Plant Identification with Convolutional Neural Networks

Shi Cao

Abstract

The classification of Flavia data set which contains 32 types of leaves has been studied in this project. Different input image sizes and image preprocessing techniques including color removal, binarization and edge detection have been used to train convolutional neural networks (CNNs) with different architectures. From our systematic study by tuning the input and architecture of CNNs, it shows that *edge* images with a size of 340*240 can lead to approximately 100% test accuracy with a CNN architecture where one convolution layer and one fully connected layer are used.

1 Introduction

Deforestation has become a very serious problem around the world. In order to protect plants, a digital database that contains a detailed description of each species should be built and the information collected in the database can be used to identify the known and new plant species [1]. However, there are many challenges to face, due to the large amount of plant species existing on earth. A lot of resources and man power are needed. Usually, it takes a long period of time to train a person to be capable of identifying plant species, which creates a shortage of experts who can assign the labels for plants and help to build the database of plants. Things can become even worse as experts on one species may not be familiar with another species. Scientists have been trying to use the techniques of machine learning to identify plants [2, 3]. Machine learning technology can be used to replace human observation and speed up the analysis.

Plants can be identified by different parts such as leaf, fruit and flower, *etc.* The parts that are mostly widely used are the leaves of plants. Leaves are present during most of the lifetime of a plant and are often abundant. Leaves contain distinctive shapes and textures that usually change very little after the leaves are mature. The leaf of one plant is usually very different from the leaf of another plant. Different types of leaves are shown in Fig. 1.

Convolutional Neural Networks (CNN) [5] are widely used by researchers to do image processing and classification. In our project, we will implement a CNN using *TensorFlow* [6] to deal with plant leaf identification. By training input leaf images with different label, which is the type of a plant, we want to build a CNN to identify the names of test images. This is a supervised learning study.

Normally it takes a long time to train a deep neural network. A deep CNN usually contains thousands of weight components. If the training is done on a central processing unit (CPU), the time consumption might be very long, which hinders the research for scientists. Here in this project

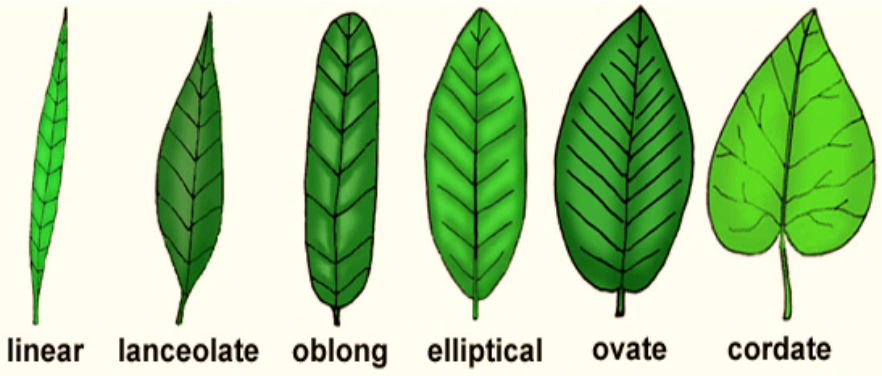


Figure 1: Different shapes of leaves [4]

we will address the acceleration of plant leaf identification with parallel computing techniques. The CUDA (Compute Unified Device Architecture) Deep Neural Network (cuDNN) [7] is a toolkit that can be used to accelerate the training of a CNN. A noticeable speedup can be achieved by taking advantage of graphics processing unit (GPU). The contribution of this projet include:

- An evaluation of the effects of different input image sizes
- An evaluation of the different architechtures of CNNs
- An evaluation of the different image pre-processing techniques

The rest of this report is organized as follows: Section 2 discusses about some related work on using machine learning approaches to analyze plant leaves. In Section 3 we briefly introduce CNN. The data set used in this project is shown in Section 4 and we discuss our experiment design to perform a systematical study of the data set. In Section 5 we show the result of our experiments. We conclude the project in Section 6 and discuss ideas for future work.

2 Related work

Machine learning methods have become increasingly popular to solve identification problems in different fields. In biology and agriculture, machine learning is also widely used to study morphological features of plants. The features of identifying a leaf should be extracted and used for training a classifier [8]. Features may be computed from 2D regions within the leaf image (region-based) [9] or from the contour of the leaf (contour-based) [10]. Both region-based and contour-based methods have been applied to leaf shape recognition. For example, a so-called Centroid Contour Distance (CCD) method is a contour-based approach [11] which calculates the distance between the midpoint and the points on the edge corresponding to interval angle. The multi-scale arch height (MARCH) method [12] was proposed to use hierarchical arch height features.

The shape, color and texture features are common features involved in several applications [13]. The characterization of texture, shape, and color properties was also proposed for leaf classification [14]. Other works have extensively analyzed the different features of leaves. The invariant recognition of binary images of tree leaves was considered [15].

Different image processing techniques were applied to leaf images to effectively extract features of leaves. Invariant moments proposed by Hue [16] were used to recognize objects [17], including leaves of plants [18]. The color information in invariant moment method [18] is missing, however, Man *et al.* used color moments for plant classification [2]. Enhancing the contrast and quality of images, histogram equalization and region of interest segmentation have been used [14, 19]. Chaki and Parekh [20] used image processing for feature extraction to classify species with a neural network. The features obtained by different methods were fed into a classifier such as probabilistic neural network [11], nearest-neighbor [21] and support vector machines[14].

The convolutional neural network (CNN) offers a new approach to learn features automatically from the data set. It can perform both feature extraction and classification. The use of CNN on the leaf identification task has become popular in recent years. For example, a mobile application for leave identification was developed by utilizing a CNN [3]. The use of parametric rectified linear units (PReLUs) instead of ReLUs with a grayscale input image was studied [22]. Reyes *et al.* [23] used images of the entire plant for classification. Lee *et al.* [24] classified 44 species using CNNs. There was also a hybrid approch where hand-crafted and CNN features are used together to feed in a CNN [25].A feedforward neural network derived from the Bayesian network [26] was used to recognize and classify 32 kinds of plants with 12 different leaf features. A recent work used 4-layer CNN and GPU-based acceleration [27] for crop plants.

3 Convolutional Neural Network

In this section the basic knowledge of CNN is introduced and the structure of CNN is explained from the top to bottom. A CNN contains many layers that can extract many features from input images. Every layer have different functions, such as convolution or sub-sampling. The most commonly used layers are discussed here.

3.1 Convolutional Layer

The convolutional layer is the most important layer in the whole structure of a CNN. The function of this layer is the same to the convolution used in the field of image processing (as shown in Fig. 3). A matrix called kernel is used to extract certain features through the operation of convolution. Different kernels are able to extract different kinds of features from the input images. It is important for a training process to find the best weight components which consist of the kernels. The kernels can extract the most useful features of identifying different images, which can be illustrated by Fig. 4 where some feature images are obtained by using Average kernel, Laplacian kernel and Sobel kernel.

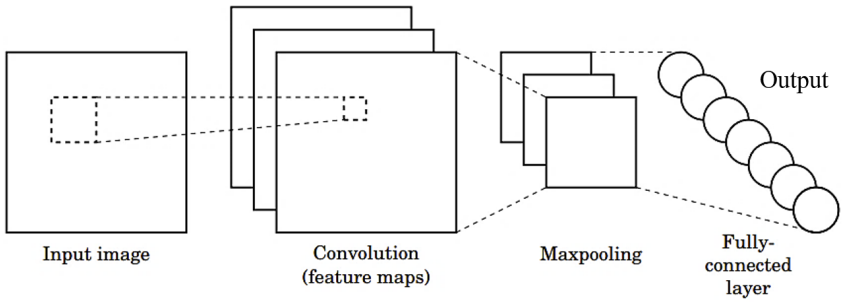


Figure 2: The structure of a simple CNN

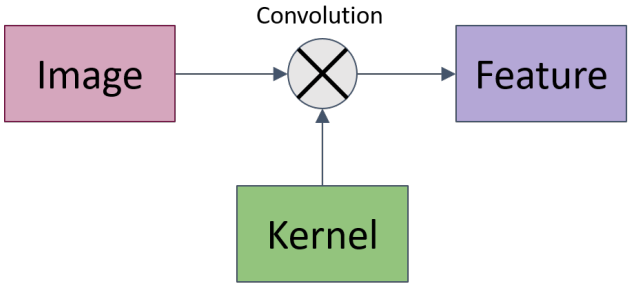


Figure 3: A diagram showing the operation of convolution

(a) (b) (c) (d)

Figure 4: Feature images obtained by using different kernels. (a)Original image. (b)Average kernel. (c) Laplacian kernel. (d) Sobel kernel.

3.2 Max Pooling Layer

The max pooling layer is used to do the down-sampling of input images of this layer. Due to the large size of the input image, it is preferred to reduce the number of nodes in the next layer. The intuition is that once a feature has been found, its exact location is not that important but the key information is the locality which gives the location of one feature relative to other features. The reduction in the spatial size maintains the relative positions of the features in the input image. The input image for this layer is divided into many small non-overlapping regions. In every region, the

Table 1: Hyperparameter used in this project

Learning rate	Batch size	Convolution layer kernel	Convolution layer kernel size	Max-Pooling layer size	Fully connected layer node
0.0001	32	8	3*3	3*3	64

maximum value in this region is used as the output.

3.3 Normalization Layer

Normalization layer is used to do a normalization of the matrix. It is usually used to increase the performance of a CNN. It is useful especially if the rectified linear unit (ReLU) is used as the activation function. As the output of ReLU is unbounded, with normalization, the values will be reduced if they are uniformly large and the relatively large value will be boosted.

3.4 Fully Connected Layer

A dense layer is fully connected to its previous layer. The output from the layer before the dense layer is still a 2D matrix which can be considered as an image. The fully connection to this image makes the input of the dense layer as a 1D vector which contains all the pixels of the image. After the dense layer, the output can be processed as a 1D vector, similar to that in a ordinary neural network. This layer is used near the end of the CNN.

4 Experiment Data Set and Design

The data set used in this project is the Flavia data set [26]. This data set contains 1907 RGB images and 32 species of plants. Each species of plant has $50 \sim 70$ images. Each image in this data set has a size of 1600*1200. The goal is to identify the leaves for these 32 species.

In this project, *TensorFlow* is used to implement a CNN for the classification. The *TensorFlow* contains APIs for different types of layers which support easy design of a CNN. A CNN is a kind of deep neural network which contains a lot of matrix operation and this type of computation can be well accelerated by cuDNN [7]. A NVIDIA GTX 980 Ti graphic card is used in this work.

The key experiment parameters are given in Table 1. These parameters are known as hyperparameters which cannot be directly learned from the regular machine learning process such as learning rate, convolution layer kernel size, nodes number in fully connected layer, *etc.* The structure of the CNN used in this project is adapted from the one used in the tutorial of *TensorFlow* for CIFAR-10 data set [28]. A diagram of this structure in shown in the left of Fig. 5. This architecture is referred as 2C+2FC model since 2 convolution (C) layers and 2 fully connected (FC) layers are used. Table 1 shows the hyperparameters used in this project. The total number of iterations is set to 6000.

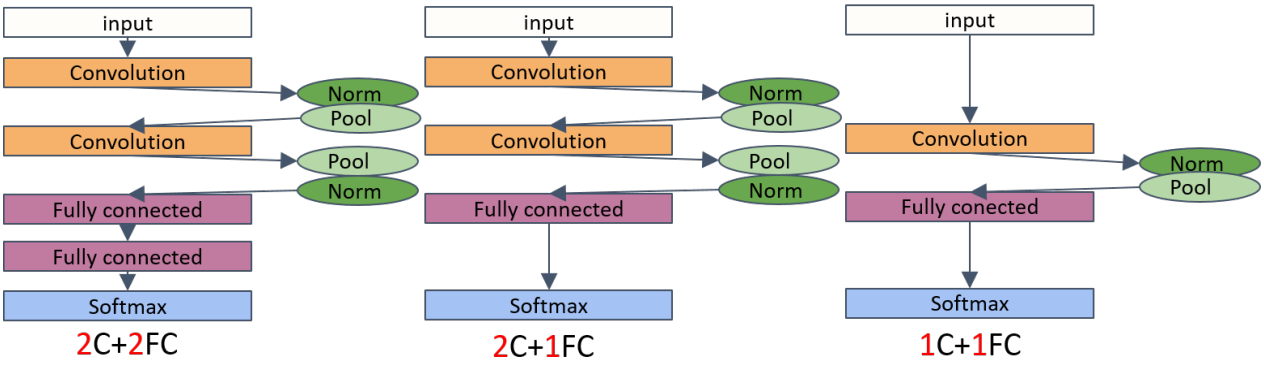


Figure 5: The structure of the CNN used in this project. The abbreviation name of the architecture is labeled below the schematic graph where C represents the convolution layer and FC means the fully connected layer.

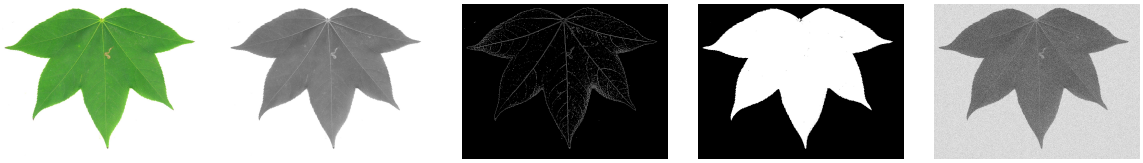
In this work, we systematically evaluate the effects from image size, architecture of CNN and image type obtained from preprocessing on the training and test results.

4.1 Image Size

The image size plays an important role in training a CNN. The original size of the image in the Flavia data set is 1600*1200, which is relatively large when the images are used as the input of a CNN. If a large image size is used, although the details in the image will be kept, the memory and computational costs will be very high. If a small image size is used, although the computational resources used can be reduced, useful information and key features in the images may be lost. A balance between the details in the images and the memory and computational costs needs to be achieved. We perform systematically studies for images with different sizes obtained by cropping or rescaling of original images.

4.2 Architecture

The architecture to start with is referred in this work as 2C+2FC model since 2 convolution layers and 2 fully connected layers are used. The reason that we choose this architecture which was used for CIFAR-10 data set is due to the similarity between our data set and CIFAR-10. Both data sets are used to do multi-classes classification for RGB images. Even if this is a good structure to start with, this may not be the optimal structure for the Flavia data set which contains only green leaves in front of a white background, which is different from CIFAR-10 data set. We want to test different architectures of CNNs and compare the training processes. We can remove one fully connected layer from the 2C+2FC model and obtain what is referred as 2C+1FC model. We can further simplify the network by removing 1 convolution layer which results in an architecture



(a) (b) (c) (d) (e)

Figure 6: Results of different image preprocessing techniques. (a) Original image. (b) Color removal. (c) Edge detection. (d) Binarization. (e) Noise corruption.

with only 1 convolution layer and 1 fully connected layer. This is referred as 1C+1FC model. The diagrams of these architectures are shown in Fig. 5. By tuning different architectures , we may be able to find the optimal one among these three.

4.3 Image Preprocessing

The CNN can be used to extract features for the input images. It is worth pointing out that it may not be able to extract some features efficiently in limited training steps. We want to test if image preprocessing where the images are processed by convoluting them with some known kernels will help train a CNN. In this experiment we will try a series of image preprocessing techniques [29], such as color removal, binarization, edge detection, and noise corruption. A comparison of the visual effects after applying these techniques is shown in Fig. 6.

5 Experiment Results

5.1 Training and Test Accuracy

First, we show the effects from image size with given architecture and input image type. RGB images with different sizes are fed into a 2C+2FC network. Table 2 shows detailed results of tuning five image sizes at the end of 6000 steps of training. In Fig. 7, the detailed training records are shown. For larger input image size, the training converges at the end of 6000 steps. For smaller input image size, a longer training process is expected. A hypothesis is that the larger images contain more information and more valid features that can be extracted by the convolution layer, which makes the training more efficient.

In order to verify this hypothesis, we use different CNN architectures to train the same input images. The experiment results are shown in Fig. 8 and Fig. 9 for CNN architecture of 2C+1FC and 1C+1FC respectively. For the five image sizes, the training and test accuracy plots follows the same tendency as that with the 2C+2FC architecture, namely that the larger image size leads to a quicker convergence of training. The training performance is improved for smaller input sizes with the removal of one fully connected layer. However, there is no noticeable change when one more convolution layer is reduced. For the same input size, it is shown by our results that smaller CNN

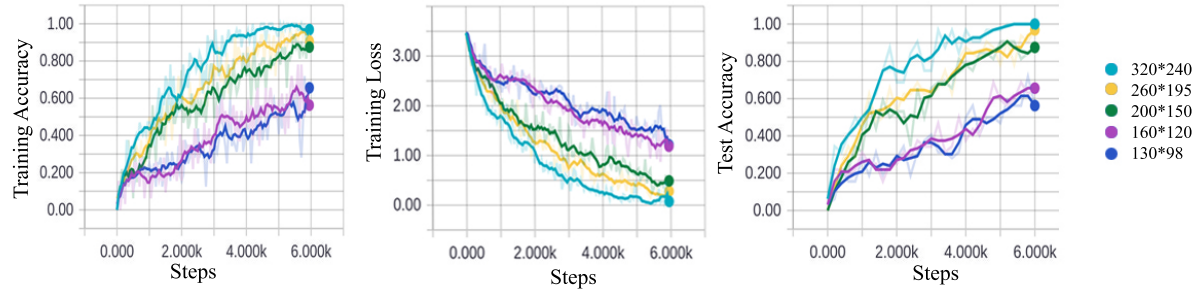


Figure 7: Accuracy and loss for different image sizes (with a CNN of 2C+2FC structure). Left panel: training accuracy. Middle panel: training loss. Right panel: test accuracy

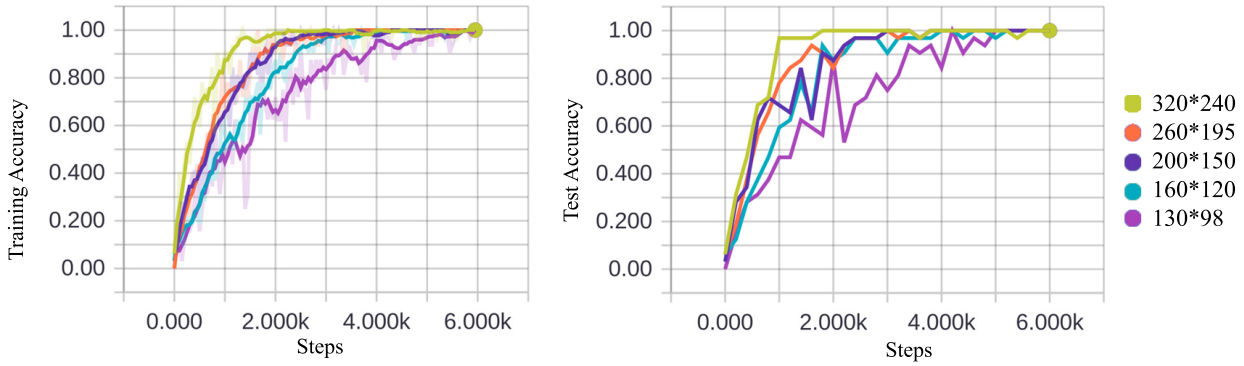


Figure 8: Accuracy for different image sizes (with a CNN of 2C+1FC structure). Left panel: training. Right panel: test

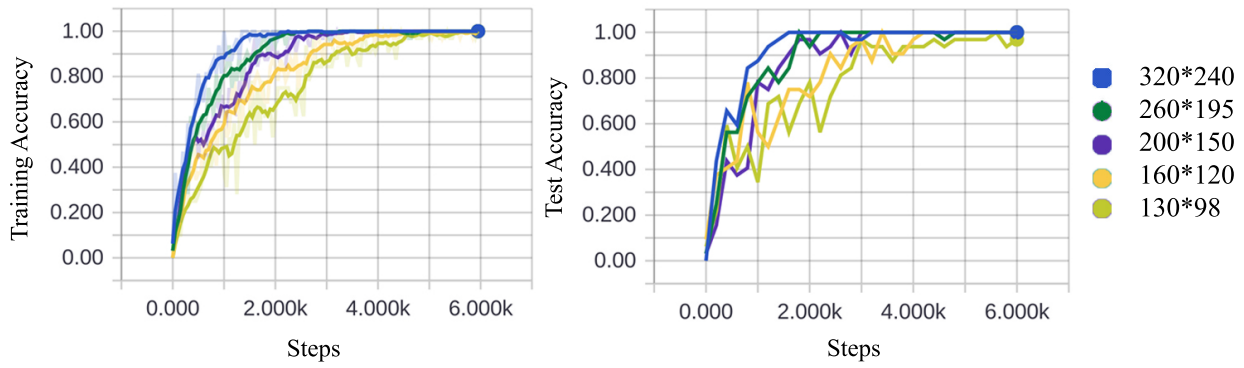


Figure 9: Accuracy for different image sizes (with a CNN of 1C+1FC structure). Left panel: training. Right panel: test

Table 2: Accuracy and loss for images with different sizes at the end of training (6000 steps)

Image size	Training accuracy	Training loss	Test accuracy
320*240	0.9688	0.07607	1.000
260*195	0.9063	0.2831	0.9688
200*150	0.8750	0.4897	0.8750
160*120	0.6563	1.118	0.6563
130*98	0.5625	1.217	0.5625

architectures give quicker convergence. All three architectures used in this work show consistent tendency of convergence, which proves our hypothesis that with a smaller image size, it takes more time to train a CNN. The implication of these results is that the CNN architecture should match the image size.

In our work, the 1C+1FC architecture gives the optimal training speed with a image size of 320*240. In the rest of this report, without specification, we will use this optimized architecture and input image size for the study of different input image types. Grayscale images are frequently used to train a CNN when the color of objects does not dominate classification. In leaf classification, since all the leaves are green, it is reasonable to feed a CNN with just grayscale images. The training and test accuracy with RGB images and grayscale images with size 320*240 are shown in Fig. 10. The training and test accuracy are high for both RGB images and grayscale images but the fluctuation is more severe for grayscale images. Although the color is in green for all the leaves, the levels of green are different. The dark or light green colors have high correlation with the types of leaves, which may contribute to the stability of the accuracy for RGB images. In addition to grayscale images, another two types of images are used. One is the binary image and the other is the edge image which is obtained by convolving the original grayscale image with the Sobel filter [29]. We plot the results for all three different image types, where the image size is 320*240 and the architecture of CNN is 1C+1FC. The training and test accuracy are shown in Fig. 11. The binary images lead to the worst classification where the training accuracy is stuck at around 50%. The curve is relative flat after 2000 steps and the accuracy is not improved even with longer training time. We thought the reason for this is that the information loss in binary images is severe as information about the veins of a leaf is lost in binarization. However, the edge image leads to the best results among all the input image types. The edge images contain all the edges and curvature information including veins of leaves. The shapes of veins are very important in identifying the leaves. The Sobel filter will filter out the white background and other homogeneous regions inside the leaves. The most important features are fed into the CNN, which makes the convergence quicker. After 2000 steps, the training accuracy is already 100%. From the results above, it can be concluded that for leaf classification, the edge images lead to the quickest convergence. The edges and gradient information in images of leaves play a vital role in leaf classification.

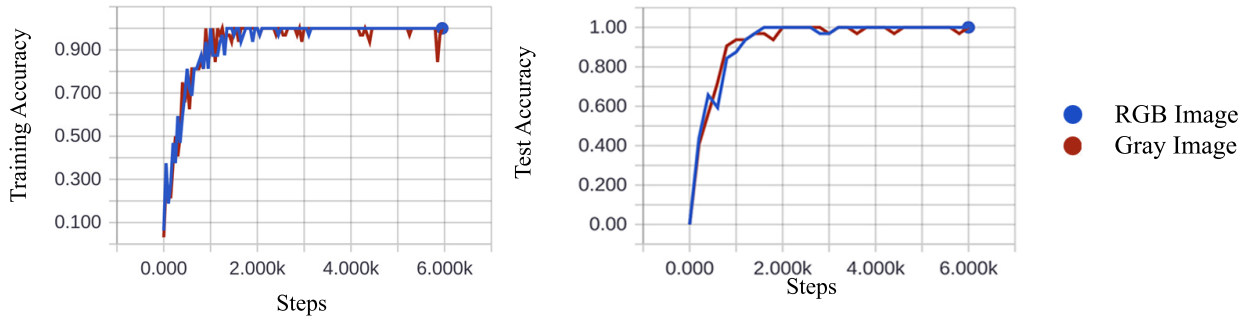


Figure 10: Accuracy for RGB images and grayscale images.

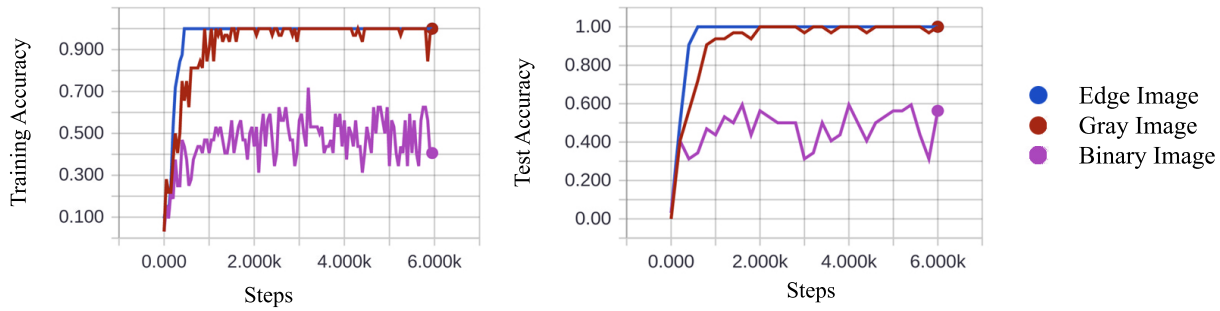


Figure 11: Accuracy for grayscale images, edge images, and binary images.

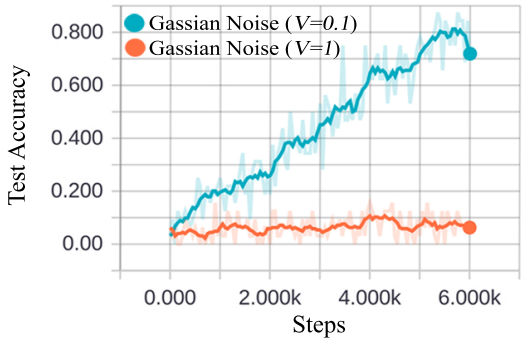


Figure 12: Test accuracy for images corrupted by Gaussian white noise. V is the variance of noise.

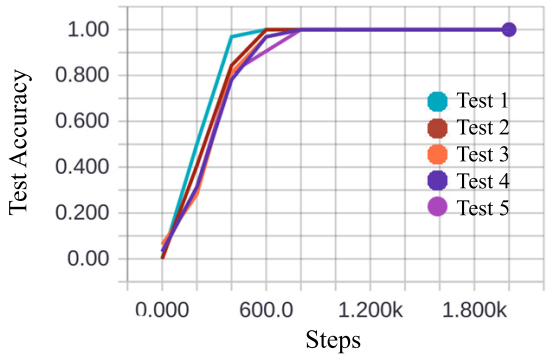


Figure 13: Test accuracy for 5-fold cross validation.

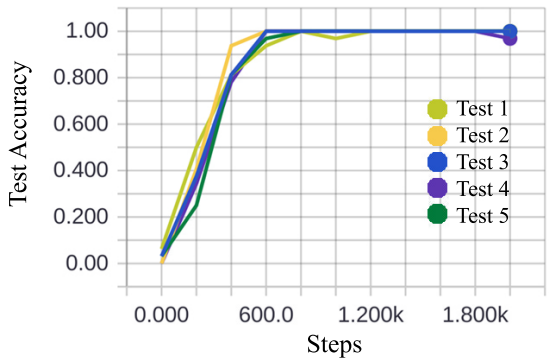


Figure 14: Test accuracy for 5 tests with randomization.

The accuracy for this data set is very high and this is mainly due to the high quality of the images. In 2014, Milan Sulc *et al.* [30] have already obtained 99.7 ± 0.3% accuracy for the same data set by using SVMs. Usually, the images obtained through cameras contain some level of noise. In order to simulate this scenario [1], Gaussian white noise is added to the grayscale images which are used as input images. Two cases are tested and the variance of the Gaussian noise is set to 0.1 and 1, respectively. The mean of Gaussian noise for each case is set to zero. The accuracy rates for these two cases are shown in Fig. 12. With a larger variance, the CNN does not converge and the test accuracy is less than 10% which is not improved with the increase of training steps. For a small noise level, the CNN can extract the right features to identify leaves and the test accuracy is slowly increasing with more training steps. The increase of the test accuracy is very slow compared with that when RGB images are used. This showed that the CNN does not have a very good resistance to noise.

5.2 Cross Validation and Randomization

From the systematical study shown above, we find the optimized CNN architecture (1C+1FC) and input image size (320*240) as well as the best input image type ("edge" images). With these set-up, we would like to evaluate the test accuracy by statistical methods. To validate the results, two methods are used. One is 5-fold cross validation, the other is randomization. A 5-fold cross-validation method is used to better measure the accuracy for each data set. The test accuracy are shown in Fig. 13. It shows the 100% test accuracy for each single run of 5-fold cross validation.

We also used a randomization method. For every session of run, 80% of randomly selected samples are used as the training set, while the rest is used as test set. The plot of test accuracy are shown in Fig. 14. There is only one test accuracy is 96.88% which is lower than 100% and the averaged test accuracy for 5 times randomization of entire data set is (4*100%+ 96.88%)/5 = 99.376%. The edge images as input images give very high test accuracy and the CNN is trained quickly.

5.3 Confidence Interval

The 95% confidence interval is found for the results. Due to the small number of samples in this data set, the lower limit of the confidence is close to 95.5%.

6 Conclusion and Future Work

In this project, we used *TensorFlow* to implement a CNN for the classification Flavia data set. We test the effects of input image sizes and find out that a larger image size usually lead to quicker convergence. The effects of different architectures of CNN have also been tested. For Flavia data set, the architecture that works best is the 1C+1FC, which means that a deeper CNN may not always lead to better results and it's necessary to find the suitable architecture through a series of

[1] We would like to thank Dr. Scott for the suggestions.

tests. Although CNN is a very good tool to extract features from images, it may not always be able to perform well. In this experiment, we tested many image preprocessing techniques, including color removal, binarization, edge detection, and noise corruption. We have found out that some operations do not have obvious effects, such as color removal but some operations like binarization lead to poor results, and the most interesting finding is that edge detection can lead to the best result. With 5-fold cross validation, the test accuracy rate can reach nearly 100%. The effects of noise corruption is also tested. It seems that the CNN has low tolerance to noise. In order to get good results, a noise reduction step may be necessary for images with noise in the future work.

We have learned from this experiment that there are various factors that may affect building a good classifier. The accuracy rates for this data set are very high which are mainly due to the high quality of the images and the relatively simple classification task. We have observed the trend of the effects of changing a particular parameter. However, more tests are needed to verify the conclusions we made in this experiment. The data set used in this experiment is relatively simple since all the images were taken in a controlled environment. In a more complex data set, if the images are taken in fields with cell phone, the preprocessing of images may make a huge difference between success and failure because of the noise or other factors introduced by the uncontrolled environmental variables. If a more complex data set is used, the optimal image size and architecture may change. It would be useful to find the trend of the effects of changing all these parameters and see if a general rule can be found through these tests.

References

[1] A. Grainger, *Controlling tropical deforestation*. Routledge, 2013.

[2] Q.-K. Man, C.-H. Zheng, X.-F. Wang, and F.-Y. Lin, "Recognition of plant leaves using support vector machine," in *International Conference on Intelligent Computing*. Springer, 2008, pp. 192–199.

[3] T. J. Jassmann, R. Tashakkori, and R. M. Parry, "Leaf classification utilizing a convolutional neural network," in *SoutheastCon 2015*. IEEE, 2015, pp. 1–3.

[4] "Botany 115 terminology," http://waynesword.palomar.edu/termlf2.htm, accessed: 2017-04-26.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[6] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[7] S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro, and E. Shelhamer, "cuDNN: Efficient primitives for deep learning," *arXiv preprint arXiv:1410.0759*, 2014.

[8] I. K. Kazmi, L. You, and J. J. Zhang, "A survey of 2D and 3D shape descriptors," in *Computer Graphics, Imaging and Visualization (CGIV), 2013 10th International Conference*. IEEE, 2013, pp. 1–10.

[9] D. Zhang and G. Lu, "Review of shape representation and description techniques," *Pattern recognition*, vol. 37, no. 1, pp. 1–19, 2004.

[10] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE transactions on pattern analysis and machine intelligence*, vol. 24, no. 4, pp. 509–522, 2002.

[11] A. Hasim, Y. Herdiyeni, and S. Douady, "Leaf shape recognition using centroid contour distance," in *IOP conference series: earth and environmental science*, vol. 31, no. 1. IOP Publishing, 2016, p. 012002.

[12] B. Wang, D. Brown, Y. Gao, and J. La Salle, "March: Multiscale-arch-height description for mobile retrieval of leaf images," *Information Sciences*, vol. 302, pp. 132–148, 2015.

[13] P. Hiremath and J. Pujari, "Content based image retrieval using color, texture and shape features," in *Advanced Computing and Communications, 2007. ADCOM 2007. International Conference on*. IEEE, 2007, pp. 780–784.

[14] B. VijayaLakshmi and V. Mohan, "Kernel-based pso and frvm: An automatic plant leaf type detection using texture, shape, and color features," *Computers and Electronics in Agriculture*, vol. 125, pp. 99–112, 2016.

[15] K. Horaisová and J. Kukal, "Leaf classification from binary image via artificial intelligence," *Biosystems Engineering*, vol. 142, pp. 83–100, 2016.

[16] S. Theodoridis, A. Pikrakis, K. Koutroumbas, and D. Cavouras, *Introduction to pattern recognition: a matlab approach*. Academic Press, 2010.

[17] M. Mercimek, K. Gulez, and T. V. Mumcu, "Real object recognition using moment invariants," *sadhana*, vol. 30, no. 6, pp. 765–775, 2005.

[18] Z. Zulkifli, P. Saad, and I. A. Mohtar, "Plant leaf identification using moment invariants & general regression neural network," in *Hybrid Intelligent Systems (HIS), 2011 11th International Conference on*. IEEE, 2011, pp. 430–435.

[19] J. S. Cope, D. Corney, J. Y. Clark, P. Remagnino, and P. Wilkin, "Plant species identification using digital morphometrics: A review," *Expert Systems with Applications*, vol. 39, no. 8, pp. 7562–7573, 2012.

[20] J. Chaki and R. Parekh, "Plant leaf recognition using shape based features and neural network classifiers," *International Journal of Advanced Computer Science and Applications*, vol. 2, no. 10, 2011.

[21] M. M. de Souza, F. N. Medeiros, G. L. Ramalho, I. C. de Paula Jr, and I. N. Oliveira, "Evolutionary optimization of a multiscale descriptor for leaf shape analysis," *Expert Systems with Applications*, vol. 63, pp. 375–385, 2016.

[22] Y.-H. Wu, L. Shang, Z.-K. Huang, G. Wang, and X.-P. Zhang, "Convolutional neural network application on leaf classification," in *International Conference on Intelligent Computing*. Springer, 2016, pp. 12–17.

[23] A. K. Reyes, J. C. Caicedo, and J. E. Camargo, "Fine-tuning deep convolutional networks for plant recognition." in *CLEF (Working Notes)*, 2015.

[24] S. H. Lee, C. S. Chan, P. Wilkin, and P. Remagnino, "Deep-plant: Plant identification with convolutional neural networks," in *Image Processing (ICIP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 452–456.

[25] D. Hall, C. McCool, F. Dayoub, N. Sunderhauf, and B. Upcroft, "Evaluation of features for leaf classification in challenging conditions," in *Applications of Computer Vision (WACV), 2015 IEEE Winter Conference on*. IEEE, 2015, pp. 797–804.

[26] S. G. Wu, F. S. Bao, E. Y. Xu, Y.-X. Wang, Y.-F. Chang, and Q.-L. Xiang, "A leaf recognition algorithm for plant classification using probabilistic neural network," in *2007 IEEE international symposium on signal processing and information technology*. IEEE, 2007, pp. 11–16.

[27] S. Donné, B. Goossens, S. Dhondt, N. Wuyts, H. Luong, D. Inzé, and W. Philips, "GPU-based maize plant analysis: accelerating CNN segmentation and voxel carving," in *NVIDIA European GPU Technology Conference (GTC)*, 2016.

[28] Google Inc. (2016) Convolutional Neural Networks. [Online]. Available: https://www.tensorflow.org/tutorials/deep_cnn

[29] C. Solomon and T. Breckon, *Fundamentals of Digital Image Processing: A practical approach with examples in Matlab*. John Wiley & Sons, 2011.

[30] M. Sulc and J. Matas, "Texture-based leaf identification," in *European Conference on Computer Vision*. Springer, 2014, pp. 185–200.