COMP3330 Assignment 1c - Object Detection

Ysobel Sims, c3256711
Aidon Jardim, c3256733
Nabil Khan c3256750

June 4, 2019

1 Introduction

Our aim was to be able to detect cans or bottles in an environment in real world images. This object detection system uses a Single-Shot Multibox Detector. This is a fast yet reliable approach to object detection.

For the training dataset, we used a mix of real images and rendered images and automatic classification from blender. This gave us an advantage such that we could create a large training dataset without having to manually classify these rendered images.

2 Dataset

The dataset is made up of around 1,700 rendered images and 300 real images. Rendered images were only used for training, alongside 200 real images. Validation used only real images. Each image has an accompanying XML file with the annotations in the PASCAL VOC format.

The rendered images were generated with Blender. This was done using a scene containing a plastic bottle and two cans. The bottle and cans were randomly placed on the scene. Then, renders would be created from this configuration with changing camera angles. 11 updates to the camera angle occurred for pitch, and then yaw. This created 11x11 renders for one random scene. Using the batch_render script, this could be done for any number of random scenes. Around 1,700 images were generated from a batch of 14 random scenes. It took around a minute to render each image. Examples of these rendered images are below.

For each render, the bounding boxes would be created. The maximum and minimum x and y values of the mesh would be retrieved and from these, the bounding boxes would be added to a JSON file. A converter was then used to translate this JSON file into PASCAL VOC files for each rendered image.

Figure 1: Rendered images used as training data.

Real images were needed to ensure the CNN would be able to detect cans and bottles in real images and not just rendered images. These images were taken on phones and then converted into the same size as the rendered images (600x400). Using Labelbox[3], a collaborative online labelling software, the bounding boxes for these images were created and then exported in PASCAL VOC format.

XnConvert[5], a free batch image processing tool was used to ensure all images were of the same size, and of the same format. All images were processed to be of size 600x400 and be of JPG format.

3 Single-Shot Multibox Detector

Single-Shot Detection(SSD)[1] is a new type of object detection initially developed for usage in real-time image detection systems where previous detection technologies like PASCAL-VOC and COCO would not work for embedded systems let alone real time systems. It was developed and released in 2016. While the accuracy of image detection does not differ greatly from more traditional approaches, the speed at which it produces images with detection boxes is greatly superior.

> "This results in a significant improvement in speed for high-accuracy detection (59 FPS with mAP 74.3% on VOC2007 test, vs. Faster R-CNN 7 FPS with mAP 73.2% or YOLO 45 FPS with mAP 63.4%)" [4]

The basic SDD approach uses the front half of a standard image recognition network, using a feed forward neural network and a collection of bounded boxes with scores for features within them. But SDD has a different end half for the network where it uses a range of smaller feature maps to effectively "zoom in" and detect small items at different ranges. Each feature map has a set of default

bounding boxes where the position of the bounding boxes relative to the feature cells are fixed. We can then predict the offset of the bounding boxes to the feature as well as detect if the object we want to recognise is in the box. During training, we look to find any bounding boxes overlapping with the locations of the desired objects in the training data and overlay as many boxes that pass a certain threshold with the labelled boxes. This means we get a lot of boxes but it also means we get better matches than if we just tried to fit a single box to the labelled box.

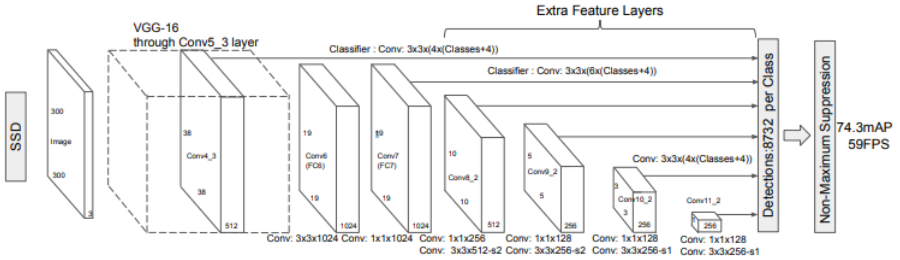

Figure 2: SSD network image taken from [4]

Given this drastic improvement in speed, we decided to work with SSD over more traditional approaches since the time taken for the neural network is already going to be extremely long, given that we have over 1000 images at 600 x 400. SSD is also simpler compared to more traditional approaches, given that we only need to train one network We are working with SSD300 as our choice of SSD implementation.

4 Implementation

For the implementation of our CNN, we used Keras. We chose Keras because it was the quickest way to implement these networks since we already had an example piece of code that implemented an SSD CNN in Keras. We used TensorFlow for the backend, as that was the backend the code was using for Keras.

Our code for training a model can be found in the ssd300_training.ipynb. Follow the instructions in the code to run the training for a model or alternatively, you can just evaluate the model using ssd300_inference.ipynb to use a pre-trained model to perform image recognition on images without the training period.

With regards to loss rate, training was halted after the loss value stopped increasing by a set delta. Once this delta was reached, the code would then run for a few more epochs to make sure this delta would not be surpassed. This is called the patience value. For most of the results, the loss delta was set to 0.001

and the patience value was set to 2. Maximum epochs wasn't reached in our testing, thus making it irrelevant.

5 Results

After training on around 1,700 rendered images and 300 real images, we got the following results.

The first image was retrieved after a 12 epoch training run. The confidence value of these boxes was very small, with boxes only appearing when the confidence value was set to 0.04 or less. We set the bounding box retrieval size to 50. There appears to be some clustering in the centre, with two boxes overlapping one bottle. There is the possibility that this happened only by chance since the other bottles were not detected.

Figure 3: An output from the prediction system.

For this next image, the loss value at the end of the training was 0.0046. This was retrieved after training for 10 epochs, where the loss value decreased by many orders of magnitude between each epoch. The confidence value of each bounding box was the same, this being 0.5. Because of this, we were unable to filter out more than the number of boxes retrieved. In this image, we retrieved the top 200 bounding boxes.

A new training run though was conducted with a lower learning rate schedule of 0.0001. The loss value of this training after 5 epochs was 0.00047. This

Figure 4: Another output from the prediction system.

finished early since the loss value was small. The top 200 bounding boxes were retrieved and all boxes with a confidence value of 0.45 or better were output. All bounding boxes had a confidence of 0.5.

Figure 5: Another output from the prediction system.

We then changed the training parameters to allow the loss value to be made smaller. The minimum delta was set to 0.00001 and patience to 10. This allowed the loss value to be tuned down to 7E-20, which can be seen in figure 6.

epochs	validation accuracy
1	0.0121
2	0.0022
3	4.00E-04
4	6.55E-05
5	9.40E-06
6	1.15E-06
7	1.19E-07
8	9.98E-09
9	6.72E-10
10	3.51E-11
11	1.38E-12
12	3.90E-14
13	7.63E-16
14	9.77E-18
15	7.69E-20

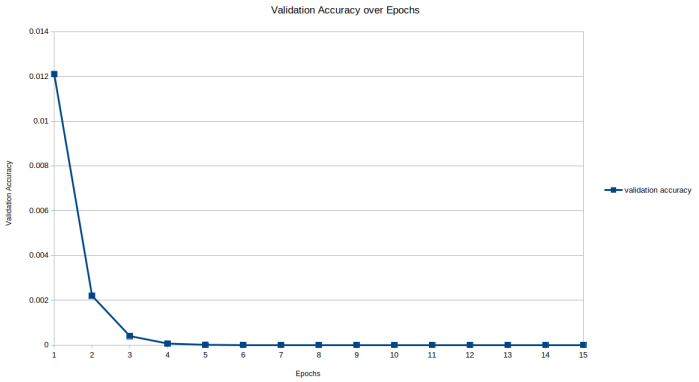

Figure 6: Table and corresponding graph of validation accuracy of the last run.

Even though this value was many orders of magnitude smaller than the previous training results, similar detection results are encountered, as seen in figure 7.

Figure 7: From the last run with low loss value (7E-20)

6 Discussion

Looking at these results, we can see that although we got a low loss value, it didn't properly convert into decent bounding boxes for the images. This could be due to a disconnect with the error function used (SSDError supplied by Keras) or it could be due to an error in adapting the code to work with a singular class 'Plastic Bottle' instead of working with the 20 classes the original SSD model was trained to detect.

From the loss graph, we can see an extremely quick rate of improvement, requiring very few epochs to approach an extremely low loss value. Yet the lack of that converting into good boxes may have been due to not enough testing data being used. In future development, setting aside a larger amount of the data to be used as testing data may have proved beneficial to having the loss value reflect actual progress.

Even though the bounding boxes are highly inaccurate, we do see a large amount of rectangular boxes. This is good because the shape of the object we are trying to detect is vaguely rectangular with only the shorter cans resembling a square. The issue is likely not with the shape of the bounding box.

More time should've been devoted towards testing the boundary box drawings. Although we had an excellent amount of test data, we should've set aside a smaller portion of that data and began work on the neural network, using that data to ensure that the SSD code was working as intended with the system learning how to draw boxes identifying the target objects.

It looks like there is a set of box sizes that the CNN is trying, much like a sliding window implementation of object detection. This is bad for objects that have varying size, like the plastic bottles do in the images.

It also seems that perhaps the loss value computation was not quite tuned for the dataset. Even though the loss value became very good, the results (the bounding boxes) were bad, with them all having a low confidence. Tweaking this function may prove beneficial in acquiring a better, more accurate and more confident result.

In the future, a more up to date object classifier should've been used. This probably would've given a better and more accurate result. Something like YOLO v3[2] would've been advantageous to us.

7 Conclusion

Even though the loss value from the validation dataset was very good (near 0) after a few epochs of training, the output from SSD-300 was not very good.

We have made an extensive training and testing dataset using both rendered and real images, and used PASCAL VOC, a standardised annotation format for object detection.

References

[1] bruceyang2012. https://github.com/bruceyang2012/ssd_keras_7.

[2] YOLO3: A Huge Improvement. https://mc.ai/yolo3-a-huge-improvement/.

[3] labelbox. https://labelbox.com/.

[4] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: single shot multibox detector. *CoRR*, abs/1512.02325, 2015.

[5] XnApps. https://www.xnview.com/en/xnconvert/.