CS230

Food Macro Ratio Calculator

Dongze Li[*]
Department of Computer Science
Stanford University
lidongze@stanford.edu

Abstract

Food classification problems have become popular for the last few years. Food calorie calculation based on food images has also become an interesting topic since it is crucial for keeping track of a healthy diet and staying fit in general. In my project, I aim to solve the problem of estimating and calculating calories intake by introducing an end-to-end approach to directly predict the macro ratio of carbohydrates, fat and protein instead of doing traditional food classification. I experimented on VGG16 and InceptionV3 architecture and my final model settles on InceptionV3. I trained the model using 10,000 images from web crawling and the labels are generated by a simple calculation after getting the nutrient information from websites. I have achieved 0.0043 for mean squared error loss for training set and it proves that the end-to-end approach for calculating macro-nutrient information also works without specific food classification and volume measurement.

1 Introduction

With the increasing health awareness among modern people, food-intake estimation and dietary assessment have also become a main concern in our daily lives. People want to know how much calories they take in on a daily basis. However, due to the fact that not all macro-nutrient information are available when people dine out and that making food at home requires a significant commitment of time and certain culinary skills, keeping track of a healthy diet have presented some difficulties, especially for people who regularly work out and need a rigid dietary control. The food macro ratio calculator provides a generic solution by calculating the ratio of the three basic macro-nutrients: carbohydrates, protein, and fat. The input of my training model are food images which may or may not contain noises. I then used inceptionV3 convolutional neural network as a pre-trained baseline model and used softmax regression layer to output a predicted ratio vector which contains three ratios. This ratio can later be used for calories calculation if the user puts in the weight of the food.

2 Related Works

During investigation phase, I saw many implementations related to the food classification and some of them aim for providing calories estimation using image segmentation along with multi-label food classification. They include but not limited to:

1. Food recognition and leftover estimation for daily diet monitoring. [1]

[*]Use footnote for providing further information about author (Ibpage, alternative address)—*not* for acknowledging funding agencies.

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

2. Highly Accurate Food/Non-Food Image Classification Based on a Deep Convolutional Neural Network [2]

3. Food classification with Deep Learning in Keras/Tensorflow [3]

4. DeepFood: Deep Learning-Based Food Image Recognition for Computer-Aided Dietary Assessment [4]

5. Deep Learning Assisted Macro-nutrient Estimation For Feedforward-Feedback Control In Artificial Pancreas Systems [5]

Unfortunately, none of them provides a generic solution since there are endless number of food classes and similar food can take multiple forms and can appear to be very different. The most common data set for food classification is Food-101 which contains 1000 images for each of the 101 most common food. I originally experimented on this and found the complexities of the project sky-rocketed since in order to make accurate calories estimation, I would need to make accurate food classification based on more food classes hence more data. Obviously, 101 food classes wouldn't be enough and large data set would also make it harder to train and iterate. Although most of the current studies regarding the food classification as well as calories estimation use this kind of traditional approach, I decided to go for another approach due to the complexity of the food classification. I used InceptionV3 baseline model with additional convoluional layers along with softmax regression layers to directly predict the macro ratio of the food within the image.

3 Data Set and Features

Due to the fact that all of the data set for food related deep learning task are prepared specifically for food classification problems, there is no existing data set can be directly used by this project. I tried to use food-101 data set for initial experiments and found labeling data by hand was rather tedious and inefficient. Later on, I discovered that there were thousands of images posted on food recipes websites. Therefore, I ended up writing a web crawler tool with alternating proxies to download 10,000 images from those websites along with all of their ingredients and macro-nutrients information. The current data set has around 9,000 images for training set, 500 images for validation set, and another 500 images for test set to follow the 90/5/5 split. Moreover, I used the image augmentation such as flipping, cropping and mirroring for all of the training images. All images used for training had a resolution of 300x300x3 and with normalization by dividing the RGB values by 255.

4 Methods

4.1 Deep Convolutional Neural Networks

Convolutional neural network is great for image analysis as it can learn complex image features. In this project, I used the InceptionV3 as a baseline model to pre-train my training data. After that, I added a few more layers along with softmax regression in the end.

4.2 Mean Square Error Loss Function

Considering that Mean Square Error (MSE) is one of the most commonly used regression loss function which calculates the sum of squared distances between target variable and predicted values, I used MSE function for loss function as well as one of the metrics. By minimizing MSE, the model can gradually learn the predicted ratio as expected.

4.3 Adam Optimizer with Decayed Learning Rate

Adam optimizer combines the best properties of the AdaGrad and RMSProp algorithms to provide an optimization algorithm that can handle sparse gradients on noisy problems. During the experiments, I did find that the MSE loss tends to go back and forth if the learning rate is kept at the same value for the entire training. Therefore, I improved the performance by using decayed learning rate which gradually decreases based on the number of epoch. As a result, I gained a very promising result from the model.

5 Experiments/Results/Discussion

5.1 Experiments with Different Baseline Models

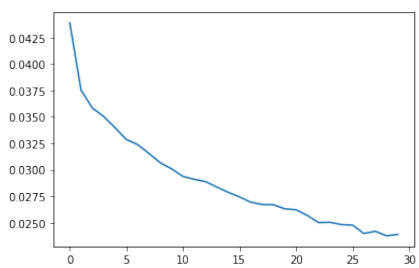
Initially, I tried lots of different pre-trained models in order to see which one worked the best for the current problem set. After experimenting with many architectures such as VGG16, ResNet, InceptionV3, and DenseNet, I concluded that InceptionV3 was more suitable for this problem, as it brought out the best performance with the lowest mean squared error loss value.

5.2 Experiments with Adding More Convultional Layers

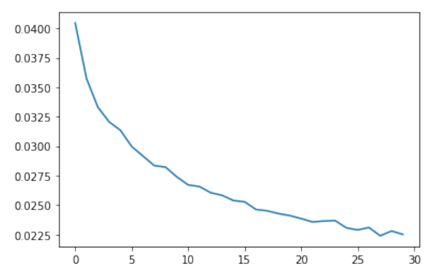
Besides my experiments with different baseline models, I also tried to add a few more convolutional layers along with the pooling layers on top of the baseline model. As in the case that deeper network would have a greater chance of success, likewise, by adding a few more layers to the model, I have improved the performance substantially by 5%.

5.3 Experiments with Batch-size for the Training Model

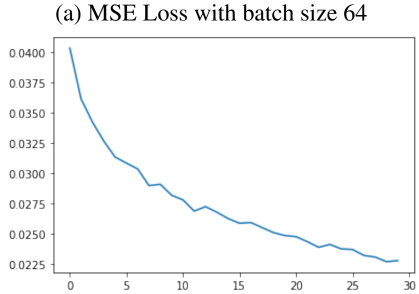
Making the batch size too big would slow down the learning process dramatically, while choosing too small of a batch size was also not ideal since it could potentially affect the final performance. Thus, I decided that a mini-batch size of 32 might be the best option. This also echoes the recommendation by Yoshua Bengio in his "Practical Recommendations for Gradient-based Training of Deep Architectures," who argues that a mini-batch size of 32 is a good default value for starting the training.[6] I experimented with batch-size of 64, 16 and 32, and compared the results altogether. The batch size of 32 still turned out to be the best option for relatively fast training without compromising the performance.



(a) MSE Loss with batch size 64



(b) MSE Loss with batch size 16



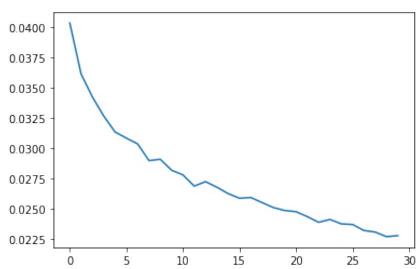
(c) MSE Loss with batch size 32

5.4 Experiments with Different Learning Rates

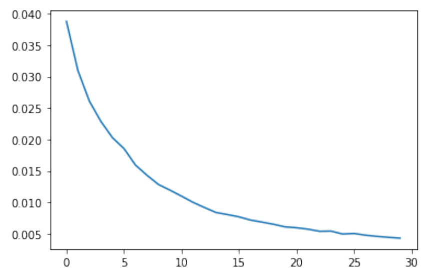
I first tried a default value of 0.0001 along with Adam optimizer just to get things started; the performance was satisfactory, although I found the MSE loss tended to jump back and forth during the training. I also tried to decrease the learning rate to 0.00005 and 0.00001, which made the learning slower and thus would not be ideal for the training at early stages. Later on I wrote a function to calculate the decayed learning rate based on the number of epochs I was running on during the training. The metric performed better, and more importantly, it did not compromise much training time as in the case of using the small learning rate all the time.

5.5 Experiments with Trainable/Non-Trainable Pre-trained Baseline Model

Initially, I kept all of the weights within the pre-trained model and used them as feature extractors for the training. Afterwards, I tried to make weights trainable for some of the later layers within InceptionV3 model and the performance improved dramatically. The final loss dropped from 0.0253 to 0.0068, and the MSE dropped from 0.0228 to 0.0043. The mean absolute error (MAE) also dropped from 0.1120 to 0.0509. Most importantly, the performance on the validation data also improved a lot. The validation loss dropped from 0.1354 to 0.03. The MSE of validation set dropped from 0.1329 to 0.0275. The MAE of validation set dropped from 0.2984 to 0.1238. The following graphs illustrate the performance improvement on the trainable baseline model:



(a) Picture 1



(b) Picture 2

Although the average training time per epoch became longer as I trained more parameters than non-trainable baseline models, it would still be beneficial for the overall result.

5.6 Experiments with Adding Food Classification as Multi-task Learning for Performance Booster

I briefly tried adding food classification as multi-task learning by using another data set of 100 images for experiments for ratio prediction along with multi-label classification. Because I did not have existing data set on the internet, I would have to label the data by hand, which cost a great amount of time and thereby became a major blocker for my task. Furthermore, usinI g only 100 images for training was not ideal and the accuracy loss for food classification would not converge regardless of the result of the ratio prediction. The main issue with the current approach was that doing multi-label classification without specifically using YOLO algorithm for object detection would be an extremely difficult learning task, and hence could not boost the performance of the original ratio prediction. Therefore, in the future I plan to write an automated system to label the corresponding food class based on the food ingredients, and in the meanwhile add the YOLO algorithm for object detection so as to give it another try.

6 Conclusion

1. Compared with many other architectures like VGG16, ResNet, DenseNet, InceptionV3 is more suitable for calculating the macro-nutrient information of food.

2. I achieved 0.0043 for mean squared error loss for training set and 0.0275 for test set, which means that there is a slight overfit on the training set. Thus, I might need more labelled data in the future.

3. Trainable layers within baseline model could be a game changer for specific task.

4. End-to-end approaches may be a good option especially when the data is limited and the problem is too complex to generalize.

5. Decayed learning rate is usually good for both performance and training iterations since it does not compromise too much training time.

6. A systematic way to get more clean labelled data is always beneficial.

7 Future Work

In terms of the future work of my project, first I need to include more labelled data, which should be the image data set where each image includes two sets of labels–one label for multi-label classification and another one for macro-nutrients ratio information.

Second, I plan to do more experiments on multi-task learning to predict the macro-nutrient ratio along with the food classification. The food classification would be multi-label classification which might require YOLO algorithm to scan through the entire image.

Next, I would also consider training another model for safety net so as to detect whether the given image input is a food image. This approach is very important for real-world applications.

Finally, I would try to train deeper convolutional neural networks by using residual blocks.

8 Contributions

I got a lot of help from Huizi as my project TA for both ideas and project directions. Other than that, I am playing solo the entire time.

9 References

[1] Ciocca G., Napoletano P., Schettini R. (2015) Food Recognition and Leftover Estimation for Daily Diet Monitoring. In: Murino V., Puppo E., Sona D., Cristani M., Sansone C. (eds) New Trends in Image Analysis and Processing – ICIAP 2015 Workshops. ICIAP 2015. Lecture Notes in Computer Science, vol 9281. Springer, Cham

[2] Kagaya H., Aizawa K. (2015) Highly Accurate Food/Non-Food Image Classification Based on a Deep Convolutional Neural Network. In: Murino V., Puppo E., Sona D., Cristani M., Sansone C. (eds) New Trends in Image Analysis and Processing – ICIAP 2015 Workshops. ICIAP 2015. Lecture Notes in Computer Science, vol 9281. Springer, Cham

[3] https://github.com/stratospark/food-101-keras

[4] Liu C., Cao Y., Luo Y., Chen G., Vokkarane V., Ma Y. (2016) DeepFood: Deep Learning-Based Food Image Recognition for Computer-Aided Dietary Assessment. In: Chang C., Chiari L., Cao Y., Jin H., Mokhtari M., Aloulou H. (eds) Inclusive Smart Cities and Digital Health. ICOST 2016. Lecture Notes in Computer Science, vol 9677. Springer, Cham

[5] A Chakrabarty, F J. Doyle, and E Dassau, "Deep Learning Assisted Macronutrient Estimation For Feedforward-Feedback Control In Artificial Pancreas Systems" in 2018 Annual American Control Conference on Milwaukee, WI, USA, IEEE, August 2018

[6] Yoshua Bengio, "Practical recommendations for gradient-based training of deep architectures", 2012