniques are involved. The implementation process, label correlations and various multi-label evaluation matrices will be discussed as well.**

I. INTRODUCTION

The techniques in processing large-scale image classification have developed rapidly in recent years. With extensive use of digit images, digital database has been taken great advantage in various fields of applications which extract information from digit images and explore opportunities to serve the society. Thereby, the high demand of such digital information stimulates research on image recognition. A majority of researches focus on single-label-multi-class image recognition. In real world, however, images are usually annotated with more than one label. We need a more generic solution that can be utilized for image recognition in a versatile way. As a result, multi-label-multi-class classification attracts many data scientists attention on handling general machine learning tasks [13] [4] [13] [11]. In this paper, I aim to construct an experiment to analyze the characteristics of multi-label task and compare it to single-label task. Moreover, this paper will discuss the limitations in implementing a multi-label task, such as multi-label correlation [6] [8], multi-label evaluation [18] [6] and label transformation [6].

The single-label task experienced a series of revolutions. Based on its history, there are SIFT based sparse coding [9], RGB-D Image view[[15], and applications of classical classifier such as SVM [24], Random Forest [25]. More significantly, after 2012 with the emergence of a famous network AlexNet [1], innovations of deep neural network, Convolutional Neural Network (CNN) broke through the bottleneck of image recognition studies. Single-label task has been dominated by CNN system since CNN achieved a great success since 2012.

In this paper, I will compare two image classification techniques in terms of multi-label task. One is chosen from a classic classifier based algorithm, XGBoost (published in 2014) [16]. Even though XGBoost is a relatively new algorithm, it is still based on traditional gradient boosting and utilizing tree structures. Nowadays, XGBoost is wildly used given its good performance and scalability [16]. In order to compare with classic classifier performance, I choose CNN as my second candidate given its recent popularity in image recognition. Its convolutional computation can extract important features effectively from digit images. As a result, it is essential to choose CNN as a comparison object in multi-label image classification. Under the same hardware platform, I analyze the structures and evaluate the performance of those two techniques in a multi-label experiment.

In the implementation section, I conduct exploratory data analysis in order to provide a multi-label dataset visualization with summarized information. This visualization process gives a direction to future implementations. Moreover, to make CNN system concrete, I develop label encoding, image standardization, and multiple modelling processes and discuss layer selection, parameter determination and epochs tuning process. For XGBoost, I employ Scale-Invariant Feature Transform(SIFT), RBG feature distributions and Sobel method to complete the feature extraction step, and continue to apply XGBoost classifier to train the fitted model. In addition, there is a discussion about how to select hyper-parameters to control the overfitting problem. In the overall evaluation part, I implement three multi-label evaluation matrices [6] including Exact Match Ratio, Hamming Loss and modified F-measure [29] . After this self-designed experiment under a restricted hardware platform, the traditional classifier based XGBoost algorithm performs even better than CNN in a multi-label image classification task.

II. LITERATURE REVIEW

Single-label image classification has achieved a great success, and many scientists and researchers did a lot of attempts and made numbers of achievements in the image classification study area.

The performance of traditional classifiers highly relies on feature extraction process. For feature extractions, there are a few techniques for key points detection, edge-corner detection, and color detection. Scale-Invariant Feature Transform(SIFT) [21] is one of the most common techniques people used to employ; in addition, the new idea that applying sparse coding which to learn dictionary of basis functions from SIFT has risen [9]. Moreover, ORB descriptors in the bag of word model [15] combines k-means [26] algorithm was introduced in image retrieval. In addition, Colour is a significant characteristic of images [17], many techniques aim to obtain color features were raised. In image classification process, RGB channels usually represent special patterns which are helpful in feature extraction process [14]. In addition, edge and corner also provide the powerful information in prediction [22]; thus, Sobel edge detector [19] is one of the commonly used methods. Sobel Edge detection uses one 3x3 kernel to calculate the horizontal changes, and it uses the other 3x3 kernel to derive the vertical changes, and the techniques combine Sobel edge detection and soft-threshold wavelet denoising was proposed to deal with the noise images [23].

Before 2012, the mainstream techniques on image classification were traditional classifiers such as support vector machine (SVM) [24] and random forest(RF) [25]. Moreover, SVM with spatial pyramid matching kernel [7] had an excellent performance on image recognition. However, the image classification development broke a bottleneck in 2012 by Convolutional Neural Network (CNN). In 2012, AlexNet [1] was proposed and won the 2012 ImageNet LSVRC competitions. AlexNet utilized GPU-net and eight layer architectures in including five convolutional layers and three fully-connected layers. Also, it took advantages of applying dropout [27] to reduce the overfitting. After 2012, CNN became wildly used in image classification studies and had great successes. Then VGG [10], GoogleNet [20] and ResNet[[28] were developed rapidly in improving the performance of CNN.

The studies of multi-label image classification were not as popular as single-label tasks; however, many scientists paid attention to developing suitable techniques on multi-label tasks. Multi-label Linear Discriminant Analysis (MLDA) [8] was proposed in 2010 to utilized the advantages of correlations between labels in classification. Also, many researchers focused on neural network deep learning in solving multi-label classification problems. Hypotheses-CNN-Pooling (HCP) [12] was developed as a flexible CNN system typically on solving multi-label tasks. Moreover, combining the probabilistic latent semantic analysis(PLSA) and milt-label multiple kernel learning [3] was come up with to develop the effective approach on multi-label tasks. The deep neural network which develops both semantic and spatial relations [4] between labels, and the proposed Spatial regularization Network (SRN) [4] was to discover the hidden relations between labels. Furthermore, weakly-supervised system [5] for multi-label image classification was developed with matrix completion. In 2016, CNN-RNN [11] system was created to solve the multi-label task. As for multi-label evaluations [18], researchers also investigated some concepts on result measurement.

III. Methodology

This paper aims to compare and analyze two different approaches on solving multi-label image classification problems. Based on the literature review, we have the insight that the methodology of implementing image recognition has two main directions: one is traditional approach such as a combination of feature extraction and traditional modelling algorithms; the other approach is convolutional neural network which has recently achieved a great success on applications of image recognition. In this paper, I utilized two recently proposed algorithms including XGBoost and convolutional neural network as my primary experimental approaches.

A. XGBoost

XGBoost was initially published in 2014 and proposed based on a scalable tree boosting system [16]. Overview XGBoost, the system provides state-of-the-art solutions on many different aspects of real-world applications. Moreover, it won the Higgs Machine Learning Challenge after which XGBoost has been wildly used in multiple types of machine learning tasks. The reason I choose XGBoost as an example of traditional approach is due to its outstanding scalability on ML studies. On the other hand, it is rare to see a combination between XGBoost algorithm applied on multi-label image classification task. As a result, I think it worth to investigate XGBoost algorithm and compare its performance with that of convolutional neural network in terms of solving multi-label image classification tasks.

A further look of XGBoost, it is a gradient boosting framework, and provides parallel processing in computation which is much faster than existing popular solutions on a single machine. The out-of-core computation offers users a more flexible usage so that people can process huge data even on personal desktop. Moreover, XGBoost has a powerful structure on tree pruning. It utilizes cross-validation in its inner design which helps to select the optimal boosting iterations and to avoid overfitting problems.

In order to effectively implement XGBoost algorithm, it is necessary to analyze its parameters. There are three kinds of parameters: general parameters, booster parameters, and learning task parameters. More specifically, general parameter is used to define overall functionality; booster parameters model every step for individual booster; and learning task parameter is used to optimize the overall performance. In XGBoost classifier implementation, correctly defining the booster parameters plays an important role in avoiding overfitting and facilitating overall evaluation. For instance, a parameter called max_depth is used to control overfitting by restricting the maximum depth of trees so that a model is forced to learn the optimal degrees of details for each sample. Another parameter called Learning_rate is used to incorporate robustness into a model by reducing the weights on every step of building tree structures.

B. Convolutional Neural Network

Convolutional Neural Network (CNN) has experienced an incredible success on large-scale image recognition since 2012. In that year, AlexNet [1] won the ImageNet LSVRC competitions, which is the start point that CNN became the trend of image processing technique. With the development of hardware and evolution of CNN's structure, both performance and training speed of CNN have been significantly improved. As a deep learning neural network, CNN generally consists of four types of layers: convolutional layer, ReLU layer, pooling layer and fully connected layer. Let us look into the functionality of those layers respectively. Firstly, the main characteristic of convolutional layers is that CNN employs convolutional computation to extract image features instead of applying connected weights between input and output neurons. There are many activation functions such as sigmoid and Gaussian. As its name suggests, ReLU layers apply the most frequently used activation functions ReLU. Selecting the most suitable activation function has big influence on overall performance of training process. Finally, Pooling layer is to reduce the spatial size and the fully connected layer is to complete the high-level reasoning process.

When employing CNN in image classification, it is important to design a suitable structure. A complex structure is computationally demanding in that it requires a high-level GPU hardware. Moreover, in designing the convolutional layers we need to take care of the embedding parameters. For instance, filters defines the output space of current convolutional layer; and kernel_size determines the width and height of 2D convolution window. In order to design a feasible CNN structure, we expect professional experience and numerous attempts, which makes building CNN model challenging.

C. Challenges

Here lists some challenges in a multi-label image classification task:

1. Multi-object leaning

Not like single object detection, in multi-label image classification each image has different number of labels. As a result, in each image objects are of different scale with different locations, which makes it more complex to capture all objects in a single image. Therefore, the learning task of multi-object learning is more challenging than single object recognition.

2. Multi-label correlation

In multi-label image classification, one image contains more than one label; and there might be correlations between two multi-labels [6]. For instance, in one image, one pixel might belong to different classes. The overlapping between two classes might deteriorate the accuracy of prediction.

3. Threshold adjustment

Suppose there are N potential labels for a single image. When predicting label(s) in the image, the fitting model will assign a probability to each potential label. In order to determine which potential label corresponds to the target label, we need to provide one threshold for those N probabilities. However, this threshold is hard to calibrate, and the cost to determine the optimal threshold is not negligible especially in implementing CNN.

4. Hardware limitation

Numerous attempts on image processing rely on powerful hardware. As we know, CNN demonstrates excellent performance generated by recently released network GoogleNet [20], but it requires high-GPU system to support its training process. In this paper, the investigation and discussions are based on a non-GPU platform. In order to compare CNN and XGBoost, the implementations are running on a CPU system. It is a challenging and time-consuming task.

D. Overall Framework

The approaches of utilizing CNN and XGBoost on multi-label image classification task are totally different. According to the principles of these two algorithms, CNN does not require too much feature engineering process before fitting the model; however, the final performance of XGBoost highly relies on pre-processing and feature engineering process. Therefore, when designing the framework, it is necessary to consider the the principle behind each algorithm. In specific, in CNN I put more emphasis on structures design and parameters selection in each layer because of limitation of hardware. While in XGBoost, I concentrate on feature extraction process and aim to extract relatively independent features as many as possible from each image.

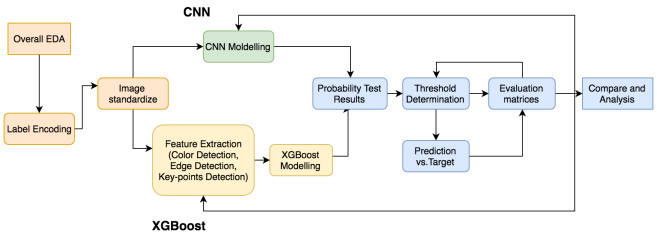

Fig. 1. Architecture Design

E. Evaluation Matrices

Evaluation matrices of Multi-label classification is worth to be introduced in this paper. It is common to use accuracy and F-measures in Single-label-multi-class evaluation. Beside that, there are some other matrices such as Precious and Recall that might be considered. However, those evaluation matrices do not work well in multi-label classification because the evaluation result for a testing sample can either be fully correct or partially correct. Suppose there are three target labels given a test sample, but the prediction rule only produces two correct labels. In this situation, we have to define evaluation matrices containing different level of correctness, which aggravates complexity in evaluation steps.

There are basically three types of measures in a multi-label task: evaluating partitions, evaluating ranking and using label hierarchy [6]. In this paper, I apply evaluating partitions measure to assess how much the predict labels are close to

the target labels for testing data. In the framework design that contains N potential labels for each image, I convert the prediction result from N probabilities to N binary representations using a threshold. In addition, for the text target labels, I also convert text labels to N binary representations in a way that 1 represents this image contains this label and 0 otherwise.

Example	Target labels	Target label convert	Predict labels Prob	Predict label Covert (Threshold = 0.6)
1	Obj-1, Obj-3, Obj-5	[1,0,1,0,1]	[0.7, 0.1, 0.67, 0.02, 0.5]	[1,0,1,0,0]
2	Obj-2, Obj-5	[0,1,0,0,1]	[0.1, 0.9, 0.4, 0.2, 0.8]	[0,1,0,0,1]
3	Obj3, Obj4	[0,0,1,1,0]	[0.6, 0.2, 0.8, 0.3, 0.1]	[1,0,1,0,0]

Fig. 2. Example of Label Encoding

In order to evaluate prediction results comprehensively, I implemented three different measures: Exact Match Ratio, Hamming Loss and the multi-label evaluations proposed by Godbole [29](Godbole measure).

1. Exact Match Ratio

Exact Match Ratio ignores the partial correctness, but considers multi-label measures the same as single-label measures. For each testing sample, as long as the prediction is not fully correct, it will be considered as a wrong prediction.

$$\text{Exact Match Ratio, MR} = \frac{1}{n}\sum_{i=1}^{n} I(Y_i = Z_i)$$

2. Hamming Loss

Hamming Loss measures prediction correctness by considering total number of labels and total number of testing samples. This measure treats partial prediction as independent prediction.

$$\text{HL} = \frac{1}{kn}\sum_{i=1}^{n}\sum_{l=1}^{k}[I(l \in Z_i \wedge l \notin Y_i) + I(l \notin Z_i \wedge l \in Y_i)]$$

3. Godbole Measure

Godbole measure is an adjustment of original measure of precious, recall, accuracy and F-measure. It takes partial correctness into consideration.

$$\text{Accuracy, A} = \frac{1}{n}\sum_{i=1}^{n}\frac{|Y_i \cap Z_i|}{|Y_i \cup Z_i|}$$

$$\text{Precision, P} = \frac{1}{n}\sum_{i=1}^{n}\frac{|Y_i \cap Z_i|}{|Z_i|}$$

$$\text{Recall, R} = \frac{1}{n}\sum_{i=1}^{n}\frac{|Y_i \cap Z_i|}{|Y_i|}$$

$$F_1 = \frac{1}{n}\sum_{i=1}^{n}\frac{2|Y_i \cap Z_i|}{|Y_i| + |Z_i|}$$

IV. Experimental Design and Analysis

The experiment involves multi-label image classification problem, I follow the methodology framework and manage to solve a real-world problem. Eventually, I will analyze the causes and results for this concrete experiment.

A. Dataset

The dataset is from a data science competition website named Kaggle, and the data was offered by a public completion which is called Planet: Understanding the Amazon from Space. This completion requires to use satellite data to track human footprint in the Amazon rainforest. The source data is from https://www.kaggle.com/c/planet-understanding-the-amazon-from-space/data. I only use this dataset to carry on the multi-label image classification research without participating the competition. Therefore, I utilized the provided images with given corresponding label(s). The training images are given in .jpg format, and there are 40479 images, their corresponding labels are provided in a .csv file containing text labels.

In Fig.3, it shows a subset of the image dataset. As it shown, every image has its corresponding label(s); but the number of labels is uncertain. Also, it is obvious that, in this dataset, there is no clear boundary between object and background; also, every pixel in a image belonging to a label. For instance, The fifth figure in Fig.3, the clear pattern can be found is a river which is labelled as water, and the left corner there is very small part shows cloud which is labelled as cloud; however, the rest pixels of this image is not useless area but represent primary. Therefore, this increases the multi-label classification challenge since the feature extraction requires a lot of effective pixels.

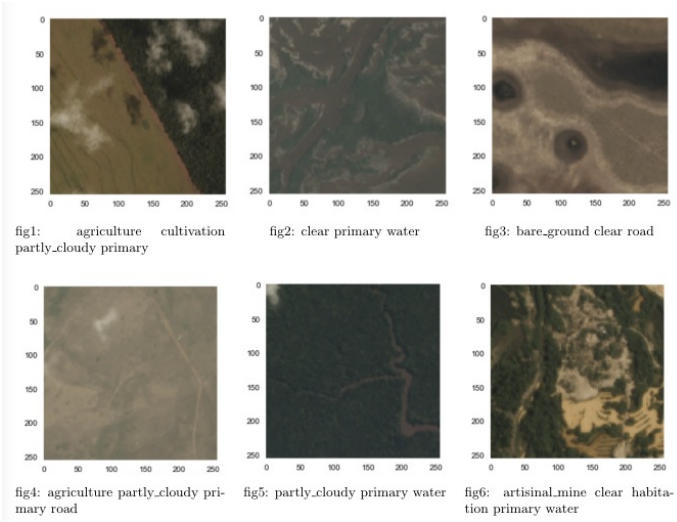

Fig. 3. Examples of Given Images

In order to better modeling and evaluate, I split the dataset into two parts. Around 60% of images (24287 images) are used for training and 40% of images (16192) are used to test the fit model.

B. Overall exploratory Data Analysis (EDA)

Before modeling, I initially did the overall EDA and this step would provide a basic information of this dataset, which helped me to get to know the major characteristics of this dataset and summarized those pieces of information visually.

There are 17 different labels Fig.4, and the labels are not evenly distributed. There are 40479 images in total, and 37513 of images contains primary, but only 100 images contain conventional_mine. The imbalanced label distribution is shown in Fig.5.

Label	Counting Summary	Label	Counting Summary
primary	37513	cloudy	2089
clear	28431	bare_ground	862
agriculture	12315	selective_logging	340
road	8071	artisinal_mine	339
water	7411	blooming	332
partly_cloudy	7261	slash_burn	209
cultivation	4547	blow_down	101
habitation	3660	conventional_mine	100
haze	2697		

Fig. 4. Labels Classes Overview

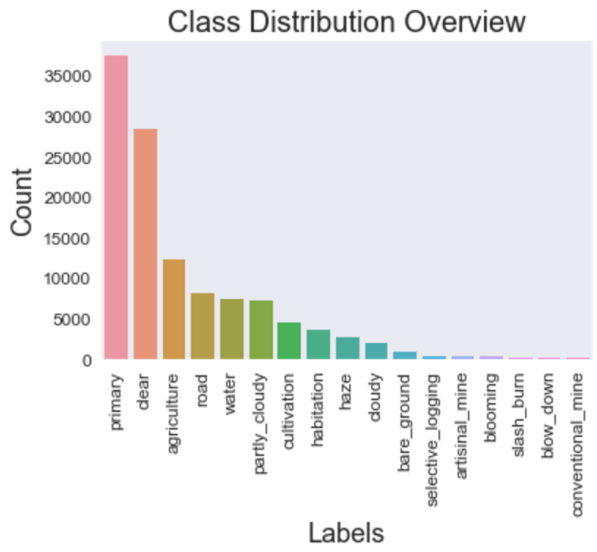

Fig. 5. Label Class Histogram

Unlike single-label task, the number of labels for every image are uncertain; thus, I created a plot to observe the distribution that how many labels in one image. From Fig.6, we can see that there are almost 20,000 images contain 2 labels, and around 7,500 images contain 3 labels and 7500 images contain 4 labels. Moreover, the maximum labels are 9 in a single image, and only around 2,500 images contains single label.

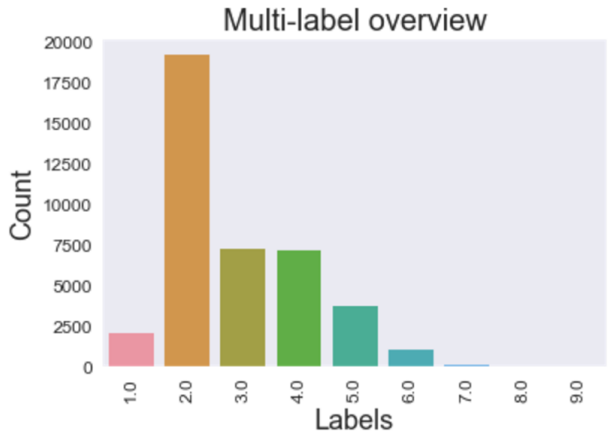

Fig. 6. Histogram Distribution – Number of Labels in Each Image

Moreover, for multi-label task, it is important to check the correlations between labels, since the labels may have correlations between each other. With the dependence of importance labels, we can benefit from this situation based on Correlation Labeling Model [8]. Fig.7 shows the correlation plot for this dataset. As we can see, the red color represents high correlation, and dark blue represents less correlation. We could conclude that, in this dataset, the correlation between labels are not strong, the most correlated term is the label primary; however, as we analyzed before, since around 3/4 images contain primary, this might be the reason that primary has high correlation with other labels.

$$C_{kl} = cos(y_{(k)}, y_{(l)}) = \frac{\langle y_{(k)}, y_{(l)} \rangle}{\|y_{(k)}\| \|y_{(l)}\|}$$

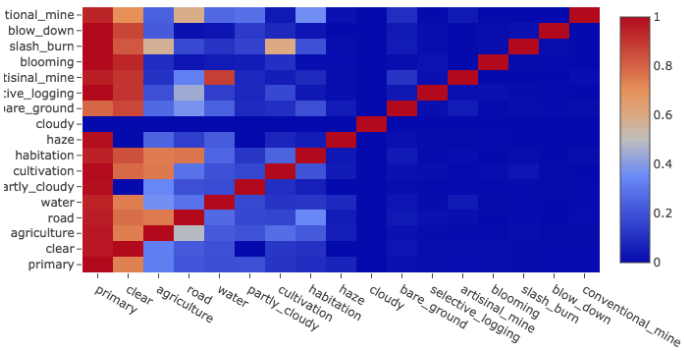

Fig. 7. Label Correlation Overview

C. Implementation

I use python as my software tool. In this experiment, the important libraries I chose include numpy, pandas, seaborn, tqdm, opencv, spicy, counter and time. In specific, when implementing CNN, the major library is keras under tensorflow

backend; when implementing XGBoost, the major library are XGBClassifier and OneVsRestClassifier.

Pre-processing

The pre-processing before CNN and XGBoost modeling is class variable label encoding. Based on my previous investigation about evaluation algorithms, I need an output matrix in size of 40479 x 17 with all binary representations. I firstly created an dictionary in python for all 17 labels, and utilize this dictionary to concrete binary relevance [6] transferring text labels into the required format.

CNN implementation

CNN algorithm does not require heavy preprocessing on given images. Before building CNN structures, I standardized the image size into 64 x 64 x 3. In addition, I split the images and labels into training and testing two groups.

In building CNN model, many types of layers can be employed; however, there are some points need to be considered when modeling. The first point is the hardware limitation, I used a laptop with 2.7 GHz Intel Core i5 processor, and it does not have GPU platform. With this limitation, the shortage of CNN occurs, since CNN highly relies on high GPU computation. AlexNet in 2012 used two-GPU system, and the training process took about 5 to 6 days. Also, other famous structures such as VGG and GoogleNet also have strict requirements on hardware. Without GPU supporting, it is almost impossible to run those famous structures even though they are generalized and usually shows the excellent performance. The second point is the unique characteristics of this image dataset. Those images have multi-labels, but there is no boundaries between object and background for every image. Technically, this is not a traditional multiple object recognition task, and every single pixel belonging to a class. In this situation, the CNN structure design map may be different from tasks that ask for recognizing one or more objects. As a result, the idea "more complex structure, better performance" might not be true. Capturing the comprehensive features from whole image is more essential.

My implementation structure of CNN involves CNN layer, max-pooling layer, ReLU layer, dropout layer, flattery layer and dense layer (forward prorogation layer). In the table CNN model 1, it shows my first self-designed model structure, and CNN model 1 Result shows the experiment results with changing number of epochs and their corresponding performance. As we can see, based on this CNN structure, the more epochs I assigned, the better results can be achieved, and the highest validation accuracy is 0.9428 achieved at epoch 17. In addition, the time consumption increased by adding more epochs. In average, adding one more epoch, the marginal time raise is around 40 seconds. Moreover, the validation accuracy is increased from 0.9114 to 0.9428 by adding more epochs from 3 epochs to 17 epochs; however, the marginal time consumption increased around 6 times which is from 145 seconds to 835 seconds. The time consumption is from a model consists of extremely simple structure which only contains two convolutional layers. Moreover, I designed another CNN model with more complex structure. I add three convolutional layers and choose filters at 64, 64 and 128 respectively. At the same time, I attempted 5 epochs in the first round, but the result was not improved at all. First, the accuracy dropped from 0.9504 to 0.8678 when adding from 1 epoch to 5 epochs. Besides, the training time took 7202 seconds in total which is almost 10 times of the best result given by the simpler structure introduced before.

These test results prove that it is not necessary to add more convolutional layers to build a complex CNN structure. Sometimes, the more complex structure might lead to worse predicting results and huge time consumption. Back to this experiment, the images do not contain clear objects; as a result, running more convolutional layers might guide the model with wrong directions on training process; in other word, over using the convolutional layers might cause to ignore the background information, but in this dataset, there is no background information and over eliminating would cause feature missing problem. Hence, this multi-label task is not suitable for using too many convolutional layers.

The advantage of implementing CNN is that it does not require heavy preprocessing or feature engineering process. Deep Neural Network provides a comprehensive system for processing feature extraction and model training at the same time. CNN, as a deep neural network, utilizes convolutional computation on feature extortion process as well as uses the network structure on the training process. The drawback of implementing CNN is that it is difficult to design a feasible model to fit a particular dataset. People first have to consider and determine from a lot of types of layers and parameters; also, they need to attempt several times modeling process to find a suitable CNN structure, and the whole process usually requires a huge amount of time consumption.

CNN Model 1	
Layer	Parameter
Conv1	filters=32 kernel_size=3x3 strides=2 activation='relu'
MaxPooling1	pool_size=2x2
Conv2	filters=48 kernel_size=3x3 activation='relu'
MaxPooling2	pool_size=2x2
Dropout1	Prob=0.5
Flatten	
Dense1	filters=128 activation='relu'
Dropout2	Prob=0.5
Dense2	filters=17 activation='sigmoid'

CNN Model1 Result					
Epoch	Time(sec)	Loss	Acc	Val_Loss	Val_Acc
3	145	0.2464	0.9070	0.2208	0.9114
5	245	0.2044	0.9194	0.1888	0.9226
10	417	0.1755	0.9315	0.1608	0.9349
15	607	0.1595	0.9369	0.1497	0.9394
20	835	0.1500	0.9400	0.1430	0.9428

XGBoost implementation

XGBoost algorithm is a gradient boosting framework; therefore, it is unlike CNN, feature engineering plays an significant role before applying XGboost classifier. The feature extraction results have a critical influence on final prediction performance.

Feature extraction

When doing feature extraction, the most common ideas include key points detection, color detection and edge detection. When implementing these feature extortion techniques, I did a lot of attempts and some of them succeeded but some of them failed. In this paper, I will introduce the feature extraction process and analyze why some techniques could be successful and others were failed.

1. Scale-Invariant Feature Transformation(SIFT) Keypoint Detection

Speaking of key points detection, SIFT is one of the most common used algorithms. SIFT finds the key points by utilize Gaussian smoothing operations to determine the local maximum or minimum, and this algorithm is based on location, local scale and orientations of key points [9]. I implemented SIFT by using library opencv in python, and the key points detection example is shown in Fig.8. This figure contains some obvious line features; thus, SIFT clearly searched them. Usually with the key points, the next steps is to apply the vector quantization methods either utilizing histogram or clustering techniques to classify the key points. Firstly, I considered SIFT as an applicable method in this multi-label task. However, after I tried a few examples I found that, some images has no key points such that SIFT finds nothing for that image; an example is shown in Fig.9. After this discovery, I realized that not all types of images have clear key points in my dataset. In addition, the key points location, scale and orientations do not contribute to feature extraction in this dataset since in this dataset key points and its belonging classes are highly independent based on previous correlation discussion. After this unsuccessful attempt, I put more attention on color detections and edge detections.

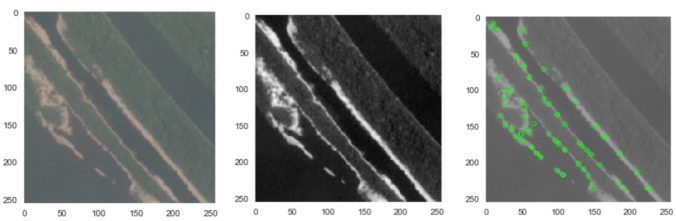

Fig. 8. SIFT Key Points Detection

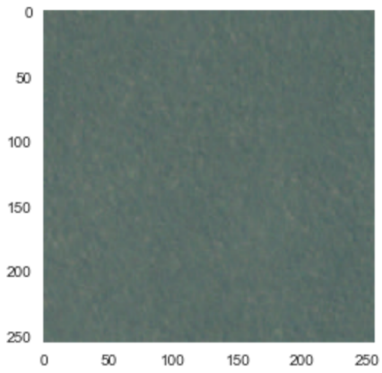

Fig. 9. Example that SIFT cannot detect any key point

2. RGB Feature Detection

In this dataset, color features are critical since different classes may not have special shapes or patterns, but every class usually has its typical color characteristics. Based on the raw pixel information, I think the generalized patterns are more important. Fig.10 shows an image itself, a blue color channel version, a green color channel version and a red color channel version. I observed that in different color channel, the overall patterns are discrepant. Hence, I did deeper investigation to observe the color distributions. In Fig.11, I plotted histograms for RGB channels and surprisingly found that for each channel, the pixel values follow a bimodal distribution. Hence, the characteristic value of the bimodal distribution can be considered as features. Hence, for each channel in every image, I produce 11 characteristic values including mean, median, mode, standard deviation, maximum, minimum, kurtosis, skewness, left-modes mode, right-modes mode and their difference. As a result, for one image with three channels, I have 33 characteristic values as features.

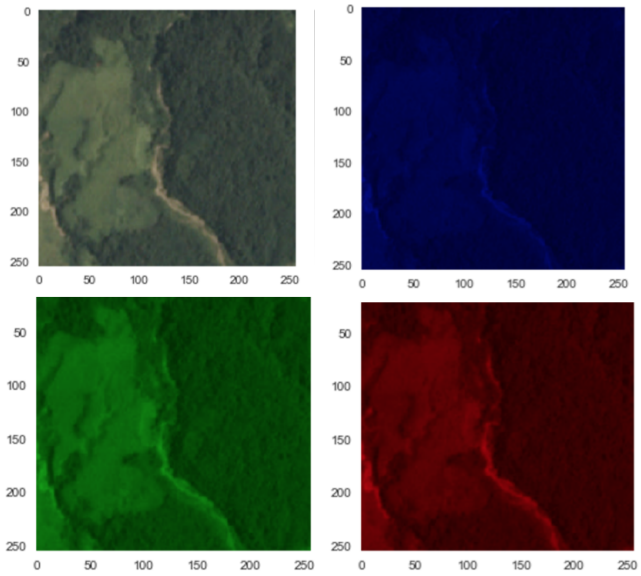

Fig. 10. RGB channels Example

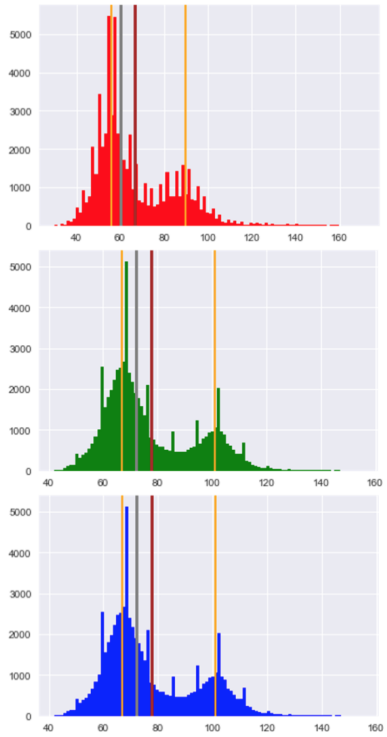

Fig. 11. RGB Bimodal Distribution Example

3. Sobel Edge Detection

With the ROB features, I considered to extract features by edges. Unlike SIFT key points detection, edges must exist in every single image, some images contains clear edges and some images has no special edges. I employed Sobel edge algorithm to do the edge detection, and it will ignore the color information and only focus on edges. Sobel Edge detection uses one 3x3 kernel to calculate the horizontal changes, and it uses the other 3x3 kernel to derive the vertical changes. As shown in Fig.12, figure sobelx shows the horizontal changes and sobely shows the vertical changes. Since the original image size is 256 x 256 x 3, Sobel algorithm will produce two colorless images with size 256 x 256 x 1. I combined them to produce a characteristic vector with 64 values. In addition, I selected 7 values from that vector as features including the mean, median, standard deviation, maximum, minimum, kurtosis and skewness.

Fig. 12. Sobel Edge Detection Example

Modeling

As I discussed, for each image, I have 40 values to represent its features. Transforming the raw image pixels to 40 feature values for a single image is fast; however, with the sample size over 40,000 images, the converting time is huge. I used library Time in Python to setup a timer to record the transformation time. It took 6969.148 seconds to transfer all 40,479 images into 40 feature values by each. Eventually, I obtained a matrix at size 40,479 x 40.

After feature engineering process, I started to train the model by applying XGBoost algorithm. I combined "XGB-Classifier" and "OneVsRestClassifier" to train the multi-label at the same time. To control the overfitting, I manipulated the tuning parameter max_depth. The table XGBoost Model shows the influence of tuning-parameter.

XGBoost Model 1			
max_depth	Training Acc	Valid Acc	Training Time(sec)
3	0.5479	0.5279	68.2491
5	0.6088	0.5480	119.4357
7	0.7055	0.5510	143.9989
10	0.9048	0.5512	199.1166

The training accuracy and validation accuracy are not as high as CNN at this moment; however, this is from the XGB score library in Python, and I will compare the prediction results with CNNs by applying the multi-label-multi-class evaluations mentioned in methodology part. Moreover, the training time of XGBoost is acceptable. The larger max_depth, the longer training time, but the marginal time consumption increase is not too bad. In this case, I finally decided the tuning parameter max_depth equals 7 since it provides relatively better accuracy and lower overfitting.

In implementing the traditional image recognition method such as feature extraction with XGBoost, there are some drawbacks: heavy implementations is required in feature extraction process, and transformation from images to feature values cost a huge amount of time. The advantage of XGBoost is that once the feature extraction has done, then training process is much easier than CNN.

V. Evaluation

For both CNN and XGBoost, I selected the best performance models respectively after several rounds of attempts and tests. Based on the previous introduction, I implemented three evaluation methods which are typically for the multi-label-multi-class task. The Evaluation Summary table presents summary results for both CNN and XGBoost.

In this evaluation summary table, it records Exact Match Ratio score, Hamming Loss, Godbole Measures with different choices of threshold.

Exact Match Ratio is the most strict measure; in specific, for each test image, it is considered as a wrong prediction as long as one of the produced labels were wrongly predicted or missed. Exact Match Ratio totally ignores the partial correct.

Evaluation Summary Table							
Algorithm	Evaluation Method	Thres=0.4	Thres=0.45	Thres=0.5	Thres=0.55	Thres=0.6	Thres=0.65
CNN	Exact Match Ratio	0.5068	0.5121	0.5107	0.4997	0.4824	0.4565
	Hamming Loss	0.0590	0.0583	0.0584	0.0593	0.0613	0.0647
	Precision (Godbole)	0.8285	0.8086	0.7208	0.7726	0.7499	0.7208
	Recall (Godbole)	0.8722	0.8871	0.9176	0.9066	0.9131	0.9176
	Accuracy (Godbole)	0.7735	0.7696	0.7106	0.7519	0.7352	0.7106
	FMeasure (Godbole)	0.8388	0.8343	0.7841	0.8186	0.8045	0.7841
XGBoost	Exact Match Ratio	0.5412	0.5479	0.5509	0.5503	0.5414	0.5294
	Hamming Loss	0.0500	0.0492	0.0488	0.0489	0.0500	0.0512
	Precision (Godbole)	0.8656	0.8521	0.8382	0.8245	0.8095	0.8245
	Recall (Godbole)	0.8897	0.9009	0.9102	0.9183	0.9228	0.9183
	Accuracy (Godbole)	0.8055	0.8046	0.8013	0.7965	0.7873	0.7965
	FMeasure (Godbole)	0.8656	0.8638	0.8602	0.8554	0.8476	0.8555

To achieve a high score by using MR measure in the multi-label task is extremely difficult. Compare the Exact Match Ratio produced by CNN and XGboost, the average accuracy was around 50%. The highest score for CNN is 51.21% when threshold equals 0.45, and the highest score for XGBoost is 55.09% when threshold equals 0.5.

Hamming loss is the most generous evaluation method. It totally considers all the partial correct predictions. Therefore, the prediction results were extremely good in my implementation. The lowest loss rate for CNN is 5.83% when threshold equals 0.45, which means the overall accuracy reached to 94.17%. In addition, the lowest loss rate for XGBoost is 4.88% when threshold equals 0.5, and the accuracy reached to 95.12%.

Godbole Measures is the modification of single-label-multi-class task. From the summary table, it is obvious that the overall recall score is over 90%, which means the false negative value is small and both techniques prediction results do not contain many missing predictions. Compare the Precious with Recall, the precious score is lower than recall, which means the prediction contains False Positive more than False Negative. In general, F-measures were over 80%, and accuracy was between 75% - 82%. For CNN, the highest accuracy achieved is 77.35% when threshold equals 0.4 and the highest F-measure reached to 83.43%. For XGBoost, the highest accuracy was 80.55% and the highest F-measure reached to 86.56% when threshold equals 0.4.

Based on this result, the XGBoost overall performance is better than CNN, which is surprised. Here are some possible reasons: 1. CNN structure may require further modifications. Running CNN on a no-GPU system is difficult, and slight adjusting filters, kernel size or adding layers require another round of training process. Therefore, running on high-GPU machines could offer a better platform to repeat the adjustment and training process in order to attain a more suitable model for this dataset. 2. XGBoost feature extraction process is more targeted to this dataset. When doing feature engineering, I considered the key points detection is not useful and RGB features play more important role in this specific dataset. Hence, XGBoost feature extraction may result in the more effective feature values.

Overall, the highest Exact Match Ratio is 55.09%, the lowest Hamming Loss is 4.88% and the highest F-measure is 86.56% for this experiment. In this paper, there are some non-negligible restrictions in the implementation process. For CNN, I believe with more powerful hardware and with more time, the final results could be significantly improved. For XGBoost, applying dimensional reduction techniques such as Principal Component Analysis (PCA) or Linear discriminant analysis(LDA) after feature extraction step might be helpful. Generally speaking, CNN is a more powerful technique in image classification. In this paper, the structure design of CNN is quite simple, but the accuracy is not much lower than XGBoosts performance, recall that XGBoost feature extraction was complicated. Therefore, CNN has the potential to reach better results after breaking through the hardware limitations.

VI. CONCLUSIONS

In conclusion, the multi-label image classification would be a trend of image processing development in the future. Multi-label task is is an extension of a single-label task, and it is more practical in real word. However, it is difficult to obtain high-quality performance in a multi-label task due to challenges such as label-correlation, label-overlapping, multi-feature extractions and hardware limitations. Based on my experiment and investigations, CNN has big potential in terms of improving the performance of a multi-label task. Moreover, solutions such as a combination of CNN and RNN, HCP, and a utilization of MLDA and SRN to discover the hidden relationship between labels would help to optimize the training process and improve the final results.

REFERENCES

[1] Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks[C]//Advances in neural information processing systems. 2012: 1097-1105.

[2] Szegedy C, Liu W, Jia Y, et al. Going deeper with convolutions[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 1-9.

[3] Pang, Yanwei, et al. "Multimodal learning for multi-label image classification." Image Processing (ICIP), 2011 18th IEEE International Conference on. IEEE, 2011.

[4] Yang, Jianchao, et al. "Linear spatial pyramid matching using sparse coding for image classification." Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on. IEEE, 2009.

[5] Cabral, Ricardo, et al. "Matrix completion for weakly-supervised multi-label image classification." IEEE transactions on pattern analysis and machine intelligence 37.1 (2015): 121-135.

[6] Sorower, Mohammad S. "A literature survey on algorithms for multi-label learning." Oregon State University, Corvallis 18 (2010). APA

[7] Yang, Jianchao, et al. "Linear spatial pyramid matching using sparse coding for image classification." Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on. IEEE, 2009.

[8] Wang, Hua, Chris Ding, and Heng Huang. "Multi-label linear discriminant analysis." Computer VisionECCV 2010 (2010): 126-139.

[9] Jia, Zhengping, et al. "SIFT-based Sparse Coding for Large-scale Visual Recognition."

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

[11] Wang, Jiang, et al. "Cnn-rnn: A unified framework for multi-label image classification." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016.

[12] Wei, Yunchao, et al. "HCP: A flexible CNN framework for multi-label image classification." IEEE transactions on pattern analysis and machine intelligence 38.9 (2016): 1901-1907.

[13] Wei, Yunchao, et al. "CNN: Single-label to multi-label." arXiv preprint arXiv:1406.5726 (2014).

[14] Saber, Eli, and A. Murat Tekalp. "Frontal-view face detection and facial feature extraction using color, shape and symmetry based cost functions." Pattern Recognition Letters 19.8 (1998): 669-680.

[15] Grana, Costantino, et al. "A fast approach for integrating ORB descriptors in the bag of words model." Proc. SPIE. Vol. 8667. 2013.

[16] Chen, Tianqi, and Carlos Guestrin. "Xgboost: A scalable tree boosting system." Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining. ACM, 2016.

[17] Kavitha, J. C., and A. Suruliandi. "Texture and color feature extraction for classification of melanoma using SVM." Computing Technologies and Intelligent Data Engineering (ICCTIDE), International Conference on. IEEE, 2016.

[18] Wu, Xi-Zhu, and Zhi-Hua Zhou. "A Unified View of Multi-Label Performance Measures." arXiv preprint arXiv:1609.00288 (2016).

[19] Kanopoulos, Nick, Nagesh Vasanthavada, and Robert L. Baker. "Design of an image edge detection filter using the Sobel operator." IEEE Journal of solid-state circuits 23.2 (1988): 358-367.

[20] Szegedy, Christian, et al. "Going deeper with convolutions." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

[21] Lowe, David G. "Distinctive image features from scale-invariant keypoints." International journal of computer vision 60.2 (2004): 91-110.

[22] Harris, Chris, and Mike Stephens. "A combined corner and edge detector." Alvey vision conference. Vol. 15. No. 50. 1988.

[23] Gao, Wenshuo, et al. "An improved Sobel edge detection." Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on. Vol. 5. IEEE, 2010.

[24] Chang, Chih-Chung, and Chih-Jen Lin. "LIBSVM: a library for support vector machines." ACM transactions on intelligent systems and technology (TIST) 2.3 (2011): 27.

[25] Breiman, Leo. "Random forests." Machine learning 45.1 (2001): 5-32.

[26] Wagstaff, Kiri, et al. "Constrained k-means clustering with background knowledge." ICML. Vol. 1. 2001.

[27] Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." Journal of Machine Learning Research 15.1 (2014): 1929-1958.

[28] Szegedy, Christian, et al. "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning." AAAI. 2017.

[29] Godbole, Shantanu, and Sunita Sarawagi. "Discriminative methods for multi-labeled classification." Advances in knowledge discovery and data mining (2004): 22-30.