Project Refeed: nutrition tracker

Bryce Yao

Introduction:

The advent of technology has taken big steps in healthcare areas over the past decade. People now have more means and ease to monitor their health conditions. Yet it is still very difficult to get accurate information of the nutrition from our everyday meals. The problem is that there are too much variety in the choice of food for each meal and too complicated nutrition facts associated with each type of food. It is the goal of this project to tackle this barrier. I propose a computer vision system that leverages the camera and computing power in a smartphone to quickly identify the type of food from a camera image and inform the user about its nutrition facts.

Dataset:

There are several public datasets online:

Open Food Facts

https://www.kaggle.com/openfoodfacts/world-food-facts

The dataset "Open Food Facts" contains nutrition facts from foods around the world.

UEC Food 256

http://foodcam.mobi/dataset256.html

The dataset "UEC FOOD 256" contains a total of 31395 food photos in 256 categories. Each food photo has a bounding box indicating the location of the food item in the photo. 10% of the images were randomly chosen to be the evaluation set and were kept the same in each of the models examined in this study.

Approaches:

The key challenge of the task is that the system should accurately identify the food from an image. Previous researches have already paved the road for some promising neural network architectures and have reported some excellent results. Using a different dataset, Lee et al has achieved 81.67% food recognition accuracy[1]. Other researchers also have found good architectures that can achieve higher than 80% accuracy[2]–[4]. These studies have provided a solid foundation for this study.

This project takes a closer look at the varies architectures. I first investigated and implemented two very deep convolutional neural network architectures in order to compare their performances and find a way to improve upon them. The best architecture reported from literatures is based on the Inception V3 concept. It is therefore investigated first. It is noticed that one challenge limiting the performance in the literature is that there is usually more than one type of food on a plate. For example, fries are usually served as a side dish with many main dishes. It is reasonable to expect that if the system has a localization function prior to the identification, the performance can be further increased. Therefore, two of the YOLO architectures are also implemented in the first stage of the study.

After accessing the performances of the two models, an error analysis suggests that each of the models has its own weaknesses with respect to the chosen dataset. Concretely, the Inception model has high potential but suffers from two problems. The first is that the dataset is relatively too small to unleash the full powers of the feature extractors; the second is that the pictures in the dataset often features multiple food in one picture. Without more consistent data to learn from, the model is easily confused. On the other hand, the YOLO model is excellent at drawing the bounding box but gives less accurate predictions.

By combining the powers of these two models, a new system is proposed in this study. The new system utilizes a smaller implementation of the YOLO system (tiny YOLO[5]) as the first half, followed by an Inception model. The YOLO system only identifies food vs non-food and crops out the food image that can be fed into the inception half. Then the inception half does the classification.

Results and discussions:

YOLO

The YOLO system is inspired and adapted from this github repository: (https://github.com/AlexeyAB/darknet). Transfer learning is used in this stage to repurpose this network for food identification. A pretrained weight from the YOLO authors website is downloaded as the initial weights. The data is preprocessed to accommodate the YOLO requirement. Specifically, the image needs to be labeled first by its category, then by the bounding box. The dataset gives the bounding box information in the format of [(x pixel of upper left corner), (y pixel of upper left corner), (x pixel of bottom right corner), (y pixel of bottom right corner)] while YOLO requires the format to be [(x of box center), (y of box center), (width of box), (height of box)] and the values are normalized to the image width and height. With the labels determined, the data is trained on two Nvidia GTX 1080 Ti GPUs.

The neural network is set to have 29 layers. It has 23 convolutional layers, 5 max pooling layers, and a final sigmoid layer. The training session is set to have a minibatch size of 64 using the Adam optimization. A plot illustrating the training loss and the mean average precision vs. the iteration number is shown below in Fig. 1.

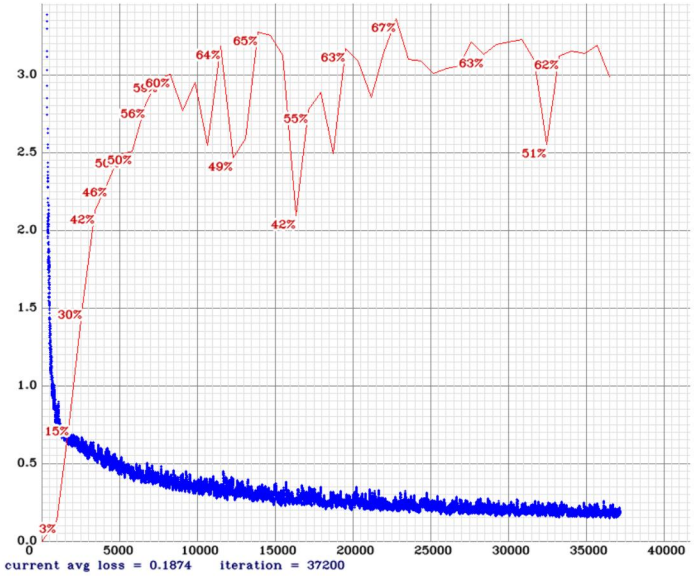

Fig. 1. Training curve of the YOLO model. Blue curve shows the training error, and the red curve shows the mean average precision (mAP). The final mAP is 60%.

After 37200 iterations, the training was terminated because the loss is converging to a constant value. For an initial model, this should be good enough to represent the model and carry out some error analysis. Figure below shows some examples of the YOLO predictions. Pictures on the first row are good examples, while pictures on the second row are imperfect.

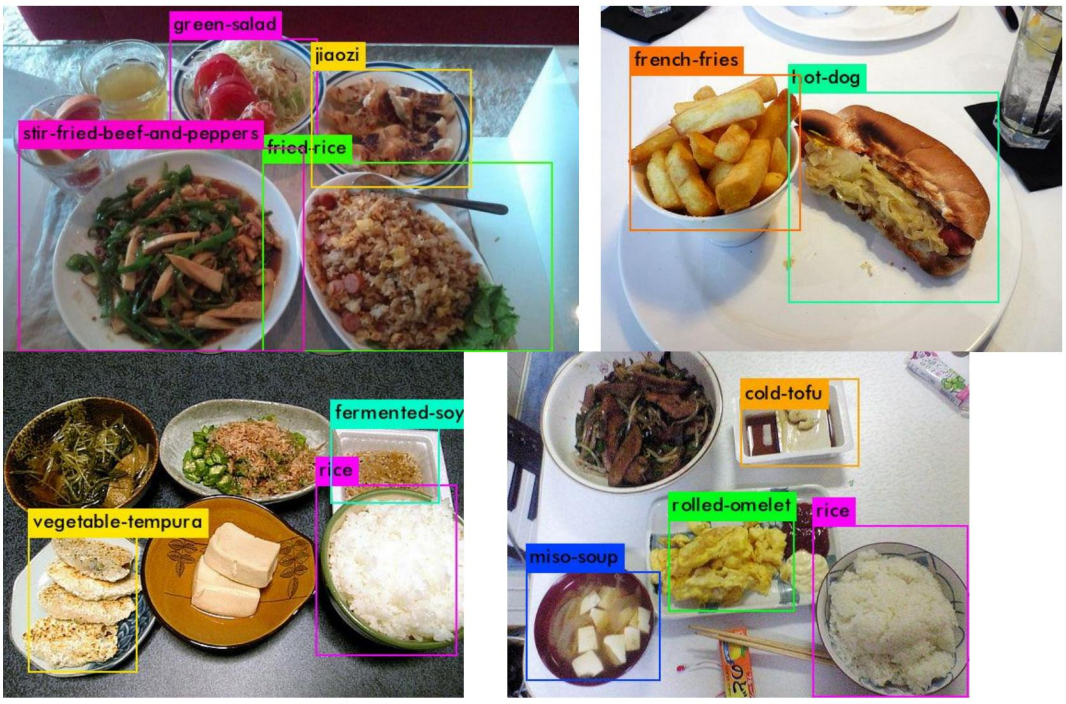

Fig. 2. example predictions of YOLO.

The localization works reasonably well. There are some items unrecognized mostly because they do not belong to the food categories seen by the model in training session. Additional data with more labels will help to alleviate this problem. It is noticeable that this dataset is the only dataset I have that labels the food with bounding boxes, other additional data will require manual labeling.

Inception V3

The inception architectures are the state-of-the-art image classification systems. The first version is illustrated in the GoogLeNet work[6], the second version and the third version added batch normalization and factorization ideas. The architecture adopted in this study is inspired by an application tutorial in the Keras documentation[7]. The InceptionV3 model is downloaded within the Keras framework without the last layer as the base model. A fully connected softmax layer with 256 outputs is added for the new classification. Fig.3 shows an illustration of the network architecture used in this work.

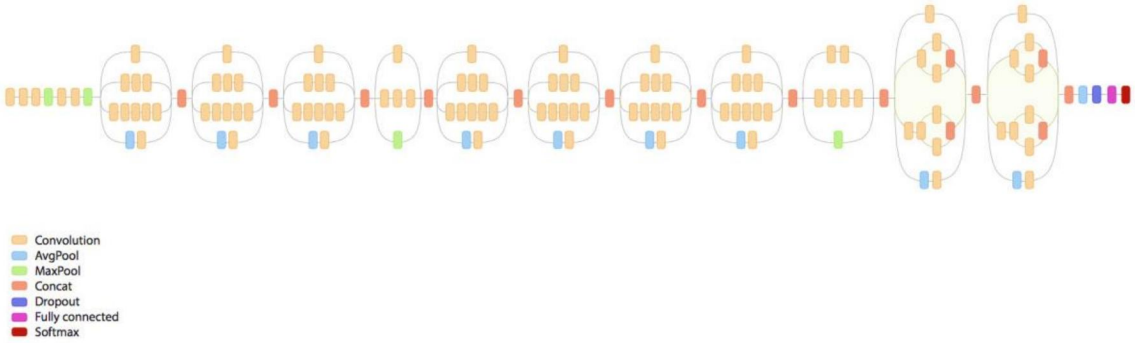

Fig. 3. The Inception V3 architecture used in this work is adapted from the author's github repository: https://github.com/tensorflow/models/tree/master/research/inception.

Pretrained weights from imagenet are loaded as the initial point of fine tuning. The actual training is divided into two phases. In the first phase of 30 epochs, only the newly added top layer is trainable, and in the second phase, the last inception block along with the last layers were trained. The system is trained using on a Nvidia GTX 1080 Ti GPU with CUDA. Fig. 4 shows the training curve of this model. After 230 epochs, the final accuracy on the evaluation set is 71.69%. This number is lower than the literature reported accuracies[3], [4].

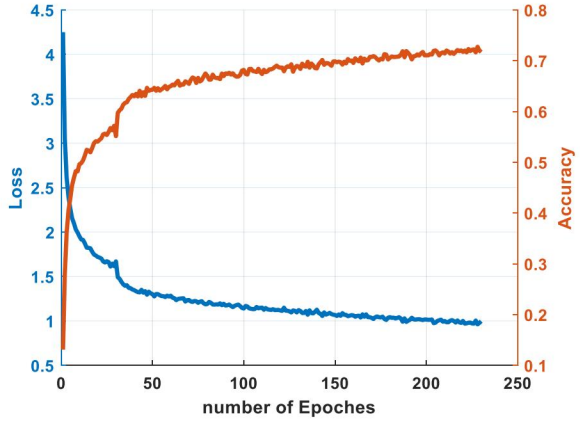

Fig. 4. Training curve of the Inception V3 model.

The discrepancy can be understood with two explanations. Firstly, the higher accuracy were gained by training on a different dataset with many more examples (over 101,000 images). It is expected to get higher accuracies with more training examples. Another problem with the current dataset is that the images sometimes contain multiple kinds of food but there is only one label associated with the image as one single training example. Fig. 5 shows two error examples predicted by the Inception V3 model. It predicts the left image as rice, and the right image as egg. Clearly it is recognizing the features of rice and egg, but not able to identify all the food in the image.

Fig. 5. Examples of wrong predictions from Inception V3 model.

Alternatively, the last layer of the model can be replaced with Sigmoid activations to allow multiple food output. But that would require the dataset to also allow multiple labels to be associated with one image. This potentially will increase the performance of this model. However, it is time consuming to change the structure of the dataset, and it concerns less about the neural network model study. Therefore, it is omitted in this project, but could be a future step. Instead, a new hybrid model is proposed using the same dataset.

tinyYOLO + Inception V3

After some thinking, there is a unique way to combine the strengths of the two system that can offer a superior overall performance. The new system is a concatenated version of the two neural networks: a small YOLO model that only does one classification: food, and a powerful Inception V3 model follows that. The YOLO system needs only take care of the bounding box now, which it does really well, and dissect the imaging into multiple smaller images that each has only one food item, such that the following Inception V3 system can predict more efficiently.

Because the initial food localization step only recognizes one category. Not that many features are needed, therefore a tiny-YOLO system is adopted here. The tiny-YOLO system is originally proposed by the YOLO authors for real-time video analyzing. Because it is small, it is very fast. An illustration of the architecture is shown in Fig. 6. It utilizes one residual path, and two detection nodes. Instead of nine anchor boxes in the full version, the tiny-YOLO uses six.

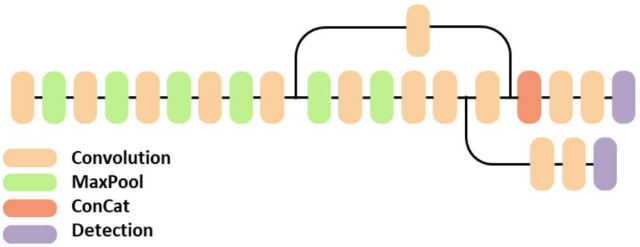

Fig. 6. Architecture of the tiny-YOLO system.

The dataset is modified to only use one label: food and retrained on this new structure. Because of the simplicity, the training is done faster with better results, as shown in Fig. 7. An example of the image dissecting is illustrated in Fig. 8.

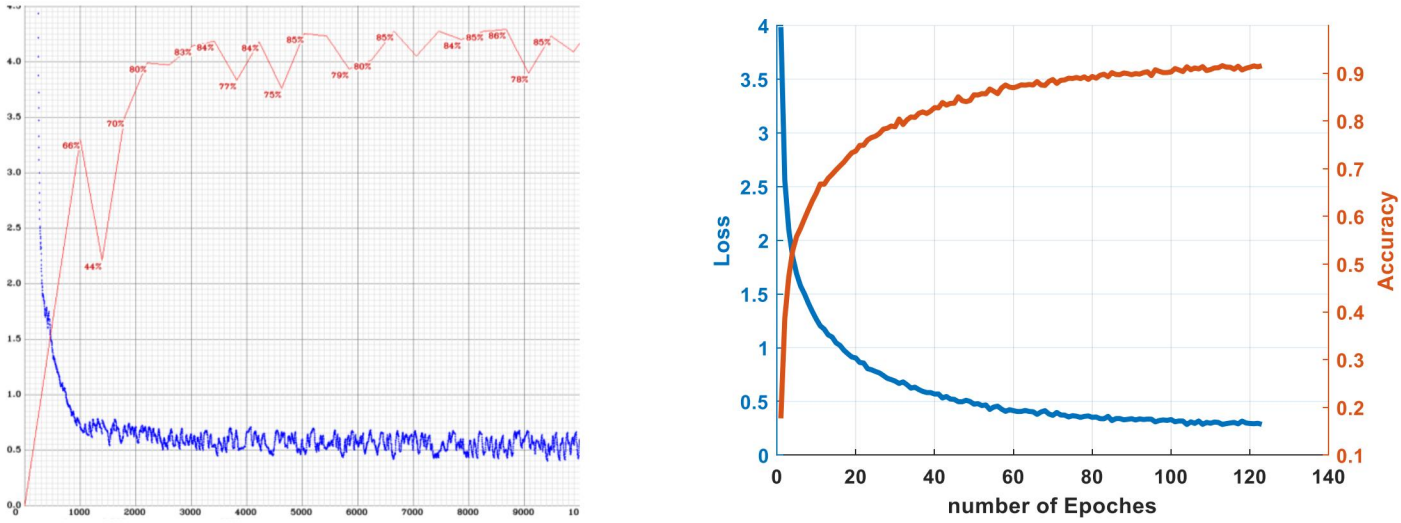

Fig. 7. Training curve of the tiny-YOLO system (left) and the Inception V3 system (right) with inputs fed from the tiny-YOLO. The overall accuracy is 91.27%.

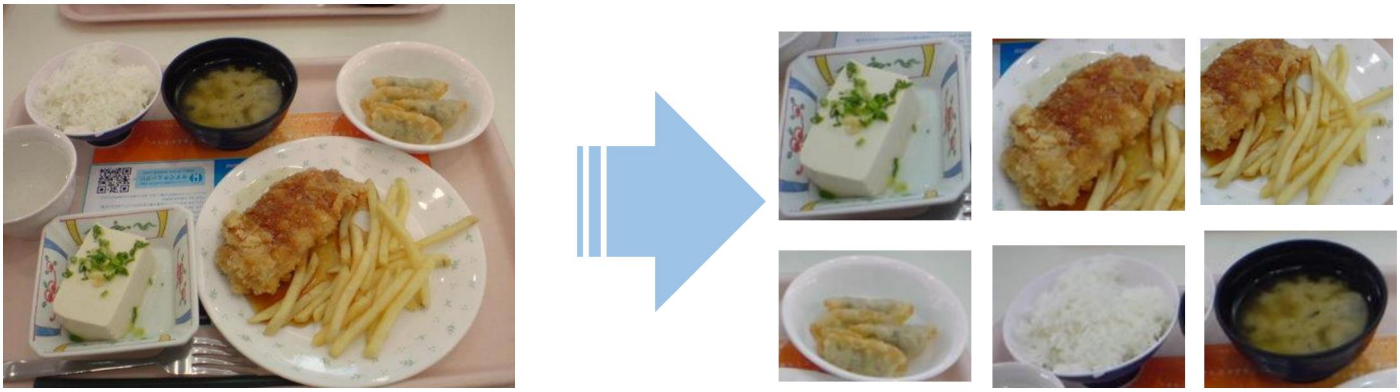

Fig. 8. Example of image dissection for individual food classification.

After the tiny-YOLO is trained, the inception V3 model is trained using the outputs of tiny-YOLO, with the original labels. A training curve is shown in Fig. 7. The overall performance of this hybrid system is better than any of the individual system, as expected. It also achieves a very high 91.27% top-1 category prediction accuracy on the validation data.

Conclusions and next steps:

This study investigates and compares the performances of several different convolutional neural network architectures. After error classifications, a hybrid system that uses a small YOLO food identifier and an InceptionV3 feature extractor to achieve the best accuracy. It is worth pointing out that the food categories in everyday meals extend far beyond 256. It requires many more labeled data for such a system to be deployed in real life. However, this system is a good starting point. To further improve the performances, several directions show great potential. The reported system is a hybrid of two individual ones, it is possible to better integrate the two system to save computational resources and training time. For example, one can incorporate Inception blocks into the YOLO system. Using anchor boxes tailored to the food can also increase its performance.

References:

[1] K.-H. Lee, X. He, L. Zhang, and L. Yang, “CleanNet: Transfer Learning for Scalable Image Classifier Training with Label Noise,” Nov. 2017.

[2] L. Bossard, M. Guillaumin, and L. Van Gool, “Food-101 – Mining Discriminative Components with Random Forests,” Springer, Cham, 2014, pp. 446–461.

[3] S. Mezgec and B. Koroušić Seljak, “NutriNet: A Deep Learning Food and Drink Image Recognition System for Dietary Assessment,” *Nutrients*, vol. 9, no. 7, p. 657, Jun. 2017.

[4] A. Singla, L. Yuan, and T. Ebrahimi, “Food/Non-food Image Classification and Food Categorization using Pre-Trained GoogLeNet Model,” in *Proceedings of the 2nd International Workshop on Multimedia Assisted Dietary Management - MADiMa ’16*, 2016, pp. 3–11.

[5] B. Cheung, “YOLO for Real-Time Food Detection.” [Online]. Available: http://bennycheung.github.io/yolo-for-real-time-food-detection. [Accessed: 06-Jun-2019].

[6] C. Szegedy *et al.*, “Going Deeper with Convolutions.” 2015.

[7] Keras, “Applications - Keras Documentation.” [Online]. Available: https://keras.io/applications/. [Accessed: 07-Jun-2019].