CHALMERS
UNIVERSITY OF TECHNOLOGY

Photo OCR for Nutrition Labels

Combining Machine Learning and General Image Processing for Text Detection of American Nutrition Labels

Master's Thesis in Complex Adaptive Systems

JULIA REIBRING

Department of Signals and Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2017

Master's thesis

Photo OCR for Nutrition Labels

Combining machine learning and general image processing for text detection of American nutrition labels

JULIA REIBRING

Department of Signals and Systems
Chalmers University of Technology
Gothenburg, Sweden 2017

Photo OCR for Nutrition Labels
Combining machine learning and general image processing for text detection of American nutrition labels
JULIA REIBRING

Supervisor: Lars Yencken, Lifesum
Examiner: Fredrik Kahl, Department of Signals and Systems

Master's Thesis
Department of Signals and Systems
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: A typical American Nutrition Label on a box of Cherrios.

Typeset in LaTeX
Gothenburg, Sweden 2017

Photo OCR for Nutrition Labels
Combining Machine Learning and General Image Processing for Text Detection of American Nutrition Labels
Department of Signals and Systems
Chalmers University of Technology

Abstract

Food journaling relies on having access to high quality nutritional data. As the number of foods are growing and changing rapidly, there is a need for an updated database of reliable food data. Journaling apps typically crowd-source this information, but manual data entry is slow and has limited accuracy.

This thesis investigates how a photo OCR algorithm can be adapted to nutrition labels for the purpose of collecting reliable nutritional data. It examines how we can exploit the regular structure of nutrition labels in order to streamline the OCR process. For this purpose, we have limited ourselves to American nutrition labels, which have regular structure due to regulations.

The algorithm combines state-of-the-art machine learning for image classification with general image processing for segmentation, in order to extract accurate nutrition information from a photo of a nutrition label. In contrast to general text, nutrition labels use a limited vocabulary. For this reason a machine learning algorithm was trained to recognize words instead of characters, unlike classical OCR algorithms. However, words which were classified as numeric were segmented once more to be able to classify each character individually. Two convolutional networks were used for word and character classification, these were obtained by fine-tuning the already existing image network VGG16.

To segment the image into words and characters, simplifying assumptions which exploit the structure and layout of nutrition labels were used. By trying to find lines between rows, curvatures and oblique angles could be found within the image, which in turn revealed the alignment of words.

Two small datasets of words and characters were manually collected and labeled. This training data was then augmented by creating copies with more and less brightness than the originals. Networks trained on this augmented data had a validation accuracy of around 80% for both words and characters. Investigation was also done around exploiting sequence information for re-classifying words. This increased the accuracy slightly, at the cost of added complexity to the algorithm.

The thesis shows that if only a limited set of words are to be found within an image, one may train a machine learning algorithm to classify words instead of characters. It also shows that if assumptions can be made about how the words are structured, this knowledge can be used for more accurate segmentation.

Keywords: Photo OCR, Machine Learning, Convolutional Neural Networks, Text Detection, Nutrition Labels

Acknowledgements

Thanks to the Data team at Lifesum and my Chalmers supervisor Fredrik Kahl.

Julia Reibring, Stockholm, May 2017

Contents

List of Figures

1

Introduction

1.1 Background

Lifesum is a mobile health app which allows its users to enter the foods and meals they have eaten in order to keep track of their calorie intake. In addition to this, the app also gives guidance on how to make healthier food choices overall, by motivational feedback and inspiration. One of the features Lifesum offers is the food rating system, which rates foods from A to E. This rating is based on the nutritional composition of the food, which means that it relies heavily on having access to nutritional data.

Lifesum has a database of millions of foods from all over the world. The database includes calories, nutritional information, serving sizes, popularity, etc. The database is to some extent based on national databases (such as Livsmedelsverket's in Sweden), but is mainly crowd-sourced by Lifesum users. However, the quality of the crowd-sourced data tend to be of bad quality, as it is difficult to check its validity.

One way to increase the quality of the data would be to have a feature in the app which lets the user to take a photo of a nutrition label. The image would then be used as an input to an algorithm which extracts the nutritional information. This would be beneficial for two reasons: (1) It would increase the quality of the data and (2) it would allow users to add more data in less time, which probably means that more nutritional data would be added.

As computers' computational power rapidly has increased, machine learning, and

Figure 1.1: Typical examples of American nutrition labels. As can be seen in the images, they all share the same structure and layout.

deep learning in particular, has experienced significant development in recent years. Deep learning involves artificial neural networks, which are built up by a net of neurons and mimics how the human brain function. A certain kind of neural network is convolutional neural network, which is inspired by the visual cortex of animals and therefore works particularly well for image recognition.

Convolutional neural networks is now state-of-the-art for image recognition and has proven to work very well for detection and classification of objects in images. It should therefore be advantageous to use for scanning photos of nutrition labels.

1.2 Purpose

This thesis explores how convolutional neural networks can be used for scanning the text content of American nutrition labels. Due to regulations, American nutrition labels are very standardized and usually share a common content, structure and layout, so effort is put into taking advantage of this additional knowledge, instead of just relying on general word and character detection/recognition.

The objective is to design an algorithm which takes an image of a nutrition label as an input and outputs a list of calories and nutrients with the corresponding amounts.

This problem is more constrained than general text detection and therefore require it's own custom dataset to train on. However, manual data collection and labeling can be a tedious task and therefore this thesis also investigate how the performance can be increased by data augmentation.

In the long run this could serve as a basis for a new feature within the Lifesum app, a feature which would allow users to enter nutritional data by merely taking a photo of the nutrition label.

In the bigger picture this would motivate people to eat healthier as it will get easier to keep track of all kinds of nutrient intakes. The gained knowledge about foods' nutritional composition would also switch focus from just counting calories to eating better overall.

1.3 Delimitations

Only nutrition labels from the US are considered. Nutrition labels in the US are standardized (common structure, content, layout, font and color scheme) and should therefore be easier to achieve higher accuracy on, in contrary to nutrition labels in other countries where content and layout may vary a lot more.

1.4 New Contributions

General photo OCR has been researched thoroughly and many improvements have been achieved in recent years. However, industry problems are usually constrained and should therefore benefit from customization. This is where this thesis contribute; it shows how a practical problem can be solved using state-of-the-art machine learning, but also leverage on the additional knowledge there is about the defined problem.

More specifically the thesis shows that if we know which words are expected to be found within the image, if we know how the words are structured and if we know in what sequences the word are expected to appear we may use such information to improve the output result. Customization like this could extend to similar problem such as scanning tickets, recipes and receipts. These problems would probably benefit from similar customization.

1.5 Outline of Thesis

In the next chapter, the theory which the thesis relies on is presented. The main focus is put on convolutional neural networks (the reader is assumed to have basic knowledge about artificial neural networks,) as it is a vital building block of the algorithm. The chapter also include an introduction to photo OCR as well as image processing techniques which were used in the algorithm.

Chapter 3 describes the method and is split up in to three different sections. The first one covers how data was collected, labeled and augmented, the second one how two convolutional neural networks were trained on words and characters respectively, and the last one describes all steps in the final algorithm.

The output results of the final algorithm are found in chapter 4. The accuracy of how the CNNs classify words and characters are shown, together with sample images and their respective top 3 classifications. Results which shows how some words and characters are more common to be interchanged with other words and characters are also presented. However, the accuracy of the final algorithm is fairly hard to analyze, as it highly depends on the resolution of the input image, the composition and so on. For that reason, a few sample outputs of the algorithm are shown to get an idea of how well it performs.

The effectiveness of the algorithm and how it can be improved is discussed in chapter 5, as well as how the algorithm compares to other more general photo OCR algorithms.

At last, conclusions are made and summarized in the final chapter.

2

Theory

This chapter introduces the theories which are relevant for the developed algorithm.

2.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a powerful machine learning tool. They share their basic structure with general artificial neural network (ANNs), with connected neurons distributed in different layers. As in ANNs, an input signal is transferred through the layers of neurons and generates an output signal.

Though in contrast to regular ANNs with fully connected layers (where every node in each layer is connected to every node in the previous and following layer), neurons in CNNs are only connected to a limited number of neurons in the neighbouring layers. This result in not just less connections, but also allows for more layers, i.e. deeper networks and deep learning. The architecture makes CNNs particularly feasible for computer vision, as images are expected to interfere more with neighbouring pixels than pixels further away in the image.

Convolutional Layer

CNNs consist of convolutional layers. These layers take a 3 dimensional volume of neurons as an input (just as an image with width, height and three color channels, which is the input to the first convolutional layer) and generally outputs another 3 dimensional volume. It does so by ”sliding” a smaller 3 dimensional filter over the input volume. Mathematically this means that the dot product between each sub-area in the image and the weights in the filter is computed, i.e. a convolution. This results in a 2 dimensional map of new values, as illustrated in figure 2.1.

More than one filter could be used on the same input volume, which means that an additional map of values is produced. These are referred to as different feature maps, as they may reveal different features in the image, as a corner, an edge or in deeper networks an eye or a nose. Feature maps which are generated from the same layer serves as an input to a new layer.

The filters share the same depth as the input volume, but usually have a width and height of 3x3 or slightly larger. The filter may be used on every sub-part of the image, but could also be moved more that 1 pixel at a time. This parameter is called the stride and is what defines the distance between each dot product. Because the filters usually are larger than 1x1, a padding parameter adds additional ”padding

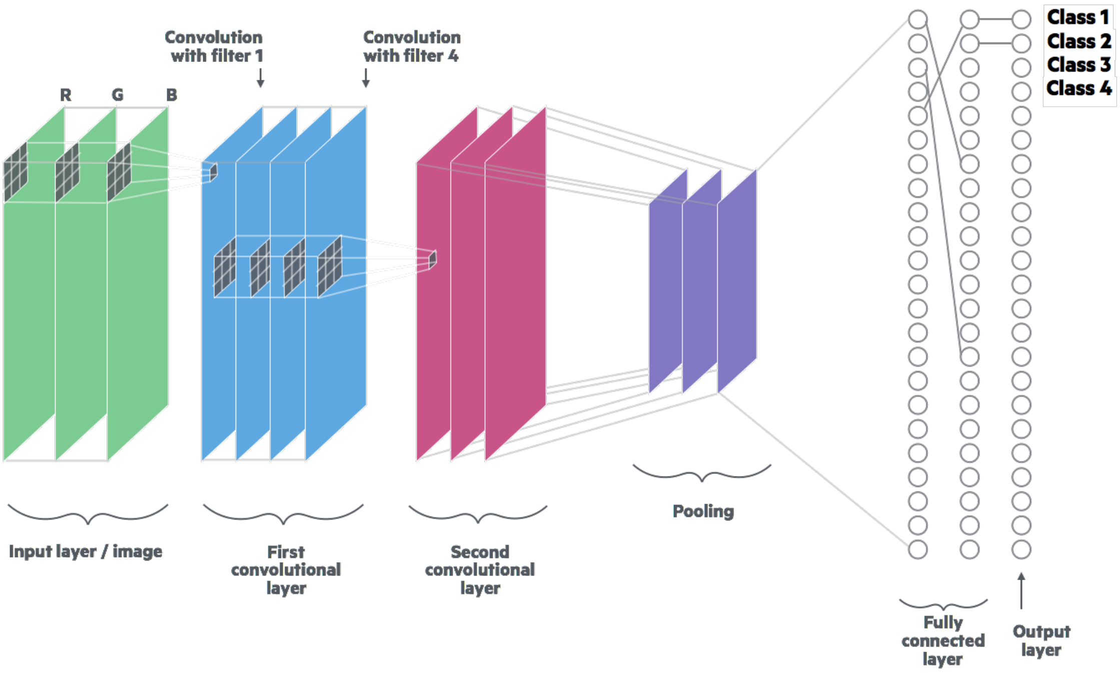


Figure 2.1: Main components of a convolutional neural network. The input is an image with a width, height and depth of 3, as the three color channels RGB. The dot product between each sub-part of the image and a filter creates a feature map in the second layer, three other filters creates three additional feature maps in the same layer. These then serves as an input to the third layer in the same manner. Pooling is illustrated as down-sizing the image.

neurons" around the image so that the number of neurons in the output layer shares the same width and height as the input layer.

If the input volume has a width w_1, a height h_1 and a depth d_1 and the number of filters is n, the filter size is f, the depth of the padding around the input volume is p and the stride is s, then the dimensions of the output volume is given by

$$w_2 = (w_1 - f + 2p)/s + 1 \tag{2.1}$$
$$h_2 = (h_1 - f + 2p)/s + 1 \tag{2.2}$$
$$d_2 = n. \tag{2.3}$$

It is important to note that the convolutions are translational invariant, meaning that the same filter is used over the whole input volume. This makes the number of parameters of the network much smaller compared to ANNs, where each connection may have its on weight.

Pooling

Another concept of CNNs which makes the network lighter are pooling and down-sampling. It might be reasonable to decrease the resolution of the network because of the large number of neurons, this can be achieved by pooling. Most often it is accomplished by taking the maximum of neighbouring neurons, using the ReLU activation function, which is given by

$$f(x) = \max(0, x),$$

but could also be done by taking the average of the input signals or using the sigmoid function. For practical and efficient use though, the ReLU-function has proven to work well [1].

Fully Connected Layer

When a number of convolutional layers and pooling layers have been connected, a network with progress from one layer to another has been generated. In the deepest layers the number of neurons are likely to be far less than in the shallow layers, which allows for fully connected layers at the end. The fully-connected layer outputs a vector with class scores of each class the network is trained to classify.

2.1.1 Vgg16

The Vgg16 image net is a deep convolutional network for image classification architectured and trained by Visual Geometry Group (VGG) at Oxford University [5].

What their research showed was that the depth of the network is a critical component for good performance. But to reduce the number of parameters in such very deep networks, it only performs 3x3 convolutions and 2x2 pooling from throughout the whole network.

For practical use the Vgg16 net can easily be finetuned for more specific computer vision tasks by fixing the weights in the first layers and allow the weights in the deeper layers to get updated when trained on new data. This is called transfer learning and is based on the fact that more general features are revealed by the network in more shallow layers, whereas more specific features appear in the deeper layers, meaning that only the last layers have to be updated.

2.1.2 Precision and Recall

For evaluation of a classification algorithm, one may use the two measures *precision* and *recall*. For a specific class of images, precision is the proportion of true positives of all images that were classified as that class, whereas recall is the proportion of all true positives among all the images that should have been classified as that class. Precision and recall are given by

$$precision = \frac{tp}{tp + fp}$$

and

$$recall = \frac{tp}{tp + fn},$$

where tp is the number of true positives, fp is the number of false positives and fn is the number of false negatives. If we have multiple classes, the formulas become

$$precision_i = \frac{M_{ii}}{\sum_j M_{ji}}$$

$$recall_i = \frac{M_{ii}}{\sum_j M_{ij}},$$

where M is the confusion matrix of all classes. The columns of the confusion matrix represents the number of times a class was predicted, while each row represents the actual number of instances there are in each class. If an overall value of precision and recall is desired, one may take the averages

$$recall = \frac{\sum_i recall_i}{n}$$

$$precision = \frac{\sum_i recall_i}{n}.$$

A measure for combining precision and recall is the harmonic mean as it combines the two values,

$$F = 2 \cdot \frac{precision \cdot recall}{precision + recall}.$$

2.2 Photo OCR

Optical character recognition (OCR) is used to be able to read handwritten or typed characters in images, which usually are generated from scanning or photos.

The classic approach of OCR is to first segment the image into characters and then read the text character by character. This is relatively simple when the image is a 2D image (as from scanning), but gets harder when the image is a photo, because of skewed angles, bad resolution, strange backgrounds, illumination etc.

OCR is generally done in two main steps, detection followed by classification. In classical OCR a binarized version (only black and white pixels) of the image is usually created for the purpose of segmentation. The image may then be straighten so that each row in the image can be found by analyzing how the number of black and white pixels vary along the vertical dimension. After the image has been split into rows, each row may then be segmented to words and characters by analyzing once again how the number of black and white pixels vary, but now in the horizontal direction.

In photos however, the text may not be very structured and analyzing a binary version of the image might not be very effective. For detecting text one may instead use a *detection by classification* approach. This means that machine learning is used to define if subparts of the image either include or do not include text.

This approach could be combined with an ”Edge Box technique” which tries to identify words by counting the number of edges there are within a box and the number of edges which are crossing the border of the box. A lot of edges within in the box and a few crossing the border would indicate that there is an object (word) contained in the box. [3].

2.3 Image Processing

Image processing is processing of images using mathematical operations.

Color images are built up by three values in each pixels: red, green and blue (RGB). The intensity ranges from 0 to 255, where 0 is black and 256 is white. Simple image processing include transforming a color image to gray-scale and black and white (binary). Grayscale images have just one value from 0 to 255 whereas pixels in binary images either are 0 or 255 (or 0 or 1).

2.3.1 Otsu's Method

Otsu's method is used to find the best threshold when transforming a grayscale image to a binary image (only black and white pixels). This is done by identifying the threshold (between 0 to 255) which minimizes the variance between the pixels' intensity within the same class (black or white). Mathematically we want to minimize

$$\sigma_w^2(t) = \omega_0(t)\sigma_0^2(t) + \omega_1(t)\sigma_1^2(t),$$

where t is the threshold, σ are the variance within the classes and ω is the portion of pixels belonging to each class respectively.

2.3.2 Adaptive Thresholding

In Otsu's method, a global value is used as a threshold value. But if for example the lightning conditions are different in different parts of the image, we may use adaptive thresholding instead, which calculates different thresholds for different areas. This method calculates the threshold differently for each pixel depending on the values of neighbouring pixels. Depending on the image size, we may use larger or smaller values of the neighbouring area to take into account.

2.4 Related Work

In recent years, convolutional neural networks has gained traction as it has become state-of-the-art for image recognition. For the purpose of localizing and reading text within photos, a fair amount of research has been done as well.

Convolutional neural networks has proven to work well for reading text in real-world images [4]. But CNNs can be used for detection as well [3]. This can be achieved by combining machine learning classification with the Edge Box technique (mentioned in section 2.2) which suggest bounding boxes which surrounds text. However, this result in a large number of false positives, which is dealt with using a CNN which is trained on not just classifying words, but also recognizes images that do not include any words. Research has also shown that instead of classifying individual characters, one may train a CNN to recognize words in a limited dictionary [3].

For a data collection purpose it has been shown that synthetic data for creating a large dataset to train a neural network on has proved to be a very successful strategy [2]. The research shows that all data may be synthetic and no manual work for labeling is needed at all.

3

Method

The main purpose was to use convolutional neural networks to extract information from images of nutrition labels printed on foods, as well as making use of the structure, content and layout American nutrition labels usually have.

The starting point was to train a CNN on images of words that are common in nutrition labels, which basically means treating the words as objects instead of *words made up by characters* (as in regular text detection). This is because the words we expect to find in nutrition labels builds up a very limited dictionary.

Nutrition labels don't just include words as for example "calories", "protein" and "sodium", they also include a lot of numbers. For that reason, the CNN was also trained on classifying images which contain some kind of amount (either just numbers, percentages, or amounts of grams or milligrams). This is because images which the CNN classifies as one of these four classes will be segmented once more. When the amounts have been segmented into individual characters, the exact amounts can be derived by using another CNN trained on identifying characters.

In addition to this, general image processing was used to detect the actual nutrition label as well as for word and character segmentation.

To increase the accuracy of the algorithm, two additional approaches were taken. The first one was using augmented data for training of the CNNs, so that the dataset gets larger. The second one was trying to do better classification based on what word sequences we expect to find in nutrition labels.

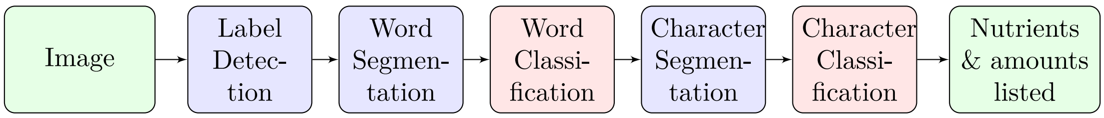


Figure 3.1: A schematic overview of the end-to-end pipeline of the photo OCR algorithm. The label detection, word segmentation and character segmentation all involves image processing, whereas word classification and character classification is done by using two CNNs.

A very general pipeline of the algorithm is presented in figure 3.1. Step by step, the algorithm can be described with the following bullet points:

- The algorithm takes an image as an input
- The nutrition label is detected within the image (but if this is not successful, the user will get to crop the image)
- The image is rotated to an upright position if needed

- The image is split into sub images with one word in each image
- The sub-images are classified as either a word or some kind of amount
- The sequences of words are analyzed and some words might be re-classified
- If a sub-image is classified as some kind of amount, it is split up in individual characters
- The characters are classified
- Output of nutrients and amounts

In section 3.3 the different steps of the algorithm are explained more in depth, but first the methodology for data collection, labeling and training the networks will be covered.

3.1 Data Collection and Annotation

One challenge of using supervised machine learning is the required amount of annotated data, which could be very time consuming to gather. For this project 81 photos of nutrition labels were taken at an American food store, using a standard mobile phone camera. The images were then segmented (using the same segmentation algorithm as described in section 3.3). All segmented sub-images were then manually evaluated; A successful segmentation of a word was defined as an image which included all characters from one word, including characters which might have been cut off a little but no more than half.

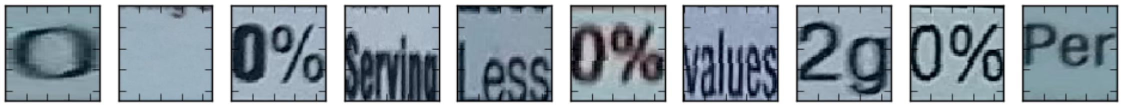
The segmented words were then labeled and resulted in around 2200 unique images of words. A simple labeling program was written in Python for this purpose. A relatively large portion of the images were not really a word, but included some kind of amount, these were labeled as either x, g, mg or *percent*, depending on which kind of characters were included. Only words that appeared at least 27 times were kept, as the accuracy is not expected to be very high if the dataset of a class is too small. At the same time, classes which are very well-represented could also cause problems (see section 2.1.2). Therefore, no class was allowed to consist of more than 99 images. This resulted in around 1800 images distributed over 37 different classes.

The words which were labeled as numeric (labeled x, g, mg or *percent*) were segmented a second time into individual characters, using again the method described in section 3.3. The images of characters were then labeled as a digit between 0 and 9, %, m, g, or a single dot. This resulted in around 3000 labeled characters.

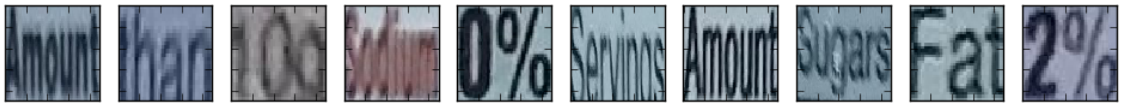
Not only images of words are trained on. The segmentation algorithm outputs some false positves, usually sup parts of vertical and horizontal lines or sub parts of a row which does not include any word at all but some distortion instead. The network is trained on this as well.

3.1.1 Data Augmentation

Collecting and labeling data is a time consuming task. Fortunately, data augmentation can leverage on a smaller dataset and create a larger one. For a dataset of images, this would typically include rotating or flipping the images.



(a) 10 samples from a word dataset with regular cropping. The classes of each image from left to right: *number, empty, percentage, Serving, less, percent, values, gram, percentage, Per.*



(b) 10 samples from a word dataset with tight cropping. The classes of each image from left to right: *Amount, than, gram, Sodium, percent, Servings, Amount, Sugars, Fat, percent*

(c) 10 samples from a word dataset with tight cropping but also versions of the images with less and more intensity so that the dataset becomes three times as big. The classes of each image from left to right: *Fat, percent, Calories, Sugars, vline, percent, Trans, gram, Less, Carbohydrate*

Figure 3.2: Samples from three different datasets of words.

However, for the purpose of getting a richer dataset of images of words, rotating and flipping doesn't seem feasible, as the words are expected to be in upright position and not reflected. Instead, the images were augmented by changing the intensity by either half or doubled the amount, which yielded three times as many images. Furthermore, the images were cropped once more, so that the images did not include any extra pixels above or below it (this is referred to as tight cropping in the thesis).

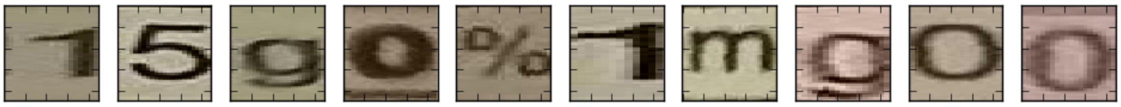
The data augmentation resulted in six different datasets, three with words and three with digits. Samples from the datasets are presented in figure 3.2 and 3.3.

3.2 Training of Neural Networks

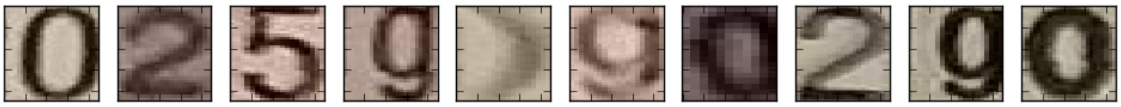
The common procedure for photo OCR is to do character segmentation followed by character recognition, but because there is a very limited number of words used in nutrition labels, it could be advantageous treat the words as objects instead and train the network on classifying a limited dictionary of words.

The approach was to use the vgg16 network which is already trained on visual recognition, but then finetune it for identifying the words which are used in nutrition labels. This is called transfer learning.

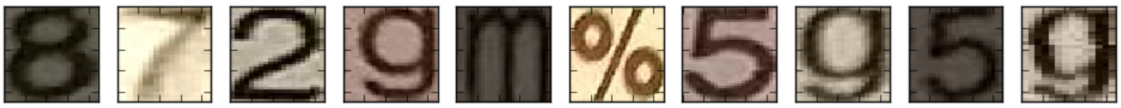
The network was trained on the three different versions of the word dataset



(a) 10 samples from a character dataset with regular cropping. The classes of each image from left to right: *1, 5, g, 0, %, 1, m, g, 0, 0*



(b) 10 samples from a character dataset with tight cropping. The classes of each image from left to right: *0, 2, 5, g,), g, 0, 2, g, 0*



(c) 10 samples from a character dataset with tight cropping. The classes of each image from left to right: *8, 7, 2, g, m, %, 5, g, 5, g*

Figure 3.3: Sample of three different datasets of characters.

presented in figure 3.2. The first dataset consisted of images which were generated from the initial segmentation, the second one consisted of images with tight cropping and the last dataset included the tight cropped images as well as the exact same images but with less and more intensity.

The CNNs were trained until the validation accuracy of the test set, which constituted 20 % of the images, didn't increase. Note that in the dataset which included the augmented images, the augmented versions were not included in the validation set and no augmented versions of the images which were part of the validation set were also included in the training set.

The resulting weights of the network were saved to be used in the final algorithm for word classification.

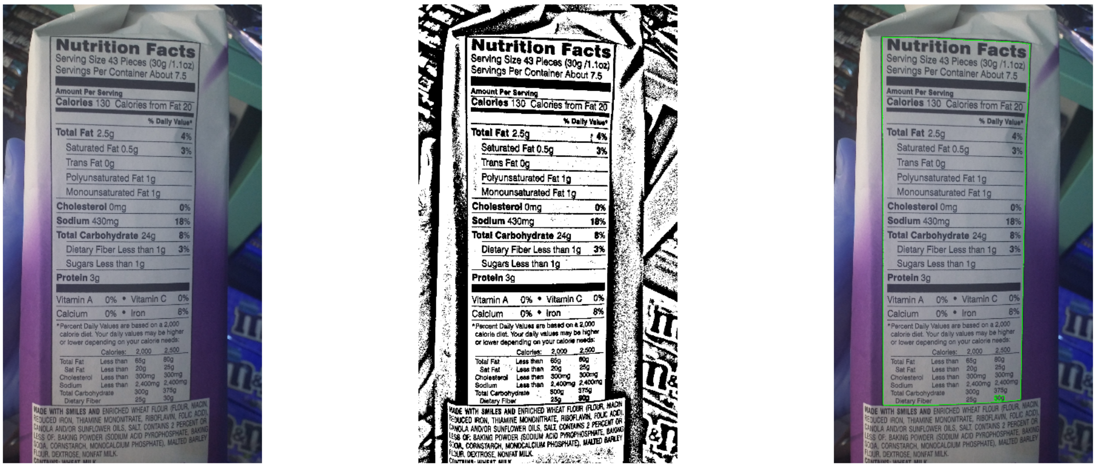
The exact same procedure was used for training another CNN on classifying characters and the weights were saved to be used in the final algorithm.

3.3 Algorithm Pipeline

The two trained networks described in section 3.2 were used for classification of the words and characters in the nutrition label. But to build up an algorithm which takes an image of a nutrition label as an input and output a list of the nutrients and the corresponding amounts more steps are necessary.

The idea is that the algorithm first crops the image so that the nutrition label is the only thing contained in the image. Then it splits the image horizontally from top to bottom, so that each sub image correspond to one row of text. After that each row is split up word by word, where-after each word is classified using the CNN trained on words. The words that are classified as some kind of amount are then segmented into characters, after which those are also classified using the CNN trained on characters. In a last step all pieces are evaluated and the algorithm outputs a list of nutrients and its corresponding amounts.

The steps builds up an algorithm pipeline and they are explained more in depth below.



(a) Original image

(b) Binary image

(c) Nutrition label has been detected.

Figure 3.4: The figure shows the original color image and the binary version of the same image, which was used for detecting the actual nutrition label.

3.3.1 Detect Nutrition Label

In practice, the user could be asked to take a photo of the nutrition label and then crop the image herself so that only the nutrition label is contained within the image. However, it would be even nicer if the this procedure is fully automated. Therefore, the algorithm tries to crop the image by analyzing the composition of the shapes within the image.

First, the image is binarized using adaptive thresholding. Shapes are then defined as connected black areas. Each shape has a bounding box which surrounds it, which gives each shape a width and a height.

The goal is to figure out which shape is the rectangle that surrounds the nutrition content. The key here is to look at the hierarchy of the shapes. The hierarchy describes which shapes are contained within other shapes and the opposite. What we expect is that a larger shape will surround smaller shapes which are all relatively wide but not very high, i.e. horizontal "line-like shapes". If we find a larger shape which contains these other line-like shapes, it is likely to be the shape which surrounds the nutrition label.

(a) Step 1, cropping

(b) Step 2, find line-like shapes

(c) Step 3, fit linear function to lines

(d) Step 4, find additional rows to split between

When the most likely shape is found, the smallest bounding box which can surround the shape is found using an opencv function. If the bounding box is tilted, it is rotated so that the nutrition labels is in an upright position. In the last step the image is cropped using the bounding box and the cropped image is then suggested to the user, who may approve it or decide to crop it herself.

3.3.2 Detect Horizontal Lines

American nutrition labels usually have horizontal lines which splits the main rows. Depending on the image quality, these lines were hopefully detected in the previous step as distinct shapes. But even if they were detected, they might not be perfect lines, because the binarization of the image might not have worked perfectly and shapes might have blended together, causing unwanted blobs. See figure 3.6 for an example.

An algorithm was written to ”clean” the lines from unwanted blobs. Each shape in the image has pixels in a two-dimensional grid, which means that the edges of a shape are either vertical or horizontal. By analyzing how the direction of the edges changes a long the contour, pieces of the contour which changed rapidly in a vertical direction or pixels which changed the direction from going right to left or left to right, were removed.

When the lines had been identified, a linear function was fit to the pixels (which all have a x and a y value). Using these lines, the image can then be split horizontally from top to bottom, generating multiple sub-images. See figure 3.7.

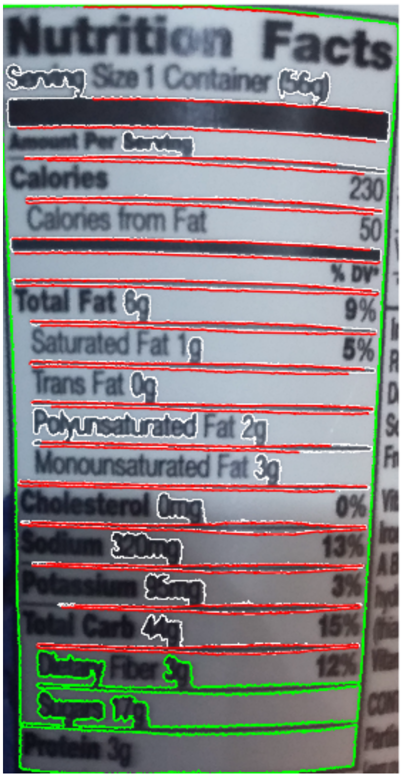


Figure 3.6: Example of when the line-detection has somewhat failed. The white contours shows how unwanted blobs are wrongly blended to the line. After the "cleaning line from blob-algorithm had run", the read lines were outputed.

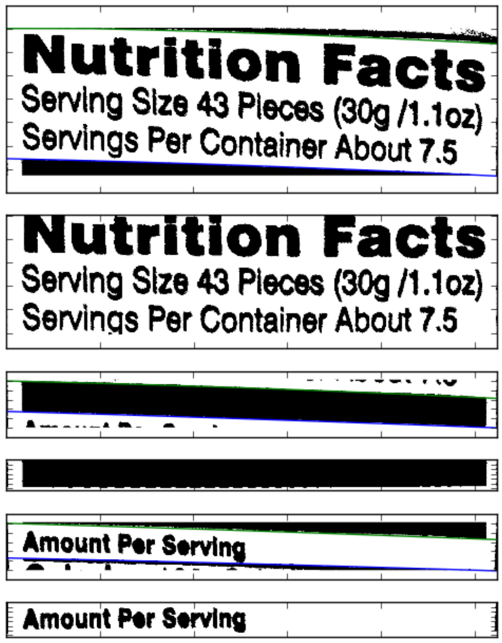


Figure 3.7: Each row is straighten out using the lines that were found in the image. This allows for easier for finding places where additional splits could be done.

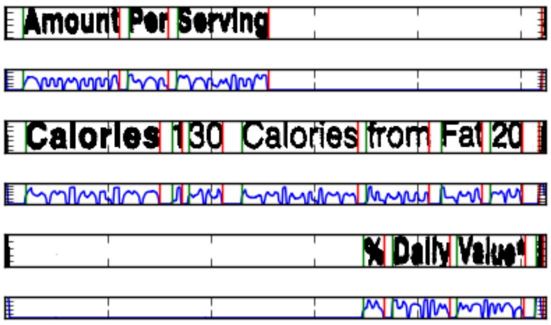


Figure 3.8: Word segmentation is done by analyzing how the intensity varies along the row.

3.3.3 Split by Row

All rows in nutrition labels are usually not separated by a line and sometimes the algorithm fails to find the algorithm. Therefore, the algorithm investigate where it is likely that there are additional splits between rows.

In the previous step, the algorithm hopefully found at least a few lines in the image and split the image into subimages. The lines are defined as some second degree linear function. These lines are used to straighten the row images by pushing all pixels so that the top pixels are all aligned. (See figure, which is not included yet.)

Using a binarized version of the image, the number of black pixels along each row of pixels is counted, which generates an intensity histogram. In addition to this, along each row of pixels, the number of times consecutive pixels turn from black to white or white to black is counted as well. These two values are then used to decide whether it is likely that the image could be split up even more vertically, as we expect the number of black pixels to be low and the number of times the pixels switch from black to white or the opposite low as well.

3.3.4 Split by Word

When the image has been split up in rows, each row is analyzed individually. Once more an intensity histogram of number of black pixels is created, but this time by checking how the intensity varies in the horizontal direction. Also, the number of times black pixels are followed by white and the opposite is counted.

To be able to split the row into words, the expected width of a gap between words is calculated using the pixel height of the row and divide it by 6. If the number of black pixels is low for more than this minimum gap value, the algorithm suggest that this is where a word begins/ends.

By splitting up the image first by row and then by word generates x and y values in the original image which, if the algorithm has been successful, correspond to the bounding boxes of the words. The content of each bounding box is saved as an image, ready to be classified.

3.3.5 Classify Words

The CNN trained on words is used to classify the words. The neural network outputs probabilities of each class. The top three classes with the highest probabilities are saved.

3.3.6 Possible Re-classification

Some words are more likely than others to appear in sequences. For example, if the algorithm classify three consecutive images as ”Amount”, ”Per”, ”Sodium”, it is likely that the last word is actually instead "Serving", especially if the output probabilities for ”Amount” and ”Per” are high but the one for ”Sodium” is low, and even more likely if the second most likely classification of the third word is actually ”Serving”. For this reason, it might be a good idea to add an additional piece of code to the algorithm which checks for these common sequences.

If a probability of a classification was higher than 0.90, the classification was taken as true, but if the probability was lower, the second and third most likely classification was considered being the correct classification. By testing all combinations of sequences using the first, second and third most likely classification, and comparing the sequences to a list of manually gathered common sequences in nutrition labels, some words might be re-classified.

3.3.7 Split by Character

Some words are classified as either "grams", "milligrams", "percent" or "numbers". These words are split up once again, to generate sub-images of each character. The same procedure is done as when splitting words. However, this time it is enough with a single blank space of any length in order to split it. The algorithm even split the image if the number of black pixels are very small and the number of times pixels turn from black to white less than twice.

3.3.8 Classify Characters

The CNN trained on characters is used to classify the digits, letters and symbols that are expected to be in nutrition labels.

3.3.9 Output Result

When all nutrients and amounts are found and identified, they need to be mapped so that each nutrient is paired with its right amount. This is fairly simple as the structure of US nutrition labels are very standardized. On each row, the name of the nutrient comes first, then the amount, and maybe the percentage.

The nutrients we are looking for are the following. The rows with the highest probability for each nutrient is picked.

When all information is extracted, some calculations may be needed before it can be stored in the database. The database store all nutrients in amounts per 100 g.

(a) Step 5, word segmentation

(b) Step 6, word classification

(c) Step 7, character segmentation

(d) Step 8, output

4

Results

4.1 Accuracy of Word Classification

The network trained on the augmented data had the best accuracy. The precision, recall and harmonic value F were calculated to

$$precision = 0.8271,$$

$$recall = 0.7202,$$

$$F = 0.7700.$$

The accuracy after each epoch are presented in figure 4.1. A sample of the classifications of the validation set is presented in figure 4.2. A confusion matrix of the validation set is presented in figure 4.3.

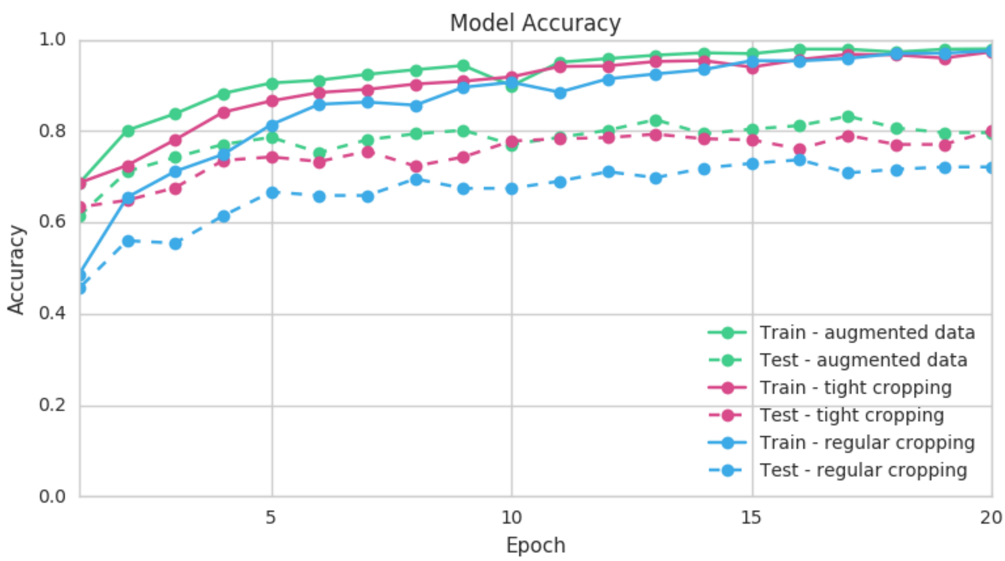


Figure 4.1: Three different word datasets were trained for 20 epochs. After each epoch the classification of the training set as well as the validation set were evaluated. It can be seen that tight cropping gave a higher accuracy than regular cropping and that the dataset with tightly cropped images including augmented data performed the best.

Figure 4.2: 20 samples of the classification of the images in the validation set. Top three classifications and the corresponding probabilities are listed. Even if the algorithm didn't pick the correct classification, the right classification is here always included as the top three.

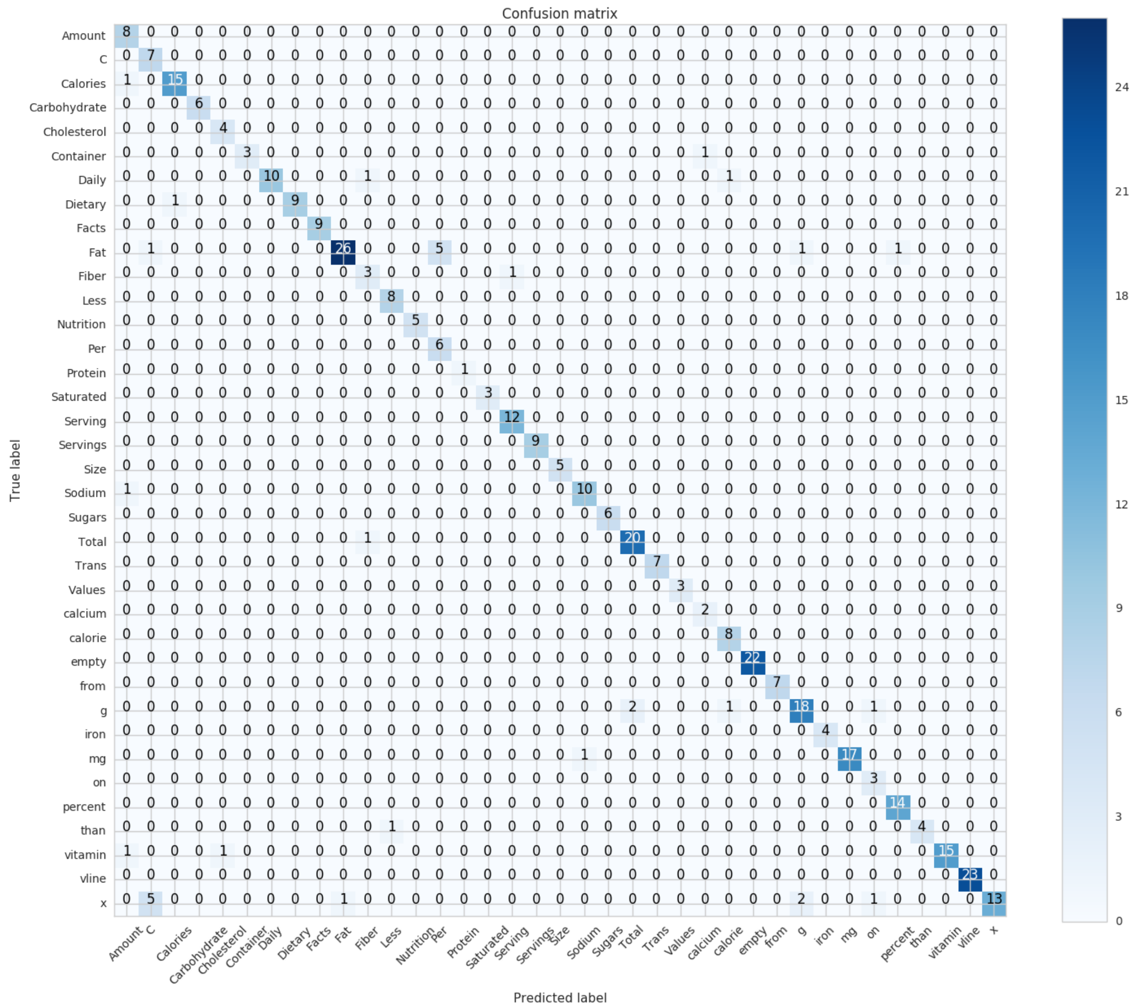


Figure 4.3: In the confusion matrix the columns of represents the number of times a class was predicted, while each row represents the actual number of instances there were in each class. As an example, the word "Fat" was mis-classified as "Per" five times.

4.2 Accuracy of Character Classification

The accuracy for the three different datasets after each epoch are presented in figure 4.4. It shows that the tight cropping or using augmented data did not have a significant impact on the results, therefore the results of the network trained on regularly cropped data will be presented.

The precision and recall for each character varied a bit, therefore they are presented individualy for each character in the table below.

Character	sample size	precision	recall	harmonic
0	24	0.8846	0.8519	0.8679
1	16	0.8889	0.8421	0.8649
2	19	0.8	0.7619	0.7619
3	20	0.8824	0.8333	0.8571
4	10	1	0.9	0.9474
5	18	0.7778	0.7368	0.7567
6	5	0.1667	0.1429	0.1539
7	7	1	0.8571	0.9231
8	8	0.5833	0.5385	0.56
9	5	1	0.6667	0.8
*	3	1	0.75	0.8571
.	1	1	0.5	0.6667
g	19	0.8235	0.7778	0.8
(	4	0.5	0.4286	0.4616
m	6	1	0.8571	0.9231
%	20	1	0.95	0.9744
)	2	1	0.5	0.6667

4.3 Sample Output

Because the accuracy of the algorithm depends highly on the input image, three sample outputs are presented. In figure 4.7 the output of the nutrition label which has served as an example throughout the thesis is shown. Two additional inputs and outputs are shown in figure 4.8a and 4.8b.

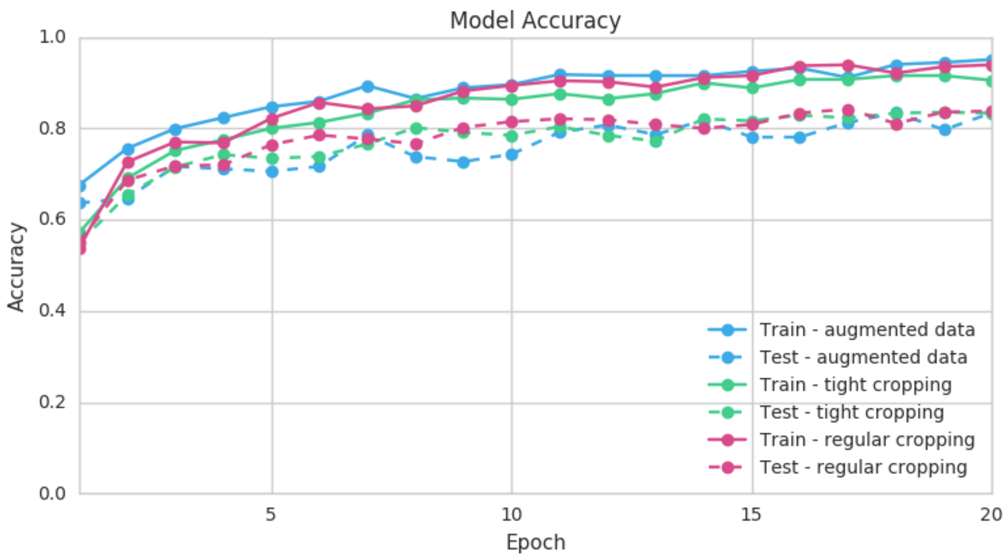


Figure 4.4: Three different character datasets were trained for 20 epochs. After each epoch the classification of the training set as well as the validation set were evaluated. Using tight cropping and augmenting the data does not seem to have a significant impact on the output result.

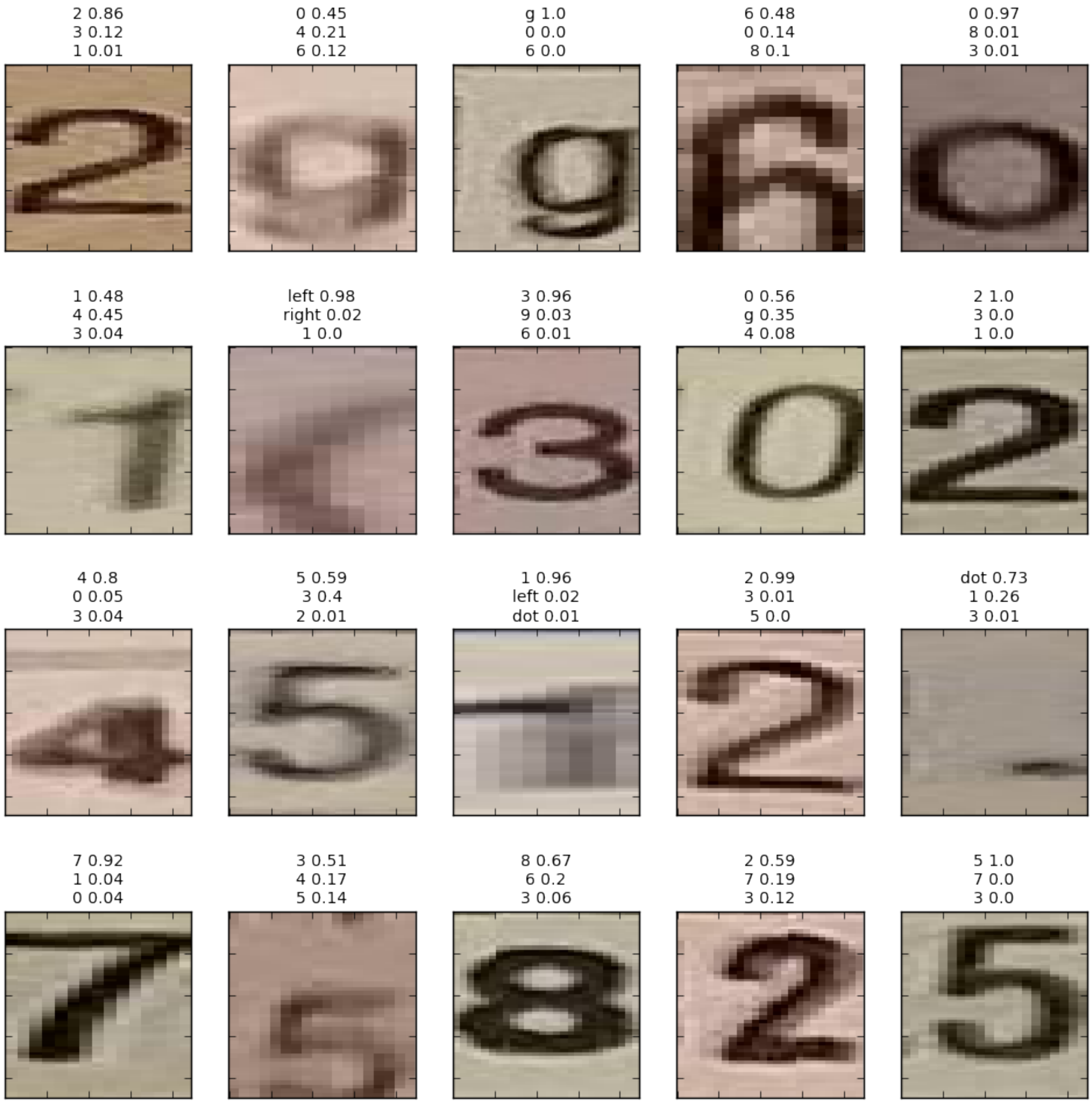


Figure 4.5: 20 samples of the classification of the images in the validation set. Top three classifications and the corresponding probabilities are listed.The algorithm classifies most characters correctly.

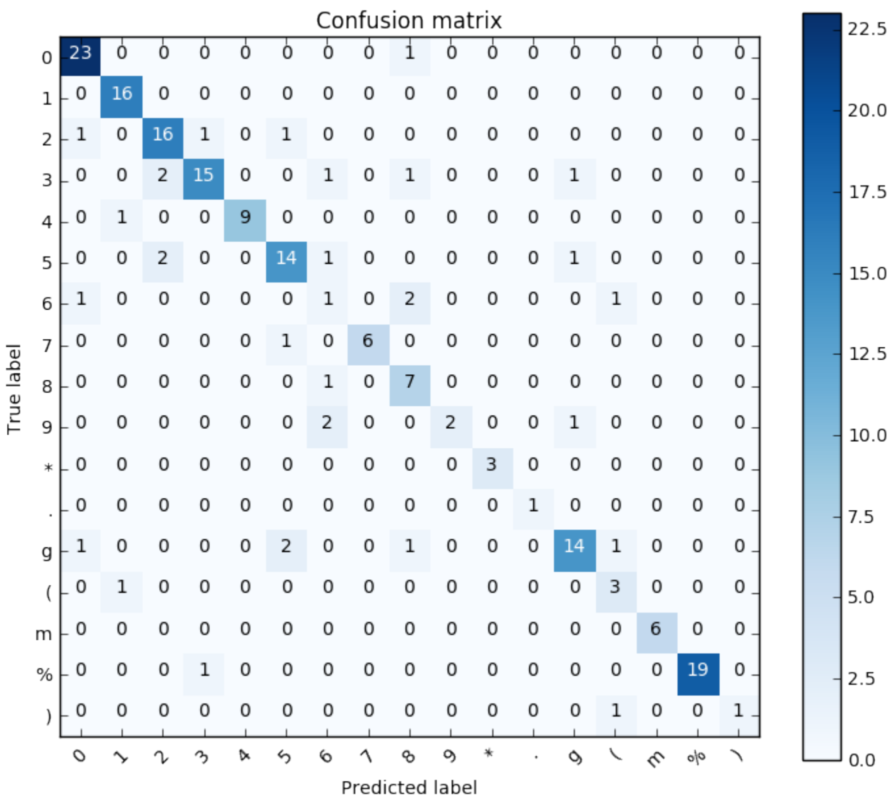


Figure 4.6: The reason many g were labeled as 0 could be due to poor segmentation or blurry images. The tail of the g is not always so distinct.

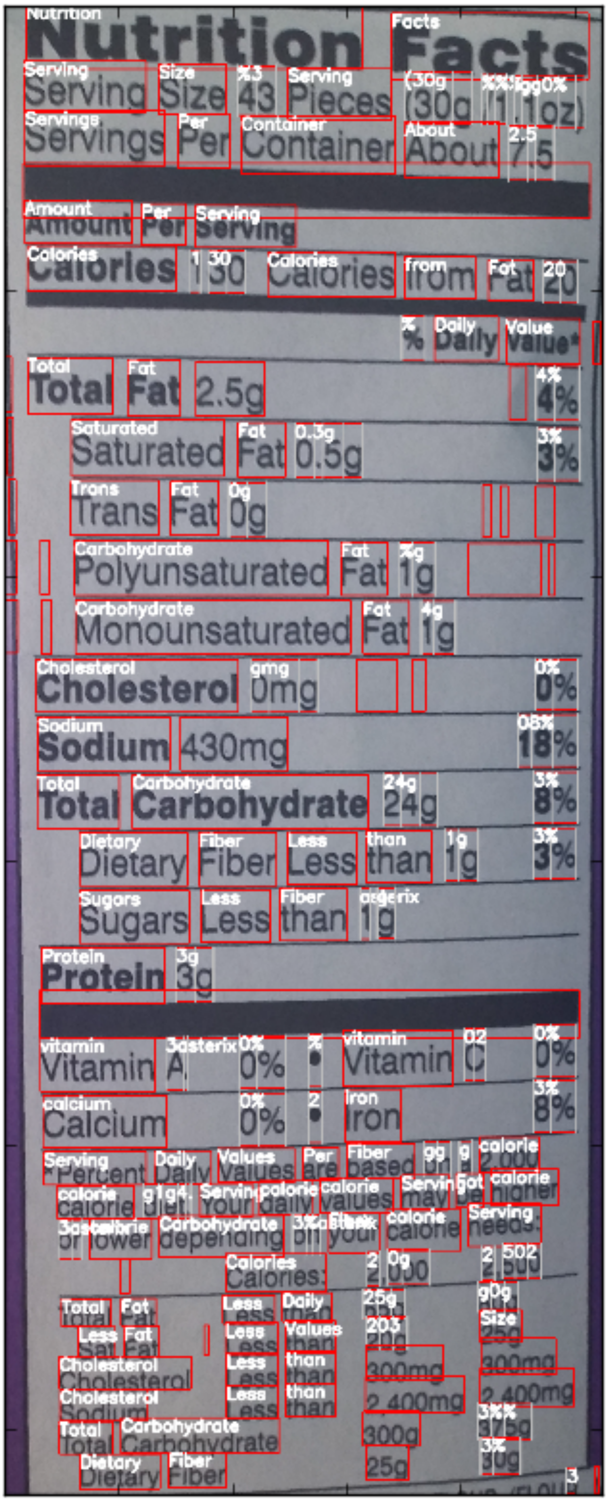
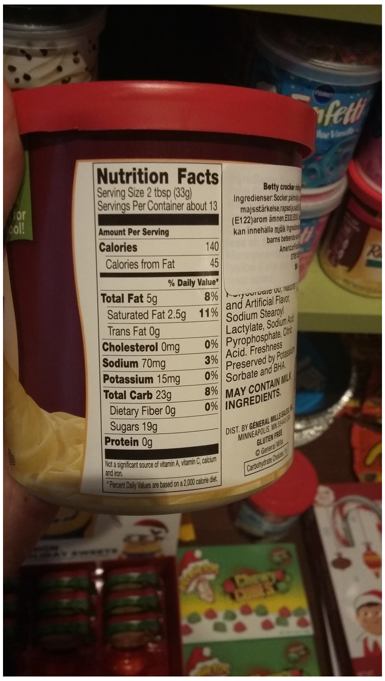
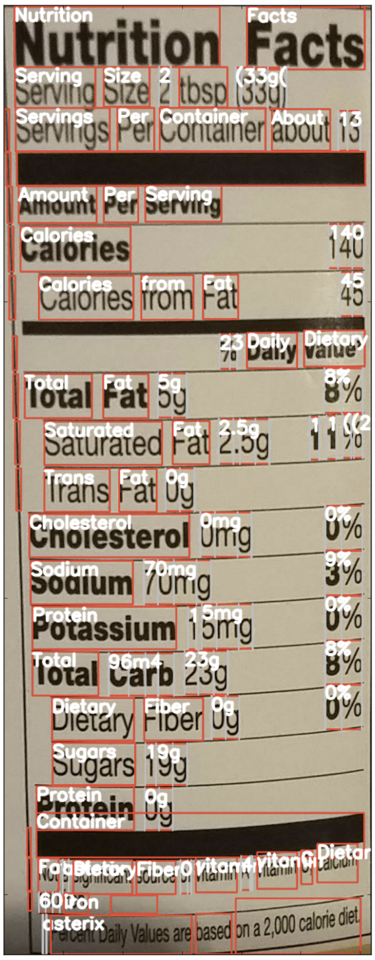


Figure 4.7: An example output of the algorithm. Segmentation worked well and most words were classified correctly. (The network was not trained on Monounsaturated and Polyunsaturated because of the limited amount of sample images.) The characters are not very well classified, might get updated.

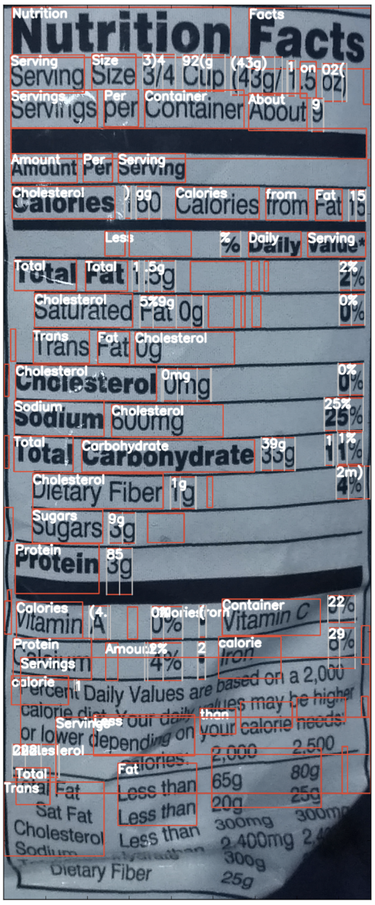


(a) Input image a.

(b) Input image b.



(c) Output result of image a.



(d) Output result of image b.

5
Discussion

The approach of this thesis has been to use common assumptions about how nutrition labels are structured and combine it with machine learning in order to be able to read the content of nutrition labels. This has been achieved by using two rather small, manually collected and labeled datasets.

The result of the algorithm is not straight-forward to evaluate, as the quality of the input image highly affect the performance of the algorithm. If the image is relatively sharp and the structure of the nutrition label is as expected, then the segmentation is likely to turn out successful, meaning that all (important) words and characters are segmented nicely. The classification of the words and characters is not as good as desired though, which is probably an effect of the small datasets that were gathered. In particular, the character classification performed poorly, which might have been caused by the imbalanced dataset.

The famous MNIST dataset classifies numbers from 0 to 9 (10 classes) and the dataset consists of a training set of 60 000 examples and a validation set of 10 000 classes. This could be compared to the two datasets used in this thesis, one of them consisting of 1800 images distributed over 37 classes and the other one 1600 images distributed over 17 classes. It is not hard to draw the conclusion that the dataset used for the networks in this thesis are far smaller than desired.

The bad accuracy has been dealt with by re-classifying words by analyzing the three top classifications of each word the algorithm outputs and compare all combinations with sequences that are expected to be found in nutrition labels. This turned out to increase the classification slightly, but doesn't seem like a sustainable method in the long run, as it makes the algorithm architecture more complex.

A similar approach was considered for re-classifying characters based on the order of the characters and redundancy we expect or not expect to see. Some example of expectations:

- We expect to find g at the end of a word which has been classified as milligrams
- We expect to find m and g at the end of a word which has been classified as milligrams
- We don't expect m and g to be at any other position than second last and last
- We don't expect more than one dot in a word
- We expect % at the end of a word

Another approach in order to increase the classification was to augment the data by simply increasing and decreasing the intensity, yielding a total of 3 version of the same image. This was a quick fix that turned out to have a positive impact on the outcome. So if new data would be considered being too time consuming to collect and label, then maybe it would be worth trying other methods to augment the data even more. Stretching the image could be one idea, (but because all inputs to the vgg16 net are transformed to images of 224 x 224 x 3, stretching the image would only bring value if the stretching wouldn't be uniform along the axis).

The input quadratic dimensions to the vgg16 network have another impact on the dataset. Because all inputs are transformed into images of size 224 x 224 x3, it means that the ratio between the width and height is lost, which might be valuable information as says something about how long a word is. Maybe the ration could be added to the classification algorithm in some manner. However, the horizontal frequency distortion of image features most likely provides the network with word-length cues.

An interesting result is that "tight cropping" compared to "regular cropping" increases the accuracy of words, whereas it's hard to tell a difference when it comes to characters. The reason is probably that when characters are crop, we lose information about where on the row the character is placed vertically. For example, if images of characters are cropped tightly, then a *g* and a 9 looks very similar.
Google made a successful AI tool which is called "Quick, draw!", where the user is asked to draw different things that the computer ask you to and then the AI is supposed to guess what you draw. A few months back, the AI was pretty good, but not breathtaking. However, when I recently tried it again, it's doing an amazing job. This is because as people has tried the AI and even if the AI made mistakes, new data was gathered while people were using it. The AI has then been re-trained and now does an amazing job.

The point I am trying to make here is, how accurate does an algorithm have to be to still be useful? If an algorithm is good enough and presented to users as a beta version then hopefully users are willing to contribute by re-labeling the classifications that the algorithm didn't manage to classify correctly, so that in a future state, the networks can be retrained on more data. If a nutrition label feature like this is launched in the right way, not promising too much, then hopefully it will serve it's purpose without any mistakes.

6
Conclusion

It can be concluded that if the number of words to be classified is very limited, it might be a good idea to consider them as objects in a Photo OCR algorithm. It can also be concluded that using assumptions about how the text content is structured can be useful for segmentation.

Bibliography

[1] *Efficient backprop - Neural networks: Tricks of the trade.*

[2] Ankush Gupta, Andrea Vedaldi, and Andrew Zisserman. Synthetic data for text localisation in natural images. *CoRR*, abs/1604.06646, 2016.

[3] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Reading text in the wild with convolutional neural networks. *CoRR*, abs/1412.1842, 2014.

[4] Lukas Neumann and Jiri Matas. A method for text localization and recognition in real-world images. In *Proceedings of the 10th Asian Conference on Computer Vision - Volume Part III*, ACCV'10, pages 770–783, Berlin, Heidelberg, 2011. Springer-Verlag.

[5] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

A
Appendix

A.1 Table of characters

Character	**samples**
0	150
1	150
2	150
3	150
4	132
5	150
6	72
7	72
8	87
9	45
g	150
m	69
*	39
.	60
%	150
(	57
)	39