International Journal of User-System Interaction 13(1) 2020, 1-17

Expiry date recognition using deep neural networks

Vlad Florea, Traian Rebedea

University POLITEHNICA of Bucharest
Splaiul Independenței 313, București 060042
E-mail: vlad.florea1709@stud.acs.upb.ro, traian.rebedea@cs.pub.ro

Abstract. This paper proposes a deep learning solution for optical character recognition, specifically tuned to detect expiration dates that are printed on the packaging of food items. This method can be used to reduce food waste, having a significant impact on the design of smart refrigerators and can prove especially useful for persons with vision difficulties, by combining it with a speech synthesis engine. The main problem in designing an efficient solution for expiry date recognition is the lack of a large enough dataset to train deep neural networks. To tackle this issue, we propose to use an additional dataset composed of synthetically generated images. Both the synthetic and real image datasets are detailed in the paper and we show that the proposed method offers a 9.4% accuracy improvement over using real images alone.

Keywords: expiry date recognition, synthetic ocr dataset, deep learning, computer vision, vision impaired

DOI: 10.37789/ijusi.2020.13.1.1

1. Introduction

Food waste is one of the worst problems currently affecting the world, needlessly exhausting natural resources and polluting our planet. For comparison, if food waste were a country, it would be the third in terms of greenhouse gas emissions, according to the Food And Agriculture Organization of the United Nations[1]. An application for smart refrigerators that allows scanning the expiration date of a product, using images captured by the camera of a mobile or fixed device, would encourage to reduce food waste and would allow visually-impaired persons to find out this information

[1]Food And Agriculture Organization of the United Nations, "Food wastage footprint & Climate Change", http://www.fao.org/3/a-bb144e.pdf, last accessed 30th March 2020

without the help of others.

Although there are several solutions proposed for expiry date detection in images, like those proposed by Zaafouri, Sayadi & Fnaiech (2015) or Peng, Peursum & Li (2012), there is no readily available commercial application on the market that uses such a solution. However, a lot of applications for expiry date monitoring are available, which have a considerable number of downloads, but use manual date input methods such as date pickers or text fields. These applications, which are addressed to both companies and individuals, send notifications to the user when a product reaches its expiry date. Unfortunately, manually entering the expiration date for each product can be very time consuming.

Another problem is that current laws do not oblige manufacturers to include the expiry date of each item in Braille format on the packaging, so people with visual impairments cannot access this important information.

2. Existing methods

Existing optical character recognition systems have very poor performance when it comes to expiry date recognition. The poor accuracy of such systems for the task at hand is most likely due to the fact that the datasets they were trained on did not include dot matrix type fonts which make up most expiration dates. These systems are also trying to obtain the highest degree of generality possible for any type of text, size, text density in the image, etc.

To visualize their performance we used the validation part of the dataset proposed in this paper (section 4. Dataset), which we've also made public on Github. We ran the existing solutions over these images, and then, for consistency, the extracted text has been passed through the filtering algorithm mentioned in section 5. Filtering candidates, which uses regular expressions and a few other techniques to extract the expiration date.

In the following section, some of the most popular systems will be presented and their applicability to this problem will be analyzed using the aforementioned method.

Tesseract OCR

Tesseract OCR is one of the most popular open-source engines for optical character recognition. Results obtained using this system (Table 1) show that it can't be applied to the task at hand. This system is more specialized in

optical character recognition for scanned documents, while the problem of expiry date detection is similar in difficulty with the problem of "scene text detection" in the state of the art literature, described in detail by Long, He & Yao (2018). For most images, the output of Tesseract OCR is empty, even if the date is written using ordinary fonts and not the dot-matrix type. Also, even if only the portion containing the expiration date is cropped from the image, results are still poor, due to the high level of noise (background, overlap with other packaging elements, etc.)

Scene Text Detectors

Given the failure of Tesseract OCR it is clear that a more robust system is needed. State of the art "scene text detector" deep learning models should be a better solution, being specialized in the detection of text "in the wild", such as house numbers, advertising panels, or traffic signs. Some of the most common are SegLink (Shi, Bai & Belongie, 2017) and TextBoxes++ (Liao, Shi & Bai, 2018). These models are usually combined with the CRNN model (Shi, Bai & Yao, 2016), to extract text from the regions in which it was detected. For these architectures, open-source implementations available on GitHub have been tested, using the weights provided with the models. The results can be found in Table 1.

Google Cloud Vision API

Google Cloud Vision API is a service which gives application developers access to ready-made computer vision models that run in the cloud. The "out of the box" performance of the Google Cloud Vision OCR system is the best out of all the pre-trained models studied in this section as can be seen in Table 1. In Figure 1 you can see an example where the expiry date detection is correct and an example where this system fails.

Table 1 – Precision obtained by existing systems on the validation set

System	Precision
Tesseract OCR	10.6%
TextBoxes++ (default) + CRNN (default)	13.6%
SegLink (default) + CRNN (default)	16.6%
Google Cloud Vision API	50.0%

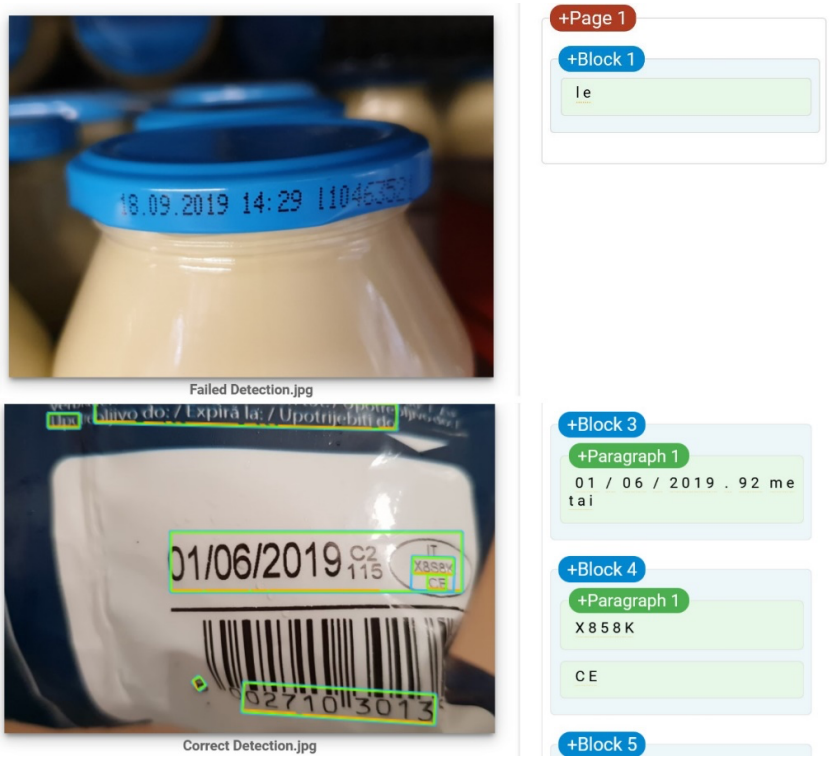

Figure 1. Examples of correct and failed expiry date detection using the Google Cloud Vision API

Considering the poor results presented above, obtained using existing systems, experts in the computer vision field have previously attempted to solve the problem of expiry date recognition bypassing some of the steps required for an end to end system:

- Peng, Peursum & Li (2012), use the relationship between the position of the barcode and the position of the expiration date on the packaging of each product. This way, the step of extracting regions of interest (detailed in section 3.1. Extracting regions of interest) is essentially eliminated and only the extraction of the text from the already set region is necessary. The major disadvantage is that the expiry date position may not necessarily be correlated with the barcode position, especially for products where it is manually printed and may not be placed the same every time. It is also necessary to build a database that stores the relationship between the two for each individual

product.

- Zaafouri, Sayadi & Fnaiech (2015) propose a method for extracting regions of interest using classical computer vision techniques and then classifying them using a single hidden layer neural network or an SVM to separate regions containing expiration dates from the rest of the image. The algorithm proposed in the current paper eliminates the need to manually generate features as the deep convolutional network presented in section 3.1. Extracting regions of interest will automatically select the best features during the training process.

3. Proposed method

Inspired by state of the art optical character recognition architectures, the algorithm proposed in this paper represents an "end to end" solution for detecting expiry dates in images (with very high accuracy) and consists of three important processing steps, which can be seen below in Figure 2.

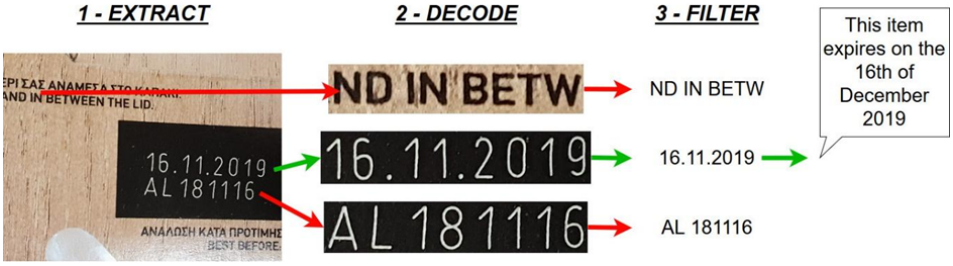

Figure 2. Main processing steps in the proposed method

The first step is to extract regions of interest that might contain expiration dates by using a deep state of the art neural network of the "scene text detector" type. Unlike classical techniques for optical character recognition in documents, neural networks of this type are specialized in the detection of text "in the wild", such as images containing house numbers, advertising panels, traffic signs, etc. Thus they can detect text in various positions, lighting conditions, backgrounds or fonts. For this reason, they are most suitable for expiry date detection, a problem that presents similar difficulties.

During the second step, quadrilaterals delimiting the regions of interest obtained in the previous step will be cropped from the image and used as input for a deep neural network of the "scene text recognition" type,

specialized in extracting sequences from images. Thus, we will obtain the contents of each region of interest as ASCII text (section 3.2. Text extraction).

Since there are no public datasets for the problem of expiry date recognition, the construction of a specialized dataset for this problem was necessary in order to train the two neural networks. This dataset consists of both real images and images that have been artificially generated using a 3D graphics engine. The dataset used is detailed in section 4. Dataset

For the last phase (section 5. Filtering candidates), text extracted in the previous step will be filtered using a series of regular expressions and logical criteria, then the actual dates will be obtained using a library that can parse time and date in most popular formats.

3.1. Extracting regions of interest

The first part of the system consists in extracting regions of interest, that have features similar to those of an expiration date, from the image. For this task we used the TextBoxes++ architecture based on the high performance for text detection and localization obtained on the ICDAR challenge, for its fast runtime compared to other models and for the existence of an official implementation published by the authors, which demonstrates how it works and justifies the results obtained. Also, the similarity between this model and the Single Shot Detector (SSD) model (Liu, et al., 2016) makes it easy to understand, as SSD is an object detection model widely used today, so there are plenty of materials available online which explain its inner workings.

The TextBoxes++ architecture

The TextBoxes (Liao, Shi, Bai, Wang & Liu, 2017) model treats the problem of scene text detection similarly to the problem of object detection, building upon the SSD architecture, which, in turn, builds upon the advancements of YOLO (You Only Look Once) (Redmon, Divvala, Girshick & Farhadi, 2016), by using a single, end to end, convolutional neural network which takes in an image and directly predicts the class of each detected object and its associated bounding box. The main difference is that in TextBoxes there are only two classes (background and text) and only the landscape format "prior boxes" are used. To take this into account, there are a few adjustments which need to be made, like for example handling the reduced density on the vertical axis by creating offset "prior boxes" to fill this empty space, and others which are further explained in the TextBoxes paper.

TextBoxes++ (Liao, Shi & Bai, 2018) extends the original TextBoxes model, published a year before, by adding the capability to detect skewed text regions. While TextBoxes could only extract rectangle shaped bounding boxes, TextBoxes++ introduces the concept of "Rotated Bounding Box", predicting an extra set of parameters that convert the detected rectangles to parallelograms (colored with green in Figure 3) by applying an offset to their corners. By using this technique, the amount of background content extracted from the original image is kept to a minimum, which is very useful, especially in the case of expiry date recognition, where backgrounds can have a lot of unwanted noise.

Figure 3. "Rotated Bounding Box" coefficients illustrated

3.2. Text extraction

The regions of interest extracted during the previous step are used as input to a recurrent neural network which specialises in optical character recognition.

The CRNN architecture

For this step the CRNN (Shi, Bai & Yao, 2016) architecture has been used as it is a model commonly used in conjunction with TextBoxes++, East Text Detector (Zhou, et al., 2017), SegLink (Shi, Bai & Belongie, 2017) and other scene text detection architectures, in order to achieve state of the art performance as reported by their authors.

The CRNN architecture is depicted in Figure 4. The first part of the network is composed of convolutional layers which have the purpose of

extracting low level features from the image. The latent space obtained using these convolutional layers is then split up into vertical chunks which are then used as input to the recurrent part of the network, composed of bidirectional „Long Short-Term Memory" (LSTM) (Hochreiter & Schmidhuber, 1997) blocks, which take into consideration multiple columns to output a character at each position. The resulting characters are then passed through a final layer, which filters possible duplicates or excess white space.

This model is very flexible, as its authors could use it to detect notes in sheet music, so finetuning it to detect characters in expiration dates should not pose any problems. The only necessary step is to finetune this model using regions containing images of expiration dates which are cropped from our dataset.

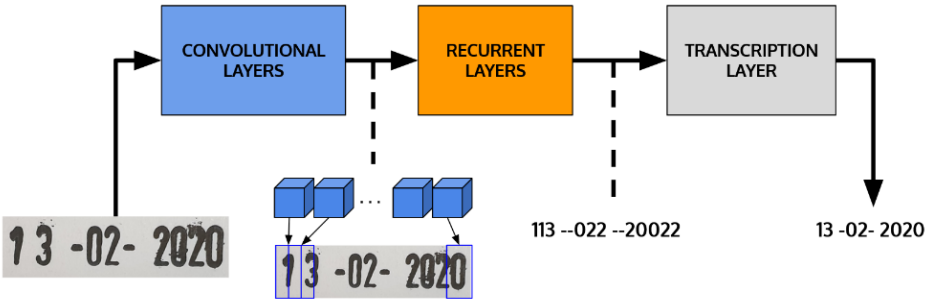

Figure 4. CRNN architecture

4. Dataset

Given the rising popularity of deep neural networks and the resulting increasing demand for training data, many methods of synthetically generating datasets are beginning to emerge. A good example could be considered the free Microsoft AirSim[2] simulator, which generates training datasets for drones or autonomous cars using a 3D graphics engine. It captures images using virtual cameras, located in 3D space, whose parameters have been adjusted to simulate their real-life hardware correspondent

[2] https://www.microsoft.com/en-us/research/project/aerial-informatics-robotics-platform/

(position in relation to the vehicle, field of view, etc.).

For training the networks mentioned in sections 3.1. Extracting regions of interest and 3.2. Text extraction, a dataset consisting of both real images with expiration dates printed on products and synthetically generated images has been built.

The dataset presented is used only for "fine-tuning", as firstly a pre-training step is needed due to the very large number of parameters in the used neural networks. For this purpose two popular datasets in the field of optical character recognition were needed, which are also employed by other approaches using TextBoxes++: SynthText (Gupta, Vedaldi, & Zisserman, 2016) and ICDAR (Shahab, Shafait, & Dengel, 2011).

4.1. Real world images

For the dataset containing real images, we have collected approximately 660 photos, taken using a smartphone, which contain expiration dates from a diverse range of food products. To annotate these images a readily available online platform has been used, where all the images were first uploaded and then a minimal interface for annotating them has been defined. All images were manually annotated using the interface in Figure 5.

After specifying the classes of objects required, the annotation platform allows users to define segmentation polygons over the image using their mouse. Also, extra fields can be added, which can be marked as mandatory or optional.

For this project the following fields have been defined: the exact date in a fixed format, to be able to test the efficiency of the final filtering step (section 5. Filtering candidates) and the exact text present in the image that defines this date to train the optical character recognition step (section 3.2. Text extraction). These two fields have been doubled in case the image also contains the manufacturing date of a product, which is identical to the expiry date in terms of features and therefore is also useful for training the models. These dates can then be removed during the filtering step, being an earlier date than the expiry. A few checkboxes were also added to the interface so that users can report problematic images that were later reintroduced or removed completely from the dataset.

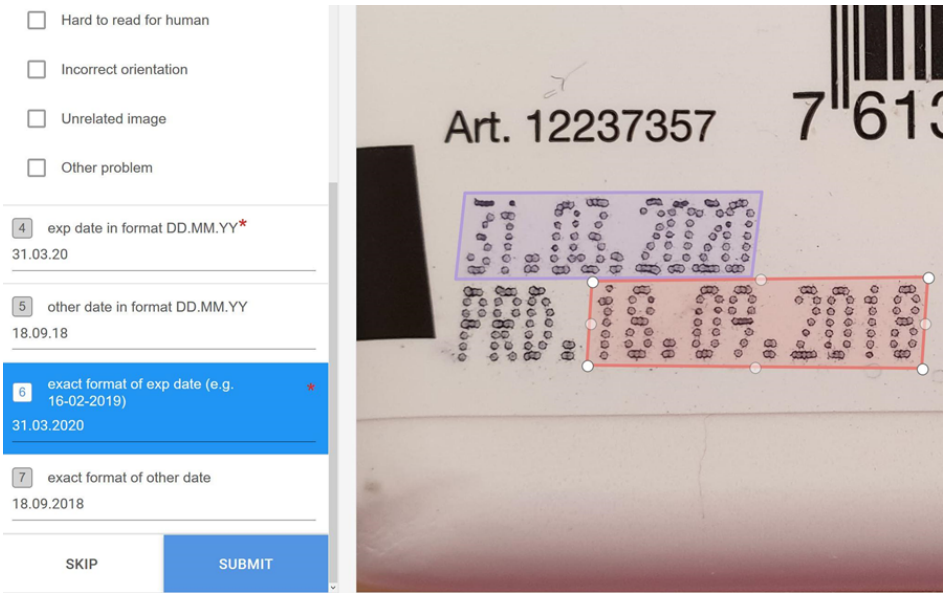

Figure 5. Annotation interface for the dataset with real expiry date images

The content of the fields and the coordinates of the segmentation polygons can be downloaded at the end of the annotation process as a JSON file. We developed a Python script to convert the generated JSON file into CSV files with the same formatting as the ones in the ICDAR dataset so as not to make changes to our input and output pipeline for .tfrecord files.

10% of the images obtained (66 real images) were used as a validation set. These were not included in the training process and were used only for performance evaluation (section 7. Results).

4.2. Synthetic training set

Development of the synthetic dataset began by downloading some freeware fonts from the internet which contain dot matrix type characters or characters modeled after thermal printers and typewriters. These fonts were then used in a Python script, in conjunction with the date and PIL libraries, to generate transparent .png images containing dates in different formats. Some examples can be seen in Figure 6.

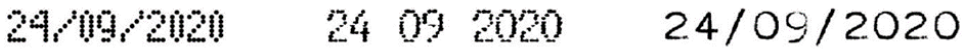

Figure 6. Examples of synthetically generated text

Next, these images were imported into the Unity3D graphics engine as transparent textures. Some .obj 3D models such as cans of beverages, boxes, bottles, etc. were downloaded from the free3d.com website with which a new scene was created, where these objects were positioned. A few directional lights were added to the scene, then a C# script was developed, which randomizes the intensity and position of the lights.

An "anchor" (a GameObject containing only one Transform element) was added to the surface of each object. The position of this anchor will become the target of a DecalProjector object that uses a projection volume (Figure 7). This way we can simulate the printing of a date on the uneven surface of the object, by projecting textures containing expiration dates, generated in Python at the previous step (Figure 6), over the surface of the 3D models.

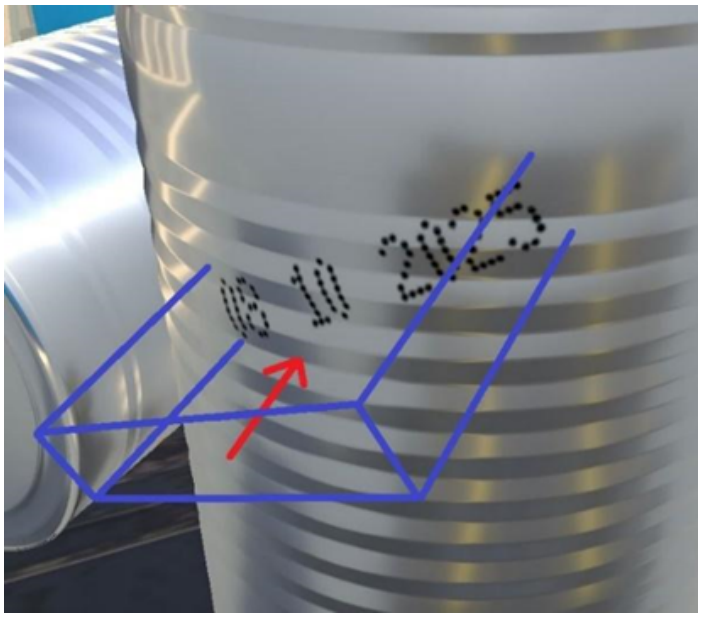

Figure 7. The projector and its associated projection volume

A "Camera" object was added to the scene, whose parameters (projection type, field of view, aperture, etc.) have been manually adjusted to best

simulate the camera of a modern smartphone. A C# script was attached to this virtual camera, to automatically bring the expiration date of a random object into view using the anchors defined earlier on the surfaces. Once the area of interest is brought into the view of the camera, a method is called to change the texture projected on the object (choosing at random from the previously generated textures), and the camera takes a shot. The coordinates of the projected texture are then calculated and transformed into the coordinate system of the camera, multiplying with the corresponding transformation matrix, to obtain the coordinates of the polygon delimiting the expiration date in the final image. The text in the image corresponds to the file name of the respective texture (example: 24s09s2020.png where "s" means the character "/"). The coordinates and text in the image are ultimately saved as a .CSV file in the same format as the ICDAR data set, to maintain consistency with the real image set. Thus, a completely synthetic training example, like the one shown in Figure 8, can be generated.

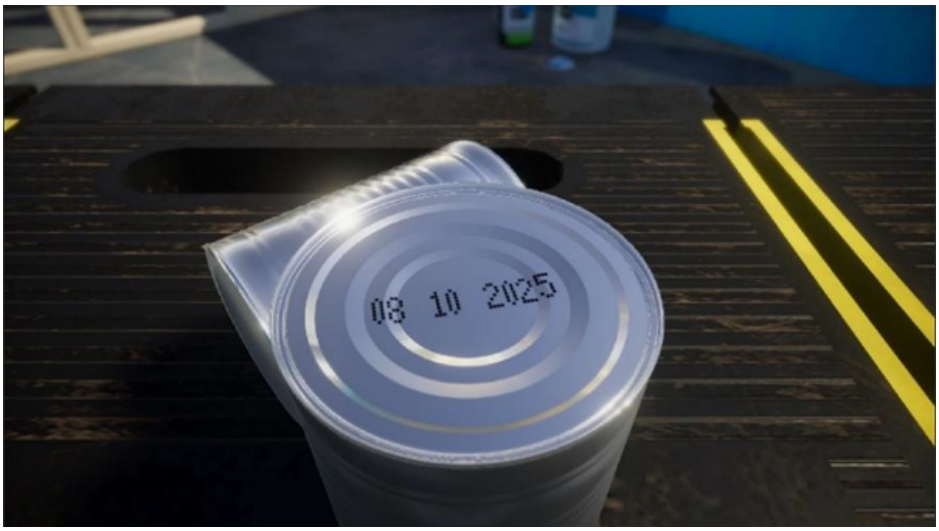

Figure 8. Synthetically generated image in the virtual 3D space

In order to obtain the best results concerning lighting, reflections, transparency of objects, etc. and to simulate a real camera's depth of field the official "HD Render Pipeline" library was used in Unity, which implements state of the art techniques in the field of computer graphics and is only available as a beta version at the time of writing.

4.3. Augmentations

During training, the images in the dataset are augmented, to prevent the phenomenon of overfitting, using transformations such as random rotation, random translation, adding noise, etc. This has been achieved using the imgaug[3] Python library. A few samples can be observed in Figure 9.

Figure 9. Original image (left) and 3 examples of augmentations

5. Filtering candidates

For filtering, a set of regular expressions is used, which extracts combinations between numbers ("\d") and the most common characters used for delimitation (space, dot, line, etc.), to extract all text spans that are similar in format to an expiration date. The regular expression used is depicted in Figure 10:

```
(?:\d{2}(?:\d{2})?(?: *)[/:\-, _.]?(?: *))?\d{2}(?: *)[/:\-, _.]?(?: *)\d{2}(?:\d{2})?
```

Figure 10. Regular expression used for filtering

Extracted text has formats like: year/month/day, day/month/year, month/year, etc. These results are then processed using the dateutil[4] library with the "fuzzy" option activated to let the parser pick out the most apropriate date format automatically. The outputs will be DateTime objects, which will be filtered using an interval of plus or minus 30 years from the current date to prevent errors. Remaining dates will be sorted and the date farthest in the future will be picked.

---

[3] https://github.com/aleju/imgaug
[4] https://github.com/dateutil/dateutil

6. Mobile application

To demonstrate how the algorithm presented in this paper can be used in the real world, to combat food waste, we built an application that runs on a smartphone and keeps track of expiry dates. The application allows users to quickly scan the expiration date (using the proposed method) and barcode of a product using the camera. They will then be able to consult their list of products at any time and will receive notifications when they are close to their expiration date.

The application consists of a backend, a REST API that runs in the cloud and a native Android client app which handles user interaction. The app also makes use of the free OpenFoodFacts[5] API to autocomplete the remaining fields based on the scanned barcode, and thus make the process of adding a new product as fast as possible. The process required for adding a new item is illustrated in Figure 11, and takes only a few seconds per product.

7. Results

The proposed method for expiry date detection proved to be a success, obtaining significantly better accuracy than existing systems (tested in section 2. Existing methods) for expiry date recognition. Table 2 shows a comparison of the performance obtained using each of the data sets presented in section 4. Dataset (the synthetically generated data set and the real data set) for each of the two neural networks used. The best results were obtained by combining the synthetically generated data set with the manually annotated image set. It can be observed that the re-training of the CRNN model generally has a greater benefit than the re-training of the TextBoxes++ model, and the re-training of both models is the one that gives the best results. The precision obtained clearly shows the difference between the synthetic and the real data, because no matter how hard we tried to make the synthetic images look as realistic as possible, the models trained with real data achieved better performance each time. The final result with the best accuracy was the model trained using both datasets, so the construction of the synthetic data set proved its usefulness in the end.

[5] https://world.openfoodfacts.org/data

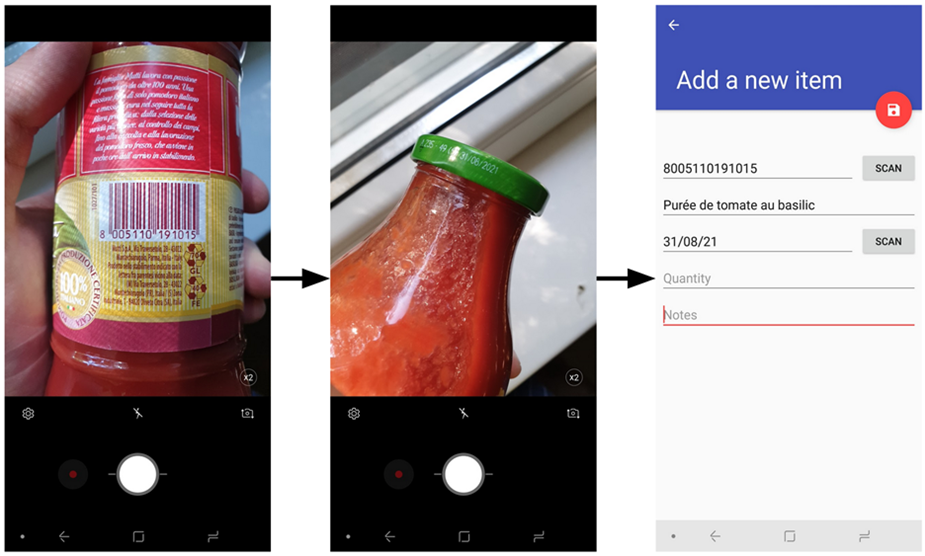

Figure 11. Adding a new product in the mobile app for expiry date tracking

Table 2 – Performance of the proposed method using synthetic and real images

Retrained TextBoxes++	Retrained CRNN	Dataset	Precision
NO	NO	-	13.6%
YES	NO	Synthetic Images	19.6%
NO	YES	Synthetic Images	28.7%
YES	YES	Synthetic Images	40.9%
YES	NO	Real Images	24.2%
NO	YES	Real Images	34.8%
YES	YES	Real Images	63.3%
YES	YES	Real + Synthetic Images	**72.7%**

Also, the waste reduction application that incorporates the presented method works perfectly, the time required for extracting the date is about 4 seconds. The total time required to insert a new product by an experienced user is approximately 15 seconds.

An example of correct and wrong detections obtained using this system can be seen below in Figure 12, and these results show the system only struggles in cases where the date is not printed correctly, or in cases where the contrast between the date and the underlying color of the packaging is very small.

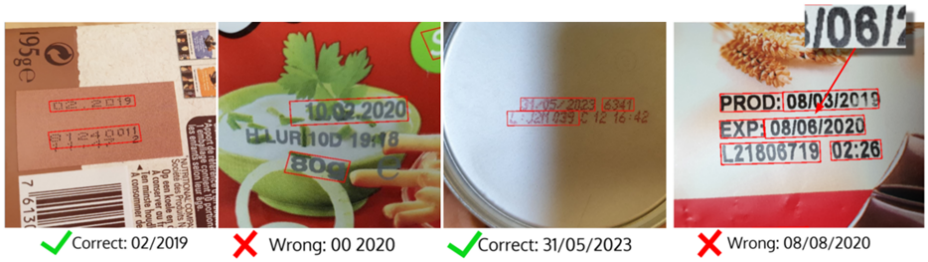

Figure 12. Examples of correct and wrong detections using the proposed method

8. Conclusions

In conclusion, this project proves that it is possible to build an "end-to-end" solution, based almost entirely on deep neural networks, which can detect expiry dates with much greater accuracy than the best optical character recognition systems available on the market. It is also shown that this solution can be reliably used in a mobile application to reduce food waste, a very serious problem considering current global statistics.

During development, it has become apparent how difficult it is to build a deep learning solution in a domain where there are no publicly available datasets of sufficient size. Fortunately for expiry date recognition, we were able to generate a synthetic dataset and gather a considerable number of real images (about 660), as food products are widely varied and easily available.

All the code and datasets presented in this paper are available on Github[6]. You can also find a video clip there which was captured from the screen of the mobile phone and illustrates the usage of the final application.

For the date detection step (extracting regions of interest), the neural architecture proposed by Dai et al. (2018) seems to be a major improvement in the future, due to the fact that it is much more rotation invariant. At the moment, however, there is no publicly available implementation to reproduce the results mentioned by the authors.

References

Dai, Y., Huang, Z., Gao, Y., Xu, Y., Chen, K., Guo, J., & Qiu, W. (2018). Fused text

[6] https://github.com/vladalexgit/expiry_date_recognition/

segmentation networks for multi-oriented scene text detection. International Conference on Pattern Recognition (ICPR). DOI: 10.1109/ICPR.2018.8546066

Gupta, A., Vedaldi, A., & Zisserman, A. (2016). Synthetic data for text localisation in natural images. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, (pp. 2315-2324). DOI:10.1109/cvpr.2016.254

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780. DOI: 10.1162/neco.1997.9.8.1735

Liao, M., Shi, B., & Bai, X. (2018). Textboxes++: A single-shot oriented scene text detector. IEEE transactions on image processing, 27(8), 3676-3690. DOI: 10.1109/TIP.2018.2825107

Liao, M., Shi, B., Bai, X., Wang, X., & Liu, W. (2017). TextBoxes: A fast text detector with a single deep neural network. Thirty-First AAAI Conference on Artificial Intelligence.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). SSD: Single shot multibox detector. European conference on computer vision. DOI: 10.1007/978-3-319-46448-0_2

Long, S., He, X., & Yao, C. (2018). Scene Text Detection and Recognition: The Deep Learning Era. arXiv preprint arXiv:1811.04256.

Peng, E., Peursum, P., & Li, L. (2012). Product Barcode and Expiry Date Detection for the Visually Impaired Using a Smartphone. 2012 International Conference on Digital Image Computing Techniques and Applications (DICTA). DOI: 10.1109/DICTA.2012.6411673

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). DOI: 10.1109/CVPR.2016.91

Shahab, A., Shafait, F., & Dengel, A. (2011). ICDAR 2011 Robust Reading Competition Challenge 2: Reading Text in Scene Images. International Conference on Document Analysis and Recognition. DOI: 10.1109/ICDAR.2011.296

Shi, B., Bai, X., & Belongie, S. (2017). Detecting oriented text in natural images by linking segments. IEEE Conference on Computer Vision and Pattern Recognition. DOI: 10.1109/CVPR.2017.371

Shi, B., Bai, X., & Yao, C. (2016). An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. IEEE transactions on pattern analysis and machine intelligence, 39(11), 2298-2304. DOI: 10.1109/TPAMI.2016.2646371

Zaafouri, A., Sayadi, M., & Fnaiech, F. (2015). A Vision Approach for Expiry Date Recognition using Stretched Gabor Features. The International Arab Journal of Information Technology, 12, 448-455.

Zhou, X., Yao, C., Wen, H., Wang, Y., Zhou, S., He, W., & Liang, J. (2017). EAST: an efficient and accurate scene text detector. IEEE conference on Computer Vision and Pattern Recognition. DOI: 10.1109/CVPR.2017.283

.