**COMPUTER VISION PROJECT**

**SICT – SIMPLE IMAGE CLASSIFIER USING TENSORFLOW**



Prepared For:

Iman H. Kartowisastro, Ph. D. (D1118)

Class: LA08

Prepared By:

Andreas Pangestu Lim – 2001540116

Andrew – 2001554166

Jonathan – 2001542651

July 2nd, 2018

**CHAPTER I – THEORETICAL FOUNDATION**

Technology evolves from time to time. Computer, as the most important part of technology, had underwent many changes since its inception. Early computers were created only as computing device (calculator). Nowadays, computer can do almost anything a human can do. For instance, computer is capable of capturing an image using web camera (webcam) with high resolution, similar to human eye capability. Not only that, computer can also process the image by resizing, cropping, recoloring, applying filter and so on to produce desired image. And even further, modern computer using Artificial Intelligence (AI) is able to detect and recognize certain objects inside an image.

The concepts of Image Capturing, Image Processing, and Object Detection/Recognition have become so popular, they are wrapped into a particular domain in Computer Science called **Computer Vision**. This report will focus more on Object Detection/Recognition: What is it, what is it for, how does it work, and how Machine Learning affects it. This report will also demonstrate the application of Object Detection/Recognition in a simple, yet useful software called SICT (Simple Image Classifier using Tensorflow).

So, what is Object Detection/Recognition? It is quite simple. **Object Detection** talks about how computer detects object in an image/video. **Object Recognition** talks about how computer recognizes object in an image/video. Both are quite similar but remain different. To truly understand the difference, one must first understand the contrast meaning between “detect” and “recognize”.

**“To detect”** means to know whether something exists or does not exist. For example, an automatic door sensor “detects” movement. If there is any, the door opens. Otherwise, the door remains closed. In this case, the sensor does not care about who or what causes the movement.



Figure 1: Two people are going through an automatic door.

**“To recognize”** means to identify something that has been encountered before. For example, a fingerprint sensor in an airport “recognizes” foreigner’s fingerprint. If the fingerprint belongs to an international criminal, an alarm will trigger. Otherwise, nothing will happen. In this case, the fingerprint sensor had already saved a database which contains collection of criminal’s fingerprint. So, when a criminal does fingerprint scan, the sensor recognizes it as a criminal’s fingerprint. But when an ordinary tourist does a fingerprint scan, the sensor will not recognize it.



Figure 2: A person is doing fingerprint scan on a fingerprint sensor.

Object Detection/Recognition is very important in Computer Vision. Moreover, they become the main factor that distinguish Computer Vision from Image Processing. And that is because Object Detection/Recognition is the basic skill needed by a computer for analyzing image/video and extracting information or features from it. This supports the goal of Computer Vision, which is to enable computer to understand the meaning of image/video. Given a photograph below:



Figure 3: A fruit stall at Barcelona market.

After performing Object Detection/Recognition on the image, computer can detect and recognize a great number of fruits. The fruits are orderly fashioned based on their types: apple, orange, banana, etc. Some price tags can also be found: one for every fruit type. These facts are then used by the computer to analyze the image until it is concluded that the image is in fact a fruit stall.

Now, how does a computer perform Object Detection/Recognition? It depends on the type of Object Detection/Recognition the computer wants to perform. According to Narotthambai and Tandel, there are 3 main stages for invariant, shape-based Object Detection/Recognition (Narotthambai and Tandel, 2016):

**1. Data Pre-processing (Image Processing)**

Image Processing includes a set of processes with one goal: enhancing image quality to produce desired image. The common processes are resizing, cropping, recoloring, filtering, smoothing/sharpening, and so on. In term of Object Detection/Recognition, Image Processing may be required to make the image noise-free or clearer for Feature Extraction. Image Segmentation may also be done to make Feature Extraction easier.

**2. Feature Extraction**

Next, the pre-processed image has its features extracted and stored into a database. Feature Extraction will ensure that Object Detection/Recognition runs easier and more accurate.

**3. Classification (Match and Search)**

Classification includes searching and matching image features using the database from the earlier Feature Extraction. Consider an apple as the image feature for “Fruit Stall”. If an apple is found in different image, computer will recognize that feature and may later classify the image as “Fruit Stall”.

For human, it is easy to understand the meaning of an image/video. Computers, however, have a hard time doing that. Object Detection/Recognition still remains a very challenging task for computer to perform. While computers excel at accurately reconstructing the 3D shape of a scene from images taken from different views, they cannot name all the objects and animals present in a picture, even at the level of a two-year-old child. This happens because the real world is made of a jumble of objects, which all occlude one another and appear in different poses, creating extreme variations in shape and appearance (Szeliski, 2010).

There are several techniques or tools that can be used to increase Object Detection/Recognition performance. One of the them is by including what is called Machine Learning. By definition, **Machine Learning** is a subset of [Artificial Intelligence](https://en.wikipedia.org/wiki/Artificial_intelligence) in the field of [Computer Science](https://en.wikipedia.org/wiki/Computer_science) that often uses statistical techniques to give [computers](https://en.wikipedia.org/wiki/Computer) the ability to “learn” with [data](https://en.wikipedia.org/wiki/Data), without being explicitly programmed. For the purpose of Object Detection/Recognition, successive trainings are given to the computer in terms of: (1) choosing and extracting image features; (2) classifying other images by searching and matching those features. Overall, Machine Learning existence definitely improves Object Detection/Recognition accuracy.

Machine Learning itself can be classified into two categories, supervised learning and unsupervised learning. In supervised learning, the output for any inputs given to the program are given by the creator. The goal in supervised learning is to find a general rule that maps the inputs into outputs. On the other hand, unsupervised learning does not have any given outputs, leaving the program to find structure in its input. Image classification uses supervised learning task. It defines a set of objects to be identified in images and train a model to recognize them using labeled photos.

There are two phases in image classification, training and testing. In training phase, the unique characteristics of the image features are isolated, and descriptions of each classification category are given. These characteristics have some criteria that needs to be fulfilled before being used in the next phase. In testing phase, these features are used to classify image features.

Another technique that could help increase Object Detection/Recognition performance is by using a convolutional neural network (CNN) architecture. There are several types of CNN that is available to use in the internet such as: MobileNets, Inception, Xception and others. Here we use the MobileNets architecture. This architecture uses depthwise separable convolutions which is introduced in Inception models. It reduces the number of parameters if compared to networks with normal convolutions that have the same depth which in result will create a light-weight deep neural-networks. However, by using depthwise separable convolutions, it might also create a low complexity deep neural network.

Depthwise separable convolution are made up of two layers: depthwise convolutions and pointwise convolutions. The former is used to apply a single filter to each input channel. The latter uses a simple 1x1 convolution to create a linear combination of the output of the depthwise layer.

**CHAPTER II – ALGORITHM EXPLANATION**

Our program uses a trained neural network that is originally designed to work on mobile devices due to our limited resources. The program works as a retraining program, meaning it uses a trained model and retrain the model to be able to identify more classes of items. To make things organized, we will explain in brief the structure of our program, then the algorithm of our program, then elaborate it thoroughly in the next paragraphs.

Before we explain the algorithm, we think it is best to understand the structure of our program first. The program has Python scripts stored in scripts folder, and the rest are stored in tf\_files folder. Files stored in tf\_files folder are the model, images for retraining, image feature vectors (bottlenecks), training summaries, and the output graph and output labels. While many files exist, we modify only the label\_image.py file and deem label\_image.py and retrain.py files as important for our application. Therefore, algorithm described in following paragraphs will be based on those files.

In short, our program must first be retrained using images located in tf\_files folder, with folder name training\_images. In training\_images folder, exists folders with images contained in them. The folders’ name is the label for each image contained in that folder. Then, the program retrieves input in the form of URL link to an image. The image must be in form of .jpg or .jpeg file as we limit our model to process only those types of file. The image is then downloaded and labeled as a class. If the class does not exist or the model cannot classify the image, the label will be an empty string output. The program then asks user for confirmation about the class of the image. If the label is empty string output, or if user deems the class of the image incorrect, the program asks for clarification of what class the image is. After receiving the image class from user, the program will collect data using Google Custom Search with the correction class as its query, and the data collected are in form of .jpg or .jpeg file, without further validation or selection process. Then, the program will have to be retrained manually as automatic retraining is still unavailable.

Now we will discuss the algorithm in detail. The algorithm of the application begins with retraining the model using images gathered for retraining process. The retraining algorithm is as follows.

The retraining process begins if the retrain.py file is called via command prompt, with some (or all) arguments passed. Several important arguments passed are the model name, image directory for retraining, bottleneck directory, and output files directories. The process then continues with creating preparations for the retraining, such as file system preparations, model information descriptions, downloading model, and creating list of images and apply distortions (if specified in parameter). By distorting the image, we are actually performing image processing as the image is processed to become better input for the training process. Then, .jpg and .jpeg files are selected from image directory and bottlenecks from each selected file are created. By creating bottlenecks, the program does feature extraction from each image and use it for training process. After that, the new layer to be retrained are created and the training process begins for default value of 4000 times, unless specified otherwise. The training process selects random bottlenecks, and from these resources the model can learn and know its accuracy level. Training and test accuracy are both calculated, with addition cross entropy value.

The algorithm the continues with user giving input in form of URL link to an either .jpg or .jpeg image. After the image is downloaded, the image is labeled. The label is assigned according to its probability. This means, for every label the model has known, each label will have a level of probability for an image with different values.

The labeling algorithm is as follows.

First it gets the image file as passed in the arguments. Then, it continues by loading the graph, which is the model of our classifier, and read tensor from the image file uploaded. The graph is then used for its input and output layer, meaning we pass the image, in form of tensor from the image file, to the input layer. When reading tensor from the image file, the program actually fits the image as dimensional input for the model to use as input. The fitting process is done by casting the image to float values, expanding the dimension of the image’s float values, resizing the expanded dimension input, and normalize the values. Then, the input layer, filled with the normalized image values processes the image until the output layer. This process is done using neural network, model from MobileNet that has been retrained. From the output layer, results, in form of probabilities of the image’s class, are shown.

We made minor modification to the labelling algorithm, in that the algorithm does not show all probabilities of the image’s class. Instead, since we would like to create a program that classifies image, we select the class with probability value above or equal to 0.6. Thus, only a class with probability value of 0.6 or more are printed. Otherwise, no class can be output (the program returns empty string).

The rest of the process is simple as they only use Python and PHP features to be completed, such as sending and retrieving query and request to Google Custom Search using JSON, saving image using os library in Python, and retraining the model again using algorithm described above.

**CHAPTER III – EXAMPLE AND DISCUSSION ON RESULTS**

Our program is located on the following link: <https://github.com/jonathan016/image_classifier> to make managing things easier. To run the program, we should have XAMPP installed, and after XAMPP has been installed and run, Apache must be started. Then, clone the repository in xampp/htdocs/<folder\_name>. Then, go to browser and type localhost/<folder\_name>/index.html.

Example

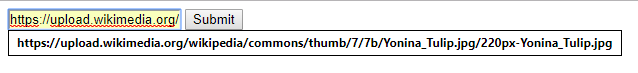


Figure 4 – Upload image page

When running the application, we can submit link such as the above example. After submitting the link, the image below will be shown.

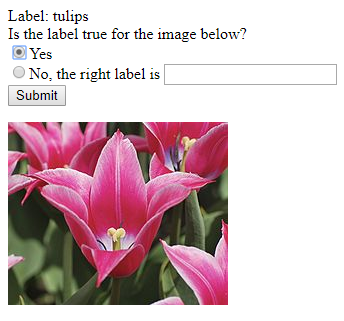


Figure 5 – Classification result page

This is the result of the classification process done by the program. We can choose if the image label/class is correct/incorrect. If we select ‘Yes’ and submit, no action is taken. Otherwise, we should see the following images.

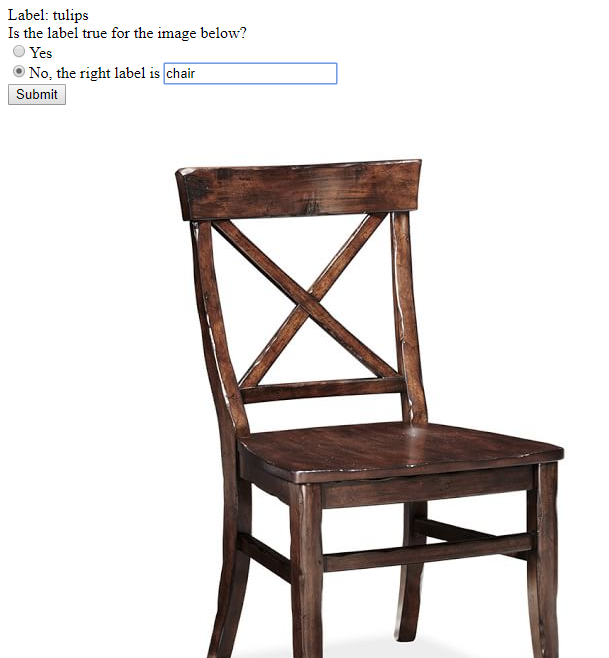


Figure 6 – Wrong classification example

The above image is an example of wrong classification. The program thinks that the image is a tulip, whereas it is a chair. We can input the correct label, and then the program will download the images for retraining automatically.

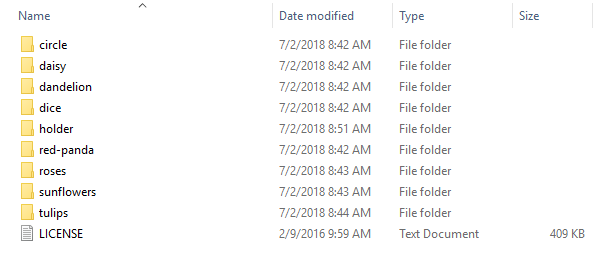


Figure 7 – Folders inside training\_images folder before correct label submission

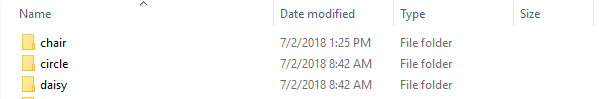


Figure 8 – Folders inside training\_images folder after correct label submission



Figure 9 – Message shown after download images finished

Result Discussion

Our program manages to classify image based on the image’s features, which is represented in normalized float values that is passed as input to the neural network and outputs a single label that best describes the image. The label output is based on all trained data that the model has.

Our program also works as our algorithm defined, in that it receives input in form of URL, download it, and classify the image. The classification result is also shown, and confirmation is asked to ensure that the program classifies fine.

When retraining the model, different accuracy levels can be obtained due to random selection of bottleneck files of the training images. Overall, we obtained over 85% accuracy level. It is also important to note that our retraining command is as follows:

python -m scripts.retrain --bottleneck\_dir=tf\_files/bottlenecks --model\_dir=tf\_files/models/ --summaries\_dir=tf\_files/training\_summaries/"mobilenet\_0.50\_224" --output\_graph=tf\_files/retrained\_graph.pb --output\_labels=tf\_files/retrained\_labels.txt --architecture="mobilenet\_0.50\_224" --image\_dir=tf\_files/training\_images --random\_crop=15

The above command is run at the root of our application folder (xampp/htdocs/<folder\_name>) and it takes some time to retrain the model.

**REFERENCES**

Szeliski, R. (2010). *Computer Vision: Algorithms and Applications.* New York: Springer.

Automatic Sliding Doors. (n.d.). Retrieved June 30, 2018, from

<http://login.gilgendoorsystems.com/Products/Automatic-Door-Systems/Sliding-doors>

A. (2017, September 22). How Do Automatic Doors Know When To Open/Close?

Retrieved June 30, 2018, from <https://www.scienceabc.com/innovation/automatic-sliding-doors-working-motion-detector-pressure-sensor-infrared.html>

Airport Security. (2013, May 26). Retrieved June 30, 2018, from

<https://raykliu.wordpress.com/2013/05/26/airport-security/>

File:Fruit stall at Barcelona market (2929344665).jpg. (n.d.). Retrieved June 30, 2018,

from <https://commons.wikimedia.org/wiki/File:Fruit_stall_at_Barcelona_market_(2929344665).jpg>

Narottambhai, M., & Tandel, P. (2016). A Survey on Feature Extraction Techniques

for Shape based Object Recognition. *International Journal of Computer*

*Applications,* *137*(6), 16-20. doi:10.5120/ijca2016908782