Deep Learning Based Single Stage Detector for On Device
Real-time Object Detection

Huy Q. Le¹, Hieu N. Thai², Kyi Thar¹ and Choong Seon Hong¹
¹Department of Computer Science and Engineering, Kyung Hee University
²SMARTEC CO., LTD, Binh Duong, Vietnam
Email: quanghuy69@khu.ac.kr, hieuthai642@gmail.com, kyithar@khu.ac.kr

Abstract
Object detection has become an integral part of computer vision used for vehicle recognition, surveillance, etc. Nowadays, there are many popular approaches to detect objects. Among them, two-stage detectors provide the best performances in detection but they are not suited to real-time applications. On the contrary, single-stage detectors do not give a good performance as the two-stage detectors but the detection speed is considerably faster than the two-stage. Thus, single-stage detectors are more suitable for practical applications that need a real-time solution. In this paper, we implement single-stage detectors with You only look once (YOLOv3) algorithm to achieve the real-time speed in object recognition on our collected datasets for evaluation.

1. Introduction
Object detection is a one of the success applications of the classification and localization problem. The technique being frequently used is to design an algorithm which can locate and classify objects correctly in rectangle boxes while avoiding false positive detections of background or multiple bounding boxes surrounding of the same object. Bounding box is a rectangular structure overlapped over an image containing all important features of a particular object. This image annotation technique decreases the scope of search for the object attributes and resources loss for the computer.

Object detection techniques has come a long way and are currently of a standard where they can be used in restrict scenarios. Current object detectors can be divided into two types: double-stage detectors and single-stage detectors. Double stage Detectors, as known as region based object detectors, have many popular approaches such as Faster-RCNN[1], R-CFN[2] that give the best performances in COCO dataset [3] detection but the overall speed is low. On the other hand, single-stage detectors, as known as single shot detectors, generally do not produce a good performance as double-stage detectors but the speed is higher.

In this paper, with the requirements of real-time processing, we proposed single stage detectors to be our object detection algorithm among double stage. This object detector uses an efficient sliding-windows approach compared to the “traditional” ones, which would literally slide different size windows over and over an image to match different objects. Instead, single stage detectors treat the sliding window as an initial guess, usually called anchors (anchors are fixed boundary box guesses).

The most popular approaches in this category are Single shot multibox detector (SSD) [4] and You only look once (YOLO) [5]. The learning accuracy is usually trade-off with real-time processing speed in single stage detectors. Too close or small objects might not be detected. Many further improvements have been added to YOLO and SSD. Later versions of YOLO method applied many small changes such as applying batch normalization, clustering box dimension, location prediction, multi-scale training to improve performance.

We can summarize our contributions in this paper as follows:
• YOLOv3 algorithm is deployed on two of most popular framework for Deep Learning: Tensorflow (from Google) and MXNet (from Amazon) to compare performance.
• The system is trained with YOLOv3 algorithm on collected datasets.
• We implement the real-time processing for our object detector algorithm on raspberry pi.

2. System Model
The proposed model is shown on Fig. 1. YOLOv3 is
deployed on two of the most popular Python framework for Deep Learning: Tensorflow (from Google) and MXNet (from Amazon) to compare performance. The training process was conducted on NVIDIA GTX 1050Ti on the computer. The training model is deployed on embedded computer (raspberry pi) after running on the GPU.

3. Single stage detectors with YOLOv3 to predict the bounding box

3.1. Network Architecture

**Backbones:** YOLOv3 uses a custom network, named Darknet–53, which has 53 convolutional layers. Darknet–53 was stacked state-of-the-art residual blocks. According to the authors, this network is much more powerful than Darknet–19 (used in YOLOv2) but still more efficient than RestNet–101 or RestNet–152. [6]

**Detection Layers:** 53 more layers are stacked after Darknet–53, which makes a 106 fully convolutional layers for YOLO v3. In the YOLO v3 architecture, 53 more layers include skip connections and up sampling blocks. [7]

3.2. Bounding Box Prediction

YOLOv3 predicts bounding boxes using anchor boxes. Anchors are initial sizes (width, height) some of which (the closet to the object size) will be resized to the object size – using some outputs from the feature extractor. As Fig. 2 shows, the input image is divided into and SxS grid of cells. Each grid cell predicts N bounding boxes as well as M class probability.

The predictions would correspond to:

\[
\begin{align*}
    b_x &= \sigma(t_x) + c_x \\
    b_y &= \sigma(t_y) + c_y \\
    b_w &= p_1 e^{t_w} \\
    b_h &= p_2 e^{t_h}
\end{align*}
\]

Where:

- \((b_x, b_y, b_w, b_h)\) are the bounding box center–x coordinate, center–y coordinate, width & height respectively.
- \((t_x, t_y, t_w, t_h)\) are predicted coordinates for each bounding box.
- \((c_x, c_y)\) is the top left corner of grid cell of the anchor.
- \((p_1, p_2)\) are anchors dimensions for the box.

3.3. Loss function

The binary cross–entropy loss

\[
H_p(q) = -\frac{1}{N} \sum_{i=1}^{N} y_i \cdot \log(p(y_i)) + (1 - y_i) \cdot \log(1 - p(y_i))
\]

Where:

- \(y_i = 1\) if input belong to class \(i\), \(y_i = 0\) if input does not belong to class \(i\). \(p(y_i)\) is the probability of the point belong to class \(i\).
4. Performance Evaluations
To perform the experimental part of our study, we trained on the custom datasets with 70 images, 1 class, learning rate $= 0.001$, optimization = ADAM. Each image contains $10-13$ objects.

As Fig. 3 shows, loss decreases after iterations, which is a good sign. At first, loss decreases fast (from around 2000 to 5 in first 800 iterations), then it struggles to pass the $800-1000$ iterations. After 1000 iterations, loss decreases slowly. The training process can be stop here.

As Fig. 4 shows, the object detector can perform well on testing set. Large objects are all detected, meanwhile small objects are detected to some extent. YOLOv3 can achieve such performance because it raises many bounding boxes during detection process. YOLOv2 only gives $13\times13\times5 = 845$ boxes, meanwhile YOLOv3 gives $13\times13\times3 + 26\times26\times3 + 52\times52\times3 = 10647$ boxes, over $10$ times higher than YOLOv2. This explains improved performance all YOLOv3 compared to YOLOv2.

Table 1 records the FPS of YOLOv3 running on the same video with different framework: Tensorflow 1.12, MXNet 1.3.1 and DarkNet – a framework written by authors in C and CUDA C. On NDVIA GTX 1050 Ti, DarkNet outperforms all other frameworks, achieve an incredibly high FPS. MXNet scores just a little higher than Tensorflow. However, because MXNet provides and executes with HybridBlock.hybridize(), the GPU would not have to work as much as Tensorflow, in return, MXNET requires considerable memory compared to Tensorflow and DarkNet.

Table 1: Performance on different framework on GPU

<table>
<thead>
<tr>
<th>Framework</th>
<th>OS</th>
<th>Hardware</th>
<th>FPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tensorflow</td>
<td>Ubuntu 18.04</td>
<td>GTX 1050 Ti</td>
<td>8.5 FPS</td>
</tr>
<tr>
<td>MXNet</td>
<td>Ubuntu 18.04</td>
<td>GTX 1050 Ti</td>
<td>10.2 FPS</td>
</tr>
<tr>
<td>DarkNet</td>
<td>Ubuntu 18.04</td>
<td>GTX 1050 Ti</td>
<td>18.1 FPS</td>
</tr>
</tbody>
</table>

5. Conclusion
In this paper, we implemented YOLOv3 algorithm and trained it on our collected datasets. Also, we conducted with different framework to find the most effective model for improving the real-time speed when running on the video. The training results show that all the objects are detected except too small or unclear objects. In future works, we will continue to develop the new algorithms to improve the speed for real-time processing and the accuracy.

Acknowledgement
This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the Grand Information Technology Research Center support program (IITP-2020-2015-0-00742) supervised by the IITP (Institute for Information & communications Technology Planning & Evaluation). Dr. CS Hong is the corresponding author.

References