# Deep Learning Food Recognition Model Final Report

APS360 Artificial Intelligence Fundamentals
Instructor: Lisa Zhang
Prepared by:

Hou, Chuyi Li, En Xu Shentu, Chengnan

August 14 2019
Word Count: 1969

# Contents

# 1 Introduction

Having a healthy and balanced diet is an important lifestyle decision that is gaining a lot of attention in the society. However, dietary risk is also a leading cause of death, due to nutrition-related diseases such as heart disease and type 2 diabetes. Unfortunately for patients with these diseases, a healthy diet is more than important, and is the necessary medication they must take.

Traditionally, building a healthy diet involves a lot of reading, calculation, and time. Some people also choose to consult with professional nutritionists, and that is no doubt an expensive approach. Our group started with the goal of developing an accurate, convenient, yet accessible solution to help people build a healthy diet. Naturally, a software approach checks these boxes, and building a food detection module is our first step. The module would identify food categories from an overview photo of the meal. As a starting point, we will use the model to target the western-style breakfast meals (Figure 1). A trained supervised machine learning model would be able to perform detection from unseen images quickly and conveniently, which serves our purpose. The result from this module would be similar to Figure 2 shown below.

Figure 1: An example image input to the model

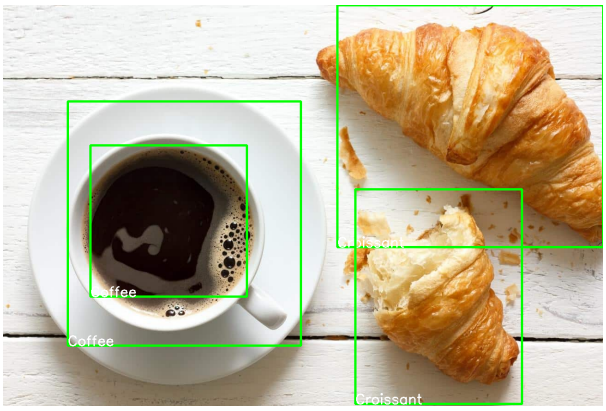


Figure 2: An example output from our model

After the functionality of this food detection module is achieved, programs can be developed to make recommendations to users on what do eat next based on the food in the current meal. Other features such as calories calculation are also possible. Ultimately, it can lead to a complete app that helps users to build a personal and healthy diet by simply taking pictures of their meals.

# 2 Background & Related Work

Background research has been conducted prior to proposing this model. We were able to find some food recognition models on the market; however, most of them could only make a decent prediction when there was only one dish in the photo. They could not handle the situation where multiple foods are present in a single picture frame. In particular, we will discuss about two of the related projects that have similar functionality.

## 2.1 NutriNet

NutriNet is a deep convolutional neural network architecture that has been used for food and drink image detection and recognition. This model was developed in 2017 and had been trained to recognize 520 different categories. The authors provided that the model achieved a classification accuracy of 86.72% on the recognition dataset, along with an accuracy of 94.47% on a detection dataset. With a real-world dataset acquired by smartphone cameras, the model was able to achieve a 55% accuracy[1].

Currently, this model is being used for the PD Nutrition Dietary-assessment application for Parkinson's disease patients[1].

### 2.2 Image AI

Image AI is a python library developed by DeepQuest AI regarding deep learning and computer vision. This library is a very powerful tool in terms of detecting objects and extracting features from the image[2]. This library uses the following 4 architectures: SqueezeNet, ResNet, InceptionV3 and DenseNet. Among these 4 models, there exists trade-offs between the processing time and accuracy. For example, SqueezeNet gives a moderate accuracy with the fastest prediction time, while DenseNet outputs the highest accuracy with a slower processing time [2].

## 3 Data Processing

Data that is 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, splitting, and arranging will be discussed.

### 3.1 Data Collection

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

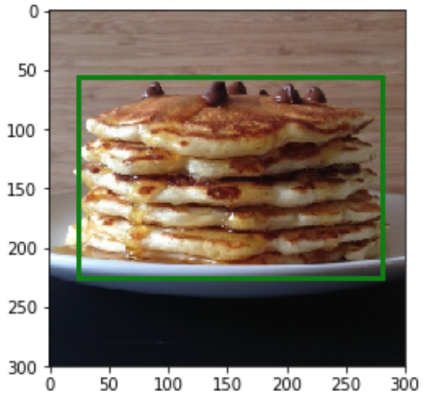


Figure 3: An Example of Raw Image from Google Open Image V5 with Bounding Box Information

### 3.2 Data Cleaning & Splitting

All of the images collected were resized to have 300 by 300 pixels to reduce memory usage and allow faster computation. Then we randomly selected 70% of the collected and resized images from each category as our training dataset. 25% Images from each class 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 70%:25%:5%.

After data splitting, we realized that the biggest issue in this method of data collection is the inconsistent number of images in each category. For instance, class bagel contains 267 images, while class coffee has 727. 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 the training dataset of each class has approximately 700 images.

### 3.3 Data Arrangement

In order to save data loading time and memory usage, we have arranged all images using the data structure in Figure 4. All images are shuffled and converted to tensor values and are stored with corresponding ground truth label and bounding box information as a Python Dictionary. These information are saved as multiple 'pth' files. During training, the model iterates over a list of these 'pth' files. The program will automatically free up the memory used to store previous data information before loading the new files. In addition, T=this way of data arrangement saves time for the model to search through other resources to find bounding box information because they are concatenated together.

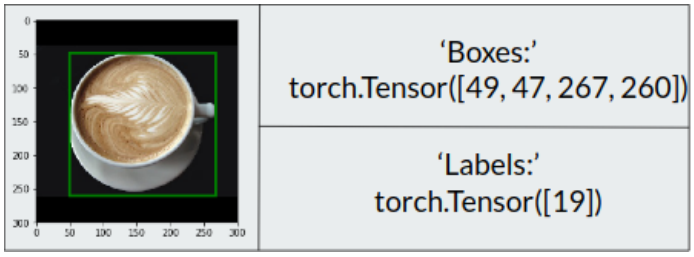


Figure 4: The processed data has the resized image, ground truth bounding boxes, and labels

## 4 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.

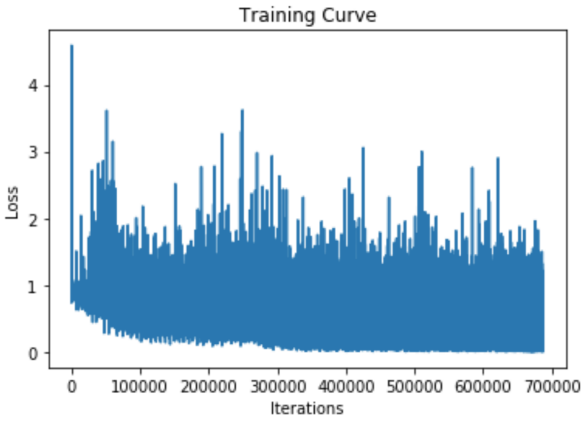


Figure 5: Baseline Iterations vs Training Loss Curve

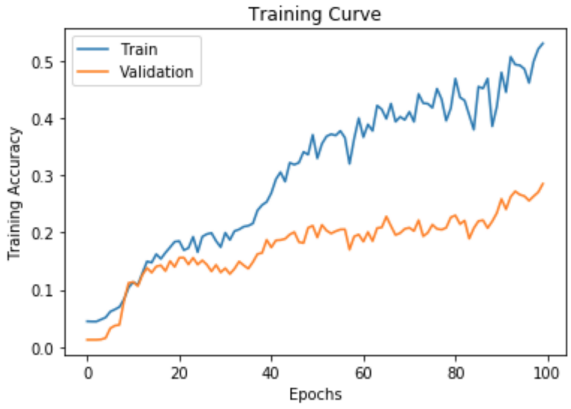


Figure 6: Baseline Epochs vs Training/Validation Accuracy Curve

# 5 Architecture

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

- Feature Network
- Region Proposal Network (RPN)
- Detection Network

It is basically a typical faster rcnn architecture. It is unnecessary to rebuild the entire network block by block. The model can be easily constructed using Pytorch [3].

Importantly, all the boxes shown in Figure 7 are running during training mode, whereas, only the blue boxes are executing during evaluation mode

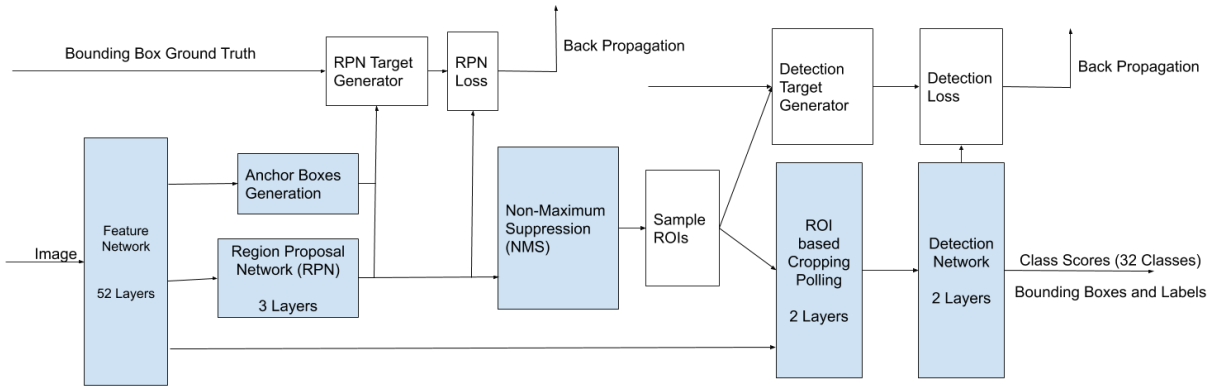


Figure 7: Faster R-CNN Model Architecture[4]

**During Training** Input image is fed into feature network. Anchor generator will generate 9 different sized boxes. With the help from Region Proposal Network (RPN), the model will generate candidate bounding boxes and compare the quality of proposed boxes with the bounding box ground truth by calculating two loss functions, objectness loss and RPN box regression loss.

Then the candidate boxes will be passed into the non-maximum suppression layer. This layer acts like a filter and reduce the number of candidates that are being passed through. The sample ROIs works as another filter and it is only functional during training. Next, the remaining cropped regions will be passed into detection target generator and compute loss based on the ground truth boudning box and label.

**During Evaluation** Input image is converted into feature maps, and RPN will propose many bounding box candidates based on the feature map. Then the remaining box candidates will be combined with the feature map forming a whole detection result in the latent space. Finally, two separate fully connected are responsible for delivering label scores and coordinates of bounding boxes with corresponding classes.

On top of this, we reduced our training effort by taking in a pretrained feature network model, MobileNetV2 [5].

# 6 Results

During training, we checkpoint 3 times each epoch and define every time we checkpoint as an iteration. The network has been trained for 10 epochs (30 iterations) and we chose to use the one trained for 5 epochs (14 iterations) as our final model since it started to show signs of overfitting after 5 epochs. In this section, the model's performance is reported and assessed using quantitative and qualitative metrics.

## 6.1 Quantitative Results

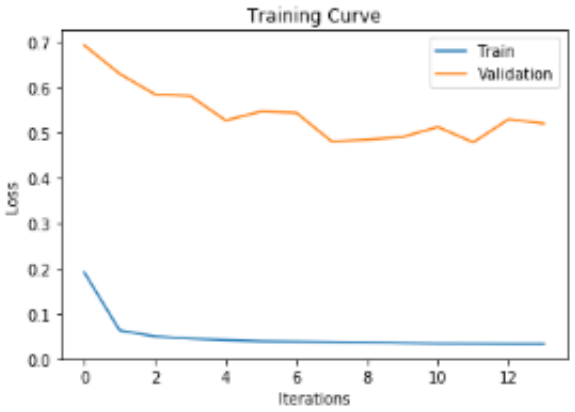


Figure 8: Iterations vs Loss Curve

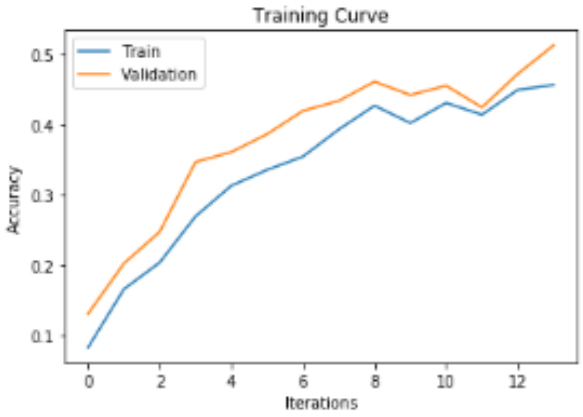


Figure 9: Iterations vs Accuracy Curve

As seen in Figure 9, the training accuracy is always lower than the validation accuracy. This is because we took the average of all computed training accuracy while training each batch, while the validation accuracy is determined at the end right before checkpointing. The model was expected to be more developed at the time of checkpointing.

While comparing the performance with the baseline model, we could see the test accuracy of the Faster R-CNN model is a lot higher. Meanwhile, the way we computed training accuracy for the final model is far more strict than the baseline. We define the model to be correct only if the predicted bounding box overlaps with the ground truth box by at least 70% with the correct label. On the other hand, the baseline only does the job of classification but does not contain any information where the object is located in the image.

Table 1: Performance Comparison with Baseline Model

| | Baseline (AlexNet) | Faster R-CNN |
|---|---|---|
| Train Accuracy | 53% | 47% |
| Validation Accuracy | 28% | 52% |
| Test Accuracy | 26% | 51% |
| Batch Size | 4 | 8 |
| Learning Rate | 6e-4 | 5e-5 |
| Optimizer | Adam | Adam |
| Trained Epochs | 100 | 5 |

### 6.2 Qualitative Results

The model is now able to correctly identify some of the breakfast-related objects in the picture. For instance, in Figure 10, it has found 3 strawberries and a waffle with precise bounding boxes; however, it has missed the juice, coffee, and cereal at the back. On the right side, the model captured coffee, juice and croissant but missed some fruits at the back and cereals in the corner.

Overall, the model is able to correctly identify typical-looking foods, such as, coffee, croissant, and strawberries. However, it has difficulties finding objects that may have variety of different combinations like cereals and salads.


Figure 10: Sample Output #1


Figure 11: Sample Output #2

## 7 Discussion

The produced training curve is quite noisy. This could be caused by using small batch sizes while training the model. We have only tried to train the model on one NVIDIA T4 GPU which provides 16 GB of memory. This limits the batch size to about 8 images in our case. One potential solution is to use multiple GPUs and allow parallel computing for having higher batch sizes to avoid the model being influenced by bad data.

As mentioned earlier, we found that the model is really good at detecting some of the classes, ignoring categories, such as salad and pear. This might due to asymmetric dataset as well as the quality of the data. One thing to mention is that during data cleaning, we found that salad pictures are quite messy because different families have completely different kinds of salads, unlike strawberries, which look mostly the same around the world.

## 8 Ethical Considerations

Our target food is particularly appeared in western breakfasts. In this case, our model would unlikely recognize food in other continents. For instance, our model would likely classify the food in Figure 13 as egg, while it is never the food shown in Figure 12, it is a traditional Chinese food called Tang Yuan.

Such bias can be reduced by expanding our data set to include more categories of foods, but we are working with the limited data set given the scope of this project.

Figure 12: Boiled Egg[6]

Figure 13: Tang Yuan[7]

Other than the recognition range of our model, we believe that there is no other major biases contained in our model. Moreover, the task itself, recognizing food, should be seen neutrally by people.

## 9 Project Difficulty Assessment

Object detection task can be done easily by taking other pretrained models, such as models trained on COCO dataset which has over 90 classes. However, COCO does not have many food classes, thus, this makes our project meaningful.

Hence, we need to collect new food dataset and train a brand new Faster R-CNN model. As the report mentioned earlier, we only trained the network for several epochs. Also, based on our understanding of Faster R-CNN, it needs more computation resources and time than other projects to have a decent model trained.

Moreover, we are not familiar with Faster R-CNN in the beginning of this project. The architecture is totally different from any networks we learned during lectures. We came through a long way to finally get here by learning the networks ourselves and seeking help from our professor and TAs.

Given the difficulty of the project and its length, we are satisfied with our final model. Our project is open source to the public and can be accessed here.

## References


[1] S. Mezgec and B. K. Seljak, "Nutrinet: A deep learning food and drink image recognition system for dietary assessment," *Nutrients*, vol. 9, no. 7, p. 657, 2017.

[2] M. Olafenwa, "Imageai," Jun 2019. [Online]. Available: https://github.com/OlafenwaMoses/ImageAI

[3] "Source code for torchvision.models.detection.faster_rcnn." [Online]. Available: https://pytorch.org/docs/master/_modules/torchvision/models/detection/faster_rcnn.html#fasterrcnn_resnet50_fpn

[4] S. Goswami, "A deeper look at how faster-rcnn works," Jul 2018. [Online]. Available: https://medium.com/@whatdhack/a-deeper-look-at-how-faster-rcnn-works-84081284e1cd

[5] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Mar 2019. [Online]. Available: https://arxiv.org/abs/1801.04381

[6] J. DeMay, "Instant pot hard boiled eggs," Dec 2018. [Online]. Available: https://www.realfoodwithjessica.com/2016/12/18/instant-pot-hard-boiled-eggs/

[7] 邪恶大师, "欢乐闹元宵黑芝麻汤圆海报psd素材," Jun 2018. [Online]. Available: https://www.16pic.com/haibao/pic_6053111.html