

# Tiny SSD: A Tiny Single-shot Detection Deep Convolutional Neural Network for Real-time Embedded Object Detection

Alexander Wong, Mohammad Javad Shafiee, Francis Li, Brendan Chwyl  
*Dept. of Systems Design Engineering*  
*University of Waterloo, DarwinAI*  
 {a28wong, mjshafiee}@uwaterloo.ca, {francis, brendan}@darwinai.ca

**Abstract**—Object detection is a major challenge in computer vision, involving both object classification and object localization within a scene. While deep neural networks have been shown in recent years to yield very powerful techniques for tackling the challenge of object detection, one of the biggest challenges with enabling such object detection networks for widespread deployment on embedded devices is high computational and memory requirements. Recently, there has been an increasing focus in exploring small deep neural network architectures for object detection that are more suitable for embedded devices, such as Tiny YOLO and SqueezeDet. Inspired by the efficiency of the Fire microarchitecture introduced in SqueezeNet and the object detection performance of the single-shot detection macroarchitecture introduced in SSD, this paper introduces Tiny SSD, a single-shot detection deep convolutional neural network for real-time embedded object detection that is composed of a highly optimized, non-uniform Fire sub-network stack and a non-uniform sub-network stack of highly optimized SSD-based auxiliary convolutional feature layers designed specifically to minimize model size while maintaining object detection performance. The resulting Tiny SSD possess a model size of 2.3MB ( $\sim 26\times$  smaller than Tiny YOLO) while still achieving an mAP of 61.3% on VOC 2007 ( $\sim 4.2\%$  higher than Tiny YOLO). These experimental results show that very small deep neural network architectures can be designed for real-time object detection that are well-suited for embedded scenarios.

**Keywords**-object detection; deep neural network; embedded; real-time; single-shot

## I. INTRODUCTION

Object detection can be considered a major challenge in computer vision, as it involves a combination of object classification and object localization within a scene (see Figure 1). The advent of modern advances in deep learning [7], [6] has led to significant advances in object detection, with the majority of research focuses on designing increasingly more complex object detection networks for improved accuracy such as SSD [9], R-CNN [1], Mask R-CNN [2], and other extended variants of these networks [4], [8], [15]. Despite the fact that such object detection networks have showed state-of-the-art object detection accuracies beyond what can be achieved by previous state-of-the-art methods, such networks are often intractable for use for embedded applications due to computational and memory constraints. In fact, even faster variants of these networks such as Faster R-CNN [13] are only



Figure 1. Tiny SSD results on the VOC test set. The bounding boxes, categories, and confidences are shown.

capable of single-digit frame rates on a high-end graphics processing unit (GPU). As such, more efficient deep neural networks for real-time embedded object detection is highly desired given the large number of operational scenarios that such networks would enable, ranging from smartphones to aerial drones.

Recently, there has been an increasing focus in exploring small deep neural network architectures for object detection that are more suitable for embedded devices. For example, Redmon et al. introduced YOLO [11] and YOLOv2 [12], which were designed with speed in mind and was able to achieve real-time object detection performance on a high-end Nvidia Titan X desktop GPU. However, the model size of YOLO and YOLOv2 remains very large in size (753 MB and 193 MB, respectively), making them too large from a memory perspective for most embedded devices. Furthermore, their object detection speed drops considerably when running on embedded chips [14]. To address this issue, Tiny YOLO [10] was introduced where the network architecture was reduced considerably to greatly reduce model size (60.5 MB) as well as greatly reduce the number of floating point operations required (just 6.97 billion operations) at a cost of object detection accuracy (57.1% on the twenty-category VOC 2017 test set). Similarly, Wu et al. introduced SqueezeDet [16], a fully convolutional neural network that leveraged the efficient Fire microarchitecture introduced in SqueezeNet [5] within an end-to-end object detection network architecture. Given that the Fire microarchitecture is highly efficient, the resulting SqueezeDet had a reduced model size specifically for the

purpose of autonomous driving. However, SqueezeDet has only been demonstrated for objection detection with limited object categories (only three) and thus its ability to handle larger number of categories have not been demonstrated. As such, the design of highly efficient deep neural network architectures that are well-suited for real-time embedded object detection while achieving improved object detection accuracy on a variety of object categories is still a challenge worth tackling.

In an effort to achieve a fine balance between object detection accuracy and real-time embedded requirements (i.e., small model size and real-time embedded inference speed), we take inspiration by both the incredible efficiency of the Fire microarchitecture introduced in SqueezeNet [5] and the powerful object detection performance demonstrated by the single-shot detection macroarchitecture introduced in SSD [9]. The resulting network architecture achieved in this paper is **Tiny SSD**, a single-shot detection deep convolutional neural network designed specifically for real-time embedded object detection. Tiny SSD is composed of a non-uniform highly optimized Fire sub-network stack, which feeds into a non-uniform sub-network stack of highly optimized SSD-based auxiliary convolutional feature layers, designed specifically to minimize model size while retaining object detection performance.

This paper is organized as follows. Section 2 describes the highly optimized Fire sub-network stack leveraged in the Tiny SSD network architecture. Section 3 describes the highly optimized sub-network stack of SSD-based convolutional feature layers used in the Tiny SSD network architecture. Section 4 presents experimental results that evaluate the efficacy of Tiny SSD for real-time embedded object detection. Finally, conclusions are drawn in Section 5.

## II. OPTIMIZED FIRE SUB-NETWORK STACK

The overall network architecture of the Tiny SSD network for real-time embedded object detection is composed of two main sub-network stacks: i) a non-uniform Fire sub-network stack, and ii) a non-uniform sub-network stack of highly optimized SSD-based auxiliary convolutional feature layers, with the first sub-network stack feeding into the second sub-network stack. In this section, let us first discuss in detail the design philosophy behind the first sub-network stack of the Tiny SSD network architecture: the optimized fire sub-network stack.

A powerful approach to designing smaller deep neural network architectures for embedded inference is to take a more principled approach and leverage architectural design strategies to achieve more efficient deep neural network microarchitectures [3], [5]. A very illustrative example of such a principled approach is the SqueezeNet [5] network architecture, where three key design strategies were leveraged:

- 1) reduce the number of  $3 \times 3$  filters as much as possible,

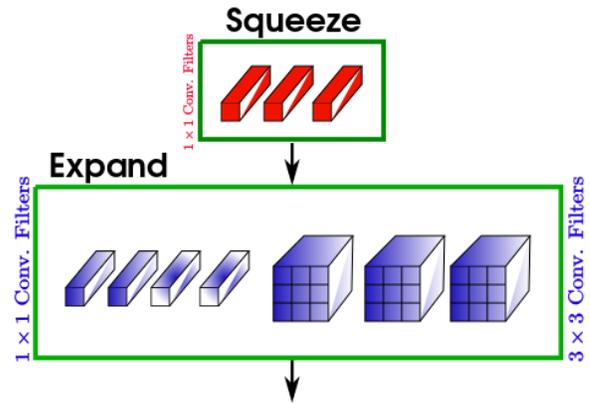


Figure 2. An illustration of the Fire microarchitecture. The output of previous layer is squeezed by a *squeeze* convolutional layer of  $1 \times 1$  filters, which reduces the number of input channels to  $3 \times 3$  filters. The result of the *squeeze* convolutional layers is passed into the *expand* convolutional layer which consists of both  $1 \times 1$  and  $3 \times 3$  filters.

- 2) reduce the number of input channels to  $3 \times 3$  filters where possible, and
- 3) perform downsampling at a later stage in the network.

This principled designed strategy led to the design of what the authors referred to as the **Fire** module, which consists of a *squeeze* convolutional layer of  $1 \times 1$  filters (which realizes the second design strategy of effectively reduces the number of input channels to  $3 \times 3$  filters) that feeds into an *expand* convolutional layer comprised of both  $1 \times 1$  filters and  $3 \times 3$  filters (which realizes the first design strategy of effectively reducing the number of  $3 \times 3$  filters). An illustration of the Fire microarchitecture is shown in Figure 2.

Inspired by the elegance and simplicity of the Fire microarchitecture design, we design the first sub-network stack of the Tiny SSD network architecture as a standard convolutional layer followed by a set of highly optimized Fire modules. One of the key challenges to designing this sub-network stack is to determine the ideal number of Fire modules as well as the ideal microarchitecture of each of the Fire modules to achieve a fine balance between object detection performance and model size as well as inference speed. First, it was determined empirically that 10 Fire modules in the optimized Fire sub-network stack provided strong object detection performance. In terms of the ideal microarchitecture, the key design parameters of the Fire microarchitecture are the number of filters of each size ( $1 \times 1$  or  $3 \times 3$ ) that form this microarchitecture. In the SqueezeNet network architecture that first introduced the Fire microarchitecture [5], the microarchitectures of the Fire modules are largely uniform, with many of the modules sharing the same microarchitecture configuration. In an effort to achieve more optimized Fire microarchitectures on a per-module basis, the number of filters of each size in each Fire

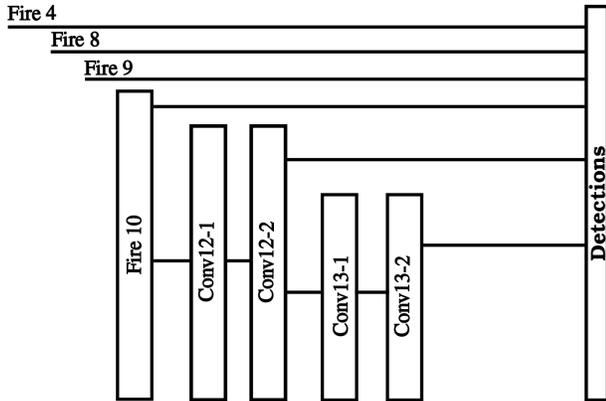


Figure 3. An illustration of the network architecture of the second sub-network stack of Tiny SSD. The output of three Fire modules and two auxiliary convolutional feature layers, all with highly optimized microarchitecture configurations, are combined together for object detection.

module is optimized to have as few parameters as possible while still maintaining the overall object detection accuracy. As a result, the optimized Fire sub-network stack in the Tiny SSD network architecture is highly non-uniform in nature for an optimal sub-network architecture configuration. Table I shows the overall architecture of the highly optimized Fire sub-network stack in Tiny SSD, and the number of parameters in each layer of the sub-network stack.

### III. OPTIMIZED SUB-NETWORK STACK OF SSD-BASED CONVOLUTIONAL FEATURE LAYERS

In this section, let us first discuss in detail the design philosophy behind the second sub-network stack of the Tiny SSD network architecture: the sub-network stack of highly optimized SSD-based auxiliary convolutional feature layers.

One of the most widely-used and effective object detection network macroarchitectures in recent years has been the single-shot multibox detection (SSD) macroarchitecture [9]. The SSD macroarchitecture augments a base feature extraction network architecture with a set of auxiliary convolutional feature layers and convolutional predictors. The auxiliary convolutional feature layers are designed such that they decrease in size in a progressive manner, thus enabling the flexibility of detecting objects within a scene across different scales. Each of the auxiliary convolutional feature layers can then be leveraged to obtain either: i) a confidence score for a object category, or ii) a shape offset relative to default bounding box coordinates [9]. As a result, a number of object detections can be obtained per object category in this manner in a powerful, end-to-end single-shot manner.

Inspired by the powerful object detection performance and multi-scale flexibility of the SSD macroarchitecture [9], the second sub-network stack of Tiny SSD is comprised of a set of auxiliary convolutional feature layers and convo-

Table I  
THE OPTIMIZED FIRE SUB-NETWORK STACK OF THE TINY SSD NETWORK ARCHITECTURE. THE NUMBER OF FILTERS AND INPUT SIZE TO EACH LAYER ARE REPORTED FOR THE CONVOLUTIONAL LAYERS AND FIRE MODULES. EACH FIRE MODULE IS REPORTED IN ONE ROW FOR A BETTER REPRESENTATION. " $x@S - y@E1 - z@E3$ " STANDS FOR  $x$  NUMBERS OF  $1 \times 1$  FILTERS IN THE SQUEEZE CONVOLUTIONAL LAYER,  $y$  NUMBERS OF  $1 \times 1$  FILTERS AND  $z$  NUMBERS OF  $3 \times 3$  FILTERS IN THE EXPAND CONVOLUTIONAL LAYER.

Type / Stride	Filter Shapes	Input Size
Conv1 / s2	$3 \times 3 \times 57$	$300 \times 300$
Pool1 / s2	$3 \times 3$	$149 \times 149$
Fire1	$15@S - 49@E1 - 53@E3$	$74 \times 74$
Concat1		
Fire2	$15@S - 54@E1 - 52@E3$	$74 \times 74$
Concat2		
Pool3 / s2	$3 \times 3$	$74 \times 74$
Fire3	$29@S - 92@E1 - 94@E3$	$37 \times 37$
Concat3		
Fire4	$29@S - 90@E1 - 83@E3$	$37 \times 37$
Concat4		
Pool5 / s2	$3 \times 3$	$37 \times 37$
Fire5	$44@S - 166@E1 - 161@E3$	$18 \times 18$
Concat5		
Fire6	$45@S - 155@E1 - 146@E3$	$18 \times 18$
Concat6		
Fire7	$49@S - 163@E1 - 171@E3$	$18 \times 18$
Concat7		
Fire8	$25@S - 29@E1 - 54@E3$	$18 \times 18$
Concat8		
Pool9 / s2	$3 \times 3$	$18 \times 18$
Fire 9	$37@S - 45@E1 - 56@E3$	$9 \times 9$
Concat9		
Pool10 / s2	$3 \times 3$	
Fire10	$38@S - 41@E1 - 44@E3$	$4 \times 4$
Concat10		

lutional predictors with highly optimized microarchitecture configurations (see Figure 3).

As with the Fire microarchitecture, a key challenge to designing this sub-network stack is to determine the ideal microarchitecture of each of the auxiliary convolutional feature layers and convolutional predictors to achieve a fine balance between object detection performance and model size as well as inference speed. The key design parameters of the auxiliary convolutional feature layer microarchitecture are the number of filters that form this microarchitecture. As such, similar to the strategy taken for constructing the highly optimized Fire sub-network stack, the number of filters in each auxiliary convolutional feature layer is optimized to minimize the number of parameters while preserving overall object detection accuracy of the full Tiny SSD network. As a result, the optimized sub-network stack of auxiliary convolutional feature layers in the Tiny SSD network architecture is highly non-uniform in nature for an optimal sub-network architecture configuration. Table II shows the overall architecture of the optimized sub-network stack of the auxiliary convolutional feature layers within the Tiny SSD network architecture, along with the number of

Table II  
THE OPTIMIZED SUB-NETWORK STACK OF THE AUXILIARY CONVOLUTIONAL FEATURE LAYERS WITHIN THE TINY SSD NETWORK ARCHITECTURE. THE INPUT SIZES TO EACH CONVOLUTIONAL LAYER AND KERNEL SIZES ARE REPORTED.

Type / Stride	Filter Shape	Input Size
Conv12-1 / s2	$3 \times 3 \times 51$	$4 \times 4$
Conv12-2	$3 \times 3 \times 46$	$4 \times 4$
Conv13-1	$3 \times 3 \times 55$	$2 \times 2$
Conv13-2	$3 \times 3 \times 85$	$2 \times 2$
Fire5-mbox-loc	$3 \times 3 \times 16$	$37 \times 37$
Fire5-mbox-conf	$3 \times 3 \times 84$	$37 \times 37$
Fire9-mbox-loc	$3 \times 3 \times 24$	$18 \times 18$
Fire9-mbox-conf	$3 \times 3 \times 126$	$18 \times 18$
Fire10-mbox-loc	$3 \times 3 \times 24$	$9 \times 9$
Fire10-mbox-conf	$3 \times 3 \times 126$	$9 \times 9$
Fire11-mbox-loc	$3 \times 3 \times 24$	$4 \times 4$
Fire11-mbox-conf	$3 \times 3 \times 126$	$4 \times 4$
Conv12-2-mbox-loc	$3 \times 3 \times 24$	$2 \times 2$
Conv12-2-mbox-conf	$3 \times 3 \times 126$	$2 \times 2$
Conv13-2-mbox-loc	$3 \times 3 \times 16$	$1 \times 1$
Conv13-2-mbox-conf	$3 \times 3 \times 84$	$1 \times 1$

parameters in each layer.

Table III  
OBJECT DETECTION ACCURACY RESULTS OF TINY SSD ON VOC 2007 TEST SET. TINY YOLO RESULTS ARE PROVIDED AS A BASELINE COMPARISON.

Model Name	Model size	mAP (VOC 2007)
Tiny YOLO [10]	60.5MB	57.1%
Tiny SSD	2.3MB	61.3%

Table IV  
RESOURCE USAGE OF TINY SSD.

Model Name	Total number of Parameters	Total number of MACs
Tiny SSD	1.13M	571.09M

#### IV. PARAMETER PRECISION OPTIMIZATION

In this section, let us discuss the parameter precision optimization strategy for Tiny SSD. For embedded scenarios where the computational requirements and memory requirements are more strict, an effective strategy for reducing computational and memory footprint of deep neural networks is reducing the data precision of parameters in a deep neural network. In particular, modern CPUs and GPUs have moved towards accelerated mixed precision operations as well as better handling of reduced parameter precision, and thus the ability to take advantage of these factors can yield noticeable improvements for embedded scenarios. For Tiny SSD, the parameters are represented in half precision floating-point, thus leading to further deep neural network model size

reductions while having a negligible effect on object detection accuracy.

#### V. EXPERIMENTAL RESULTS AND DISCUSSION

To study the utility of Tiny SSD for real-time embedded object detection, we examine the model size, object detection accuracies, and computational operations on the VOC2007/2012 datasets. For evaluation purposes, the Tiny YOLO network [10] was used as a baseline reference comparison given its popularity for embedded object detection, and was also demonstrated to possess one of the smallest model sizes in literature for object detection on the VOC 2007/2012 datasets (only 60.5MB in size and requiring just 6.97 billion operations). The VOC2007/2012 datasets consist of natural images that have been annotated with 20 different types of objects, with illustrative examples shown in Figure 4. The tested deep neural networks were trained using the VOC2007/2012 training datasets, and the mean average precision (mAP) was computed on the VOC2007 test dataset to evaluate the object detection accuracy of the deep neural networks.

##### A. Training Setup

The proposed Tiny SSD network was trained for 220,000 iterations in the Caffe framework with training batch size of 24. RMSProp was utilized as the training policy with base learning rate set to 0.00001 and  $\gamma = 0.5$ .

##### B. Discussion

Table III shows the model size and the object detection accuracy of the proposed Tiny SSD network on the VOC 2007 test dataset, along with the model size and the object detection accuracy of Tiny YOLO. A number of interesting observations can be made. First, the resulting Tiny SSD possesses a model size of 2.3MB, which is  $\sim 26X$  smaller than Tiny YOLO. The significantly smaller model size of Tiny SSD compared to Tiny YOLO illustrates its efficacy for greatly reducing the memory requirements for leveraging Tiny SSD for real-time embedded object detection purposes. Second, it can be observed that the resulting Tiny SSD was still able to achieve an mAP of 61.3% on the VOC 2007 test dataset, which is  $\sim 4.2\%$  higher than that achieved using Tiny YOLO. Figure 5 demonstrates several example object detection results produced by the proposed Tiny SSD compared to Tiny YOLO. It can be observed that Tiny SSD has comparable object detection results as Tiny YOLO in some cases, while in some cases outperforms Tiny YOLO in assigning more accurate category labels to detected objects. For example, in the first image case, Tiny SSD is able to detect the chair in the scene, while Tiny YOLO misses the chair. In the third image case, Tiny SSD is able to identify the dog in the scene while Tiny YOLO detects two bounding boxes around the dog, with one of the bounding boxes incorrectly labeling it as cat. This significant improvement



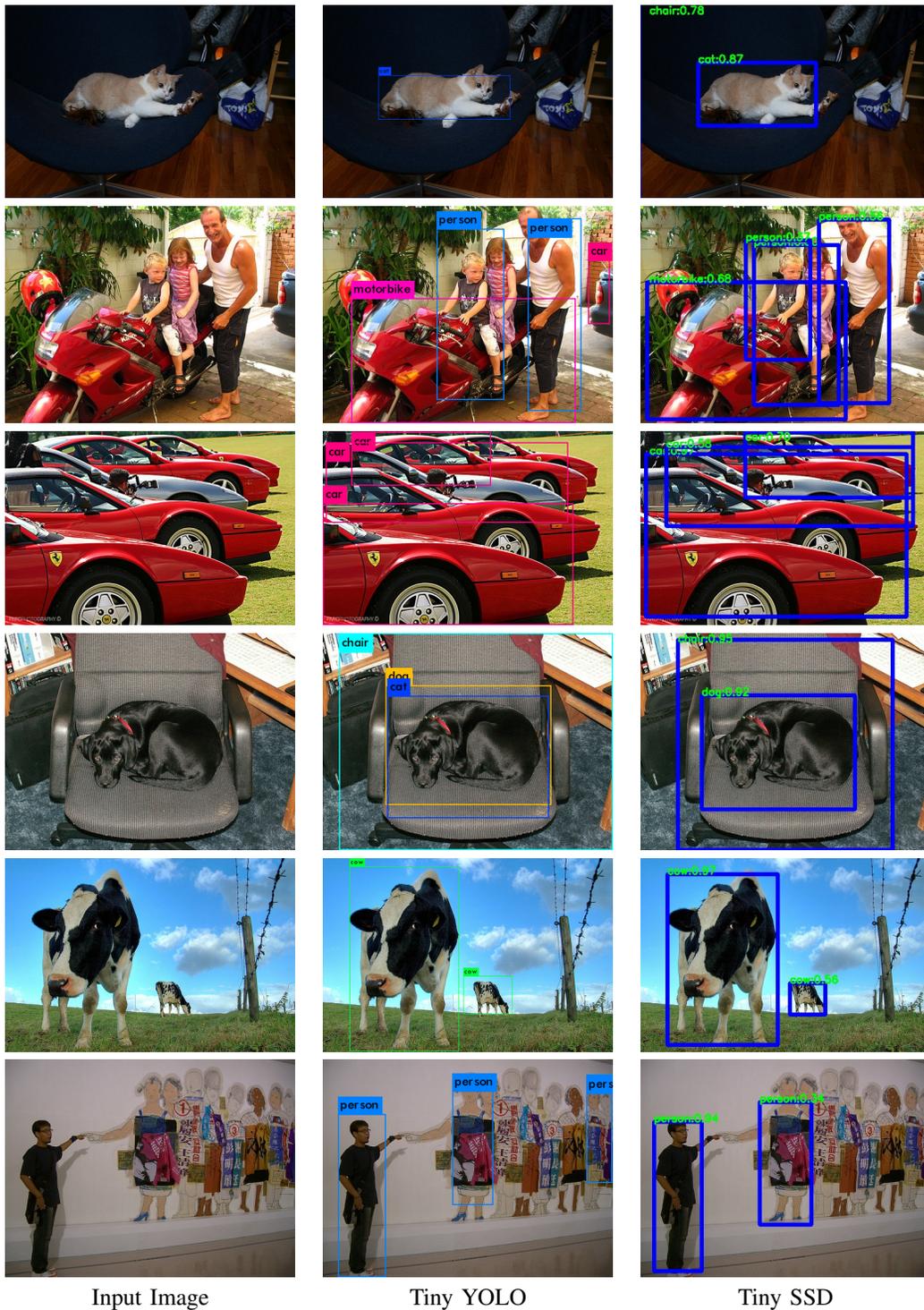


Figure 5. Example object detection results produced by the proposed Tiny SSD compared to Tiny YOLO. It can be observed that Tiny SSD has comparable object detection results as Tiny YOLO in some cases, while in some cases outperforms Tiny YOLO in assigning more accurate category labels to detected objects. This significant improvement in object detection accuracy when compared to Tiny YOLO illustrates the efficacy of Tiny SSD for providing more reliable embedded object detection performance.

- [15] Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. Training region-based object detectors with online hard example mining. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 761–769, 2016.
- [16] Bichen Wu, Forrest Iandola, Peter H Jin, and Kurt Keutzer. Squeezedet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving. *arXiv preprint arXiv:1612.01051*, 2016.