Technical Memorandum

TO: SYSC 3010 Students
FROM: Jonathan Arava
Subject: Image Classification
Date: November 26, 2017

1.0 INTRODUCTION

Imagine you have a cluster of photos on your computer and you want to organize them by having a folder of photos for every person in the photos. Traditionally you would create folders under different people's names and one by one, open each photo and see who is in it and drag the photo to the appropriate folder. For a few photos this might be manageable but for many photos this process can be extremely tedious. A solution to a problem like this would be image processing and more specifically image classification.

1.1 Summary

This report is about the topic image classification. Image classification is a special case of image processing where the image being processed is autonomously identified to be part of a specific category from a range of pre-categorized images. Image classification has machine learning and statistics analysis as the basis of the image processing algorithm.

2.0 METHOD

There are a number of different open-source libraries that can help us such as OpenFace, BoofCV, OpenCV, TensorFlow and a few others. For this technical memo I will be demonstrating using TensorFlow (TF).

2.1 Why TensorFlow

The reason for using TF is that it is supported by Google which gives it a huge advantage compared to its competitors since it is ever evolving and new versions of algorithms are updated regularly. Also there is a relatively large community of coders who use TF so this will help when debugging. Most importantly, TF has easy step-by-step tutorial videos to get TF up and running on your computer. In addition, TF has the best visualization of even complex neural networks which will come in handy for intricate image classification problems.

2.2 How Image Classification on TensorFlow Works

To understand how image classification on TF works we need to first have an understanding on deep neural networks. Imagine we were reading handwritten letter, e.g a, b, or c. We see the handwritten sample, "input" and our brain then "deciphers" what was written to equate it to the most familiar letter that we now as the "output".

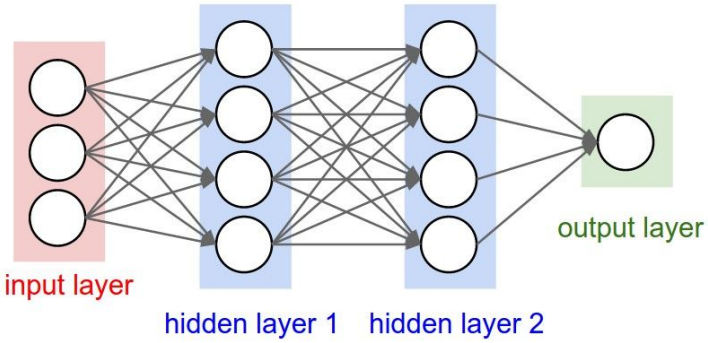

Figure 1: simple neural network

Our brain manages to process the input to give out an output. The region in between the input and the output where our brain does the reasoning to identify and classify what the handwritten letter is, is is called the “black box”. In figure 1, the hidden layers would be considered the “black box”. Over the years computer scientists have developed ways to computationally replicate the “black box” so the computer can reason and distinguish between objects/ images.

So basically, TF convulates through layers of data called nodes to learn the image. The first layer, Layer 0 on figure 2, will be determined by looking at the most general attribute of the image such as the basic shape. The program then moves to the next data set, layer 1 on figure 2 to determine more detailed attributes of the image such as the curvature of the object in the image. It repeats this process till it can confidently distinguish which category/ class the image belongs to.

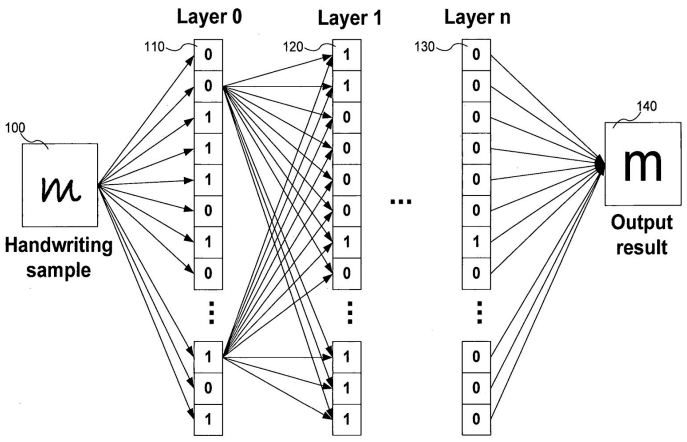

Figure 2: Neural network for Handwritten sample

3.0 PROCEDURE

To demonstrate the procedure I will use the TensorFLow for poets tutorial made by Google but I will concise the tutorial for our needs. The following procedure will be broken down to 4 steps: setup, download the training images, training the network, using the retrained model.

3.1 Setup

Step 1: install tensorflow onto the computer.

Step 2: clone the git repository.

GIves the most updated version of TensorFlow and the training model for us to work on.

```
git clone https://github.com/googlecodelabs/tensorflow-for-poets-2

cd tensorflow-for-poets-2
```

3.2 Download the training images

Step 3: The folder being downloaded consists of pre-categorized photos of flowers for the program to train itself to distinguish each category of flower. So when a flower is inputted the model will categorize it to the appropriate flower category. For this case, a folder named flower_photos is downloaded which consists of 5 folders of different flowers, i.e. daisy, dandelion, roses, sunflowers, tulips. To train photos of our choice we will replace the flowers with what images we want to classify. Below is the line of code to download the training images.

```
curl http://download.tensorflow.org/example_images/flower_photos.tgz \
    | tar xz -C tf_files
```

3.3 Training the Network

This is the part where the the program trains the system to distinguish the images by breaking them down in layers and classifying them.

Step 4: The below line puts constraints on the training model. The greater the image size configuration the more accurate the results but also the greater the learning time.

```
IMAGE_SIZE=224
ARCHITECTURE="mobilenet_0.50_${IMAGE_SIZE}"
```

Step 5: Open tensorboard, a background monitoring and inspecting tool used to monitor the training process.

```
tensorboard --logdir tf_files/training_summaries &
```

Step 6: Initiating the model training process.

```
python -m scripts.retrain \
  --bottleneck_dir=tf_files/bottlenecks \
  --how_many_training_steps=500 \
  --model_dir=tf_files/models/ \
  --summaries_dir=tf_files/training_summaries/"${ARCHITECTURE}
  --output_graph=tf_files/retrained_graph.pb \
  --output_labels=tf_files/retrained_labels.txt \
  --architecture="${ARCHITECTURE}" \
  --image_dir=tf_files/flower_photos
```

3.4 Using the Trained Model

Step 7: Select an image you want the model to classify. In our case we will use a photo from the daisy folder downloaded earlier in the flower_photos folder (the image selected is 2165746_cc379e0eea_m.jpg).

Step 8: Execute the model to classify the selected image, the daisy photo in our case, using the below code.

```
python -m scripts.label_image \
    --graph=tf_files/retrained_graph.pb  \
    --image=tf_files/flower_photos/daisy/21652746_cc379e0eea_m.jpg
```

Step 9: View the results. The below code shows the percentage match of the selected daisy image to the categories. The model indicates a 99% match of the selected image to be a daisy and a low confidence for any other category.

```
daisy (score = 0.99071)
sunflowers (score = 0.00595)
dandelion (score = 0.00252)
roses (score = 0.00049)
tulips (score = 0.00032)
```

4.0 RESULTS and DISCUSSION

4.1 Results

As mentioned above, TensorFlow gives a percentage match of the selected image to categories from the training images. The higher the match percentage the greater the possibility the image belongs to that category.

4.2 Discussion

A limitation to this method would be that the image classification process will only be as good as the images you feed it in the training process. So that means you will have to initially classify images on your own in separate folders. Another limitation would be that the more images used in the training process the better the accuracy. This means that to get accurate results for classification you will need to repeat the step of dragging images to the appropriate folder

several times. If image classification is a frequent task then the future benefit and time saved later on out weighs initial work to set it up. A possible solution for this is online where on the command line you can download a selected number of images from google for the selected word search. But this is not always accurate as sometimes images that does not belong to your search make their way to the google images page. For example you want to have a folder of images for the computer brand apple but every now and then the fruit shows up in the images. You can specify the search by being more precise such as “apple computer” but at the cost of losing some images to that search.

5.0 CONCLUSIONS

To fully understand image classification and more specifically the image processing aspect, I would highly recommend to do further research on the topic. But for the purpose of the technical memo, I believe this report is competent to give a beginner to image classification the basics for him/ her to get a general idea of the application of such tool as well as the set up for it. In conclusion, when done right, image classification can be a very useful tool especially if either the image classification is a regular task for your project or the number of images needed to be classified are significantly large to the number of images required to train the model.

6.0 REFERENCES

“TensorFlow,” *TensorFlow*. [Online]. Available: https://www.tensorflow.org/. [Accessed: 22-Nov-2017].

Figures

Figure 1:

A. Abdelfattah, “Image Classification using Deep Neural Networks - A beginner friendly approach using TensorFlow,” *Medium*, 27-Jul-2017. [Online]. Available: https://medium.com/@tifa2up/image-classification-using-deep-neural-networks-a-beginner-friendly-approach-using-tensorflow-94b0a090ccd4. [Accessed: 22-Nov-2017].

Figure 2:

“Patent US20070047802 - Training convolutional neural networks on graphics processing units,” *Google Patents*. [Online]. Available: https://www.google.com/patents/US20070047802. [Accessed: 22-Nov-2017].

Procedure

Snapshots of code from the following reference:

“TensorFlow For Poets,” *Google*. [Online]. Available: https://codelabs.developers.google.com/codelabs/tensorflow-for-poets/#0. [Accessed: 22-Nov-2017].