pic2kcal: End-to-End Calorie Estimation From Pictures of Food

Robin Ruede, Lukas Frank, Verena Heußer
Karlsruhe Institute of Technology

Abstract

We estimate kcal directly from a picture. It good.

1. Introduction

Food is life.

2. Related Work

There's some other papers like [1]–[3]. Ours is more end to end and also BETTER

3. Dataset Extraction and Preprocessing

3.1. Collection

We collected a dataset from the a popular German recipe website that contains ingredient lists, cooking instructions, and pictures of the resulting meals. The recipes are from many different cuisines and also include things like cakes, cocktails, and other. Most recipes have at least one picture. Pictures can come both from the original author of the recipe, as well as from third parties. Some of the pictures are of a single plate of food, others are for example of a whole casserole. We do not have any information about whether a picture contains a single portion. Around 10% of recipes contain a user-given value for how many calories per portion the recipe has.

3.2. Matching / Preprocessing

Since the dataset only has user-given calorie information for a small part of the data and doesn't include any details regarding the macronutrient composition, and since the user given information is often inaccurate [todo: compare to ours??], we decided to match the list of ingredients against a database of nutritional values to sum up the proportions of macronutrients as well as the total calories.

To facilitate this, we collected a secondary dataset from a German website of nutritional values. The website contains values for the amount of fat, protein, and carbohydrates in grams per 100g of product. Additionally, it contains user-sourced food amounts like "1 medium-sized apple = 130g". The data is partially sourced from the USDA Food Composition Database [4], and partially crowd sourced from manufacturer-given data of specific products.

Matching the recipe ingredients to the nutritional database has two main problems.

Firstly, the given ingredient name often includes information that is not relevant to the product itself, but rather to its preparation or visual qualities. These additional text snippets are hard to separate from information that is relevant. For example: `3 onions, diced` and `3 onions, in slices` refer to the same product, while `500g pasta, cooked` and `500g pasta, raw` vary significantly in their caloric density. We experimented with three approaches [reformulate] to solve this problem. Firstly, we tried simply matching the ingredient to the nearest ingredient based on character edit distance (Levenshtein distance). This resulted in very bad matchings because of missing handling of synonyms and the above issue. To solve this we tokenize the ingredient name to words, embed each word to a vector with Word2Vec or FastText [5], and then use average the word vectors to get a ingredient vector. This is the same method as used in the fasttext library for extracting sentence vectors. This still lead to unsatisfactory results, since each word in the ingredient name has the same weight, even though some specify less important details. For example in "red onion" vs "red apple", the word "red" is much less important than "onion" and "apple". We got the best result by using the Google Universal Sentence Encoder [6], which creates 512-dimensional embeddings of any amount of text. We find the best matches for a ingredient by comparing the embedding of user-given free text from the recipe to the embeddings for all food items for which we have nutritional data using the cosine distance, and then try to find a conversion for the given amount to a normalized gram or milliliter amount.

The second problem is matching the amounts. For ingredients given in grams this is trivial, but for many items the recipe authors use other units of measure like e.g. can, piece, tablespoon, "some", "2 large X", "salt 'by taste' ". Since spices usually have little impact on the nutritional values, we exclude ingredients that are "by taste" and similar. For the other amounts, we match the unit name (like table-

spoon or "medium large") exactly and multiply it with the given amount. We also add some special cases like matching "can" to "can (drained weight)" and similar.

The amount matching is applied to all possible ingredient matches that are similar by more than 84% (measured by cosine distance) [why lol] in decending order, or to the single closest ingredient if there is no match more accurate than 84%.

If the amount matching fails, the ingredient is marked as unmatched. If a recipe has at least one unmatched ingredient, it is discarded. With this method we have full nutritional information for around 38% of all recipes. This could be improved with further tweaking.

3.3. Dataset Statistics

- statistics about dataset
- number of pictures per recipe
- number of recipes
- most common ingredients
- cake bias

4. Models

We followed an end-to-end approach to solve the calorie prediction problem of food images. To do so we used a pretrained ResNet and DenseNet architecture. We kept the feature extractor layers and replaced the last fully-connected classification layer. We try to solve the problem interpreting it on the one hand as a classification task and on the other hand as a regression problem. Furthermore we introduced additional learning feedback following a multi-task approach.

We describe in the following only the last layer of the neural network.

In the regression case we trained a model predicting only the kcal information with one output neuron and another to predict additionally protein, fat and carbohydrates information using four neurons. The two models were trained using a L1 and smooth L1 loss.

We transformed the two already described models to a classification problem quantizing the regression outputs. So we introduced 50 class buckets for each regression output. The models were trained using a cross entropy loss.

The multi-task model is based on the regression model including the nutritional information with additional binary outputs to predict the top n ingredients. The resulting layer has four regression outputs with 50 binary outputs. The used loss combines a smooth L1 loss for the regression outputs and an binary cross entropy for the top 50 ingredients. To get the same scaling of the two learning signals we scaled the binary cross entropy loss with a factor of 400.

$$loss = L1 + 400 * BCE$$

5. Experiments

We divided the generated dataset into train/test/validation (xx/xx/xx) splits. Our training set contains xxx samples with around xxx images each recipe. The network was trained 40 epochs using a batch size of 50 samples each batch. The samples of the batches were shuffled every epoch and we evaluated the performance of the model every fiftieth batch. We implemented all networks using Pytorch.

To evaluate the performance of the model we trained several networks and run several experiments to evaluated them using firstly the evaluation data set to get quick feedback. To measure the performance of the model we only compared the given kcal information with the prediction of the network.

Firstly we used our raw data set to train the kcal-model. We wanted the network to predict the kcal information of the recipe visualized on the given input image. To perform well in this task the model needs to learn the concept of the recipe size and predict the calories according it. We assumed due to the amount of samples and the capacity of the model the problem is well learnable. Unfortunately the trained regression model performed not well on the task probably because of outlier recipes in our dataset with not valid kcal information provided by the users. Even after outlier removal, prediction of normalized kcal information of portion and trying a classification approach the model was only slightly better than a baseline model.

Second we evaluated if the additional nutritional information supports the networks capability to generalize on the recipe and portion size. Both the classification and regression objectives performed not well with the further information.

Lastly we reformulated the training objective to a slightly easier problem. We trained the network to predict the calory density of the visualized image. Because of the normalization the network only needs to grasp how many calories are in for instance 100g of the meal. This modification led to significant better results.

We could furthermore improve the results of the model using the multi-task approach. The top 50 ingredients of the recipes were injected as further information to support the model predicting the kcal information. We report the results of this model in the result section.

Method	kcal relative error
baseline	0.464
ours (kcal only)	0.361
ours (w/ macros)	0.352
ours (w/ macros+ings)	0.328

Table 1. Results per 100g. Note that multitask learning improves performance.

6. Results

Our results can be seen in tbl. 1. Example outputs can be seen in fig. 1.

Figure 1. Some example results, showing predicted calories, fat, protein, carbohydrates and ingredients.

References

[1] M. Chokr and S. Elbassuoni, "Calories Prediction from Food Images," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 2017, pp. 4664–4669.

[2] T. Ege and K. Yanai, "Image-Based Food Calorie Estimation Using Knowledge on Food Categories, Ingredients and Cooking Directions," in *Proceedings of the on Thematic Workshops of ACM Multimedia 2017*, 2017, pp. 367–375.

[3] A. Romero, X. Giro-i-Nieto, M. Drozdzal, and A. Salvador, "Inverse Cooking: Recipe Generation from Food Images," Dec. 2018.

[4] *USDA Food Composition Databases*..

[5] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching Word Vectors with Subword Information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, Dec. 2017.

[6] D. Cer *et al.*, "Universal Sentence Encoder," in *In submission to: EMNLP demonstration*, 2018.