SI670: Applied Machine Learning – Final Project
An Alternative Multi-Stage Approach for Calorie Prediction on a New Recipe-Based Dataset

Dang, Khoa Tan
ktdang@umich.edu

Wong, Eman
emanwong@umich.edu

Abstract

With the ongoing global obesity pandemic, it is evident that current methods of weight loss are inadequate. In this piece of work, we seek to investigate an alternative method to traditional calorie tracking, using deep learning techniques to obtain nutritional information from an image of food. To facilitate this, we propose a new dataset, obtained through web scraping of an online recipe website in order to obtain images of foods, and their respective ingredients, then querying the API of a nutrition database to obtain the nutritional information of each ingredient and consequently that of the food in the image obtained. We evaluate various approaches leveraging pre-trained models in order to obtain regression estimates for not only calorie content, but also that of carbohydrates, proteins, and fats. Our proposed approach is a multi-stage one, and we see that it provides significant improvements over our baseline models as well as other tested approaches.

1 Introduction

Obesity is a major health concern, with the age-adjusted obesity prevalence rate amongst adults in the United States being at 42.4% from 2017-2018 (Bryan et al., 2021), according to the CDC. With the onset of the COVID-19 pandemic in 2020, we see this situation worsen, with a reported 3% increase in obesity prevalence just within the first few weeks of March 2020 (Department of Agriculture, 2022). Thus, we believe that it is important for people to make health-conscious decisions and modify their diets. Apart from the multitude of fad diets, a tried and true method is calorie tracking. In a year-long study investigating the effect of adherence to dietary tracking on weight loss, it was found that only consistent trackers ($> 66\%$ total days tracked) achieved significant weight loss over the course of the study (Ingels, Misra, Stewart, Lucke-Wold, & Shawley-Brzoska, 2017).

However, the main issue with calorie tracking is its cumbersome nature. Either the nutritional values of food need to be known (e.g. pre-packaged foods or food establishments providing nutritional values), or each individual ingredient along with its corresponding amount needs to be known. This is an extremely tedious process in the real world and poses a large barrier for many people. Therefore, we seek to create a predictive model which provides a caloric estimation for an image of food. Additionally, we provide estimates of the number of grams of carbohydrates, proteins, and fats for the image, as these individual macro-nutrients are not only important for weight loss, but also for individuals practicing certain diets, athletes who require the monitoring of their protein intake, and for general health.

In this work, data from one of the largest online recipe websites, "AllRecipes.com" was used, due to its extremely wide range of recipes, covering a multitude of cuisines from different countries and regions. Recipes on the website were scraped, obtaining their ingredients, and images.

Meal types, as well as the originated region/sub-regions, were collected as possible necessary features in the food recognition process. An API from Nutritionix is then queried with each recipe's ingredients in order to obtain its corresponding nutritional information. The Nutritionix API stores nutrition data for restaurants, food manufacturers, and USDA's common foods. It includes over 230K UPCs/barcodes, 100K restaurant foods, and 10,000 common foods from the USDA.

We leverage our scrapped-and-prepared dataset as well as the Food101 dataset (described in section 3.1.4 and 3.2.2) and apply different deep learning architectures to accomplish nutrient estimation tasks. Our propose is to consider this task as a regression problem and utilize convolutional neural networks (CNNs) models for image recognition with Huber Loss Function to make predictions so that the mean absolute error is minimized. To achieve better performance, we suggest another approach to use a pre-trained model to retrieve the estimated "weights" of specific food classes, which then can produce a better prediction of the amounts of calories, protein, fat, and carbohydrates.

2 Related Work

In the paper, "Multi-Task Learning for Calorie Prediction on a Novel Large-Scale Recipe Dataset Enriched with Nutritional Information" (Ruede et al., 2020), the authors went through a similar process of scraping a German recipe website for images and ingredients of a recipe. NLP tokenizing and embedding techniques were then used to process the ingredients, and their caloric values are obtained from a German website of nutritional values. Following this, they then ran experiments with different ResNet and DenseNet variants. The paper states that their code is publicly released on GitHub to foster further research in this area; however, the GitHub repository provided has been taken down, and it appears that the creator(s) are attempting to create an app (DishDetective), building off of their previous work.

While the focus of Ruede et al. (2020) was the formulation of an end-to-end approach, we see other works by Myers et al. (2015) and Chokr and Elbassuoni (2017) leverage multi-stage approaches on this same task. The paper "Im2Calories: towards an automated mobile vision food diary" by Myers et al. (2015) uses restaurant data and images, combining multiple models to detect different ingredients seen in the image, segment the different ingredients, estimate their respective sizes, and using the USDA National Nutrient Database to obtain the nutritional information of the dish. One limitation of this paper is the lack of end-to-end testing outside of restaurant settings. The authors noted that they found the calorie values provided by USDA were unreliable, with most entries focused on raw ingredients and not cooked foods. The paper "Calories Prediction from Food Images" by Chokr and Elbassuoni (2017) uses the combination of a size predictor along with a food classifier in order to estimate calories. However, the dataset used is extremely clean, with food photographed against a white background, covering only 6 classes of food – burger, chicken, doughnuts, pizza, salad, and sandwich. We find that this serves more as a proof of concept, since there is no noise in their dataset alongside the narrow scope of foods covered, and is unrealistic in real-world applications.

3 Methods

3.1 Dataset

3.1.1 AllRecipes.com Scraping

We manually collected a total of 322 Allrecipes URL links and categorized them into groups by meal types, regions, and sub-regions. Then, we started to scrape the data from the collected URLs and managed to retrieve 13,319 recipes. However, there are a lot of duplicated recipes because certain dishes are common in multiple regions (for example, the dish "A Number One Egg Bread" is listed under recipes from Hawaii as well as Australia), or are categorized as different food types (for example, the dish "Chicken Tortilla Soup" are listed under both main dish and stews) depending on the regions. To handle this issue, we merged those duplicated values and updated their food types, and regions, which reduces the number of observations to 8883. The scraped images were resized to 224×224 pixels. This is believed to help with the training process when keeping up with our computing limitations (Paul, 2021).

One issue with the images obtained from Allrecipes is that the image does is not always representative of the serving size of the recipe. For example, a pie recipe could have an image of an entire pie while the serving size (and relevant nutritional information obtained) is for a single slice. We were unable to address this in the scope of this project and these images are simply left as noisy inputs.

3.1.2 Obtaining Nutritional Information

The nutrition information is available for the majority of recipes on Allrecipes. However, we wanted to create a methodology for building a larger dataset, which is applicable to multiple different websites where nutrition data may not be available. After querying Nutrionix's API for the nutritional information of the recipe, we then divide by the number of servings to obtain each individual nutritional value per serving.

The Nutritionix API has a powerful capability in interpreting the provided string of ingredients. For example, the query "1 handful of spinach, salt to taste, 5lbs peeled potatoes", returns the nutritional information of 2 ounces of spinach, 0.25 teaspoons of salt, and 5 pounds of potatoes. We see that it is able to correctly interpret the ingredients and their respective measurements, giving approximations for vague quantifiers like "handful" and "to taste", and ignoring irrelevant ingredient descriptions such as "peeled". Additionally, the API can automatically filter out irrelevant keywords to provide a more precise estimation of the nutrition information, such as "1 (16 ounces) package wide egg noodles" to "1 cup of egg noodles". The returned result from the API was then cleaned to obtain our desired information covering calories, fat, and protein amounts. Lastly, the API does not require us to specify the amount of 'targets'. By default, the API will return the smallest and most consistent measure unit of each ingredient type as the serving (for example, the serving size for "pizza" is slice).

To make an API call to Nutritionix, we removed special characters and then queried the whole list of ingredients of each recipe. Despite the natural language processing done by Nutrionix, there is a level of noise in the returned values due to ingredients being interpreted incorrectly. Thus, we would ideally be able to query each ingredient separately to obtain better results; how-

ever, for the scope of this work, we were limited by the number of daily API calls that we could make.

3.1.3 Data Cleaning

As mentioned in subsection 3.1.2, there were several instances of Nutrionix's backend's natural language processing failing to interpret certain ingredients correctly. It appears that the API still requires correct English grammar to a certain extent to produce better results. For example, the phrase, "4 Sweet bread buns", is interpreted as "4 slices of sweet bread" and "1 bun". Misinterpretations like this create noisy labels in our dataset.

Meal Type	Count
Unknown	3427
main	1926
dessert	853
side	568
soups_and_stews	524
appetizer	499
salad	469
bread	348
drinks	148

Table 1: Counts of recipes by meal type

Therefore, we utilized the nutrition data from Allrecipes to validate what we obtained from the Nutritionix API by eliminating any observations retrieved from Nutritionix with abnormal nutrition values. We did this by using nutrition data from Allrecipes as a reference, and dropping observations that had a caloric value that was 20% greater than or 20% less than that provided by Allrecipes. This cleaning step reduced the size of our dataset to 5219 observations. Lastly, we manually checked the recipes left that had caloric values 19% greater than or 19% less than Allrecipes in order to validate that our edge cases were accurate. The respective counts of recipes split by meal type are shown in table 1.

3.1.4 Food101

This dataset consists of 101 food categories, with over 101,000 images. Each category, as known as a food type, includes 250 well-prepared test images and 750 train images with little intended noise and can be easily imported from TensorFlow's datasets library. We do not utilize this dataset directly, but instead make use of a model fine-tuned on it, which we expand on in section 3.2.2.

3.2 Models

3.2.1 Initial Modelling

For our initial modeling, we began with a simple calorie regression task, by performing transfer learning with DenseNet121, our pre-trained CNN of choice. Densenet121 was chosen as it was the best-performing model on the same task when compared with other ResNet and Densenet models (Ruede et al., 2020). In terms of pre-processing, the images would first go through two augmentation layers in order to randomly flip and rotate the images. This was done in order to reduce the likelihood of overfitting, by creating a more robust model through the added diversity in our data (Shorten & Khoshgoftaar, 2019). These augmented inputs would then be passed into the pre-trained DenseNet121 model and the outputs are passed through a global average pooling layer, followed by a dropout layer, and finally, a single node linear layer. This model was subsequently used as a "base" which the rest of our models was built on.

Subsequently, we attempted a multi-task approach as suggested by Ruede et al. (2020). This model was identical to our simple calorie regression model, except for the last layer which was changed to contain 4 linear nodes; one each for calories, protein, carbohydrates, and fats.

Next, we attempted a more complex model which also took in 2 categorical variables as input, namely meal-type, and region. These categorical inputs were one-hot encoded and passed through a simple feedforward neural network (FNN). We then concatenate the outputs of this FNN with that of our CNN "block" before passing them through our final linear layer to obtain our predictions for the 4 variables.

3.2.2 Proposed Model Architecture

For our proposed model architecture we suggest a novel multi-stage procedure, in which we also pass the image into a food classification model, obtain the top 5 food predictions and their respective probabilities, then query the Nutritionix API for the nutritional information of the 5 food items and take its weighted average based on their probabilities. This weighted average is then passed into the FNN alongside the categorical features mentioned earlier, and similarly, the FNN's output is concatenated with that of the CNN, before passing through a final linear layer.

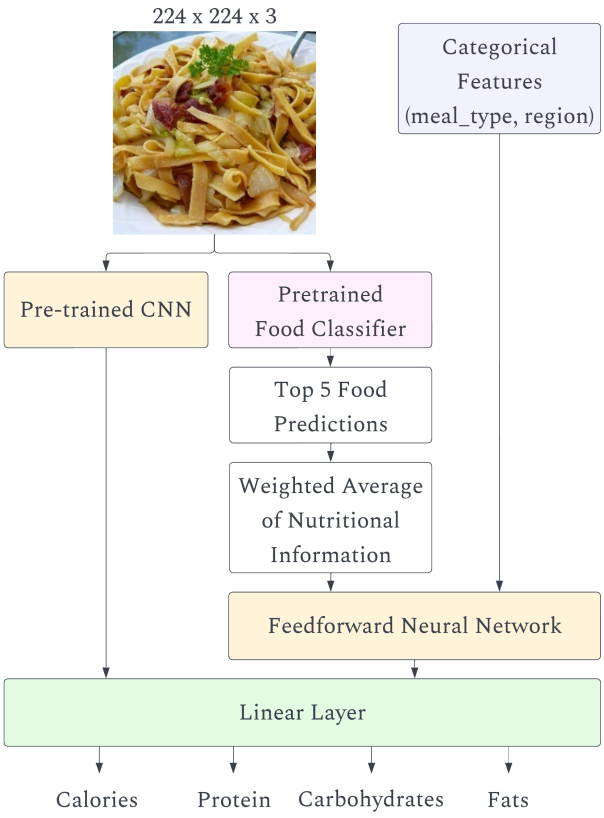

Figure 1: Overview of proposed model architecture

In this case, the pre-trained model we used for food classification was HuggingFace user `skylord`'s "swin-finetuned-food101" model, from HuggingFace's model hub (HuggingFace, 2022). This model is based on Microsoft's state-of-the-art Swin Transformer model and was fine-tuned on the Food101 dataset mentioned earlier in section 3.1.4. The food in our dataset encompasses much more than the 101 categories in the Food101 dataset, but we hope that the predicted foods will be a close approximation to our actual dish. Additionally, by averaging their corresponding nutritional values as found in the Nutritionix database, we would be able to reduce the variance of this estimation.

As seen from Figure 2, when the pre-trained classifier is presented with an image of Mexican Cornbread, the top 5 classes returned are carrot cake, bread pudding, cheesecake, apple pie, and risotto. Evidently, the prediction "risotto" is very far removed from cornbread, but its nutritional values will have a very low weight in the engineered nutritional information features. Additionally, de-

Figure 2: Example output of the pre-trained Swin model finetuned on the Food101 dataset, for an image of Mexican Cornbread

spite inaccurately classifying the image, with cornbread not being included in the 101 categories, we are able to obtain a very rough approximation of the item's nutritional information, which the model can then learn from.

For all of our models, the loss function used was Keras's Huber loss, which is similar to PyTorch's Smooth L1-loss (the choice of loss for Ruede et al. (2020)), and equivalent depending on the choice of loss hyper-parameters. Huber loss was chosen as it combines the desirable properties of both Mean Absolute Error and Mean Squared Error, giving a more robust loss function (Gokcesu & Gokcesu, 2021).

4 Evaluation and Analysis

4.1 Setup

For all models, the transfer learning was done with Adam as the optimizer with a default learning rate, for 100 epochs with early stopping patience of 3 (none of the models ran for 100 epochs), and a batch size of 32. Fine-tuning of the pre-trained CNNs used for transfer learning (unfreezing some of the pre-trained weights) was not performed as our primary goal was to investigate the models and approaches that are effective for this regression task. 20% of our data was used as our final test set, with 20% of the remaining training data used as a validation set for training.

For our proposed model architecture, we also experimented with multiple other pre-trained CNNs other than DenseNet121, namely MobileNetV2, EfficientNetV2B0, and NASNetMobile. These models were selected as they are all lightweight models suitable for deployment on mobile devices. This matches the intended use case of our models, where an end user could take an image of their meal on their phone, and a mobile application utilizing said model could return the nutritional information of their meal.

In order to evaluate the performance of our models, we implement three different baselines. The first one uses the average of each nutritional value in the training set, while the other two take these same averages after grouping the data by either meal type or region.

4.2 Results

The evaluation of the quality of our predictions is done with mean absolute error as our metric. This was chosen over other metrics such as mean squared error, as it is easily interpretable, and is on the same scale as our target variables (kcal for calories, and grams for carbohydrates, proteins, and fats).

Models	Calories	Protein	Fat	Carbohydrate
Mean Baseline	169.23	13.28	10.88	18.27
Mean Baseline (Meal type)	156.56	10.20	10.24	17.28
Mean Baseline (Region)	168.18	13.01	10.86	18.28
Calories Only	169.52	NA	NA	NA
Multi. Outputs	170.64	12.06	10.68	18.19
Multi. Outputs with Categorical Variables	154.60	10.32	10.01	18.07
Proposed approach	**147.79**	9.43	9.84	17.35

Table 2: Mean absolute errors of different models utilising DenseNet121

Table 2 shows that among the 3 baseline approaches, the baseline which averages nutritional information by meal type yielded the best performance. Additionally, we see that the regression model which had multiple outputs did not see improvements over the calories-only regression model, contrary to the findings of Ruede et al. (2020). However, since the difference in MAE between the two models was so small, we proceeded with a multiple output approach, which also provided additional valuable information. In our next model, where meal-type and region were also used as inputs, we see significant improvements compared to our previous models, also beating all of our baselines. This suggests that our model is able to learn the differences between different meal types, as well as regions, to augment the outputs from the image regression to provide more accurate predictions. Lastly, we see that our proposed approach is able to further improve on this significantly, indicating that rough nutritional estimates used as inputs can improve results in such a task. We believe that such improvements are due to the model essentially refining these nutritional estimates using the categorical and image data provided.

Pre-trained CNN	Calories	Protein	Fat	Carbohydrate
DenseNet121	147.79	9.43	9.84	17.35
MobileNetV2	149.90	10.16	10.2	17.88
EfficientNetV2B0	150.21	9.54	9.99	17.74
NASNetMobile	147.19	9.84	9.90	17.42

Table 3: Mean absolute errors of our proposed approach utilizing different pre-trained CNN architectures

With our proposed approach, we wanted to compare its performance using a variety of pre-trained CNN models and their results are shown in table 3. We see that DenseNet121 and NASNetMobile provide the best results in general, with NASNetMobile achieving slightly better performance in terms of calorie prediction.

5 Discussion and Conclusion

There are several avenues that this work can be improved upon in the future. In terms of the dataset, the noisy inputs due to images containing a different number of servings, as outlined in section 3.1.1, could be addressed using machine learning techniques such as distant supervision, or through services such as Amazon Mechanical Turk. Additionally, as mentioned in section 3.1.2, since our number of API calls was limited, we queried long strings of ingredients instead of each ingredient separately, which could be changed in future works.

As for the modeling, we believe the model can be improved using a classifier trained on a broader dataset, covering more than 101 food types. We believe that other techniques such as weight estimation and image segmentation can be combined with our proposed approach to improve the model even further.

In conclusion, we have introduced a brand new dataset of food images and nutritional information, containing 5219 unique recipes. We then tested and evaluated several modeling approaches and CNN architectures and showed that our proposed multi-stage approach, utilizing a food classifier, is able to yield significant improvements.

References

Bryan, S., Afful, J., Carroll, M., Te-Ching, C., Orlando, D., Fink, S., & Fryar, C. (2021, June). *NHSR 158. national health and nutrition examination survey 2017–march 2020 pre-pandemic data files* (Tech. Rep.). Retrieved from https://doi.org/10.15620/cdc:106273 doi: 10.15620/cdc:106273

Chokr, M., & Elbassuoni, S. (2017). Calories prediction from food images. In *Proceedings of the thirty-first aaai conference on artificial intelligence* (p. 4664–4669). AAAI Press.

Department of Agriculture, U. (2022). *Adult obesity prevalence increased during the first year of the covid-19 pandemic.* Retrieved from https://www.ers.usda.gov/amber-waves/2022/july/adult-obesity-prevalence-increased-during-the-first-year-of-the-covid-19-pandemic/#:~:text=The%20study%20found%20that%2C%20compared,of%20the%20COVID%2D19%20pandemic.

Gokcesu, K., & Gokcesu, H. (2021). *Generalized huber loss for robust learning and its efficient minimization for a robust statistics.* arXiv. Retrieved from https://arxiv.org/abs/2108.12627 doi: 10.48550/ARXIV.2108.12627

HuggingFace. (2022). *Model hub, skylord/swin-finetuned-food101.* https://huggingface.co/skylord/swin-finetuned-food101. (Accessed: 2022-12-08)

Ingels, J. S., Misra, R., Stewart, J., Lucke-Wold, B., & Shawley-Brzoska, S. (2017). The effect of adherence to dietary tracking on weight loss: Using HLM to model weight loss over time. *Journal of Diabetes Research*, *2017*, 1–8. Retrieved from https://doi.org/10.1155/2017/6951495 doi: 10.1155/2017/6951495

Myers, A., Johnston, N., Rathod, V., Korattikara, A., Gorban, A., Silberman, N., ... Murphy, K. (2015, December). Im2calories: Towards an automated mobile vision food diary. In *2015 IEEE international conference on computer vision (ICCV).* IEEE. Retrieved from https://doi.org/10.1109/iccv.2015.146 doi: 10.1109/iccv.2015.146

Paul, S. (2021, Apr). *Keras documentation: Learning to resize in computer vision.* Retrieved from https://keras.io/examples/vision/learnable_resizer/

Ruede, R., Heusser, V., Frank, L., Roitberg, A., Haurilet, M., & Stiefelhagen, R. (2020). *Multi-task learning for calorie prediction on a novel large-scale recipe dataset enriched with nutritional information.*

Shorten, C., & Khoshgoftaar, T. M. (2019, July). A survey on image data augmentation for deep learning. *Journal of Big Data*, *6*(1). Retrieved from https://doi.org/10.1186/s40537-019-0197-0 doi: 10.1186/s40537-019-0197-0