Deep Learning Food Recognition Model Progress Report

Hou, Chuyi Li, En Xu Shentu, Chengnan

July 24 2019

Supervisor: Lisa Zhang
University of Toronto

Contents

1 Introduction

Our team is pleased to submit this paper to report the progress of the food-recognition model. The goal of this model is to identify what kinds of food are consumed by the person from a overview photo of the meal. In particular, we will use the model to target the western-style breakfast meals (Figure 1). We will use supervised learning techniques and use labelled data to train the neural network. The trained model is expected to recognize various kinds of foods in the photo and report them back to the user (Figure 2).

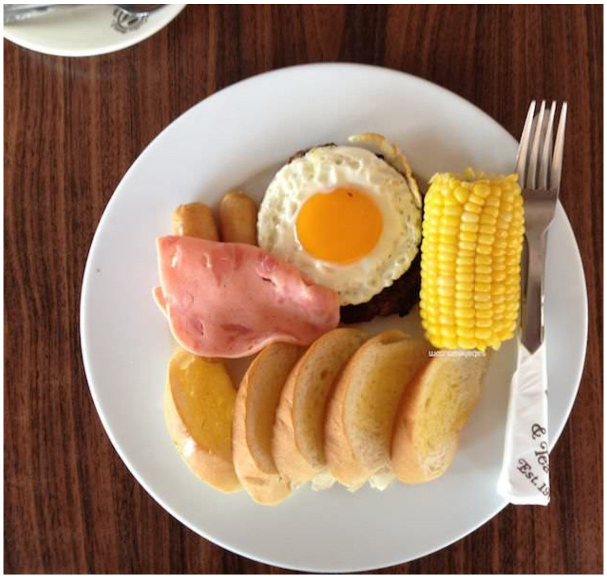


Figure 1: An example of image input to the model

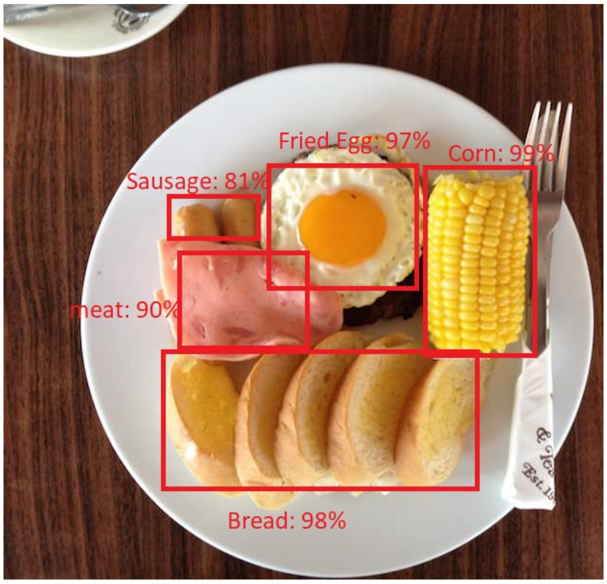


Figure 2: An example of output from model that has box information

Essentially, by having this base functionality, future developments will implement recommendation feature to accompany the food recognition feature. Thus, the model would be able to make diet recommendations for meals later in the day based on the person's current meal.

2 Data Processing

Data that are used to train, validate and test the model will be in the forms of RGB values of each pixel in an image. In this section, the methods of data collecting, cleaning, and splitting will be discussed.

2.1 Data Collection

32 categories of popular western style breakfast foods will be used for training the model, including bagel, coffee, omelette, pancake, etc. A python script was written to search through Google Open Image Dataset V5 and extracted photos that belong to these 32 categories with the corresponding box information (Figure 3). All images collected were then stored into 32 folders according to their labels.

2.2 Data Cleaning

All of the images collected were cropped according to the box information of the labeled feature using python openCV library. Meanwhile, they were resized to have 300 by 300 pixels (Figure 4). The biggest issue in this method of data collection is inconsistent quantities of images in each category. For instance, class bagel contains 267 images, while class coffee has 1027. In order to avoid model overfitting to the labels with more training images, we used data augment techniques (such as horizontal and vertical flip) to create more images to make sure each class has approximately 1000 images.

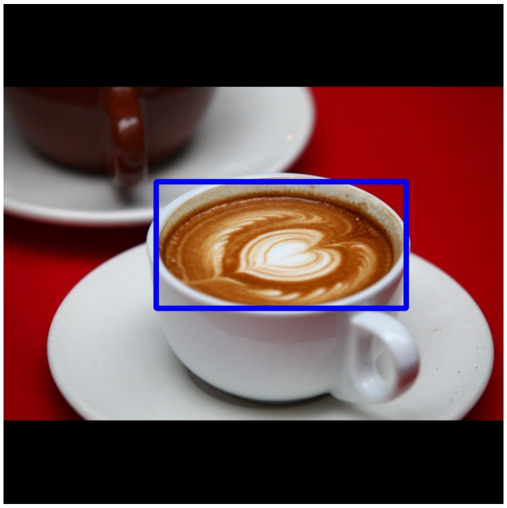


Figure 3: An example of images extracted fromo Google Open Image Dataset V5

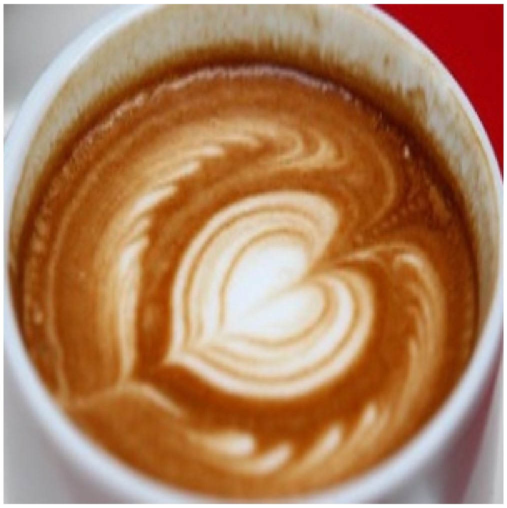


Figure 4: An example of the image that feeds into the classification model

2.3 Data Splitting

After collecting and cleaning phase, there are 32,000 images in total (1000 for each category). We randomly selected 800*32 collected and processed images as our training dataset. 150*32 Images were chosen randomly for the validation dataset, and the rest was used for testing purposes. Therefore, train:validation:test ratio of the data for implementing this model is 80%:15%:5%. For additional testing purposes, we will be using photos taken using smartphones and resized to 300 by 300 pixels.

3 Baseline Model

We used pretrained Alexnet model with one fully connected layer that outputs 32 channels as the baseline model. While training the baseline model, each image was fed to the pretrained Alexnet and was extracted 256*5*5 features. Then these features were laid flat in a fully connected layer to output 32 channels. The baseline model was trained for 100 epochs and was able to achieve 53% training accuracy, 28% validation accuracy, and 26% test accuracy.

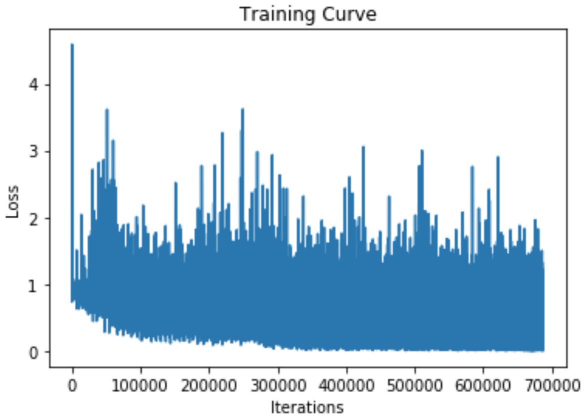


Figure 5: Baseline Iterations vs Training Loss Curve

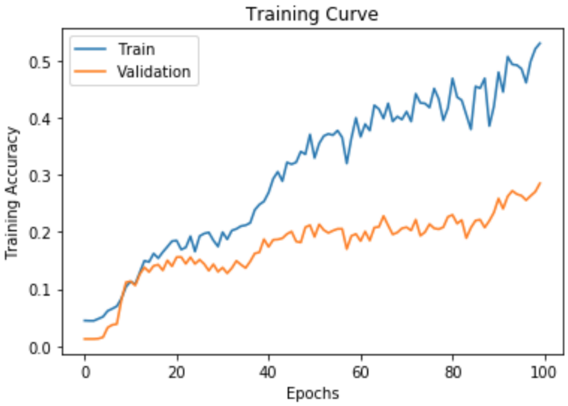


Figure 6: Baseline Epochs vs Training/Validation Accuracy Curve

4 Architecture

Our best model architecture that we came up so far consists a combination of two neural networks:

Object detection neural network on top of a convolutional neural network with fully connected layers producing 32 classification results.

On top of this, we reduced our training effort by taking in a pretrained detection model, which is acceptable and ideal to fit our current computing power.

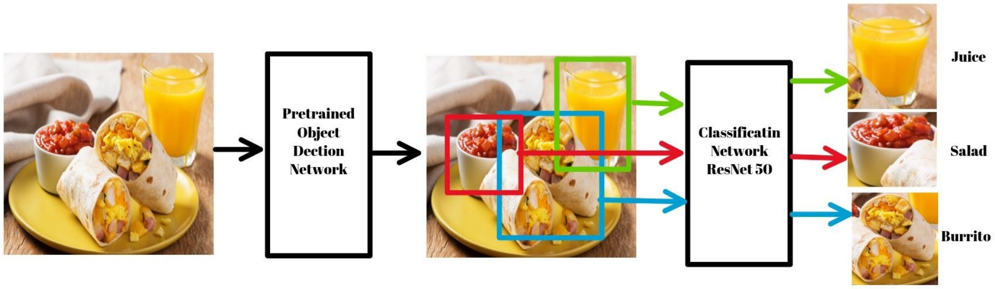

Pretrained Object Dection Network
Classificatin Network ResNet 50
Juice
Salad
Burrito

Figure 7: The general idea of our model's architecture: a detection model followed by a CNN.

4.1 Object Detection Neural Network

We planned to use a pretrained object detection model which has been trained on COCO (Common Objects in Context) dataset.

Input An image (300x300 pixels) contains multiple classes of food.

Output Ideally, we want the output to be the pixel coordinates (i.e. x/y-max/min) of each class detected. This will help us crop the points of interest and do classifications on each cropped image later by feeding the it to a CNN.

4.2 Classification Neural Network

The CNN that we planned to use to take in the result from upper object detection network to produce multi-class classification outputs is ResNet (Deep Residual Network).

Pytorch?Torch has all types of ResNet. Due to current limited computational power, the type of ResNet that we planned to use is ResNet50 which contains 50 layers.

Input One image cropped based on the pixel coordinates provided by the upper detection network, and the image will be resized (by stretching) to 300x300 pixels. There will be multiple images feeding through, one image at a time.

Output One classification output of the input image. It will be a 1x32 tensor, one possibility for each class. Ideally, there will only be one class having a significantly higher possibility than the rest. Thus, the model can tell what kind of food it is.

4.3 Connection between the Two Networks

As you can see from above descriptions of the two networks, they can be independent from each other. Hence, there should be some connections between them in order to complete the entire food detection task.

We need to do something that can connect the output from the detection model to the input of our trained ResNet. The connection process can be done by the following briefly described steps:

- Crop the initial input image using the produced points of interest. (There will be multiple cropped images from one initial input image)
- Indexing the cropped images and points of interests so that we can retrieve the image's corresponding location in the input image when we produce the final output.
- Resize each cropped images to 300 by 300 pixels by method of stretching (using OpenCV).
- Feed one resized image one at a time through our classification network, ResNet. Listing the result with the the image and its location in the initial input image.

4.4 Final output of our model

The idea of the final output of our model would be similar to the output proposed in the project proposal (i.e figure 2).

5 Result

As of July 23, 2019, the working model has been trained for 12 epochs. It has achieved a training accuracy of 43%, validation accuracy of 37%, and test accuracy of 35%. As seen, the working model has already had higher validation and test accuracy comparing to the baseline model. More training and hyperparameter tuning will take place.

6 Discussion

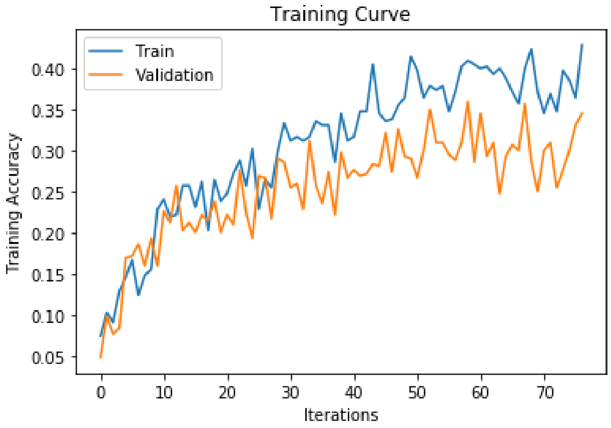


Figure 8: Training Curve

The produced training curve is quite noisy. This could be caused by using small batch sizes while training the model. We relied on Google Colab to train the model; however, one critical issue about Colab is that it only provides about 12 GB of GPU memory which limits the batch size to about 20 images in our case. One potential solution for having higher batch size to avoid the model being influenced by bad data is to use Google VM engines with customized GPU and memory.

7 Project Progress

This project was launched on July 1, 2019 and is expected to be complete by August 10, 2019. As of July 23, 2019, the following tasks have been completed.

All contributions are updated regularly on the Github page: https://github.com/nbjameslee/APS360_FOOD

- Image Collection: Completed
- Image Cleaning: Completed
- Data Splitting: Completed
- Baseline Model Built: Completed
- Building the neural network: Completed
- Debugging neural network using small data set: Completed

Tasks have been divided equally among three members.

En Xu Li (Thomas)

- Implemented a Python script to extract food images from Google Open Image Dataset V5
- Cropped and Resized all images to have the same shape with Python OpenCV
- Used pretrained Alexnet as the baseline model to extract features and trained to achieve reasonable accuracy
- Currently training resnet-50 as the object classification model

Chengnan Shentu

- Worked on recreating another faster R-CNN model from GitHub as a back-up plan.
- Worked with Google Open Image Dataset v5 as possible data collection source.

Chuyi Hou (Sky)

- Helped with data cleaning: went through the training dataset and removing unrelated/random images from each food class.
- Worked with faster r cnn: downloaded a pretrained faster r cnn model, tried to run the code from github repo in google colab. There has been several struggles.
- Finding the detecting model: after working with faster r cnn, a realization came out. Instead of continuing trying out the entire network, a detection model trained on COCO dataset should be considered.

8 Project Plan

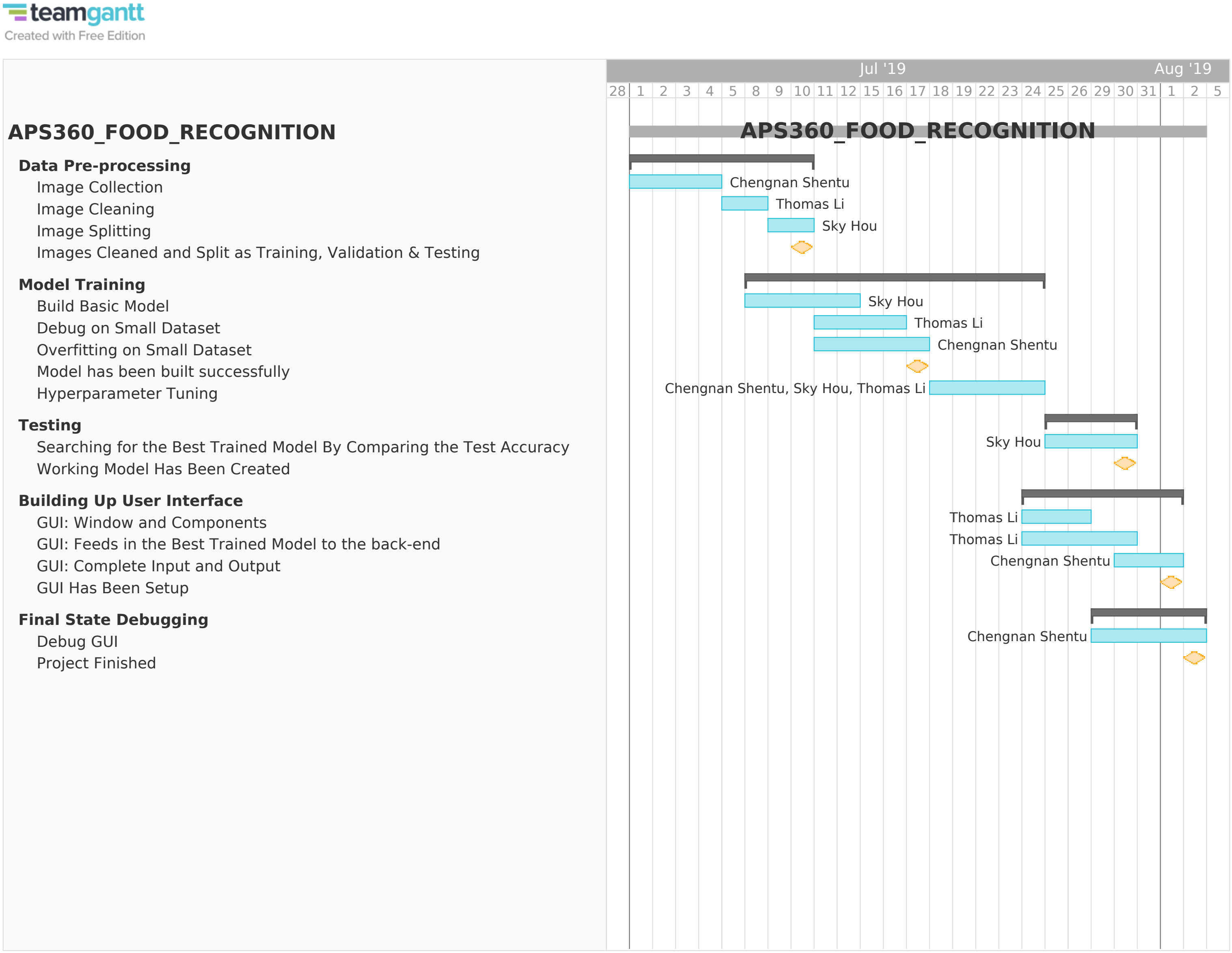
The remaining tasks are planned as the following

- Hyperparameter tuning including modification to the neural network architecture: Expected to finish by Aug 1 (In Progress)
- Testing and verification of the classification model: Expected to finish by Aug 3
- Connection between detection model and classification model: Expected to finish by Aug 5
- GUI: Expected to finish by Aug 9

8.1 Project Schedule

A team Gantt chart has been revised according to the current progress and the team tries to follow the detailed scheduled plan presented below:

Figure 9: Gantt Chart



9 Learning from TA

Insufficient Computational Power After a quick discussion with the other TA, Huan, we realized that it is impossible to train a object detection model using either our own machines or Google Colab. It would take enormous amount of time to train without the support from multiple GPUs. Therefore, we decided to use pretrained detection models (trained on COCO dataset) used in faster r cnn models.

Use of Google Cloud According to Andrew, the credit we have on Google Cloud Computation Resource can run out quickly, so we have to spend it wisely. He suggested we use Google Cloud only for the purpose of training and hyper-parameter tuning of the completed model. As a result, we have planned to do all the initial model building, debugging and testing either on our own computer or on google colab, and train on Google Cloud until the model is finalized.

Time Management Andrew emphasized on the importance of time management in this project. The neural networks we use in the project takes far longer time to train than the ones we encountered in the lab. Complex neural networks that deal with images can take many hours and even days to train, and there are hyper-parameter tuning steps that follow. Thus we tried to start training our model as soon as possible in order to use the limited time we have before the project is due.