George Washington University

Group Final Report

DATS 6203

The Bounding Shape Generator

Authors:

Charles Garrett Eason

Binbin Wu

Instructor:

Amir Jafari

December 3, 2019

1 Introduction

The purpose of our project was to determine if there was a quick and simple way to generate bounding boxes for object detection using an already trained artificial neural network. Specifcally, we wanted a way to do this without the need of human annotation or the need of backpropagation to calculate bounding boxes. Our solution was to use a sliding window that is fed into an already trained ANN to generate binary maps. These binary maps are then mapped onto the original image and summed to generate a full map that reports object location and shape, we call this the Cropping-Predicting-Mapping (CPM) algorithm. Our solution is unique in its simplicity, efficiency, and the ability to use any already constructed ANN for object boundary identification purposes. Furthermore, there are a few limitations and many avenues for further research to make this approach robust.

This report will consist of several sections that detail our work. First, we will discuss the data that was used for the fruit classifiers followed by a discussion of the classifiers themselves. Next, we will provide an overview of our CPM procedure. Then will follow a discussion of the results and the limitations of our approach. Finally, this will be followed by a conclusion section, references, and appendix with our project code.

2 Datasets

The data used for training our fruit classifier consisted of 120 unique fruits and vegetable types. Each training and testing image presented a single fruit or vegetable in the center of the image with a white background. These images were 100 x 100

pixels across three channels (RGB), which amounted to a total of 82,213 images. Additionally, there were 103 multi-fruit images that contained multiple fruits. Some of these multi-fruit images were used in our CPM procedure (discussed below) after the classifier was trained.

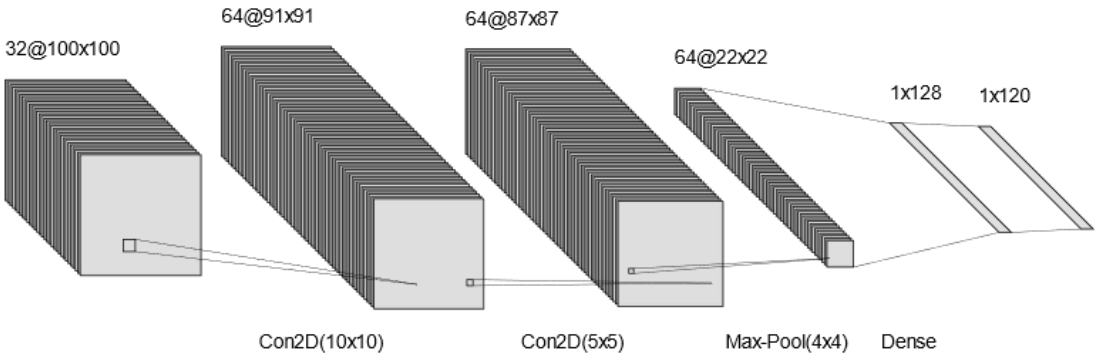
The dataset was constructed by planting fruits and vegetables in the shaft of a low speed motor (3 rpm) while recording a short 20 second movie using a Logitech C920. A white sheet of paper was used as the background. Due to variations in lighting conditions, the authors of the dataset used a dedicated algorithm to extract the fruits from the background. Pixels that were marked as background by the algorithm were then filled in as white.

Our second Cells dataset consisted of 1364 images (80,000 cells) across 7 classes. The images had a median size of 1200x1600x3 and consisted of blood smear photos of different cell samples, some containing different stages of malaria. The dataset was highly imbalanced with most sells being classified as "Red Blood Cell" (RBC). The dataset used to train our classifier was from a subset of cell images taken using the bounding boxes to extract individual cells.

3 Description of Classifiers

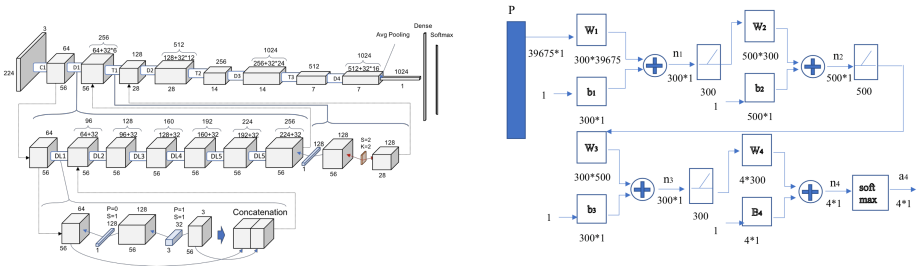
After the decision to proceed with this project was made and the theoretical foundations put together, Garrett created a simple CNN to classify the fruit data. He also augmented the data by: 1) shifting images, 2) zooming images, 3) flipping images, and 4) rotating images. The network architecture for this model was a sequential CNN with the following input and hidden layers:

Figure 1: Fruit-360 CNN



1. Input with size 100 x 100 x 3
2. 2D Convolution, kernel size of 10 x 10, channel output of 32, and ReLu activation
3. 2D Convolution, kernel size of 5 x 5, channel output of 64, and ReLu activation
4. Max Pooling, pool size of 4 x 4, dropout of 25%, and flattening of the outputs
5. Dense layer, output of size 1 x 128, dropout of 50%, and ReLu activation
6. Dense layer, output of size 1 x 120, and activation of softmax

The loss was calculated using categorical crossentropy, the optimizer used was adam, and the metric for model performance was accuracy. After training this network, the accuracy on the hold out set amounted to approximately 98%. While it



(a) Pre-trained DenseNet (b) Cell MLP

Figure 2: *Binbin'sPre-trained DenseNet and Cell MLP models.*

performed well on this fairly "sterile" dataset, this model was rather mediocre using real world images. For example, the model could identify unobstructed apples, but any obstruction caused an immediate decline in model performance. Looking back, generating obstructed images would likely assist with this issue. This was the classifier used to produce the fruit-360 results.

Meanwhile, Binbin used DenseNet121 to train on the fruit dataset. DenseNet121 is a CNN network that passes the output of previous network layers to the next layer. He added an additional dropout layer and global average polling layer at the end to prevent overfitting. The result was 98.3% accuracy and 0.96 F1-score. However, due to limitations of our dataset, little background noise was present which again preventing strong classification results on out-of-sample data. This classifier was trained on 120 fruit classes, but was not used in the fruit-360 results below.

For the cells dataset, Binbin used his own Exam 1 multi-layered perceptrion (MLP) model. His MLP predicted well given the unbalanced nature of the training data, with 96% accuracy and 76% F1-score. Binbin's model contained dropout and batchnormalization layers to help reduce overfitting, help with neuron imbalance,

and speed up training. His model output good results for RBC's as demonstrated below and was the classifier used in the cell results.

4 Description of the CPM Algorithm

After some deliberation and theorizing, Binbin and I came up with the following CPM procedure. The general idea is to crop a sliding window of images from a main image of interest. These cropped images are then fed into some classifier of choice, producing a binary predicted output. This output is then used to generate a "binary map" (matrix of 0's or 1's), which are in turn mapped to the original image producing a "full map". Once this is done for all the cropped images, all the full maps are summed to produce the bounding shapes.

The process of the CPM is more precisely described below:

Given K_{size} and K_{shift} s.t.

$$K_{size} \in \mathbb{R} : K_{size} \in (0, 1)$$

$$K_{shift} \in \mathbb{R} : K_{shift} \in (0, 1)$$

And an Image of MxN dimensions,

$$Im = (im_{xy}) \in \mathbb{Z}^{MxN}$$

Define,

$$Im_{size} = min(N, M)$$

$$a_{0,0} = (0, 0)$$

$$b_{0,0} = (Im_{size} * K_{size}, Im_{size} * K_{size})$$

$$a_{(i,0)} = a_{(i-1,0)} + (Im_{size} * K_{shift}, 0)$$

$$b_{(i,0)} = b_{(i-1,0)} + (Im_{size} * K_{shift}, 0)$$

$$a_{(0,j)} = a_{(0,j-1)} + (0, Im_{size} * K_{shift})$$

$$b_{(0,j)} = b_{(0,j-1)} + (0, Im_{size} * K_{shift})$$

$$a_{(i,j)} = a_{(n-1,j-1)} + (Im_{size} * K_{shift}, Im_{size} * K_{shift})$$

$$b_{(i,j)} = b_{(n-1,j-1)} + (Im_{size} * K_{shift}, Im_{size} * K_{shift})$$

where $i \in [1, ..., I]$, $b_{(I,0)} \leq (M, 0)$, $j \in [1, ..., J]$, and $b_{(0,J)} \leq (0, N)$

Thus, each i, j define a coordinate pair a, b, which in turn define a cropped image,

$$C_{(i,j)} = Im[a_{(i,j)}; b_{(i,j)}]$$

a submatrix of the original image Im.

The notation is defined:

$$Im = (im_{xy}) \in \mathbb{Z}^{MxN} : Im[(a, b); (c, d)] = \begin{bmatrix} im_{a,b} & & im_{a,d} \\ & \ddots & \\ im_{c,b} & & im_{c,d} \end{bmatrix}$$

where $a, c \in Z : a, c \in [0, M]$ and $b, d \in Z : b, d \in [0, N]$

Using an ANN classifier,

$$C_{(i,j)} \rightarrow P_{(i,j)}$$

where $P_{(i,j)} \in \{0, 1\}$

$P_{(i,j)}$ is then used to construct the binary maps, $M_{(i,j)}$, where,

$$M_{(i,j)} = [m_{xy}]$$

$$m_{xy} = P_{(i,j)}$$

$$dim(M_{(i,j)}) = dim(C_{(i,j)})$$

Define a fullmap as,

$$F(i,j) = (f_{xy}) \in \mathbb{Z}^{MxN} : F![a_{(i,j)}; b_{(i,j)}] = 0 \wedge F[a_{(i,j)}; b_{(i,j)}] = M_{(i,j)}$$

And the set of fullmaps as,

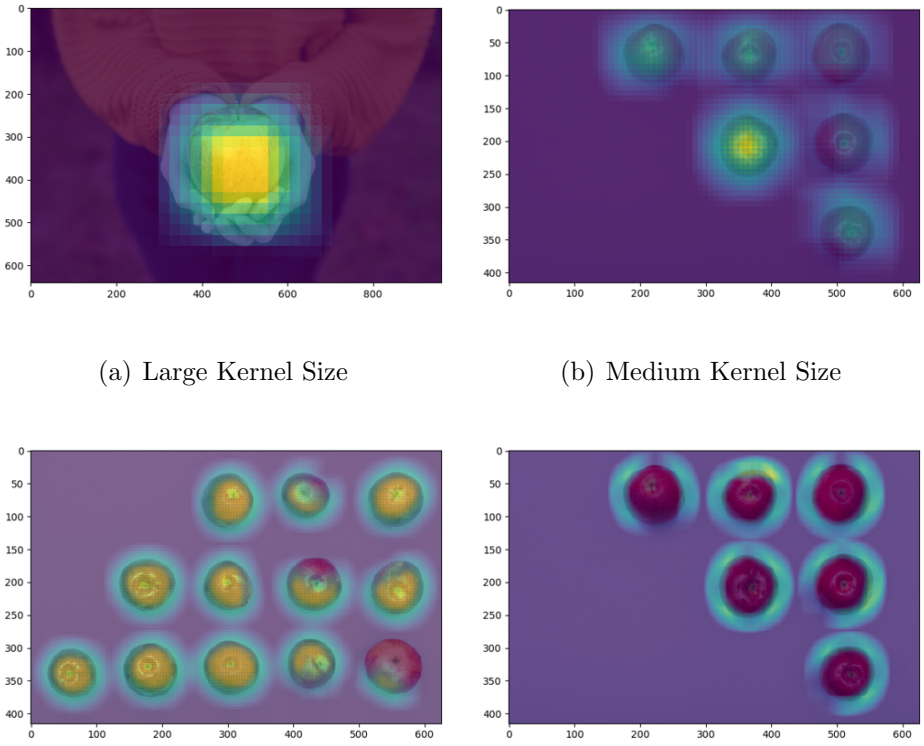
$$S = \{F(i,j) : i \in [0, ..., I], j \in [0, ..., J]\}$$

Finally produce the Sum of fullmaps A,

$$A = \sum S$$

5 Results: Fruit-360

Given the simplicity of our approach, our results look promising. We were able to bound, locate, and identify the shape of objects after applying CPM (see Fig. 2 (a-c)). The process we designed is very quick and does not require large computational resources (after the classifier is trained). We expect that this process can be used to construct bounding boxes, or even bounding shapes for larger object detection algorithms.



(a) Large Kernel Size

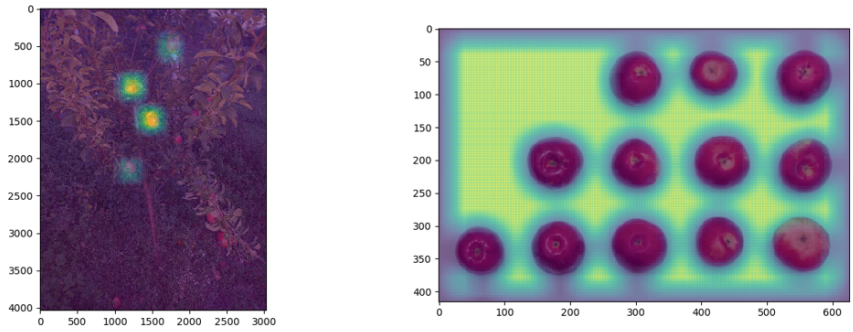
(b) Medium Kernel Size

(c) Small Kernel Size

(d) Kernel Size as Edge Detection

Figure 3: *Several examples of bounding shapes over apples generated using our CPM procedure. Results vary with classifier ability.*

While the results for separated objects were reasonable, we found that our approach had trouble identifying the independence of objects too close together. For example, if two objects are over lapping, while our algorithm can bound both objects, it cannot determine independence between the two objects directly. There may be ways around this (incorporating edge detection methods, using very small kernel sizes, etc.), but that is a topic for future research.



(a) Real-world application (b) Background extraction

Figure 4: *Two examples demonstrating the additional applications of our CPM procedure. These examples also demonstrate some limitations of CPM. In (a) only the unobstructed apples can be identified; in (b) another class was used to obtain the background, demonstrating the need for a null class when training a classifier.*

The kernel size was found to be important and is probably the second largest limiting factor in our approach. As the kernel size decreases, a finer classification of objects can be obtained; however, the smaller the kernel size, the worse a classifier is able to identify an object (as less features are presented to the classifier). Further research is needed to explore how multiple kernel sizes can be used to circumvent this limitation.

Unexpectedly, we found that reducing the kernel size eventually allowed the kernels to directly detect edges (see Fig. 2 (d)). With a better classifier we expect that finer edges can be extracted using the features of the object itself (as opposed to using color contrast, etc.) Of course, this approach is likely limited to object with fairly uniform features, but more experimentation is needed.

We also found that the ability of the classifier to discriminate input images was

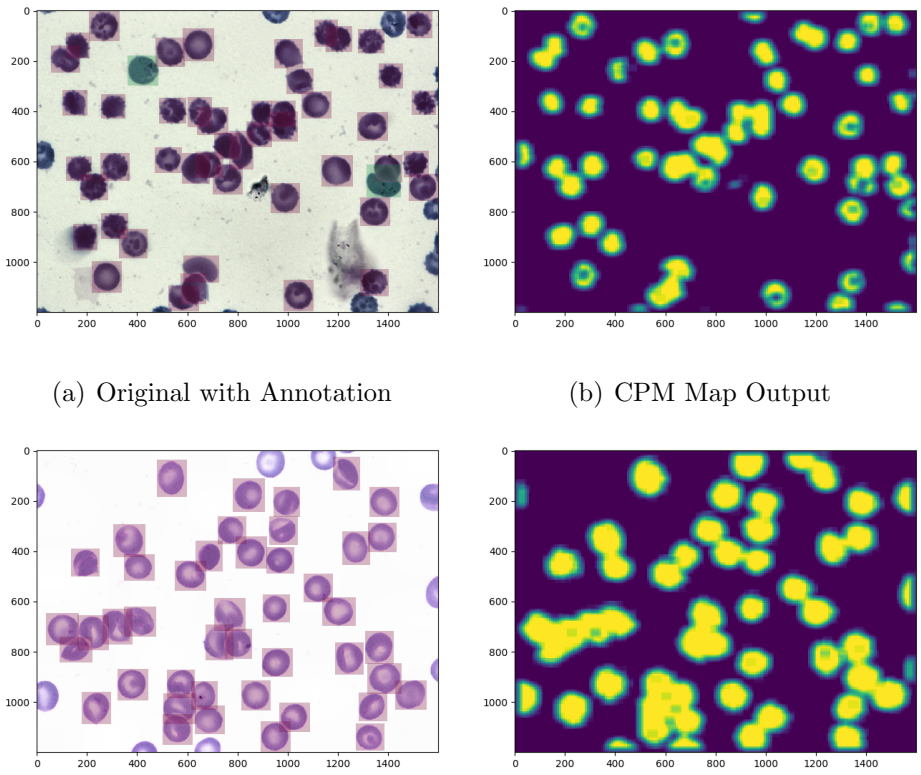
a key factor in being able to successfully apply CPM. If the classifier was not fully functional or trained on poor data, the ability of CPM to identify bounding shapes was reduced. For example, due to the limitations of our classifier sometimes only a partial image could be identified, or the "strength" of identification was low (see Fig. 2 (b-d)); Further, our real-world apple identification example was only able to identify unobstructed apples in the scene (see Fig. 3 (a)).

Finally, the need for a null class was also necessary to allow for a "background" class. If no null class was given, the least activated class would be used directly as the background class (see Fig. 3 (b)). While this could be useful to extract backgrounds, it should be noted the price is a lost class.

6 Results: Cells

Similarly to the fruit-360 results, the results of our CPM procedure provided good preliminary results. We find that the CPM procedure can map out cells fairly accurately when the kernels are well trained, while also ignoring background noise. In addition we find that our procedure, with a strong classifier, is able to bound shapes that were not bounded by the original annotators because they were only partially represented (see Fig. 5 (c-d)).

However, there are some limitaitons. As can be seen in the green bounding boxes of Fig. 5 (a), some portions of these cells are incorrectly mapped in Fig. 5 (b). This was likely due to the small kernel size and our classifier identifying small potions of these cells as RBC. We suspect with further kernel tuning and a higher/ more appropriate F1 score this problem can be addressed. Furthermore, because



(a) Original with Annotation

(b) CPM Map Output

(c) Original with Annotation

(d) CPM Map Output

Figure 5: *Several examples of bounding shapes over apples generated using our CPM procedure. Results vary with classifier ability.*

our classifier did not contain a null class, it classified the background as one of our classes. In our case the background was classified as "Trophozoite".

7 Conclusion

Simple trained neural network classifiers can be used to identify both the location and shape of objects, without the need of bounding boxes or additional annotation

beyond what is required for the classifier. This statement holds only under the assumptions that the neural network classifier is powerful enough and assuming an appropriate kernel size. Furthermore, we find that our CPM process is computationally efficient once training has occurred. Finally for proper classification, it is recommended that a null class is defined.

Future research will consist of using CPM in a CUDA setting to improve image processing speed, using multiple kernels for more robust detection, and combining edge detection methods with our CPM procedure. As for the classifiers, we'd like to explore different forms of image modification, specifically: training classifiers to detect image edges directly and generating obstructed images and partial images. Finally, we think it would be worth exploring recurrent techniques to take into account the relationship between cropped images produced during our CPM technique.

8 References

Horea M., Mihai O. (2018) *Fruit recognition from images using deep learning.* Acta Univ. Sapientiae, Informatica Vol. 10, Issue 1, pp. 26-42.

Oquab, M., Bottou, L., Laptev, I., & Sivic, J. (2015) *Is object localization for free? - Weakly-supervised learning with convolutional neural networks.* IEEE Conference on Computer Vision and Pattern Recognition (CVPR). doi: 10.1109/cvpr.2015.7298668

Papadopoulos, D. P., Uijlings, J. R. R., Keller, F., & Ferrari, V. (2016). *We Don't Need No Bounding-Boxes: Training Object Class Detectors Using Only Human Verification.* IEEE Conference on Computer Vision and Pattern Recog-

nition (CVPR). doi: 10.1109/cvpr.2016.99

Ribera, J., Guera, D., Chen, Y., & Delp, E. J. (2019). *Locating Objects Without Bounding Boxes.* Computer Vision and Pattern Recognition. Retrieved from https://arxiv.org/abs/1806.07564

Ljosa et al. (2012) *Image Set BBBC041v1* Broad Bioimage Benchmark Collection, Nature Methods.

9 External Code Appendix

1. Fruit_360_Classifier_3.py
    - The code used to train the Fruit-360 classifier.
2. Cropping.py
    - The code used to crop the images to be fed into the Fruit-360 classifier.
3. Predicting_crop_class.py
    - The code used to classify the cropped images using the Fruit-360 classifier.
4. Mapping_crop_class.py
    - The code used to generate the full map (summed individual maps) from the Fruit-360 predicted outputs.