Plant Leaf Classification Using Convolutional Neural Network

Md. Al Maruf

Department of Electrical and Computer Engineering,

University of Ontario Institute of Technology(UOIT), Canada

E-mail: md.maruf@uoit.net

Abstract

The primary interest of this project is to classify plant leaf images using the convolutional neural network (CNN). A supervised learning process is used to design the CNN model. The defined model uses two types of data, textual and image data for the leaf classification. In this experiment, a kaggle leaf dataset of 1,584 images is used to classify 99 species of plants accurately. To train a CNN model using relatively small training dataset doesn't show a good result at the end. This motivates to create a new model design that combines a convolutional and multilayer perceptrons neural network using the images and the pre-extracted features to train the model end to end. Initially, I create a CNN sequential model that takes only the image inputs to train the model. After that, a comparison is shown with the combined model that takes the images and the pre-extracted features as input in the different stage of the model. I fed the raw binary images to extract features and pass the extracted features combining with pre-extracted features through a multilayer perceptron neural network to identify the classes precisely. The combined designed system works in two states. In the beginning, it applies Keras functional API to extract features, and next uses the pre-extracted features with it in the input layer of the neural network to train the

system. The experimental result using the training dataset confirms an accuracy of 99% in this proposed method which is quite adequate to classify leaf images correctly.

Introduction

There are a large number of species of plants are available all over the world. Plant recognition is an important component of typical workflows in biological research area.[1] It is historically difficult to classify different species and often it provides the wrong classification. Therefore, automatic plant leaf classification helps in various applications. For example: plant-based medicine research and food industry,[2] crops monitoring and management,[3] and also for flower species population tracking and preservation.[4]

According to plant taxonomy, it describes that any species can be easily identified from the leaves rather than other parts of the plant. An experiment is conducted using kaggle's new data set. The new dataset consists of 1,584 images of leaf specimens (16 samples each of ninety-nine species) which have been converted to binary black leaves against white backgrounds. Each image contains three sets of features where each feature has a 64-attribute vector. To simplify the experiment, a study analyses the common features of leaf classification, like a shape descriptor, an interior texture histogram, and a fine-scale margin histogram.

In the early stage of this project, I build a CNN model using Keras Sequential API. This sequential model uses the image inputs only to train the model. The image dataset is divided into training and validation set which leads to a training accuracy of 97% and validation accuracy of 80%. As the number of images(16) per classes is not enough to train a CNN model, a combined CNN model is introduced that uses both images and pre-extracted features at the different stage of the model. The two-layer convolutional neural network extracts the features from the image dataset. These features are called self-extracted features. After that, the self-extracted features and the pre-extracted features are merged

and passed through a fully connected neural network layers to train the model properly. This combined design model shows an improvement in both training and validation accuracy. The combined dataset in CNN model results in a good accuracy where other models do not have this result.

The project report is organized as the following: Related Work section discusses the previous work related to plant leaf features. The dataset, consisting of images and CSV file of pre-extracted features, are described in Dataset section. Then I demonstrate the model design part in Architecture and Methodology section. Lastly, the Result section explores and discusses the effectiveness of the method for the designed model in leaf classification.

Related Work

Classification of plant species from images is a challenging task in computer vision. Many types of research have been done on this topic using different classification technique. The most common classifiers that are used in this area are K-Nearest Neighbour, Support Vector Machine, Decision Tree, Random Forest, and Convolutional Neural Network. The convolutional neural network is familiar for image, speech, and high dimensional data classification[5] or recognition.[6] Many studies are shown using several algorithms that can be used to detect objects in an image. Jassmann[7] proposes a method to classify leaf images utilizing a convolutional neural network. This paper describes the classification of leaves for 29 predefined categories. The system developed for this research utilizes a convolutional neural network in an android mobile application to classify natural images of leaves. For preprocessing, the original leaf image is resized to 60x80 pixels to reduce computing time. Next, the Zero Components Analysis (ZCA) whitening technique is applied to the image to make the features less redundant.

A similar approach using deep learning method is used to identify plants. Sue Han Lee[8] presents an architecture of CNN model that identifies 44 different plant species with the

performance of 99.6%. Cope et al.[9] achieved an 85% identification rate on 32 species of a single plant species using the co-occurrences of different scale Gabor filters.

Inspired by deep learning breakthrough in image classification, many researchers are analyzing their experiment using CNN with Keras to achieve a good performance. To go with the recent trends, in this project a similar kind of analysis is performed to compare the overall performance and the experimental results show that the CNN model using Keras achieves superior performance than traditional solutions.

Dataset

The kaggle dataset consists of 16 samples of 99 species which gives a total of 1,584 images. The dataset also contains textual data in CSV file that describes the features of each image. There are three kinds of features shape, margin, and texture for each image and each feature contains 64 attributes of vector value. The textual training dataset has four fields id, species name, shape, margin, and texture. Here, Id is an anonymous id unique to an image, and species name defines the class of the image. The fields description is shown below:

- **id:** unique id to an image
- **margin1, margin2, ..., margin64:** each margin feature has 64 attribute vectors.
- **shape1, shape2, ..., shape64:** each shape feature has 64 attribute vectors.
- **texture1, texture2, ..., texture64:** each texture feature has 64 attribute vectors.

The training dataset contains 990 images where I train the CNN model using 792 images (80% of training dataset) for training and 99 images for cross-validation of the model as the test dataset does not have class lebels for their own purpose.

Architecture and Methodology

This project work is splited into two phases Phase 1 and Phase 2. Here, the architecture of the model and methods are described below.

Phase 1 (Sequential Model): In this phase, a two-layer convolutional neural network is used where it considers the image inputs only to train the model. The basic neural network architecture model that I use for this scenario is Keras Sequential model API. In this design, the input layer takes the input as an array of image ids and pass it through different layers of the convolutional neural network. To rectify the input feature maps after each convolution layer a nonlinear Relu operation is performed. In addition, to categorize the image species, a fully connected layer(Multi-layer Perceptron) is added at the end where this traditional Multi-Layer Perceptron (MLP) uses softmax activation function in the output layer.

Phase 2 (Functional Model): In this second phase of the project, the plan is to combine image and pre-extracted features to train the model. As the inputs are different, in this scenario, a combined Keras Functional API model is used which takes these two different data in two different locations. The output of last convolutional layer is merged with the pre-extracted features and passes through the fully connected layer. A Keras generator trains neural network where it takes the image augmenter generator and the array of the pre-extracted features. The following CNN will describe the details.

Convolutional Neural Network

Convolutional Neural Network (ConvNet or CNN) is one of the most common methods in deep learning and also familiar as an effective class model for image recognition and classification problems.[10] There are four main operations in the CNN model that includes 1. Convolution, 2. Non Linearity (ReLU), 3. Max Pooling, and 4. Classification (Fully Connected Layer).[11] Figure 1 shows the designed architecture of the convolutional neural network.

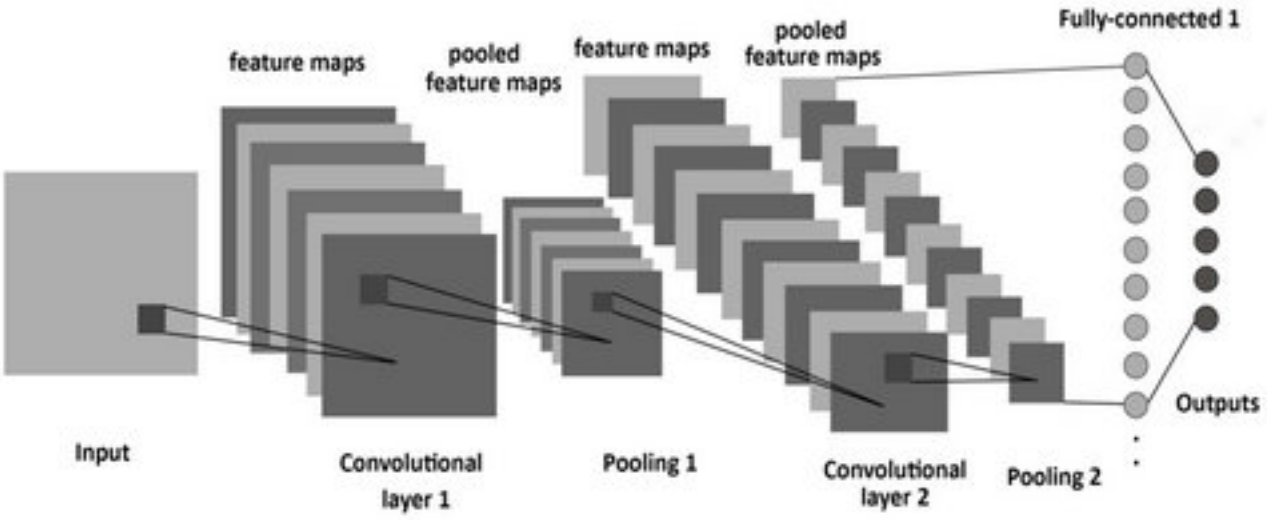


Figure 1: Architecture of Convolutional Neural Network (CNN)

Convolution: The main task of Convolution of a CNN is to extract features from the input images. In this experiment, two convolutional layer modules are used. The first layer module takes a 96x96x1 size input image and a 5x5 filter matrix to extract features. The number of filters increases the features to get extracted from the image which helps to gain the better network model.

ReLU: After completion of a Convolutional operation, a ReLU function is run to ensure the non-linearity among the extracted features. As we know, the most real-time data are non-linear and to maintain the nonlinearity in CNN, ReLU replaces all negative pixel values in the feature map by zero.

Max Pooling: Max Pooling layer take the largest element from the rectified feature map and reduces the dimensionality.Though the pooling operation decreases the image feature size, it retains the most important information. It makes the input representations (feature dimension) smaller and more manageable.

Fully Connected Layer: The Fully Connected layer is a traditional Multi-Layer Perceptron (ANN) that uses a softmax activation function in the output layer. This layer uses high-level features (final output of the convolutional layers) as input and classifies the input image into various classes based on the training dataset.

Experimental Setup

The experiment first loads a grey image with the dimension of 96 x 96. The first convolutional layer uses eight filters with the size of a 5 x 5 window, and the immediate max pooling layer changes the input image to 46 x 46. The second convolutional layer uses the 32 5 x 5 kernels with stride one that makes the image size to 42 x 42. A following another max pooling layer then finally reduces the image size to 21 x 21. The following equation (1) helps to calculate the actual output size of each convolutional layer where W means the input volume size, K is the filter size, P is the padding and S is called the stride.

$$\text{Output} = (\frac{\text{W} - \text{K} + 2\text{P}}{\text{S}}) + 1 \tag{1}$$

Table 1 demonstrate the input, output and others parameters for architectural implementation of convolutional neural network layers.

Table 1: Convolutional Neural Network Parameters

Layer	Input	Filter Size	Stride	#Filters	Activation	Output	Parameter
Convolution 1	96x96x1	5x5	1	8	ReLU	92x92x8	5x5x1x8
MaxPool 1	92x92x8	2x2	2			46x46x8	
Convolution 2	46x46x8	5x5	1	32	ReLU	42x42x32	5x5x8x32
MaxPool 2	42x42x32	2x2	2			21x21x32	
Fully Connected	21x21x32			100	ReLU	100	
Output	100			99	Softmax	99	

Learning

Optimization

This Keras API uses RMSProp optimizer with categorical cross-entropy loss function which is used to optimize the model loss.[12] In case of experimental purpose, a mini-batches of 64 examples, with learning rates of 0.001 is also initialized in this model.

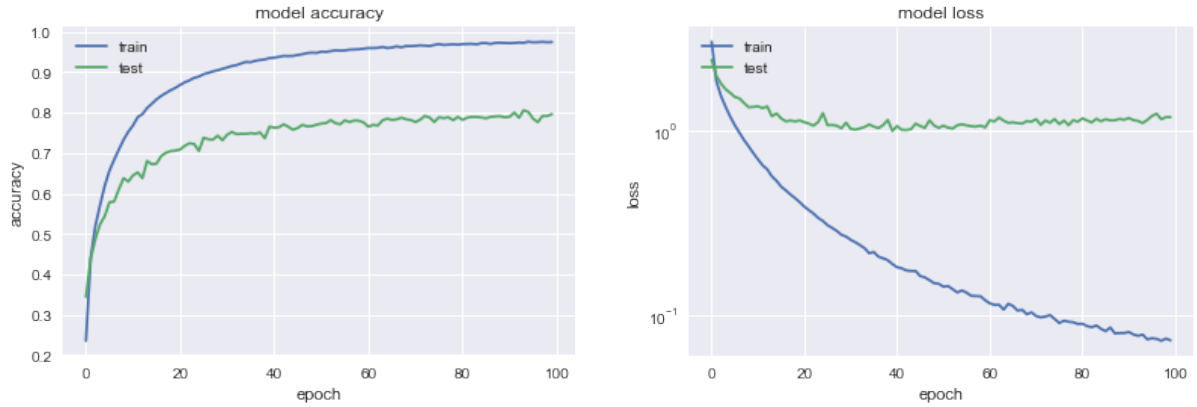
Image Data Augmentation

The kaggle dataset images are mostly similar to each other for a specific species. The difference is slightly in rotation or larger in scale. Before presenting an input to a neural network, each image is resized by 96 x 96 dimensions and place the image at the center of the output image array or the top left corner of the array. To achieve the robustness of the model, the image data augmentation part also performs a random rotation, zoom and general transformation. This technique reduces the effects of overfitting.

Experimental Results Analysis

In this section, the accuracy of leaf classification for two different Keras models are reported, and the results are also analyzed according to the input structure:

Sequential Model with Image Dataset: The first phase experiment with Keras Sequential API shows a training accuracy of 97% where the validation test doesn't gain this amount of accuracy. Figure 2 shows the model accuracy and model loss in terms of training and validation for 100 epochs.



(a) Training and Validation Accuracy (b) Training and Validation Loss

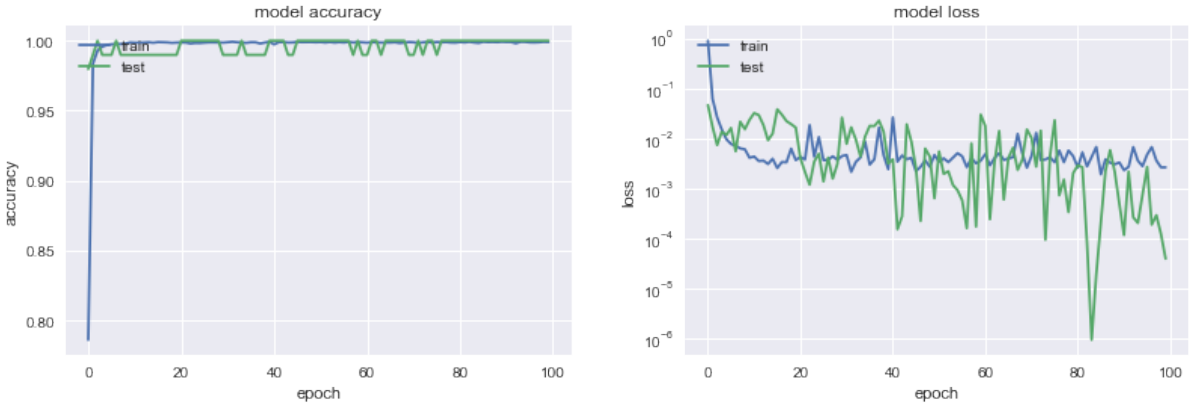
Figure 2: Keras Sequential Model Result Analysis

This designed model gives us the following best accuracy and loss information:

- **Training Accuracy:** 0.975397943867

- **Training Loss:** 0.0730303285847
- **Validation Accuracy:** 0.805976262748
- **Validation Loss:** 0.993863867401

Functional API Model with Image and Pre-extracted Features: The combined model that is used for two different input image and pre-extracted features in CNN shows a significant improvement in both accuracy and loss over the sequential model. From, Figure 3, it can be visualized the overall performance of the designed system. Overall training accuracy rate of the system is about 99.0%. This result outperforms the previous sequential model that takes the image inputs only. In this experiment, I find the following accuracy and test loss for training and testing dataset.



(a) Training and Validation Accuracy (b) Training and Validation Loss

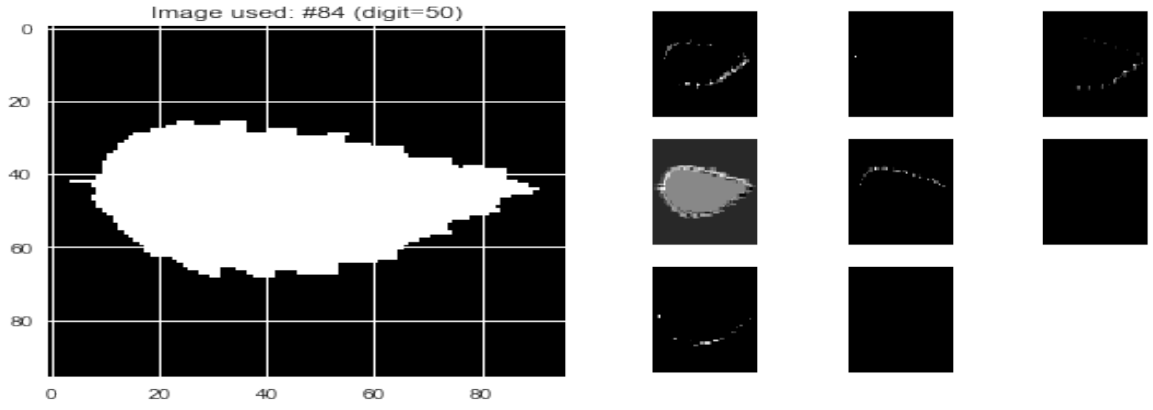
Figure 3: Keras Functional API Model Result Analysis

- **Training Accuracy:** 0.998282754224
- **Training Loss:** 0.00192553164506
- **Validation Accuracy:** 1.0
- **Validation Loss:** 0.00879054215

As the training dataset is small according to a good CNN model, it shows a low performance in the sequential model where the combined model improves a lot. The main reason behind the significant improvement in the combined model is the pre-extracted features addition in Multi-layer Perceptron (MLP) layer. It helps to preserve the learning accuracy and shows an improved result at the end.

Image Visualization:

At the end of the experiment, the CNN combined model learns to classify the raw images correctly. To visualize the prediction, a random leaf from validation set is passed through the CNN model, and the CNN model extracts the features by the filtration process. The combined model applies 8 and 32 filters in convolutional layer 1 and layer 2 respectively. Figure 5 shows the filtered image where the white color indicates the CNN filter effects and the black portion says unfiltered.



(a) Visualization of Input Image at layer 0 (b) Visualization of Image at Layer 1

Figure 4: Visualizatition of an Image in different Convolution Layer

Prediction: The kaggle test dataset does not have class lebels to predict the actual result. In this scenario, I use the cross validation approach and the model predicts to identify the species of the leaf. For the above random image, the actual class name is Quercus_Afares and the model also predicts the leaf class correctly. This gives the top 3 predictions for an

image from which it can be identified in which class it belongs to. Here, the top 3 predictions for image number 84 are shown.

Top Three prediction:

Quercus_Afares: 1.0

Quercus_Trojana: 3.72187e-21

Prunus_Avium: 2.01649e-22

Actual Species Name: Quercus_Afares

Future Work

In this stage, a plant leaf classification will help to extend the work for counting the number of plants in a field. In future, I believe that the colorful image classification with different background object might be helpful to enrich my knowledge.

Conclusion

One of the prime objectives of this project is to recognize plant leaf images from 99 species. Here, the designed framework presents two different scenarios for sequential and functional API of a convolutional neural network model for leaf classification. The second phase of the project demonstrates two layers convolutional neural network model which combines the pre-extracted features at fully connected layer with the input image array and train the entire model using Keras functional API to classify the plant species.

The outputs illustrate that using pre-extracted features Metric, the proposed approach of the combined model provides a significant performance improvement. The network model shows a training loss of almost 0.0019255 and training accuracy of nearly 99%. In addition, to visualize the training process and the image extracted features, a visualization of different layer's image filtration is also presented. Finally, it concludes that the addition of more convolution layer will help to extract more features from the input images and ultimately

add values for precise prediction.

References

1. Barré, P.; Stöver, B. C.; Müller, K. F.; Steinhage, V. LeafNet: A computer vision system for automatic plant species identification. *Ecological Informatics* **2017**,

2. Du, J.-X.; Wang, X.-F.; Zhang, G.-J. Leaf shape based plant species recognition. *Applied mathematics and computation* **2007**, *185*, 883–893.

3. Persson, M.; Åstrand, B. Classification of crops and weeds extracted by active shape models. *Biosystems Engineering* **2008**, *100*, 484–497.

4. Hong, A.-x.; Chen, G.; Li, J.-l.; Chi, Z.-r.; Zhang, D. A flower image retrieval method based on ROI feature. *Journal of Zhejiang University-Science A* **2004**, *5*, 764–772.

5. others,, *et al.* Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks* **1995**, *3361*, 1995.

6. Lawrence, S.; Giles, C. L.; Tsoi, A. C.; Back, A. D. Face recognition: A convolutional neural-network approach. *IEEE transactions on neural networks* **1997**, *8*, 98–113.

7. Jassmann, T. J.; Tashakkori, R.; Parry, R. M. Leaf classification utilizing a convolutional neural network. SoutheastCon 2015. 2015; pp 1–3.

8. Lee, S. H.; Chan, C. S.; Wilkin, P.; Remagnino, P. Deep-plant: Plant identification with convolutional neural networks. Image Processing (ICIP), 2015 IEEE International Conference on. 2015; pp 452–456.

9. Cope, J. S.; Remagnino, P.; Barman, S.; Wilkin, P. Plant texture classification using gabor co-occurrences. International Symposium on Visual Computing. 2010; pp 669–677.

10. Karpathy, A.; Toderici, G.; Shetty, S.; Leung, T.; Sukthankar, R.; Fei-Fei, L. Large-scale video classification with convolutional neural networks. Proceedings of the IEEE conference on Computer Vision and Pattern Recognition. 2014; pp 1725–1732.

11. Deep-Leafs-Operational-Classification-Model. 2017; `http://nefy.org/`.

12. Vij, D.; Aggarwal, N. Performance Evaluation of Deep learning Architectures for Acoustic Scene Classification.

Appendix A:

An experimental result of training the model is also shown to visualize the process clearly. The training dataset of 792 images were fed into CNN and the model was run for 100 epochs in CPU.

```
Epoch 91/100
791/792 [============================>.] - ETA: 0s - loss: 0.0023 - acc: 0.9991Epoch 00090: val_loss did not improve
792/792 [==============================] - 257s - loss: 0.0023 - acc: 0.9991 - val_loss: 1.1828e-04 - val_acc: 1.0000
Epoch 92/100
791/792 [============================>.] - ETA: 0s - loss: 0.0028 - acc: 0.9991Epoch 00091: val_loss did not improve
792/792 [==============================] - 257s - loss: 0.0028 - acc: 0.9991 - val_loss: 0.0022 - val_acc: 1.0000
Epoch 93/100
791/792 [============================>.] - ETA: 0s - loss: 0.0067 - acc: 0.9982Epoch 00092: val_loss did not improve
792/792 [==============================] - 288s - loss: 0.0067 - acc: 0.9982 - val_loss: 2.6698e-04 - val_acc: 1.0000
Epoch 94/100
791/792 [============================>.] - ETA: 0s - loss: 0.0036 - acc: 0.9992Epoch 00093: val_loss did not improve
792/792 [==============================] - 283s - loss: 0.0036 - acc: 0.9993 - val_loss: 2.0625e-04 - val_acc: 1.0000
Epoch 95/100
791/792 [============================>.] - ETA: 0s - loss: 0.0029 - acc: 0.9992Epoch 00094: val_loss did not improve
792/792 [==============================] - 278s - loss: 0.0029 - acc: 0.9992 - val_loss: 7.4669e-04 - val_acc: 1.0000
Epoch 96/100
791/792 [============================>.] - ETA: 0s - loss: 0.0049 - acc: 0.9988Epoch 00095: val_loss did not improve
792/792 [==============================] - 283s - loss: 0.0049 - acc: 0.9988 - val_loss: 0.0027 - val_acc: 1.0000
Epoch 97/100
791/792 [============================>.] - ETA: 0s - loss: 0.0067 - acc: 0.9988Epoch 00096: val_loss did not improve
792/792 [==============================] - 276s - loss: 0.0067 - acc: 0.9988 - val_loss: 1.9081e-04 - val_acc: 1.0000
Epoch 98/100
791/792 [============================>.] - ETA: 0s - loss: 0.0036 - acc: 0.9989Epoch 00097: val_loss did not improve
792/792 [==============================] - 275s - loss: 0.0036 - acc: 0.9989 - val_loss: 2.9517e-04 - val_acc: 1.0000
Epoch 99/100
791/792 [============================>.] - ETA: 0s - loss: 0.0026 - acc: 0.9992Epoch 00098: val_loss did not improve
792/792 [==============================] - 279s - loss: 0.0026 - acc: 0.9993 - val_loss: 1.2358e-04 - val_acc: 1.0000
Epoch 100/100
791/792 [============================>.] - ETA: 0s - loss: 0.0026 - acc: 0.9991Epoch 00099: val_loss did not improve
792/792 [==============================] - 273s - loss: 0.0026 - acc: 0.9991 - val_loss: 3.9894e-05 - val_acc: 1.0000
```

Figure 5: Experimental Result of Training CNN Model