Deep Neural Networks for Image-Based Dietary Assessment

Deep Neural Networks for Image-Based Dietary Assessment

Simon Mezgec[1], **Barbara Koroušić Seljak**[2]

[1] Jožef Stefan International Postgraduate School [2] Computer Systems Department, Jožef Stefan Institute

Corresponding Author

Simon Mezgec
simon.mezgec@gmail.com

Citation

Mezgec, S., Koroušić Seljak, B. Deep Neural Networks for Image-Based Dietary Assessment. *J. Vis. Exp.* (169), e61906, doi:10.3791/61906 (2021).

Date Published

March 13, 2021

DOI

10.3791/61906

URL

jove.com/video/61906

Abstract

Due to the issues and costs associated with manual dietary assessment approaches, automated solutions are required to ease and speed up the work and increase its quality. Today, automated solutions are able to record a person's dietary intake in a much simpler way, such as by taking an image with a smartphone camera. In this article, we will focus on such image-based approaches to dietary assessment. For the food image recognition problem, deep neural networks have achieved the state of the art in recent years, and we present our work in this field. In particular, we first describe the method for food and beverage image recognition using a deep neural network architecture, called NutriNet. This method, like most research done in the early days of deep learning-based food image recognition, is limited to one output per image, and therefore unsuitable for images with multiple food or beverage items. That is why approaches that perform food image segmentation are considerably more robust, as they are able to identify any number of food or beverage items in the image. We therefore also present two methods for food image segmentation - one is based on fully convolutional networks (FCNs), and the other on deep residual networks (ResNet).

Introduction

Dietary assessment is a crucial step in determining actionable areas of an individual's diet. However, performing dietary assessment using traditionally manual approaches is associated with considerable costs. These approaches are also prone to errors as they often rely on self-reporting by the individual. Automated dietary assessment addresses these issues by providing a simpler way to quantify and qualify food intake. Such an approach can also alleviate some of the errors present in manual approaches, such as missed meals, inability to accurately assess food volume, etc. Therefore, there are clear benefits to automating dietary assessment by developing solutions that identify different foods and beverages and quantify food intake[1]. These solutions can also be used to enable an estimation of nutritional values of food and beverage items (henceforth 'food items'). Consequently, automated dietary assessment is useful for multiple applications - from strictly medical uses, such as allowing dietitians to more easily and accurately track and analyze their patients' diets, to the usage inside well-being apps targeted at the general population.

Automatically recognizing food items from images is a challenging computer vision problem. This is due to foods being typically deformable objects, and due to the fact that a large amount of the food item's visual information can be lost during its preparation. Additionally, different foods can appear to be very similar to each other, and the same food can appear to be substantially different on multiple images[2]. Furthermore, the recognition accuracy depends on many more factors, such as image quality, whether the food item is obstructed by another item, distance from which the image was taken, etc. Recognizing beverage items presents its own set of challenges, the main one being the limited amount of visual information that is available in an image. This information could be the beverage color, beverage container color and structure, and, under optimal image conditions, the beverage density[2].

To successfully recognize food items from images, it is necessary to learn features of each food and beverage class. This was traditionally done using manually-defined feature extractors[3,4,5,6] that perform recognition based on specific item features like color, texture, size, etc., or a combination of these features. Examples of these feature extractors include multiple kernel learning[4], pairwise local features[5] and the bag-of-features model[6]. Due to the complexity of food images, these approaches mostly achieved a low classification accuracy - between 10% and 40%[3,4,5]. The reason for this is that the manual approach is not robust enough to be sufficiently accurate. Because a food item can vary significantly in appearance, it is not feasible to encompass all these variances manually. Higher classification accuracy can be achieved with manually-defined feature extractors when either the number of food classes is reduced[5], or different image features are combined[6], thus indicating that there is a need for more complex solutions to this problem.

This is why deep learning proved to be so effective for the food image recognition problem. Deep learning, or deep neural networks, was inspired by biological brains, and allows computational models composed of multiple processing layers to automatically learn features through training on a set of input images[7,8]. Because of this, deep learning has substantially improved the state of the art in a variety of research fields[7], with computer vision, and subsequently food image recognition, being one of them[2].

In particular, deep convolutional neural networks (DCNNs) are most popular for food image recognition - these networks are inspired by the visual system of animals, where individual neurons try to gain an understanding of the visual input by reacting to overlapping regions in the visual field[9]. A convolutional neural network takes the input image and performs a series of operations in each of the network layers, the most common of which are convolutional, fully-connected and pooling layers. Convolutional layers contain learnable filters that respond to certain features in the input data, whereas fully-connected layers compose output data from other layers to gain higher-level knowledge from it. The goal of pooling layers is to down-sample the input data[2]. There are two approaches to using deep learning models that proved popular: taking an existing deep neural network definition[10,11], referred to as a deep learning architecture in this article, or defining a new deep learning architecture[12,13], and training either one of these on a food image dataset. There are strengths and weaknesses to both approaches - when using an existing deep learning architecture, an architecture that performed well for other problems can be chosen and fine-tuned for the desired problem, thus saving time and ensuring that a validated architecture has been chosen. Defining a new deep learning architecture, on the other hand, is more time-intensive, but allows the development of architectures that are specifically made to take into account the specifics of a problem and thus theoretically perform better for that problem.

In this article, we present both approaches. For the food image recognition problem, we developed a novel DCNN architecture called NutriNet[2], which is a modification of the well-known AlexNet architecture[14]. There are two main differences compared to AlexNet: NutriNet accepts 512x512-pixel images as input (as opposed to 256x256-pixel images for AlexNet), and NutriNet has an additional convolutional layer at the beginning of the neural network. These two changes were introduced in order to extract as much information from the recognition dataset images as possible. Having higher-resolution images meant that there is more information present on images and having more convolutional layers meant that additional knowledge could be extracted from the images. Compared to AlexNet's around 60 million parameters, NutriNet contains less parameters: approximately 33 million. This is because of the difference in dimensionality at the first fully-connected layer caused by the additional convolutional layer[2]. **Figure 1** contains a diagram of the NutriNet architecture. The food images that were used to train the NutriNet model were gathered from the Internet - the procedure is described in the protocol text.

For the food image segmentation problem, we used two different existing architectures: fully convolutional networks (FCNs)[15] and deep residual networks (ResNet)[16], both of which represented the state of the art for image segmentation when we used them to develop their respective food image segmentation solutions. There are multiple FCN variants that were introduced by Long et al.: FCN-32s, FCN-16s and FCN-8s[15]. FCN-32s outputs a pixel map based on the predictions by the FCN's final layer, whereas the FCN-16s variant combines these predictions with those by an earlier layer. FCN-8s considers yet another layer's predictions and is therefore able to make predictions at the finest grain, which is why it is suitable for food image recognition. The FCN-8s that we used was pre-trained on the PASCAL Visual Object Classes (PASCAL VOC) dataset[17] and trained and tested on images of food replicas (henceforth 'fake food')[18] due to their visual resemblance to real food and due to a lack of annotated images of real food on a pixel level. Fake food is used in different behavioral studies and images are taken for all dishes from all study participants. Because the food contents of these images are known, it makes the image dataset useful for deep learning model training. Dataset processing steps are described in the protocol text.

The ResNet-based solution was developed in the scope of the Food Recognition Challenge (FRC)[19]. It uses the Hybrid Task Cascade (HTC)[20] method with a ResNet-101[16] backbone. This is a state-of-the-art approach for the image segmentation problem that can use different feature extractors, or backbones. We considered other backbone networks as well, particularly other ResNet variants such as ResNet-50[16], but ResNet-101 was the most suitable due to its depth and ability to represent input images in a complex enough manner. The dataset used for training the HTC ResNet-101 model was the FRC dataset with added augmented images. These augmentations are presented in the protocol text.

This article is intended as a resource for machine learning experts looking for information about which deep learning architectures and data augmentation steps perform well for the problems of food image recognition and segmentation, as well as for nutrition researchers looking to use our approach to automate food image recognition for use in dietary assessment. In the paragraphs below, deep learning solutions and datasets from the food image recognition field are presented. In the protocol text, we detail how each of the three approaches was used to train deep neural network models that can be used for automated dietary assessment. Additionally, each protocol section contains a description of how the food image datasets used for training and testing were acquired and processed.

DCNNs generally achieved substantially better results than other methods for food image recognition and segmentation, which is why the vast majority of recent research in the field is based on these networks. Kawano et al. used DCNNs to complement manual approaches[21] and achieved a classification accuracy of 72.26% on the UEC-FOOD100 dataset[22]. Christodoulidis et al. used them exclusively to achieve a higher accuracy of 84.90% on a self-acquired dataset[23]. Tanno et al. developed DeepFoodCam - a smartphone app for food image recognition that uses DCNNs[24]. Liu et al. presented a system that performs an Internet of Things-based dietary assessment using DCNNs[25]. Martinel et al. introduced a DCNN-based approach that exploits the specifics of food images[26] and reported an accuracy of 90.27% on the Food-101 dataset[27]. Zhou et al. authored a review of deep learning solutions in the food domain[28].

Recently, Zhao et al. proposed a network specifically for food image recognition in mobile applications[29]. This approach uses a smaller 'student' network that learns from a larger 'teacher' network. With it, they managed to achieve an accuracy of 84% on the UEC-FOOD256[30] and an accuracy of 91.2% on the Food-101 dataset[27]. Hafiz et al. used DCNNs to develop a beverage-only image recognition solution and reported a very high accuracy of 98.51%[31]. Shimoda et al. described a novel method for detecting plate regions in food images without the usage of pixel-wise annotation[32]. Ciocca et al. introduced a new dataset containing food items from 20 different food classes in 11 different states (solid, sliced, creamy paste, etc.) and presented their approach for training recognition models that are able to recognize the food state, in addition to the food class[33]. Knez et al. evaluated food image recognition solutions for mobile devices[34]. Finally, Furtado et al. conducted a study on how the human visual system compares to the performance of DCNNs and found that human recognition still outperforms DCNNs with an accuracy of 80% versus 74.5%[35]. The authors noted that with a small number of food classes, the DCNNs perform well, but on a dataset with hundreds of classes, human recognition accuracy is higher[35], highlighting the complexity of the problem.

Despite its state-of-the-art results, deep learning has a major drawback - it requires a large input dataset to train the model on. In the case of food image recognition, a large food image dataset is required, and this dataset needs to encompass as many different real-world scenarios as possible. In practice this means that for each individual food or beverage item, a large collection of images is required, and as many different items as possible need to be present in the dataset. If there are not enough images for a specific item in the dataset, that item is unlikely to be recognized successfully. On the other hand, if only a small number of items is covered by the dataset, the solution will be limited in scope, and only able to recognize a handful of different foods and beverages.

Multiple datasets were made available in the past. The Pittsburgh Fast-Food Image Dataset (PFID)[3] was introduced to encourage more research in the field of food image recognition. The University of Electro-Communications Food 100 (UEC-FOOD100)[22] and University of Electro-Communications Food 256 (UEC-FOOD256)[30] datasets contain Japanese dishes, expanded with some international dishes in the case of the UEC-FOOD256 dataset. The Food-101 dataset contains popular dishes acquired from a website[27]. The Food-50[36] and Video Retrieval Group Food 172 (VireoFood-172)[37] datasets are Chinese-based collections of food images. The University of Milano-Bicocca 2016 (UNIMIB2016) dataset is composed of images of food trays from an Italian canteen[38]. Recipe1M is a large-scale dataset of cooking recipes and food images[39]. The Food-475 dataset[40] collects four previously published food image datasets[27,30,36,37] into one. The Beijing Technology and Business University Food 60 (BTBUFood-60) is a dataset of images meant for food detection[41]. Recently, the ISIA Food-500 dataset[42] of miscellaneous food images was made available. In comparison to other publicly available food image datasets, it contains a large number of images, divided into 500 food classes, and is meant to advance the development of multimedia food recognition solutions[42].

Protocol

1. Food image recognition with NutriNet

1. **Obtaining the food image dataset**
    1. Gather a list of different foods and beverages that will be the outputs of the food image recognition model. A varied list of popular foods and beverages is preferred, as that will allow the training of a robust food image recognition model.
    2. Save the food and beverage list in a text file (e.g., 'txt' or 'csv').
       NOTE: The text file used by the authors of this article can be found in the supplemental files ('food_items.txt') and includes a list of 520 Slovenian food items.
    3. Write or download a Python[43] script that uses the Google Custom Search API[44] to download images of each food item from the list and saves them into a separate folder for each food item.
       NOTE: The Python script used by the authors of this article can be found in the supplemental files ('download_images.py'). If this script is used, the Developer Key (variable 'developerKey', line 8 in the Python script code) and Custom Search Engine ID (variable 'cx', line 28 in the Python script code) need to be replaced with values specific to the Google account being used.
    4. Run the Python script from step 1.1.3 (e.g., with the command: 'python download_images.py').
2. **(Optional) Cleaning the food image dataset**
    1. Train a food image detection model in the same way as in section 1.4, except use only two outputs (food, non-food) as opposed to the list of outputs from step 1.1.1.
       NOTE: The authors of this article used images combined from recipe websites and the ImageNet dataset[45] to train the food image detection model. Since the focus here is on food image recognition and this is an optional step for cleaning the recognition dataset, further details are omitted.

Instead, more details about this approach can be found in Mezgec et al.[2].

2. Run the detection model from step 1.2.1 on the food image dataset that is the result of step 1.1.4.

3. Delete every image that was tagged as non-food by the detection model from step 1.2.1.

4. Manually check the food image dataset for other erroneous or low-quality images, and for image duplicates.

5. Delete images found in step 1.2.4.

3. **Augmenting the food image dataset**

   1. Create a new version of each image from the food image dataset by rotating it by 90° using the CLoDSA library[46] (lines 19 to 21 in the included Python script).

      NOTE: The Python script containing all the CLoDSA commands used by the authors of this article can be found in a file included in the supplemental files ('nutrinet_augmentation.py'). If this script is used, the Input Path (variable 'INPUT_PATH', line 8 in the Python script code) and Output Path (variable 'OUTPUT_PATH', line 11 in the Python script code) need to be replaced with paths to the desired folders.

   2. Create a new version of each image from the food image dataset by rotating it by 180° using the CLoDSA library (lines 19 to 21 in the included Python script).

   3. Create a new version of each image from the food image dataset by rotating it by 270° using the CLoDSA library (lines 19 to 21 in the included Python script).

   4. Create a new version of each image from the food image dataset by flipping it horizontally using the CLoDSA library (lines 23 and 24 in the included Python script).

   5. Create a new version of each image from the food image dataset by adding random color noise to it using the CLoDSA library (lines 26 and 27 in the included Python script).

   6. Create a new version of each image from the food image dataset by zooming into it by 25% using the CLoDSA library (lines 29 and 30 in the included Python script).

   7. Save images from steps 1.3.1-1.3.6, along with the original images (lines 16 and 17 in the included Python script), into a new food image dataset (in total, 7 variants per food image). This is done by executing the command in line 32 of the included Python script.

4. **Performing food image recognition**

   1. Import the food image dataset from step 1.3.7 into the NVIDIA DIGITS environment[47], dividing the dataset into training, validation and testing subsets in the NVIDIA DIGITS user interface.

   2. Copy and paste the definition text of the NutriNet architecture[2] into NVIDIA DIGITS as a custom network.

      NOTE: The NutriNet architecture definition text can be found in the supplemental files ('nutrinet.prototxt').

   3. (Optional) Define training hyperparameters in the NVIDIA DIGITS user interface.

NOTE: Hyperparameters are parameters that are used to define the training process prior to its start. The hyperparameters used by the authors of this article can be found in a file included in the supplemental files ('nutrinet_hyperparameters.prototxt'). While experimentation is needed for each dataset to find the optimal hyperparameters, the file contains a hyperparameter configuration which can be copied into the NVIDIA DIGITS user interface. Furthermore, NVIDIA DIGITS populates the hyperparameters with default values which can be used as a baseline. This step is therefore optional.

4. Run the training of the NutriNet model.

5. After training is complete, take the best-performing NutriNet model iteration. This model is then used for testing the performance of this approach.

   NOTE: There are multiple ways to determine the best-performing model iteration. A straightforward way to do this is as follows. NVIDIA DIGITS outputs a graph of accuracy measures for each training epoch. Check which epoch achieved the lowest loss value for the validation subset of the food image dataset - that model iteration can be considered best-performing. An optional step in determining the best-performing model iteration is to observe how the loss value for the training subset changes from epoch to epoch and if it starts to drop continuously while the loss value for the validation subset remains the same or rises continuously, take the epoch prior to this drop in training loss value, as that can signal when the model started overfitting on the training images.

2. Food image segmentation with FCNs

1. Obtaining the fake-food image dataset

1. Obtain a dataset of fake-food images. Fake-food images are gathered by researchers conducting behavioral studies using food replicas.

   NOTE: The authors of this article received images of fake food that were collected in a lab environment[18].

2. Manually annotate every food image on a pixel level - each pixel in the image must contain information about which food class it belongs to. The result of this step is one annotation image for each image from the food image dataset, where each pixel represents one of the food classes.

   NOTE: There are many tools to achieve this - the authors of this article used JavaScript Segment Annotator[48].

2. Augmenting the fake-food image dataset

1. Perform the same steps as in section 1.3, but only on images from the training subset of the food image dataset.

   NOTE: With the exception of step 1.3.5, all data augmentation steps need to be performed on corresponding annotation images as well. If the script from section 1.3 is used, the Input Path (variable 'INPUT_PATH', line 8 in the Python[43] script code) and Output Path (variable 'OUTPUT_PATH', line 11 in the Python script code) need to be replaced with paths to the desired folders. In addition, set the Problem (variable 'PROBLEM', line 6 in the Python script code) to 'instance_segmentation' and the Annotation Mode (variable 'ANNOTATION_MODE', line 7 in the

Python script code) and Output Mode (variable 'OUTPUT_MODE', line 10 in the Python script code) to 'coco'.

3. **Performing fake-food image segmentation**

 1. Perform the same steps as in section 1.4, with the exception of step 1.4.2. In place of that step, perform steps 2.3.2 and 2.3.3.

 NOTE: Hyperparameters are parameters that are used to define the training process prior to its start. The training hyperparameters used by the authors of this article for the optional step 1.4.3 can be found in a file included in the supplemental files ('fcn-8s_hyperparameters.prototxt'). While experimentation is needed for each dataset to find the optimal set of hyperparameters, the file contains a hyperparameter configuration which can be copied into the NVIDIA DIGITS[47] user interface. Furthermore, NVIDIA DIGITS populates the hyperparameters with default values which can be used as a baseline.

 2. Copy and paste the definition text of the FCN-8s architecture[15] into the NVIDIA DIGITS environment as a custom network.

 NOTE: The FCN-8s architecture definition text is publicly available on GitHub[49].

 3. Enter the path to the pre-trained FCN-8s model weights into the NVIDIA DIGITS user interface.

 NOTE: These model weights were pre-trained on the PASCAL VOC dataset[17] and can be found on the Internet[49].

3. Food image segmentation with HTC ResNet

1. **Obtaining the food image dataset**

 1. Download the food image dataset from the FRC website[19].

2. **Augmenting the food image dataset**

 1. Perform steps 1.3.1-1.3.4.

 NOTE: The Python[43] script containing all the CLoDSA[46] commands used by the authors of this article can be found in a file included in the supplemental files ('frc_augmentation.py'). If this script is used, the Input Path (variable 'INPUT_PATH', line 8 in the Python script code) and Output Path (variable 'OUTPUT_PATH', line 11 in the Python script code) need to be replaced with paths to the desired folders.

 2. Create a new version of each image from the food image dataset by adding Gaussian blur to it using the CLoDSA library (lines 26 and 27 in the included Python script).

 3. Create a new version of each image from the food image dataset by sharpening it using the CLoDSA library (lines 29 and 30 in the included Python script).

 4. Create a new version of each image from the food image dataset by applying gamma correction to it using the CLoDSA library (lines 32 and 33 in the included Python script).

 5. Save images from steps 3.2.1-3.2.4, along with the original images (lines 16 and 17 in the included Python script), into a new food image dataset (in total, 8 variants per food image). This is done by executing the command in line 35 of the included Python script.

 6. Save images from steps 3.2.2-3.2.4, along with the original images (lines 16 and 17 in the included

Python script), into a new food image dataset (in total, 4 variants per food image). This is done by deleting lines 19 to 24 of the included Python script and executing the command in line 35.

3. **Performing food image segmentation**

    1. Modify the existing HTC[20] ResNet-101 architecture[16] definition from the MMDetection library[50] in sections 'model settings' and 'dataset settings' of the architecture definition file so that it accepts the food image datasets from steps 3.1.1, 3.2.5 and 3.2.6.

    2. (Optional) Modify the HTC ResNet-101 architecture definition from step 3.3.1 to define training hyperparameters: batch size in section 'dataset settings', solver type and learning rate in section 'optimizer', learning policy in section 'learning policy' and number of training epochs in section 'runtime settings' of the architecture definition file.
    NOTE: The modified HTC ResNet-101 architecture definition file can be found in the supplemental files ('htc_resnet-101.py'). Hyperparameters are parameters that are used to define the training process prior to its start. While experimentation is needed for each dataset to find the optimal set of hyperparameters, the file already contains a hyperparameter configuration which can be used without modification. This step is therefore optional.

    3. Run the training of the HTC ResNet-101 model on the food image dataset from step 3.1.1 using the MMDetection library (e.g., with the command: 'python mmdetection/tools/train.py htc_resnet-101.py').

    4. After the training from step 3.3.3 is complete, take the best-performing HTC ResNet-101 model iteration and fine-tune it by running the next phase of training on the food image dataset from step 3.2.5.
    NOTE: There are multiple ways to determine the best-performing model iteration. A straightforward way to do this is as follows. The MMDetection library outputs values of accuracy measures for each training epoch in the command line interface. Check which epoch achieved the lowest loss value for the validation subset of the food image dataset - that model iteration can be considered best-performing. An optional step in determining the best-performing model iteration is to observe how the loss value for the training subset changes from epoch to epoch and if it starts to drop continuously while the loss value for the validation subset remains the same or rises continuously, take the epoch prior to this drop in training loss value, as that can signal when the model started overfitting on the training images.

    5. After the training from step 3.3.4 is complete, take the best-performing HTC ResNet-101 model iteration and fine-tune it by running the next phase of training on the food image dataset from step 3.2.6.
    NOTE: See note for step 3.3.4.

    6. After the training from step 3.3.5 is complete, take the best-performing HTC ResNet-101 model iteration and fine-tune it by again running the next phase of training on the food image dataset from step 3.2.5.
    NOTE: See note for step 3.3.4.

    7. After the training from step 3.3.6 is complete, take the best-performing HTC ResNet-101 model

iteration. This model is then used for testing the performance of this approach.

NOTE: See note for step 3.3.4. Steps 3.3.3-3.3.7 yielded the best results for the purposes defined by the authors of this article. Experimentation is needed for each dataset to find the optimal sequence of training and data augmentation steps.

Representative Results

NutriNet was tested against three popular deep learning architectures of the time: AlexNet[14], GoogLeNet[51] and ResNet[16]. Multiple training parameters were also tested for all architectures to define the optimal values[2]. Among these is the choice of solver type, which determines how the loss function is minimized. This function is the primary quality measure for training neural networks as it is better suited for optimization during training than classification accuracy. We tested three solvers: Stochastic Gradient Descent (SGD)[52], Nesterov's Accelerated Gradient (NAG)[53] and the Adaptive Gradient algorithm (AdaGrad)[54]. The second parameter is batch size, which defines the number of images that are processed at the same time. The depth of the deep learning architecture determined the value of this parameter, as deeper architectures require more space in the GPU memory - the consequence of this approach was that the memory was completely filled with images for all architectures, regardless of depth. The third parameter is learning rate, which defines the speed with which the neural network parameters are being changed during training. This parameter was set in unison with the batch size, as the number of concurrently processed images dictates the convergence rate. AlexNet models were trained using a batch size of 256 images and a base learning rate of 0.02; NutriNet used a batch size of 128 images and a rate of 0.01; GoogLeNet 64 images and a rate of 0.005; and ResNet 16 images and a rate of 0.00125. Three other parameters were fixed for all architectures: learning rate policy (step-down), step size (30%) and gamma (0.1). These parameters jointly describe how the learning rate is changing in every epoch. The idea behind this approach is that the learning rate is being gradually lowered to fine-tune the model the closer it gets to the optimal loss value. Finally, the number of training epochs was also fixed to 150 for all deep learning architectures[2].

The best result among all the parameters tested that NutriNet achieved was a classification accuracy of 86.72% on the recognition dataset, which was around 2% higher than the best result for AlexNet and slightly higher than GoogLeNet's best result. The best-performing architecture overall was ResNet (by around 1%), however the training time for ResNet is substantially higher compared to NutriNet (by a factor of approximately five), which is important if models are continuously re-trained to improve accuracy and the number of recognizable food items. NutriNet, AlexNet and GoogLeNet achieved their best results using the AdaGrad solver, whereas ResNet's best model used the NAG solver. NutriNet was also tested on the publicly available UNIMIB2016 food image dataset[38]. This dataset contains 3,616 images of 73 different food items. NutriNet achieved a recognition accuracy of 86.39% on this dataset, slightly outperforming the baseline recognition result of the authors of the dataset, which was 85.80%. Additionally, NutriNet was tested on a small dataset of 200 real-world images of 115 different food and beverage items, where NutriNet achieved a top-5 accuracy of 55%.

To train the FCN-8s fake-food image segmentation model, we used Adam[55] as the solver type, as we found that it performed optimally for this task. The base learning rate was set very low - to 0.0001. The reason for the low number is the fact that only one image could be processed at a time, which is a consequence of the pixel-level classification process. The GPU memory requirements for this approach are significantly greater than image-level classification. The learning rate thus had to be set low so that the parameters were not being changed too fast and converge to less optimal values. The number of training epochs was set to 100, while the learning rate policy, step size and gamma were set to step-down, 34% and 0.1, respectively, as these parameters produced the most accurate models.

Accuracy measurements of the FCN-8s model were performed using the pixel accuracy measure[15], which is analogous to the classification accuracy of traditional deep learning networks, the main difference being that the accuracy is computed on the pixel level instead of on the image level:

$$PA = \frac{\sum_i n_{ii}}{\sum_i t_i}$$

where PA is the pixel accuracy measure, n_{ij} is the number of pixels from class i predicted to belong to class j and $t_i = \Sigma_j n_{ij}$ is the total number of pixels from class in the ground-truth labels[1]. In other words, the pixel accuracy measure is computed by dividing correctly predicted pixels by the total number of pixels. The final accuracy of the trained FCN-8s model was 92.18%. **Figure 2** shows three example images from the fake-food image dataset (one from each of the training, validation and testing subsets), along with the corresponding ground-truth and model prediction labels.

The parameters to train the HTC[20] ResNet-101 model for food image segmentation were set as follows: the solver type used was SGD because it outperformed other solver types. The base learning rate was set to 0.00125 and the batch size to 2 images. The number of training epochs was set to 40 per training phase, and multiple training phases were performed - first on the original FRC dataset without augmented images, then on the 8x-augmented and 4x-augmented FRC dataset multiple times in an alternating fashion, each time taking the best-performing model and fine-tuning it in the next training phase. More details on the training phases can be found in section 3.3 of the protocol text. Finally, the step-down learning policy was used, with fixed epochs for when the learning rate decreased (epochs 28 and 35 for the first training phase). An important thing to note is that while this sequence of training phases produced the best results in our testing in the scope of the FRC, using another dataset might require a different sequence to produce optimal results.

This ResNet-based solution for food image segmentation was evaluated using the following precision measure[19]:

$$P_{IoU\geq 0.5} = \frac{TP_{IoU\geq 0.5}}{TP_{IoU\geq 0.5} + FP_{IoU\geq 0.5}}$$

where *P* is precision, *TP* is the number of true positive predictions by the food image segmentation model, *FP* is the number of false positive predictions and *IoU* is Intersection over Union, which is computed with this equation:

$$IoU = \frac{Area\ of\ Overlap}{Area\ of\ Union}$$

where *Area of Overlap* represents the number of predictions by the model that overlap with the ground truth, and *Area of Union* represents the total number of predictions by the model together with the ground truth, both on a pixel level and for each individual food class. Recall is used as a secondary measure and is calculated in a similar way, using the following formula[19]:

$$R_{IoU\geq 0.5} = \frac{TP_{IoU\geq 0.5}}{TP_{IoU\geq 0.5} + FN_{IoU\geq 0.5}}$$

where *R* is recall and *FN* is the number of false negative predictions by the food image segmentation model. The precision and recall measures are then averaged across all classes in the ground truth. Using these measures, our model achieved an average precision of 59.2% and an average recall of 82.1%, which ranked second in the second round of the Food Recognition Challenge[19]. This result was 4.2% behind the first place and 5.3% ahead of the third place in terms of the average precision measure. **Table 1** contains the results for the top-4 participants in the competition.

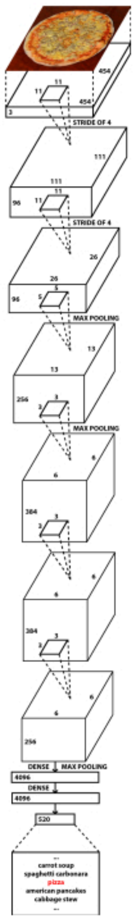

Figure 1: Diagram of the NutriNet deep neural network architecture. This figure has been published in Mezgec et al.[2].

Please click here to view a larger version of this figure.

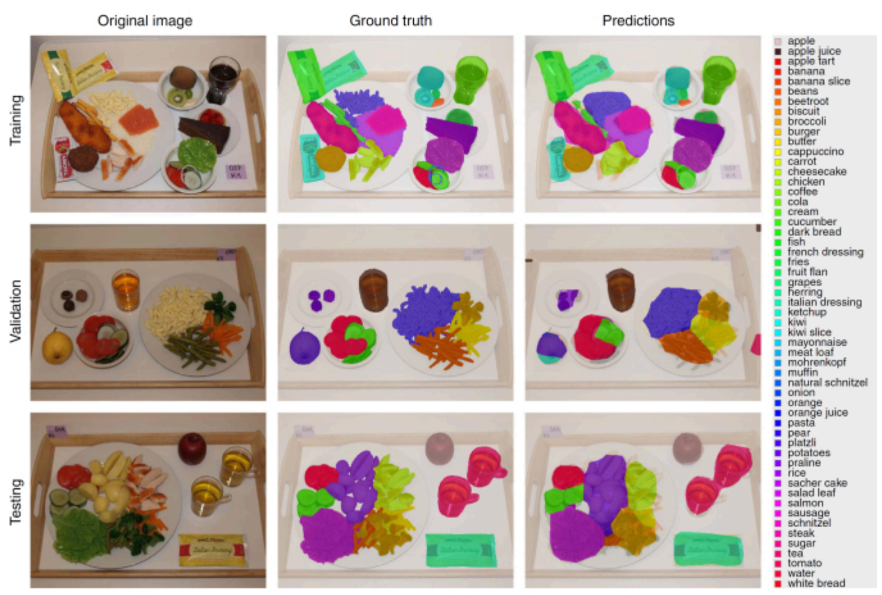

Figure 2: Images from the fake-food image dataset. Original images (left), manually-labelled ground-truth labels (middle) and predictions from the FCN-8s model (right). This figure has been published in Mezgec et al.[1]. Please click here to view a larger version of this figure.

Team Name	Placement	Average Precision	Average Recall
rssfete	1	63.4%	88.6%
simon_mezgec	2	59.2%	82.1%
arimboux	3	53.9%	73.5%
latentvec	4	48.7%	71.1%

Table 1: Top-4 results from the second round of the Food Recognition Challenge. Average precision is taken as the primary performance measure and average recall as a secondary measure. Results are taken from the official competition leaderboard[19].

Supplemental Files. Please click here to download this File.

Discussion

In recent years, deep neural networks have been validated multiple times as a suitable solution for recognizing food images[10,11,12,21,23,25,26,29,31,33]. Our work presented in this article serves to further prove this[1,2]. The single-output food image recognition approach is straightforward and can be used for simple applications where images with only one food or beverage item are expected[2].

The food image segmentation approach seems particularly suitable for recognizing food images in general, without any restriction on the number of food items[1]. Because it works by classifying each individual pixel of the image, it is able to not only recognize any number of food items in the image, but also specify where a food item is located, as well as how large it is. The latter can then be used to perform food weight estimation, particularly if used with either a reference object or a fixed-distance camera.

There has been some work done regarding the availability of food image datasets[3,22,27,30,36,37,38,39,40,41,42], and we hope more will be done in the future, particularly when it comes to aggregating food image datasets from different regions across the world, which would enable more robust solutions to be developed. Currently, the accuracy of automatic food image recognition solutions has not yet reached human-level accuracy[35], and this is likely in large part due to the usage of food image datasets of insufficient size and quality.

In the future, our goal will be to further evaluate the developed procedures on real-world images. In general, datasets in this field often contain images taken in controlled environments or images that were manually optimized for recognition. This is why it is important to gather a large and diverse real-world food image dataset to encompass all the different food and beverage items that individuals might want to recognize. The first step towards this was provided by the Food Recognition Challenge, which included a dataset of real-world food images[19], but further work needs to be done to validate this approach on food images from all around the world and in cooperation with dietitians.

Disclosures

The authors have nothing to disclose.

Acknowledgments

The authors would like to thank Tamara Bucher from the University of Newcastle, Australia, for providing the fake-food image dataset. This work was supported by the European Union's Horizon 2020 research and innovation programs (grant numbers 863059 - FNS-Cloud, 769661 - SAAM); and the Slovenian Research Agency (grant number P2-0098). The European Union and Slovenian Research Agency had no role in the design, analysis or writing of this article.

References

1. Mezgec, S., Eftimov, T., Bucher, T., Koroušić Seljak, B. Mixed Deep Learning and Natural Language Processing Method for Fake-Food Image Recognition and Standardization to Help Automated Dietary Assessment. *Public Health Nutrition.* **22** (7), 1193-1202 (2019).
2. Mezgec, S., Koroušić Seljak, B. NutriNet: A Deep Learning Food and Drink Image Recognition System for Dietary Assessment. *Nutrients.* **9** (7), 657 (2017).
3. Chen, M. et al. PFID: Pittsburgh Fast-Food Image Dataset. *Proceedings of the ICIP 2009.* 289-292 (2009).
4. Joutou, T., Yanai, K. A Food Image Recognition System with Multiple Kernel Learning. *Proceedings of the ICIP 2009.* 285-288 (2009).
5. Yang, S., Chen, M., Pomerlau, D., Sukthankar, R. Food Recognition using Statistics of Pairwise Local Features. *Proceedings of the CVPR 2010.* 2249-2256 (2010).

6. Anthimopoulos, M. M., Gianola, L., Scarnato, L., Diem, P., Mougiakakou, S. G. A Food Recognition System for Diabetic Patients Based on an Optimized Bag-of-Features Model. *IEEE Journal of Biomedical and Health Informatics.* **18** (4), 1261-1271 (2014).

7. LeCun, Y., Bengio, Y., Hinton, G. Deep Learning. *Nature.* **521**, 436-444 (2015).

8. Deng, L., Yu, D. Deep Learning: Methods and Applications. *Foundations and Trends in Signal Processing.* **7** (3-4), 197-387 (2014).

9. Hubel, D. H., Wiesel, T. N. Receptive Fields, Binocular Interaction and Functional Architecture in the Cat's Visual Cortex. *The Journal of Physiology.* **160** (1), 106-154 (1962).

10. Singla, A., Yuan, L., Ebrahimi, T. Food/Non-Food Image Classification and Food Categorization using Pre-Trained GoogLeNet Model. *Proceedings of the MADiMa'16.* 3-11 (2016).

11. Yanai, K., Kawano, Y. Food Image Recognition using Deep Convolutional Network with Pre-Training and Fine-Tuning. *Proceedings of the ICMEW 2015.* 1-6 (2015).

12. Liu, C. et al. DeepFood: Deep Learning-Based Food Image Recognition for Computer-Aided Dietary Assessment. *Proceedings of the ICOST 2016.* 37-48 (2016).

13. De Sousa Ribeiro, F. et al. An End-to-End Deep Neural Architecture for Optical Character Verification and Recognition in Retail Food Packaging. *Proceedings of the ICIP 2018.* 2376-2380 (2018).

14. Krizhevsky, A., Sutskever, I., Hinton, G. ImageNet Classification with Deep Convolutional Neural Networks. *Proceedings of the NIPS'12.* 1097-1105 (2012).

15. Long, J., Shelhamer, E., Darrell, T. Fully Convolutional Networks for Semantic Segmentation. *Proceedings of the CVPR 2015.* 3431-3440 (2015).

16. He, K., Zhang, X., Ren, S., Sun, J. Deep Residual Learning for Image Recognition. *Proceedings of the CVPR 2016.* 770-778 (2016).

17. PASCAL VOC Project. *PASCAL Visual Object Classes.* http://host.robots.ox.ac.uk/pascal/VOC (2020).

18. Bucher, T., van der Horst, K., Siegrist, M. Fruit for Dessert. How People Compose Healthier Meals. *Appetite.* **60** (1), 74-80 (2013).

19. AICrowd. *Food Recognition Challenge.* https://www.aicrowd.com/challenges/food-recognition-challenge (2020).

20. Chen, K. et al. Hybrid Task Cascade for Instance Segmentation. *Proceedings of the CVPR 2019.* 4974-4983 (2019).

21. Kawano, Y., Yanai, K. Food Image Recognition with Deep Convolutional Features. *Proceedings of the UbiComp 2014.* 589-593 (2014).

22. Matsuda, Y., Hoashi, H., Yanai, K. Recognition of Multiple-Food Images by Detecting Candidate Regions. *Proceedings of the ICME 2012.* 25-30 (2012).

23. Christodoulidis, S., Anthimopoulos, M. M., Mougiakakou, S. G. Food Recognition for Dietary Assessment using Deep Convolutional Neural Networks. *Proceedings of the ICIAP 2015.* 458-465 (2015).

24. Tanno, R., Okamoto, K., Yanai, K. DeepFoodCam: A DCNN-Based Real-Time Mobile Food Recognition System. *Proceedings of the MADiMa'16.* 89-89 (2016).

25. Liu, C. et al. A New Deep Learning-Based Food Recognition System for Dietary Assessment on An Edge

Computing Service Infrastructure. *IEEE Transactions on Services Computing.* **11** (2), 249-261 (2017).

26. Martinel, N., Foresti, G. L., Micheloni, C. Wide-Slice Residual Networks for Food Recognition. *Proceedings of the IEEE WACV 2018.* 567-576 (2018).
27. Bossard, L., Guillaumin, M., Van Gool, L. Food-101-Mining Discriminative Components with Random Forests. *Proceedings of the ECCV'14.* 446-461 (2014).
28. Zhou, L., Zhang, C., Liu, F., Qiu, Z., He, Y. Application of Deep Learning in Food: A Review. *Comprehensive Reviews in Food Science and Food Safety.* **18**, 1793-1811 (2019).
29. Zhao, H., Yap, K.-H., Kot, A. C., Duan, L. JDNet: A Joint-Learning Distilled Network for Mobile Visual Food Recognition. *IEEE Journal of Selected Topics in Signal Processing.* **14** (4), 665-675 (2020).
30. Kawano, Y., Yanai, K. Automatic Expansion of a Food Image Dataset Leveraging Existing Categories with Domain Adaptation. *Proceedings of the ECCV'14.* 3-17 (2014).
31. Hafiz, R., Haque, M. R., Rakshit, A., Uddin, M. S. Image-Based Soft Drink Type Classification and Dietary Assessment System using Deep Convolutional Neural Network with Transfer Learning. *Journal of King Saud University - Computer and Information Sciences.* In Press (2020).
32. Shimoda, W., Yanai, K. Weakly-Supervised Plate and Food Region Segmentation. *Proceedings of the ICME 2020.* 1-6 (2020).
33. Ciocca, G., Micali, G., Napoletano, P. State Recognition of Food Images using Deep Features. *IEEE Access.* **8**, 32003-32017 (2020).
34. Knez, S., Šajn, L. Food Object Recognition using a Mobile Device: Evaluation of Currently Implemented Systems. *Trends in Food Science & Technology.* **99**, 460-471 (2020).
35. Furtado, P., Caldeira, M., Martins, P. Human Visual System vs Convolution Neural Networks in Food Recognition Task: An Empirical Comparison. *Computer Vision and Image Understanding.* **191**, 102878 (2020).
36. Chen, M.-Y. et al. Automatic Chinese Food Identification and Quantity Estimation. *SA'12 Technical Briefs.* 1-4 (2012).
37. Chen, J., Ngo, C.-W. Deep-Based Ingredient Recognition for Cooking Recipe Retrieval. *Proceedings of the MM'16.* 32-41 (2016).
38. Ciocca, G., Napoletano, P., Schettini, R. Food Recognition: A New Dataset, Experiments, and Results. *IEEE Journal of Biomedical and Health Informatics.* **21** (3), 588-598 (2017).
39. Salvador, A. et al. Learning Cross-Modal Embeddings for Cooking Recipes and Food Images. *Proceedings of the IEEE CVPR 2017.* 3020-3028 (2017).
40. Ciocca, G., Napoletano, P., Schettini, R. CNN-Based Features for Retrieval and Classification of Food Images. *Computer Vision and Image Understanding.* **176 - 177**, 70-77 (2018).
41. Cai, Q., Li, J., Li, H., Weng, Y. BTBUFood-60: Dataset for Object Detection in Food Field. *Proceedings of the IEEE BigComp 2019.* 1-4 (2019).
42. Min, W. et al. ISIA Food-500: A Dataset for Large-Scale Food Recognition via Stacked Global-Local Attention Network. *Proceedings of the MM'20.* 393-401 (2020).

43. Python Software Foundation. *Python.* https://www.python.org (2020).

44. Google. *Google Custom Search API.* https://developers.google.com/resources/api-libraries/documentation/customsearch/v1/python/latest/customsearch_v1.cse.html (2020).

45. Stanford Vision Lab. *ImageNet.* http://www.image-net.org (2020).

46. Heras, J. *CLoDSA.* https://github.com/joheras/CLoDSA (2020).

47. NVIDIA. *NVIDIA DIGITS.* https://developer.nvidia.com/digits (2020).

48. Yamaguchi, K. *JavaScript Segment Annotator.* https://github.com/kyamagu/js-segment-annotator (2020).

49. Shelhamer, E. *Fully Convolutional Networks for Semantic Segmentation.* https://github.com/shelhamer/fcn.berkeleyvision.org (2020).

50. Multimedia Laboratory, CUHK. *MMDetection.* https://github.com/open-mmlab/mmdetection (2020).

51. Szegedy, C. et al. Going Deeper with Convolutions. *Proceedings of the CVPR 2015.* 1-9 (2015).

52. Bottou, L. Large-Scale Machine Learning with Stochastic Gradient Descent. *Proceedings of the COMPSTAT'2010.* 177-186 (2010).

53. Nesterov, Y. A Method of Solving a Convex Programming Problem with Convergence Rate O(1/k^2). *Doklady Akademii Nauk SSSR.* **27**, 372-376 (1983).

54. Duchi, J., Hazan, E., Singer, Y. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research.* **12**, 2121-2159 (2011).

55. Kingma, D. P., Ba, J. Adam: A Method for Stochastic Optimization. *arXiv Preprint.* arXiv:1412.6980 (2017).