

# Palmately Lobed Leaf Image Classifier Using TensorFlow

Joshua Aditya Kosasih - 13514012

Informatics/Computer Science

School of Electrical Engineering and Informatics

Institut Teknologi Bandung, Jl. Ganesha No 10 Bandung, Indonesia

13514012@std.stei.itb.ac.id

**Abstract**—Nowadays the technology has brought us closer to smarter computer that can ‘see’ and perceive the world like us. Powered by machine learning, computer can identify and classify an image its ‘looking at’. Whilst classifying simple and distinguishable objects are easy, the classification for leaf with the same structure can be very tricky. To create the model for classifying the leaves from scratch can be really hard; and that is where a tool like TensorFlow can help.

**Keywords**—*machine learning, classification, leaf, TensorFlow,*

## I. INTRODUCTION

The world is advancing fast and digital technology is the main contributor for revolutionizing it. In the early 2000s simple tasks like calculating a number, doing some simple procedure, etc. were replaced by computer. But throughout the decade, more complex and challenging task – that we thought only a man can do – also gets replaced by computer. Artificial intelligence enables computers to be smarter and capable of doing complex tasks. In general use, the term Artificial Intelligence means a machine which mimics human cognition.

One of the most basic cognition a human have is when we react or think after we see something; for eyes is our most frequently used sensor. When we see something, we can acknowledge it, identify it, analyze it, classify it. Our brains make vision seem easy. It doesn't take any effort for humans to tell apart a lion and a jaguar, read a sign, or recognize a human's face. But these are actually hard problems to solve with a computer: they only seem easy because our brains are incredibly good at understanding images.

Computer vision is one branch of Artificial intelligence, the science of computers that can see. Computer vision is concerned with the theory behind artificial systems that extract information from images. With that way, computers can ‘see’ things and able to identify and classify it.

The ability for computers to recognize images has many benefits for human life. Computer can do automatic inspection on parts repeatedly without getting tired. They can help human in identification tasks. They could also monitor and detect events for hours non-stop. This paper will further discuss the use of

computer's image recognition for identification task which is a classification problem.

## II. THEORY AND CONCEPT

### A. Machine Learning

Machine learning according to Arthur Samuel in 1959, gives "computers the ability to learn without being explicitly programmed." Machine learning explores the study and construction of algorithms that can learn from and make predictions on data. Machine learning tasks are typically classified into three broad categories:

- Supervised learning
- Unsupervised learning
- Reinforcement learning

Classification problem are mostly handled supervised way. With an algorithm that is not explicitly programmed, machine learning algorithm can classify wide range of images without the need to be coded specifically for each classification task. There are many approaches for machine learning, such as using:

- Decision tree learning
- Artificial neural networks
- Deep learning
- Inductive logic programming
- Support vector machines
- Clustering
- Bayesian networks
- Genetic algorithms
- Rule-based machine learning
- Learning classifier systems

The approach that will be used for this paper is deep learning. Deep learning is closely related to Artificial neural network, which is a learning algorithm that is inspired by neurons that construct a brain. Deep learning consists of multiple hidden layers in an artificial neural network. This approach tries to model the way the human brain processes light and sound into vision and hearing. Deep Learning methods have provided a way to automatically learn good representations of data from labeled or unlabeled samples. Deep learning enables end-to-end training of these architectures, from raw inputs to ultimate outputs. The

use of deep learning in this paper is caused by its versatility and sophistication.

### B. Convolutional Neural Network

The convolutional network model (CNN) is a particular type of deep learning somewhat inspired by animal visual cortex, which consists of multiple stages of filter banks, interspersed with non-linear operators, and spatial pooling. Convolutional neural networks (CNNs) consist of multiple layers of receptive fields. These are small neuron collections which process portions of the input image. The outputs of these collections are then tiled so that their input regions overlap, to obtain a more abstract representation of the original image; this is repeated for every such layer. CNN have become the record holder for a wide variety of benchmarks, including object detection, localization in image, face recognition, handwriting recognition, biological image segmentation, etc.

Convolutional nets consist of several component layers:

1. Convolutional layer
2. RELU layer
3. Pooling layer
4. Fully-connected layer

Convolutional layer filters through the image for a specific pattern. Convolutional layer can be composed of several filter layers each looking for unique pattern different from the other filters. Each filter look for the same pattern in different region area. This can be done because of the same weight and biases the neurons in the same filter share. RELU or rectified linear unit layer allows the net to be properly trained and safe from vanishing gradient problem. Pooling layer is used for dimensionality reduction. It ensures that the net focused on only the most relevant patterns discovered by convolution and RELU. Fully-connected layer is used to equip the net with the ability to classify data samples as the previous layers can discover complex patterns but will have no understanding of what those patterns mean.

CNN typically has sets of these three layers (convolutional, RELU, and pooling) and several fully-connected layers in between the sets – all of which repeated several times. Each set usually takes into account a level of abstraction desired for recognizing the images. From the first set with the lowest abstraction like recognizing edges and points, then to the next set to recognize shapes, until the last set to recognize the object, thus classifying it.

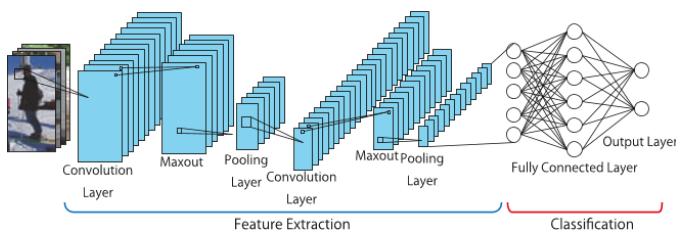


Figure 1. Convolutional neural nets layers

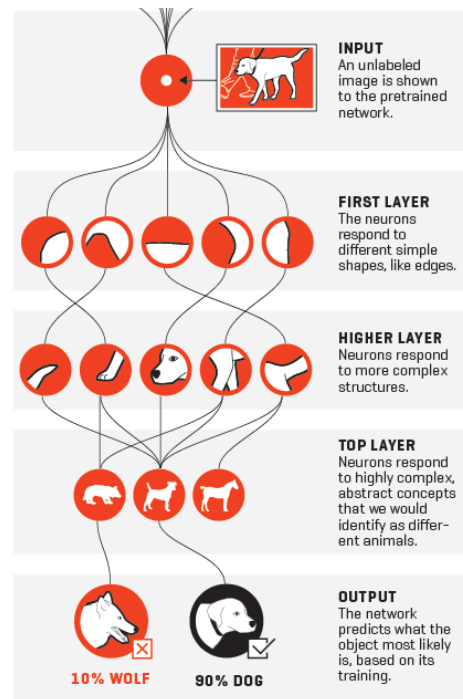


Figure 2. Steps of image processing in neural network.

### C. Palmately Lobed Leaf

A leaf belongs into one of the four basic leaf structure: simple, lobed, dissected, and compound. These structures made it easier to classify a leaf just by a glance. But when the leaves share the same structure, it would be harder to classify them.

Lobed leaves are leaves with distinct protrusions, either rounded or pointed. Palmately lobed leaves have the lobes spreading radially from a point, like fingers on a hand. All lobed leaves share the same pattern that differs them from other leaves which is their diverging leaf structure.

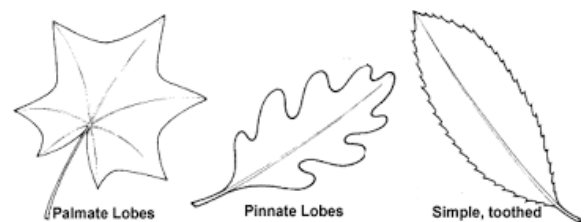


Figure 3. Leaf classification by structure

Palmately lobed leaves that will be used for classification in this paper are:

- Papaya leaf
- Cassava leaf
- Cannabis leaf

All of the above leaves are quite identical because they have the same structure. The most obvious differences they have are their margins. Even as a human it is quite hard to distinguish those leaves in some conditions. Cassava don't look much like cannabis up close, but they do have a similar leaf shape.



Figure 4. Side by side cannabis(left) and cassava(right) leaves

### III. SOLUTION ANALISYS

To be able to identify and classify images from those three types of leaf, we need a CNN model to become the classifier. But to create it from scratch would be very time consuming and we need a long-dedicated time to develop it. And even though somehow, we successfully managed to build the model, there comes new problem which is to train the model. The training process needs lots of time and computing power to get a high accuracy of predictions. From that problem on, we decided to take a look at a tool for machine learning: TensorFlow.

#### A. TensorFlow

TensorFlow is an open source software library for machine learning across a range of tasks, and developed by Google to be capable of building and training neural networks to detect and decipher patterns and correlations, analogous to the learning and reasoning which humans use. TensorFlow uses data flow graphs for its numerical computation. Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) communicated between them.

TensorFlow provides multiple APIs. It provides a set of primitives from which Machine Learning engineers and researchers can construct trainable models—as well as a framework to run these computations in an efficient way. With TensorFlow we can create a CNN model without the need to build it from scratch. And with that, eliminating our first problem in making the palmate leaf classifier. But there is still the second problem, training the model.

#### B. Inception

Inception is a pre-trained neural network model. Inception was trained by Google on one hundred thousand images with over a thousand category. But even Inception was not trained to classify papaya, cassava, and cannabis leaves. So, we need to do a process called transfer learning.

Transfer learning means applying a learning from previous training session to a new training session. Like all other CNN model, the Inception has the last layer devoted to classify the object it sees. With transfer learning, we just need to retrain that

last layer with features of palmate leaves so we can add a representation of the leaves in its repository of knowledge.

So, finally we can train our model with such short time because we only need to train the last layer of the model with transfer learning. And with that, we have solved the two main problems of creating a palmate leaf image classifier.

## IV. IMPLEMENTATION AND TESTING

### A. Implementation

To make the classifier we need to install the TensorFlow to get the tools we need and then get the sufficient dataset to train the Inception model using transfer learning. After the model has been trained, it then can be used to classify test cases.

#### 1) Install TensorFlow

Installing TensorFlow can be done in two ways, either using *Docker* or without using Docker (with the first one being the easier). For this paper we will use Docker.

Docker is a software container platform. Using containers, everything required to make a piece of software run is packaged into isolated containers. With Docker we don't have to install and configure complex databases nor worry about switching between incompatible language toolchain versions.

So we need to install Docker to the system and then install TensorFlow in Docker. Installing TensorFlow in Docker can be done by simply entering a line of code which request to create a container that contains tensorflow code base.

#### 2) Get Dataset

For training the final layer of the Inception we need at least a hundred images of the leaf or plant for each category: papaya, cassava, and cannabis. The source of images is from Google Image Search. After all the data has been downloaded, we collected a total of 188 photos of papaya leaves and plants, 127 photos of cassava leaves and plants, and 179 photos of cannabis leaves and plants. All these photos need to be stored in folders that has been named according to the class' name.

#### 3) Retrain Model

Before retraining the model, the dataset that has been collected need to be linked to the Docker with the following command:

```
docker run -it -v $HOME/tf_files:/tf_files/
gcr.io/tensorflow/tensorflow:latest-devel
```

After the dataset has been linked to the docker, the inception model can be retrained afterwards using *retrain.py* script with these command:

```
python
tensorflow/examples/image_retraining/retrain.py \
--bottleneck_dir=/tf_files/bottlenecks \
--how_many_training_steps 500 \
--model_dir=/tf_files/inception \
--output_graph=/tf_files/retrained_graph.pb \
--output_labels=/tf_files/retrained_labels.txt \
```

```
--image_dir /tf_files/leaves
```

After awhile, the transfer learning process will finish and produce an accuracy number which tells us how confident the model is.

```
2017-05-04 16:35:34.102570: Step 450: Train accuracy = 100.0%
2017-05-04 16:35:34.102685: Step 450: Cross entropy = 0.160108
2017-05-04 16:35:34.199242: Step 450: Validation accuracy = 89.0% (N=100)
2017-05-04 16:35:35.205954: Step 460: Train accuracy = 98.0%
2017-05-04 16:35:35.206041: Step 460: Cross entropy = 0.185200
2017-05-04 16:35:35.300108: Step 460: Validation accuracy = 84.0% (N=100)
2017-05-04 16:35:36.271430: Step 470: Train accuracy = 98.0%
2017-05-04 16:35:36.271519: Step 470: Cross entropy = 0.155087
2017-05-04 16:35:36.364909: Step 470: Validation accuracy = 90.0% (N=100)
2017-05-04 16:35:37.367411: Step 480: Train accuracy = 96.0%
2017-05-04 16:35:37.367505: Step 480: Cross entropy = 0.264250
2017-05-04 16:35:37.466151: Step 480: Validation accuracy = 91.0% (N=100)
2017-05-04 16:35:38.422923: Step 490: Train accuracy = 98.0%
2017-05-04 16:35:38.423000: Step 490: Cross entropy = 0.199555
2017-05-04 16:35:38.520582: Step 490: Validation accuracy = 90.0% (N=100)
2017-05-04 16:35:39.524505: Step 499: Train accuracy = 98.0%
2017-05-04 16:35:39.524618: Step 499: Cross entropy = 0.186212
2017-05-04 16:35:39.629564: Step 499: Validation accuracy = 82.0% (N=100)
Final test accuracy = 93.4% (N=61)
Converted 2 variables to const ops.
```

Figure 5. Percentage of accuracy




#### 4) Create Classifier Script

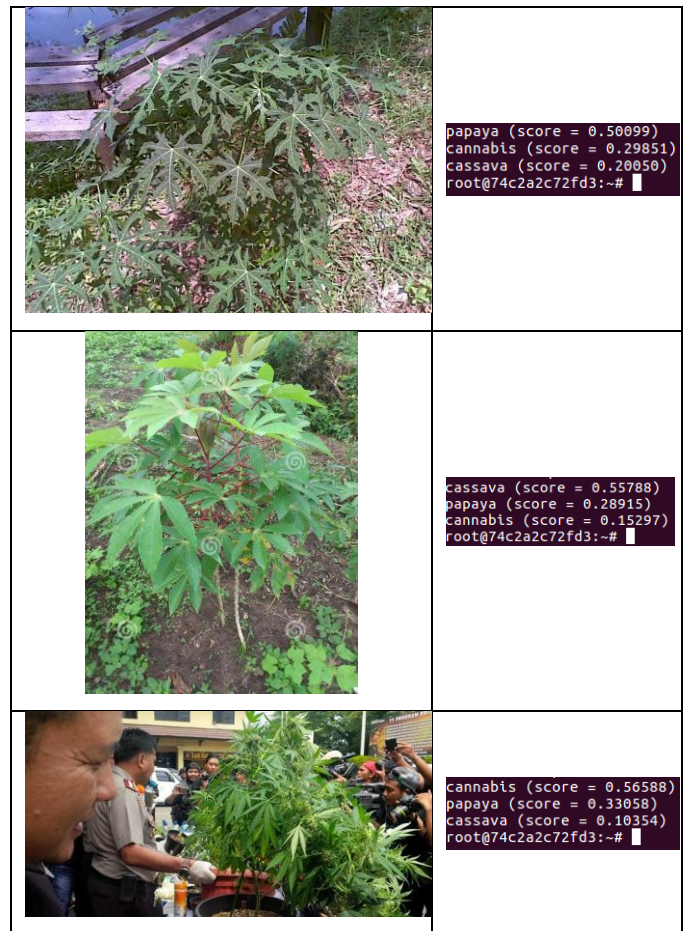
We then create a python script to read in the image\_data, loads label file, strips off carriage return, unpersists graph model from file, and feed the image\_data as input to the graph model and get the predictions.

#### B. Testing

To evaluate the model that we have trained with the dataset, we ran several tests to see its classification scores. Below are the results of the tests:

Table 1. Test results

Image_data	Classification score
	papaya (score = 0.91547) cassava (score = 0.08230) cannabis (score = 0.00224) root@74c2a2c72fd3:~#
	cassava (score = 0.96908) papaya (score = 0.01816) cannabis (score = 0.01276) root@74c2a2c72fd3:~#
	cannabis (score = 0.86581) papaya (score = 0.08988) cassava (score = 0.04432) root@74c2a2c72fd3:~#



#### V. CONCLUSION

A classification problem needs a model to map an input to its corresponding class. To create a model for palmate leaf classifier, we can either build it from scratch or build it using open source library such as TensorFlow. Big amount of good data also needed to train the model so that it has high prediction accuracy.

From the test results listed in chapter 4, we can conclude that the image classifier created for this paper has successfully passed the test and we admit that the system is able to classify a leaf correctly into its corresponding classes.

#### REFERENCES

- CS231n: Convolutional Neural Networks for Visual Recognition <http://cs231n.github.io/convolutional-networks/>, accessed 4<sup>th</sup> May 2017
- Leaf Description Glossary <http://www.cs.rochester.edu/u/nelson/wildflowers/glossaries/leaves/index.html>, accessed 5<sup>th</sup> May 2017
- TensorFlow [www.tensorflow.org](http://www.tensorflow.org), accessed 5<sup>th</sup> May 2017
- Docker [www.docker.com](http://www.docker.com), accessed 4<sup>th</sup> May 2017