Use of Rotated and Noisy Images to train and test Fast Region-based Convolutional Networks for Object Detection

Ihimu Ukpo

iuu1@columbia.edu

ABSTRACT

Object detection is a fundamental task in computer vision. While some of the first object detection frameworks, such as the Viola Jones framework, have helped to make this task possible, we move past basic object detection and into the realm of object recognition. Deep neural networks have made object recognition a reality, but require hundreds of thousands of images for training and testing in order to be useful. This paper describes a method of increasing the number of images in a small sample set by copying, rotating, and adding random noise to the sample images. Such a method could be particularly useful in training a neural network to recognize an object of which there are few photos.

The purpose of this paper is to discuss the state of the art in object detection and recognition, the process of generating the noisy images and choice of neural network system, results in training and detection, and future work.

INTRODUCTION

One of the prime motivators for object detection is face detection. The ability to detect faces is important not just for aiding cameras in setting autofocus, but also for video surveillance. One of the earliest object detection frameworks was the Viola-Jones framework. This framework provided three key contributions: the introduction of an image representation called the “integral image” which allows the features detected by the detector to be computed quickly, an Adaboost based learning algorithm that selects a number of critical visual features and yields a number of efficient classifiers, and a method for combining classifiers in a “cascade” which allows background regions of the image to be quickly discarded in order to spend more time on detecting object like regions [1] . The “integral image” is similar to the summed area table used in computer graphics for texture mapping; using only a few operations per pixel, it can be computed from an image [1]. The integral image contains Haar-like features that can be computed at any scale or location in the image in constant time, thus making computation quick. However, the number of Haar-like features within any sub-window of an image is very large. Because these features are used to identify objects, their great number makes object classification very time consuming. The second contribution, the Adaboost based learning algorithm, speeds up the classification process. In order to ensure fast classification, the learning process must exclude a large majority of the available features and focus on a small set of critical features-in order to do this, a simple modification is made to the Adaboost procedure: the weak learner is constrained in such a way that each weak classifier returned can only depend on a single feature; this process acts as a kind of feature selection process [1]. The more complex classifiers resulting from this process are then used to construct a cascade structure that dramatically increases the speed of the detector by focusing attention on more promising regions of the image [1].

While the Viola-Jones framework has provided a good basis for object detection, it has a few critical weaknesses. First, in order to

realize a very low false positive rate (10^{-6} or less), hundreds of millions to billions of negative samples would need to be processed during the training procedure; it could take months to build an object detector using the Viola Jones framework at such a false positive rate [2]. The other weakness lies within the Haar features themselves: they have a very limited capacity for representation. This limitation makes object detection difficult in cases of varying illumination and object pose. To compensate for this, researchers have extended HAAR features to include rotated Haar-like features, joint HAAR features and the like. While such features have compensated for this weakness, their inclusion increases the feature space, thus increasing the amount of spaces to search through to detect an object [2]. This increases the amount of time required to detect an object. The SURF cascade framework proposed by Jianguo Li and Yimin Zhang and derived from the Viola Jones framework overcomes those initial weaknesses in three ways. First, the framework adopts multi-dimensional SURF (Speeded Up Robust Features) instead of single dimensional Haar features to describe local patches [2]. Second, it uses logistic regression as a weak classifier instead of an Adaboost derived classifier and decision trees [2]. Finally, in place of false positive rate/hit-rate, the SURF cascade uses AUC (Area Under ROC-curve) as a single criterion [2]. The SURF framework places an emphasis on training efficiency to achieve faster detection results. The SURF is identified in the following manner: first, 'interest points' are selected from distinctive locations in the image such as blobs and T-junctions; The most valuable property of the interest point detector is repeatability, i.e., reliably finding the same interest points under different viewing conditions [3]. Then the neighborhood of every interest point is represented by a feature vector that must be distinctive and robust against noise [3]. The resulting vectors are then matched between different images; the matching is often distance based [3].

Other kinds of object detection focus on deformable models and aim to detect an object when it is deformed; this is the method employed by Object detection with Discriminatively Trained Part Based Modeling (DTPBM) to detect objects. This approach builds on a pictorial structure that represents objects by a collection of parts arranged in a deformable configuration; each such configuration is characterized by spring-like connections between certain pairs of parts [4]. The advantage of object detection with deformable models is that such models provide a natural framework for sharing information and computation between different object classes [4]; different models could share reusable parts, thus reducing the number of parts needed to detect an object.

While the object detection frameworks mentioned work, these frameworks rely on "shallow" models; the models are shallow in the sense that they require some prior knowledge of specific features of the image to detect an object. Moreover, the models must be sometimes hand designed to capture parts and their relations explicitly [5], particularly in the case of DTPBM. Deep neural networks can do object detection using "deeper" (more complex) models. Deep neural networks do not require creating models from hand and are capable of learning features which are good for classification; Moreover, such neural networks are also capable of capturing strong geometric information as well [5]. In implementation, DNNs are massively parallel structures built out of thousands to millions of identical units which perform the same computation on different data [6]. Most of these units have no data interdependency, and thus they can be computed simultaneously [6]. Because of this, neural networks are particularly suited to implementation by Graphics Processing Units (GPUs).

Experiment Design

Several deep neural network based object detection frameworks were considered for this project. The first such framework was YOLO (You Only Look Once). YOLO works by dividing the input image into an SxS grid. If the center of an object falls into a grid cell, that grid cell is responsible for object detection [7]. Object detection with this framework is framed as a single regression problem that spans from image pixels to bounding box coordinates and class probabilities [7]. Object detection speed is an advantage with the YOLO framework; YOLO is capable of performing real-time object detection at a rate of forty five frames per second. YOLO was not selected for this project because of its accuracy; While it can quickly identify objects in images, it struggles to find objects in images with unusual aspect ratios and has trouble in precisely localizing objects [7]. Another neural network framework under consideration was Fast RCNN (Region based Convolutional Neural Networks). Fast RCNN streamlines the neural network training process for convolutional neural network based object detectors by using a single stage training algorithm that jointly learns to classify object proposals and refine their spatial locations [8]. Fast RCNN was superseded by Faster RCNN, which sped up the process of object detection by using a Region Proposal Network (RPN) that shares full-image convolutional features with the detection network. Fast RCNN was chosen for this project because it provided the best balance of speed and accuracy among the examined neural network based object detection frameworks [9]. Faster RCNN was not chosen because of greater system requirements than Fast RCNN and incompatibility with the visual search engine application framework.

Sixty eight images of three different objects-a shoe, a mobile phone, and a perfume box-were captured for the training set, bringing the total training set size to 204 images. The testing images were created by copying the training images and applying random noise to the images. Please see figures one and two for an example of a captured image and its noisy clone.

Figure 1: Training image.

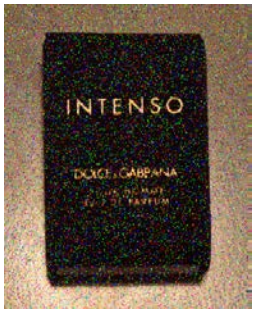

Figure 2: Testing image. This image is a copy of the training image in figure one, but with noise added to make it distinct.

Results

Training loss convergence occurred quickly for object detection, but fluctuated for the region (bounding box) where the object was detected (see figure 3). Convergence occurs when the neural network is no longer learning anything. When tested with the resulting CAFFE object detection model was loaded into the Visual Search Engine application, the application yielded many false positives with high probabilities (see figure 4). To ensure that the application was not at fault, the model built was replaced by the CAFFE model used for

object detection in the Android Object Detection GitHub project [10]. The application was able to perform object detection correctly with this model (see figure 5)

There were two possible problems with this approach. The first problem is the similarity of the testing images to the training ones. Despite the noise, the testing images are nearly identical to their training counterparts. While this allows a neural network to quickly learn, the neural network does not learn much.

The second problem was that there were no negative images-images not containing the object to be detected-included in training. Each image used for training or testing must have an annotation file in the Fast RCNN framework; each file needs to specify a class number and coordinates for the bounding box describing the object's location within the image. It is unclear how to specify the annotation for a negative image.

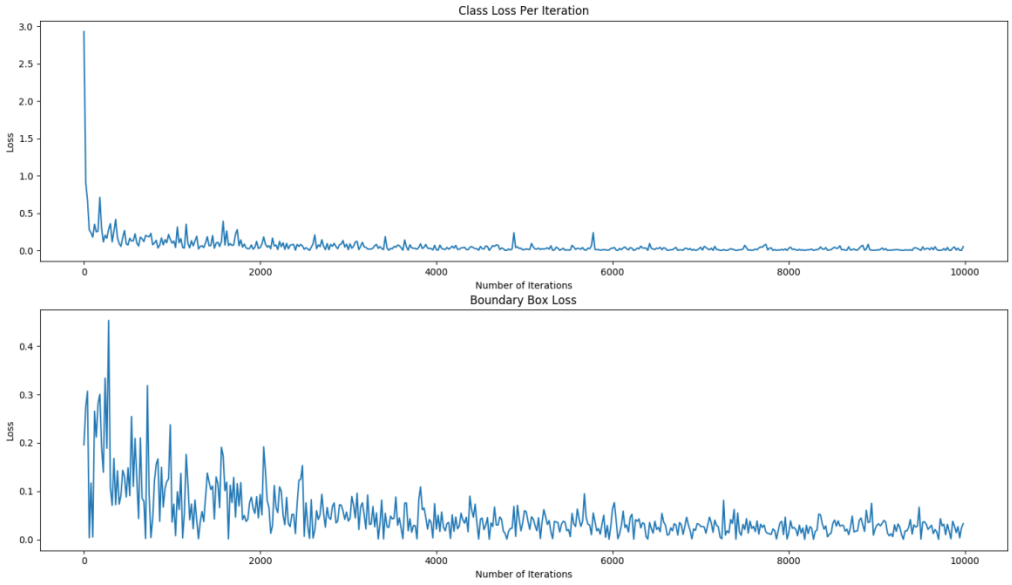

Figure 3: Training loss per number of training iterations.

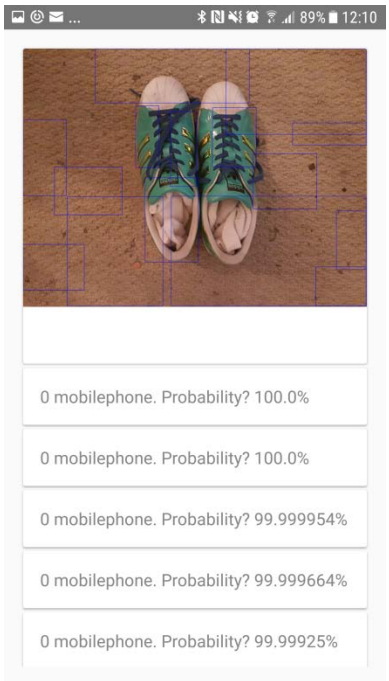

Figure 4: Object detection with created image model. Rectangles are drawn around detected objects; not only are there many rectangles not containing any objects trained, but a high probability that the falsely detected object is within the rectangle is given.

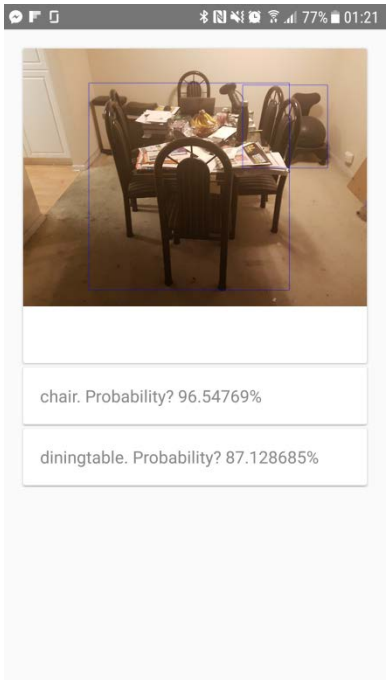

Figure 5: Correct object detection with the object detection model used with the Android Object Detection application.

Conclusion and Future Work

This method of generating images for testing and training a neural network is unusable in its current form. Despite learning quickly, the neural network does not learn much. Moreover, due to issues with specifying negative images, the learning is incomplete.

A future direction away from this work involves copying object pixel data from an image and pasting it into other images not containing the object. The images that the object will be pasted into can be created from random noise to ensure that no prior instance of the object is present in the receiving image. A means of specifying annotations for negative images will have to be determined as well if this new direction is to be implemented with Fast or Faster RCNN.

Bibliography

[1] P. Viola and M. Jones, "Robust real-time object detection," *International Journal of Computer Vision,* vol. 4, pp. 43-47, 2001.

[2] J. Li and Y. Zhang, "Learning surf cascade for fast and accurate object detection.," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[3] H. Bay, T. Tuytelaars and L. Van Gool, "SURF: Speeded Up Robust Features," *Computer vision–ECCV,* pp. 404-417, 2006.

[4] P. F. Felzenszwalb, "Object detection with discriminatively trained part-based models.," *IEEE transactions on pattern analysis and machine intelligence,* vol. 32, no. 9, pp. 1627-1645, 2010.

[5] C. Szegedy, A. Toshev and D. Erhan, "Deep neural networks for object detection.," *Advances in Neural Information Processing Systems,* 2013.

[6] Nvidia, "CUDA Spotlight: GPU-Accelerated Deep Neural Networks," 30 April 2014. [Online]. Available: https://devblogs.nvidia.com/parallelforall/cuda-spotlight-gpu-accelerated-deep-neural-networks/.

[7] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[8] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015.

[9] S. Ren, K. He, R. Girshick and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," 2015.

[10] T. Lin, "Fast-RCNN and Scene Recognition using Caffe," [Online]. Available: https://github.com/tzutalin/Android-Object-Detection.