Spotting and Transcribing Structured Nutrition Information from Product Images.

Joseph Okonda L.
Stanford University

Abstract

We apply computer vision to the task of reading structured information from noisy natural images. In particular, we develop and train models to spot and transcribe nutrition information from the packaging of processed food products. We begin by collecting and labelling a dataset containing ≈ 42000 images. Then we develop and train a YOLO9000 based detection system to spot the nutrition fact-box and the list of ingridients in an input image. Finally, we crop out the detected regions of interest and use them as input to an encoder-decoder transcription model. The detection module achieves 0.8 mAP at 0.7 IoU on our dev set. However, although the transcription module learns to successfully model our target grammar, it has a hard time conditioning on the input images.

1. Introduction

For the average consumer, it is hard to factor in a product's healthfulness when making purchase decisions. This is because of the convolutated nature of the nutrition information available to the consumer. The information is usually available in the form of a list of ingridients and a nutrition fact-box as shown in Figure 1. The ingridient list usually includes names of items that are unfamiliar and thus hard to judge their healthfulness. The numbers in the nutrition fact-box on the other hand are hard to correctly reason about.

We posit that, a computer program, if furnished with the values in the nutrition facts box and the list of ingridients, would do a a much better job of objectively assessing whether a product is healthy or not. However, for this to happen, we need to first develop a way for such a program to ingest the required information. We thus develop models to transcribe the needed data. We break the task into two subtasks. The first is to detect the nutrition fact-box and the list of ingridients. The second is to transcribe the needed information from the closely cropped segments detected by the first task. We adopt this approach because a baseline transcription system (not discussed further) that uses an off-the-shelf tool (tesseract) performs better when the input image is a closely cropped text section. For the first task, we train a detection system derived from YOLO9000 [13]. For the transcription task we train a sequence2sequence [17] model with a convolutional encoder.

Figure 1. What is isolated soy protein? Is it good or bad? How about Carrageenan? Is 320 calories per bottle healthy? What about 5 miligrammes of cholesterol or 33 grammes of sugar?

With these tasks comes one main challenges: We must collect or synthesize and label a dataset large enough for the models to learn. We discuss or data pipeline in the methods section.

2. Related Work

Object Detection sytems fall into two main recent categories: Region Proposal based detectors (R-CNN) [3, 4, 14] detect objects by first running a proposal generator which outputs regions of the image that are likely to contain an object of interest. The proposals are then further processed to classify any objects that they contain and to refine the objects' bounding boxes. Such systems give the best accuraccy but also have a high latency. Single Shot Detectors (SSDs) [12, 9] on the other hand frame object detection as a regression problem. By directly predicting the coordinates of the bounding boxes, SSDs are able to detect objects without extracting the preliminary region proposal as in R-CNNs. This makes them quite fast. However, their low latency comes at the cost of reduced accuracy.

In this project we use a single shot detector to spot nutrition information. We make this choice because we'd like our models to be able to perfomrm inference in real time.

Machine Reading systems range from traditional OCR systems such as [1] to recent models capable of recognizing text in unconstrained scene images like [7, 6]. Traditional systems impose strict contraints on their inputs. In particular, they assume that input images are printed documents on a clear backgroud without much noise. This makes them unfavorable for the task at hand because even though the texts we'd like to read are machine generated, they occur in very noisy environments with auxilary text that we'd like to ignore. Furthermore, the data we'd like to extract has structure that we'd like to preserve. Uncontstrained text spotting and classification is a much harder problem than simple optical character recognition. Unlike OCR, this task imposes no constraints on the input image [6]. Models that solve this problem usually cast it as an object detection task where various approaches are used to localize and classify the characters present in the input image. Such models offer much versatility in terms of their inputs but have high latency especially when the input images are rich in text.

The task in this project is much more complex than traditional OCR but simpler than unconstrained text spotting. We cast the transcription task as an image captioning task where the 'caption' to be generated is the text contained in the image.

3. Methods

3.1. Data

We begin by collecting data in the form of short (15 - 20 sec.) videos shot from a cellphone camera [1]. We collect 700 such videos. We then turn the videos into images using FFMPEG ensuring that we only sample a few images (5) every second. This is done to ensure that images from the same video have as much variance as possible. After this operation, every video is reduced to approximately 80 images giving us a dataset of ≈ 40000 images. Each image is labelled with ground truth bounding polygons as shown in Figure 2. Additionally, we transcribe every *video's* list of ingridients and its nutrition facts dictionary. We split our data into train-dev-test sets using a 95-2.5-2.5 scheme. Crucially, this split is done at the video level instead of the image level. This is to ensure that our test and development sets are truly unseen during training.

3.2. Spotting

For spotting, we repurpose the YOLO9000 model [12] to fit our task. In particular, we only adopt the first 13 layers. We add on a final convolution layer with as many filters as the size of our target tensor. The receptive field of the kernels in this layer is the entire input filtermap. We do this in order to avoid including fully connected layers in the architecture. After every convolution layer we add Batch Normalization [5] in order to speed up training. We also include dropout [16] after every batch normalization module for regularization. Table 1 shows our architecture. We initialize the models with YOLO9000 weights trained on the MS-COCO [8] dataset. We freeze the first 7 layers of the network. Our goal is train this network to predict a tensor of dimension $S \times S \times 7 \times B$ where in our case $S = 3$, i.e the number of grid cells along a single dimension of the input image and $B = 2$ i.e each grid cell predicts 2 bounding boxes. The 7 in the output tensor comes from the fact that we predict an 'object score' telling us if we have an object of interest in a given grid cell, 2 center coordinates, a length, a width, and the probability of the object in that grid cell belonging to one of 2 classes. We train the network to minimize YOLO's multipart MSE loss [12] using the Adam optimizer with a learning rate of 1.5×10^{-6}. To evaluate our model's performance we measure the mean average precision of our bounding box predictions with 0.7 intersection over union (IoU) as our correctness threshold.

3.3. Transcribing

For transcription, we begin by extracting the regions of interest. We do this by using the ground truth bounding boxes of our data to crop out the list of ingridients and the nutrition fact-box. This yields two new images per image, each of which is resized to be 256×256 and then fed *separately* into our encoder. Figure 3 shows example input images to our transcription module. Our encoder is a 14-layer convolutional network whose first 10 layers are initialized from a pretrained YOLO model. However, unlike in the localization module, we do not freeze any layers in the network. Our decoder is a 3-layer GRU model [2] with a hidden state of size 1024. We begin by pre-training the encoder to solve the binary classification task of differ-

[1] https://youtu.be/VssRmq-MHBc Here is a sample video

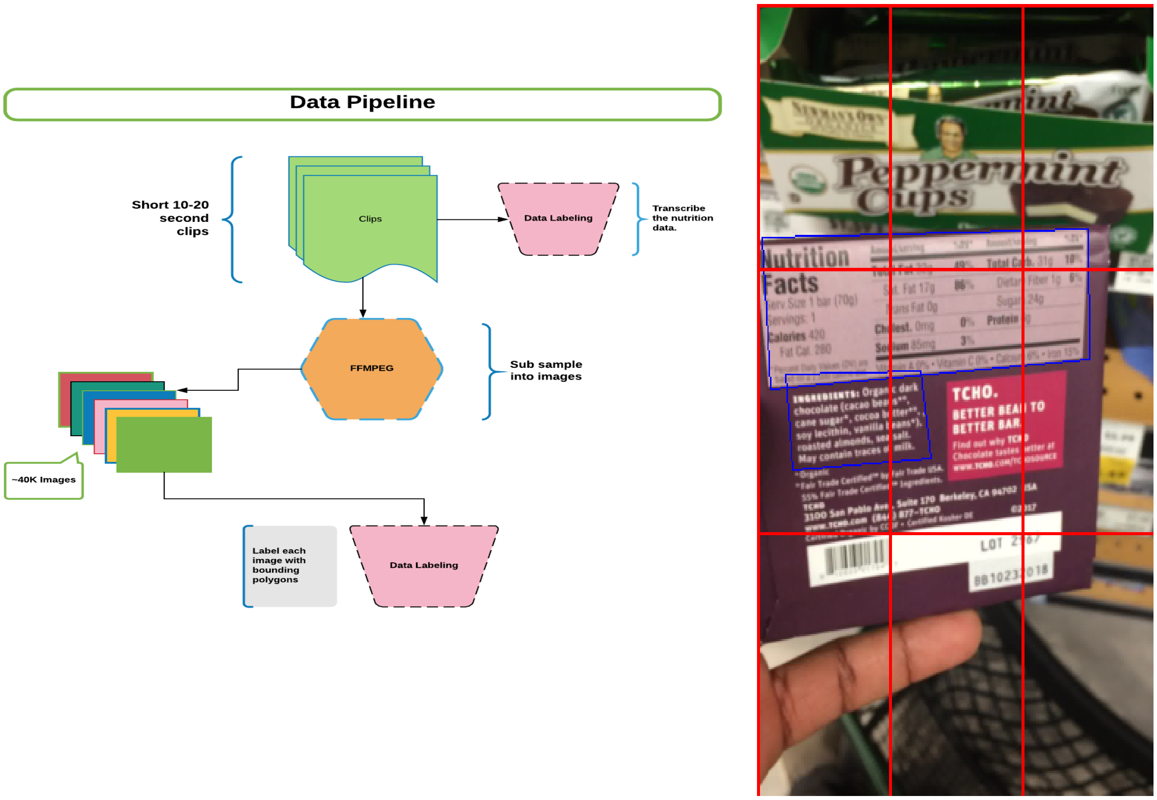

Figure 2. **Left:** Our Data Collection Pipeline. **Right:** An example image from our dataset with Bounding Polygons in blue and YOLO's grid cells in red.

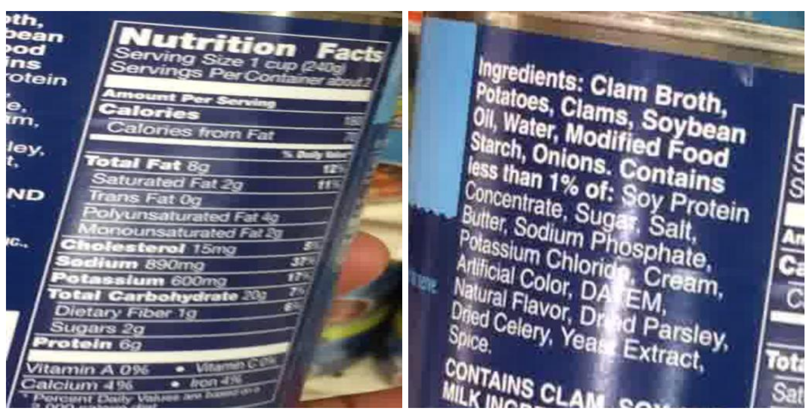

Figure 3. The input images to the Transcription Network. We use the ground truth bounding boxes to extract the regions of interent and resize them to be 256×256. **Top:** A closely cropped nutrition facts section. The **Bottom:** A closely cropped list of ingridients. They are 2 different images.

entiating between a list of ingridients and a nutrition factbox. Without this pre-training step, end-to-end optimization quickly becomes unstable. We train the network to minimize per-word cross entropy loss using Adam with a learning rate of 1.5×10^{-3}.

$$\ell = -\sum_{i=0}^{m}\sum_{t=0}^{T} y'^{<t>} \log\left(y^{<t>}\right)$$

During training we employ 'teacher forcing' [15] where at time t, we feed in the ground truth word vector of time $t-1$. Therefore, the transcription module is attempting to model the following conditional probability:

$$\log p\left(Y|F\left(X\right)\right) = \sum_{t=0}^{T} \log p\left(y^{<t>}|F\left(X\right), y^{<t-1>}\right)$$

Type	Filters	size or stride
Conv	32	3×3
MaxPool		$2 \times 2/s = 2$
Conv	64	3×3
MaxPool		$2 \times 2/s = 2$
Conv	128	3×3
Conv	64	3×3
Conv	128	3×3
MaxPool		$2 \times 2/s = 2$
Conv	256	3×3
Conv	128	3×3
Conv	256	3×3
MaxPool		$2 \times 2/s = 2$
Conv	512	3×3
Conv	256	3×3
Conv	512	3×3
Conv	256	3×3
Conv	512	3×3
MaxPool		$2 \times 2/s = 2$
Conv	S*S*B*C	$K \times K$

Table 1. Architecture of the Localization Network

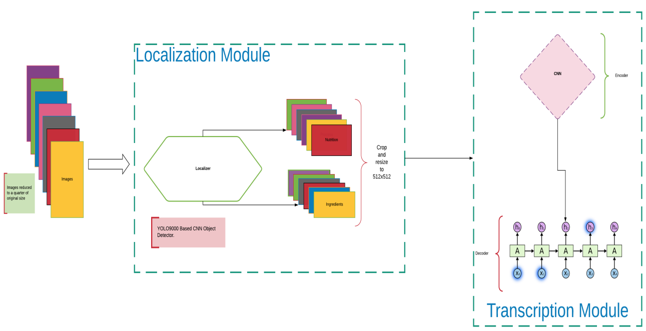

Figure 4. A cartoon of our Entire pipeline.

Where F is the encoder and X is the raw input image. To evaluate our module, we measure the BLEU score [11] using samples from our model and the ground truth annotations. Figure 4 shows an image of our full pipleine.

4. Results

4.1. Spotting Results.

Our trained detection module successfully learns to predict the bounding boxes. Figure 5 shows the results from training the detection network for 100 epochs. When calculating the mAP scores, we set the 'objectness threshold' for each grid cell to 0.9 and use 0.7 as the threshold for each class. We qualitatively evaluate our model by visualizing some of the predicted bounding boxes. Figure 6 shows some of these qualitative results.

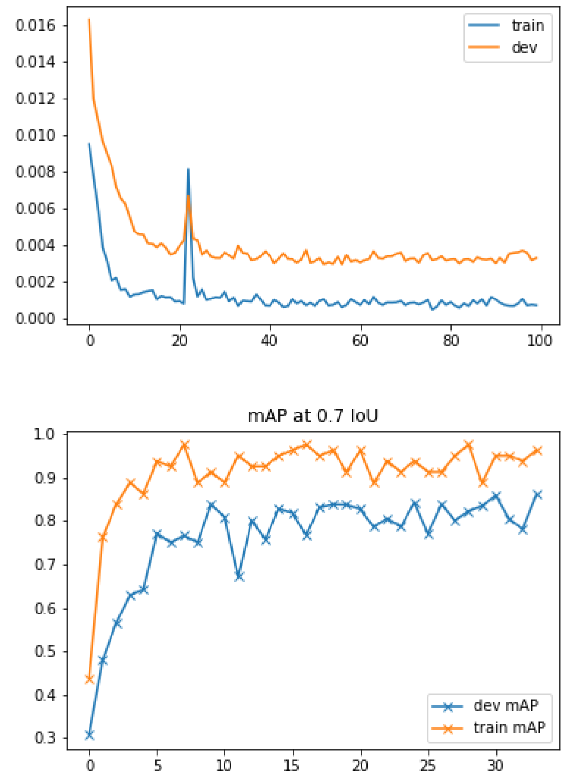

Figure 5. **Left:** Loss curves from both our training and development sets. **Right:** Mean Average Precision Curve measured at 0.7 IoU

4.2. Transcription Results.

The results of our transcription module are not impressive. Figure 8 shows the BLEU scores on both the training and development set. While our model learns to generate text that is consistent with our target grammar, as shown in figure 7, it fails to correctly condition on the input image. A symptom of this is that the transcriptions the model struggles when transcribing numeric values. These are values that the model cannot transcribe by solely relying on a simple language model.

5. Conclusion & Future Work

So far, we have successfully trained a YOLO based network capable of detecting patches of an input image that contain either the list of ingridients or the nutrition factbox. Additionally, we train an Encoder-Decoder model that

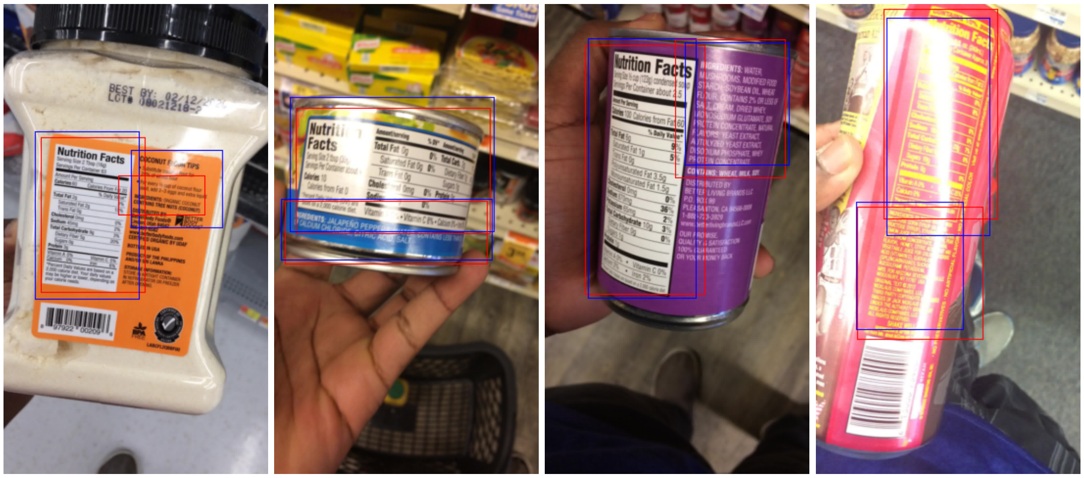

Figure 6. Example bounding boxes generated by our model. The red boxes are the ground truth bounding boxes while the blue boxes are predicted by our detector.

```
Truth
---
<start> {
  num_servings : 2 8 serving_size : 4 0 g
  calories_per_serving : 1 6 0
  calories_from_fat : 4 0
  total_fat : 4 . 5 g sat_fat : 1 . 5 g
  trans_fat : 0 g poly_fat : 1 g
  mono_fat : 2 g cholesterol : 0 mg
  sodium : 4 1 0 mg potassium : 6 0 mg
  total_carb : 2 5 g dietary_fiber : 1 g
  total_sugar : 1 g added_sugar : n/a
  protein : 3 g
} <end>
Prediction
---
<start> {
  num_servings : 4 serving_size : 1 4 0 g
  calories_per_serving : 1 4 0
  calories_from_fat : 4 5
  total_fat : 4 . 5 g sat_fat : 0 . 5 g
  trans_fat : 0 g poly_fat : n/a
  mono_fat : n/a
  cholesterol : 0 mg
  sodium : 1 0 0 mg potassium : n/a
  total_carb : 2 4 g dietary_fiber : 2 g
  total_sugar : 1 3 g added_sugar : n/a
  protein : 1 4 g
} <end>
```

Figure 7. Example Transcription Results **Top:** Ground truth transcription, **Bottom:** Sampled transcriptions.

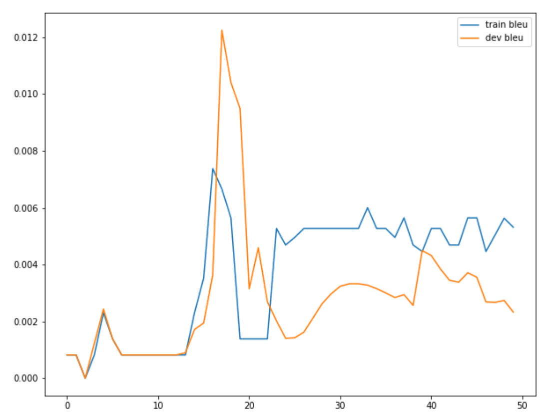

Figure 8. Train and Dev BLEU scores

attempts to transcribe the nutrition information present in a closely cropped image patch. We are less successful at the second task.

From our analysis, we discovered that the transcription network has a hard time conditioning its on the input image's encodings. One way to solve this problem would be by incorporating an attention mechanism [10] in the decoder. Moreover, at the moment, due to the way we collect and annotate our transcription data (We annotate each video), we effectively end up with a very small vocabulary. This means that even if the transcription model improves significantly, we'll be overfitting to a very small vocabulary. Our transcription module also struggles when attempting to transcribe blurry input images. The transcription task requires the model to learn fine-grained details in the input, therefore having blurry input makes this much harder.

Unfortunately, most images sampled from videos have motion blur. This could be remedied by having an explicit preprocessing module that does the deblurring.

Finally, the main bottleneck of our project is the Localizer module. This task requires us to label bounding boxes for a large number of images. Therefore, a future work will focus on ways of replacing this module with a different model that does not require bounding box annotations.

Acknowledgments
We acknowledge Google Cloud and AWS for providing compute infrastructure and the entire 231N teaching team for organizing the class.

References

[1] T. M. Breuel. The ocropus open source ocr system. In *DRR*, 2008. 2

[2] K. Cho, B. van Merrienboer, aglar Gülehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*, 2014. 2

[3] R. B. Girshick. Fast r-cnn. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, 2015. 2

[4] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick. Mask r-cnn. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017. 2

[5] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 2

[6] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Reading text in the wild with convolutional neural networks. *International Journal of Computer Vision*, 116:1–20, 2015. 2

[7] M. Jaderberg, A. Vedaldi, and A. Zisserman. Deep features for text spotting. In *ECCV*, 2014. 2

[8] T.-Y. Lin, M. Maire, S. J. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 2

[9] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *ECCV*, 2016. 2

[10] T. Luong, H. Pham, and C. D. Manning. Effective approaches to attention-based neural machine translation. In *EMNLP*, 2015. 5

[11] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: a method for automatic evaluation of machine translation. In *ACL*, 2002. 4

[12] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016. 2

[13] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6517–6525, 2017. 1

[14] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39:1137–1149, 2015. 2

[15] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural networks : the official journal of the International Neural Network Society*, 61:85–117, 2015. 3

[16] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014. 2

[17] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *NIPS*, 2014. 1