Deployment of Deep Learning Models on Resource-Deficient Devices for Object Detection

Ameema Zainab*, Dabeeruddin Syed*
Department of Electrical & Computer Engineering
*Texas A&M University
College Station, Texas, U.S.A.
{azain, dsyed}@tamu.edu

***Abstract*—Detection of images or objects in motion have been highly worked upon, and the detection has been incorporated, required and utilized in applications. A few of the limitations include the lack of computational resources, lack of strategic and methodological data analysis of the observed trained data. The accuracy of detection is dependent on movement, velocity of the objects and Illuminacy. Therefore, it is required that new techniques and strategies of detection are drafted, applied and recognized. In this work, we have worked upon a model based on scalable object detection, using Deep Neural Networks to localize and track people, cars, potted plants and other categories in the camera preview in real-time. The trained model from google inception model has been implemented in an android application. This integrated application works real-time and can be used handy in a mobile phone or any other smart device with minimal computational resources.**

***Keywords*—Deep learning, mobile device, android, object detection, neural network**

I. Introduction

It is the digital world we are living in and there is a lot to achieve in the field of robotics. A human eye takes a glance at a picture and subconsciously, classifies the objects in the image, processes all the information in the picture and all these stages occur in a fraction of second. The computer vision, if implemented with such precision and accuracy as the human visual system, can solve the application of autonomous cars, robots and others. Hence, the fast and accurate algorithms are required so that the computers can process the visual information in the images and can work in real-time enabling the robots to work autonomously without the aid of humans [1]. Also, the fast algorithms will enable the computers to convey the real-time information and enhance the potential of autonomous responsive robotic systems.

The computing paradigm is changing with machine learning, and an emerging trend of new use cases on mobile and embedded devices are observed. The process of classifying objects in digital image or video is called object recognition. Analysis of images involves various steps and poses great deal of challenges for machine learning. The used algorithms rely on matching, learning, or pattern recognition using feature-based or appearance-based techniques. According to data from digital analysts at GSMA Intelligence, there are officially more mobile devices than the population in this tech-savvy globe [2]. Implementing a machine learning model on a mobile device, without connection to the cloud, will be highly useful for real-time applications. The project focuses on the implementation of real-time object detection model on a mobile device using deep learning techniques.

In traditional methods, object detection is performed using sliding window classifiers [3] [4] [5] that train a discriminant function and scan over the locations of the image at multiple scales to verify if the object is present in the sub-windows of the image. Deformable Parts Model (DPM) [4] and Principal Component Analysis (PCA) are usually applied on Histogram of Oriented Gradients (HOG) features to decrease the dimensionality of the feature vector which gets significantly reduced without any noticeable loss. The approach is efficient in many cases with following disadvantages: 1) the discriminant function can not always be optimally trained for localization, 2) it is computationally not efficient to scan over the locations of the image at multiple scales.

More recent approaches like Regions with CNN (R-CNN) [6] generates bottom-up region proposal networks i.e. high capacity convolutional neural networks (CNN) for learning. This method involves training each component separately and employs complex pipelines. The above disadvantages in R-CNN are overcome by state-of-the-art object detection method called You Look Only Once (YOLO) [7] which considers detection as a regression problem rather than classification problem. It does not consider a complex pipeline of bounding boxes and class probabilities. Instead, it uses a single convolutional network to predict the classes of objects present in the image and the location of the objects. The novelty in this method is that it encodes contextual information about classes as well as their appearances running through the entire image during training and test. YOLO is extremely fast, approximately two times faster than the other real-time systems. The implementation of deep learning in resource-deficient devices use YOLO model for faster execution times and real-time object detection.

A. Research Objectives

The primary aim of this research is the deployment of deep learning models on resource-deficient devices. The research was successful in obtaining real-time object detection on mobile devices and also, in enhancing the classification accuracy while reducing the detection times. Deep learning models usually require high computations to be performed and deployment of these models on a mobile device stands out as a challenge. The study focuses on using deep learning model built on CNN for object detection. The trained classification model is used in the form of weights file which is then transformed using graph transformations. These allow for the

978-1-7281-4821-2/20/$31.00 ©2020 IEEE

transfer of a trained CNN model to a mobile device and the developed app uses the model for the classification of objects in real time.

II. Background Work

A. Historical Approach to Object Detection

Object detection has been modeled as classification where windows of fixed sizes of an image are fed to a classifier. Each window is given as an input to the classifier which predicts the location and class of the object. There are few problems, one of which is to know the optimal size of the window. The object may or may not be a part of the selected window size. To overcome this challenge, down sampling has been utilized. Each image's sliding windows are fed to the classifier and it is common to have at-least 64 sampled sizes. This helps to solve the issue and provides both the object classification and its location. Another issue in the field is the aspect ratio. A lot of objects are in various shapes and positions. Many works have undertaken these challenges and solved the constraints.

HOG Features - Groundbreaking research in [4] helped computer vision and gave a new direction to the classification of objects in the images or videos. They introduced the HOG features calculated on each window and fed to the Support Vector Machines (SVM) classifier. This method yielded very good results for person detection and also reduced false positive rates.

R-CNN - With the rise of deep learning the HOG features were replaced with CNN [6]. CNNs are a category of Neural Networks (NN) which have proven to obtain effective results in the recognition and classification of objects. Data that comes in the form of multiple arrays are processed by ConvNets. For example, a colour image consisting of three 2D arrays in the three colour channels (Red, Green and Blue) contain pixel intensities. Krizhevsky et al. [8] rekindled the research community's interest in deep learning and it was noted that deep learning produces ground-breaking results in image classification. However, CNNs were slow and computationally expensive as widely varying aspect ratios and slides with different shapes tuned to different appearance modes of the target object. R-CNN solved this problem by using recognition using regions [9], a selective search algorithm, that produces around 2000 region proposals being category-independent. In CNN, the fully connected part takes fixed input and so, a simple technique has been used to compute a constant (fixed) length feature vector for every proposal, irrespective of the shape of region. Hence, RCNN performs Selective Search (SS) [10], in order to generate probable patches which are then feeded to CNN followed by SVM to predict each class. This finally optimizes patches by training bounding box regression separately. R-CNN increased the detection and classification accuracy by more than 30% when compared to the previous state-of-the-art methodologies. ConvNets have been successful in classifying objects, people and traffic signs. These are also used in self driving cars and to generate vision in robots.

SPP-net - Kaiming et al. proposed the computation of feature maps from the entire image at once [11]. The features are then pooled in arbitrary regions (sub-images) which results in generating fixed-length portrayals for training and preparing the detectors. This methodology overcame the issue of slow execution which existed with CNNs on 2000 region propositions created by selective search and avoiding the repeated computation of convolutional features. The proposed method also solved the issue of providing fixed size (i.e., 224*224) input to the CNN. They also equipped the networks with another pooling strategy called Spatial Pooling. This method uses spatial pooling instead of traditional max pooling feature after the last convolutional layer. An SPPnet network structure contains a spatial pyramid pooling layer. $Conv_5$ is the final convolutional layer and the number 256 is the filter number of the $conv_5$ layer

Fast-RCNN - Girshick [12] fixed the key problem in R-CNN and SPP-net and made it possible to train end-to-end. Fast-RCNN has an additional step of adding bounding box regression to the NN itself resulting in the network now having two heads, 1) the classification head and 2) the bounding box regression head. This eventually avoids additional training of the network for localization and classification independently. These methods have enhanced classification accuracy by a huge margin when compared to RCNN and SPPnet.

Faster-RCNN - Shaoqing Ren et al., [13] proposed faster network in which full-image convolutional features are shared with the detection network, thus empowering almost cost-free region proposals. The selective search or edge boxes were replaced by a very small convolutional network called Region Proposal Network (RPN). It handles the variation ratio and scaling of the object by introducing the idea of anchor boxes. Each anchor takes its place at the center of a sliding window under consideration, and is related with a scale and an aspect ratio. This uses 3 aspect ratios and 3 scales, yielding k = 9 anchors at each sliding position. Hence the RPN produces the bounding boxes of different sizes with their respective probabilities for each class. The rest of the network remains the same. The Faster R-CNN is 10 times faster than the Fast RCNN.

B. You Look Only Once(YOLO)

An image is taken as an input and YOLO methodology learns the bounding box coordinates and the class probabilities. The novelty in this procedure is that it encodes contextual data about classes and also their appearances while running through the entire image during training and test. YOLO is extremely fast and can process in real-time. Furthermore, it is also twice as fast as the other real-time systems. The architecture of YOLO is a combination of 24 CNN layers followed by two fully connected layers. The features space from preceding layers is reduced by alternating 1 X 1 convolutional layers. The convolutional layers are trained previously on the ImageNet classification task at 1/2 resolution (224 X 224 input image) followed by the resolution doubled for detection. This model has been utilized in our implementation. In our methodology, Tiny YOLOv2 based off the Darknet[14] reference network has been used for the implementation. The reference network consists of only 9 convolutional networks rather than 24 convolutional networks and smaller number of filters employed in the layers. The smaller number of layers makes the model much faster however, accuracy obtained is less than the normal YOLOv2 [7] model. The final output of the tensor is 7 X 7 X 30 tensor of predictions. YOLO basically divides input (image) into an S X S grid, and if a grid cell contains the

center of object then that grid cell is considered responsible for recognizing that object. For those boxes, each grid cell predicts B bounding boxes and the confidence score. It also predicts the classification score for each score in every training class. A total of S X S X N boxes are predicted. These predictions are encoded as an S X S X (B * 5 + C) tensor.

As YOLO has the limitation of predicting only one class per grid, it struggles with very small objects and has localization errors. For instance, a flock of birds in an image might be considered as one object if they fall under a single grid cell.

III. Related Work

This section outlines the related work in areas of real-time object detection and localization. Analyzing images and videos play a large part in the world of information. Detection of objects in real-time has a lot of applications. If computers are able to understand the images as a human brain does, then it will be a revolution and entrance into a whole new world of information processing. It will help analyze a large bank of information from the repositories of images i.e., in the areas of hospitality, crime, space and others.

Very deep convolutional networks for large-scale image recognition was a research outcome of Imagenet 2014 challenge where multiple teams participated and contributed towards localization and classification of images using deep convolutional networks [15]. The work of Simonyan et al. focused on thorough evaluation of networks with increasing depth with the architecture starting with very small (3x3) convolution filters. By increasing the depth of weight layers to 16 or 19, significant improvement on the prior configurations was achieved. Their proposed model generalized well to a wide range of data sets and tasks, with almost equal or better performance, when compared to complex recognition pipelines built around shallow image networks. The use of 19 weight layers confirms the importance of depth in visual representation which opens up further research in the field of robot vision.

Shaoqing Ren et al. [13] introduced a Region Proposal Network (RPN) that enables almost cost-free region proposals. These proposals share full-image convolutional features with detection network. It is fully-convolutional network that simultaneously predicts object detection scores and object bounds at every position. The end-to-end trained RPNs generate high quality region proposals. These are used by the fast R-CNNs for detection. The detection system has performed very well for the very deep VGG-16 model [15] with a frame rate of 5fps (including all steps) on a GPU. Furthermore, the system achieved an improved detection accuracy of 73.2 % mAP (Mean Average Precision) on PASCAL VOC 2007 [16] and 70.4% mAP on 2012 datasets using 300 proposals per image.

Redmon et al. [7] proposed state-of-art real time object detection model called YOLO. YOLO is faster and more accurate when compared to other detection models. It is based on the GoogleNet model for image detection [17]. Novelty of this approach is in improving the utilization of computing resources by increasing depth and width of the network, while keeping the computational resources constant. The architectural decision was inspired by the Hebbian principle in neuroscience which explains the adaption of neurons during learning process in the brain. The work focused on deep neural network architecture for robotic vision, codenamed Inception. The network is 22 layers deep whose quality is evaluated in the context of classification. Their network utilized 24 convolutional networks followed by 2 fully connected layers. However, instead of the Google's inception model, a simple 1 x 1 reduction layer followed by 3 x 3 convolutional layer was utilized. Fast YOLO has also been worked upon which uses 9 instead of 24 convolutional networks. Fast YOLO has fewer filters in the layers. The rest of the training and testing parameters remained unchanged. A linear activation function for the final layer and leaky rectified linear activation (ReLU) have been used for all the other layers.

At testing time, unlike classifier-based methods, YOLO is extremely fast since it requires a single network evaluation. However, it has a few limitations. Since each grid cell only predicts two boxes and can only have one class, strong spatial constraints are imposed on bounding box predictions. This results in incorrect localization. The model struggles in recognizing small objects that appear in groups in a single cell.

Redmon et al. [18] proposed YOLO 9000 which can identify, detect and classify 9000 object categories. They have used a novel and multi-scale training method that can run at varying sizes, offering an easy trade-off between accuracy and speed. They considered detection datasets like Imagenet and COCO to validate the accuracy of the updated model. While YOLO basically had relatively low recall compared to region proposal-based methods such as Fast R-CNN, the proposed YOLOv2 maintains classification accuracy while improving recall and localization.

IV. Methodology

Training of a network needs high computational resources and sometimes, connection to cloud. The most complex models take weeks to train on machines even with highly equipped GPUs. Keeping in view the model complexities and model size, the trained models are usually hosted in the cloud [19]. For a mobile-device to use this model, the connection with the network is required. The connection requirement increases the response time. There has been a considerable amount of progress in the field of deep Learning to help mobile devices be able to train the models. These mobile devices will be equipped with GPU to run the models. The current work focuses on device-based modeling where the weights of a pre-trained model are moved from a highly equipped machine to a mobile device. With the help of protobuf (.pb) file (android) and core ML (iOS), the trained machine learning models can be integrated into a mobile application. Tensorflow, a deep learning library, supports this feature in Android. The design in Figure 1 illustrates the process flow of the study.

The motivation of this project is to develop a robotic vision system in mobile devices which mimic the human visual system. The detection system should look at the image once and be able to classify, identify and locate the objects in images and videos in real-time without any internet access. This requires the trained deep learning model to be moved to the mobile device. The implementation uses a trained model from tiny yolov2. The trained weight file is transferred to the mobile device. The challenge after the transfer are 1) to

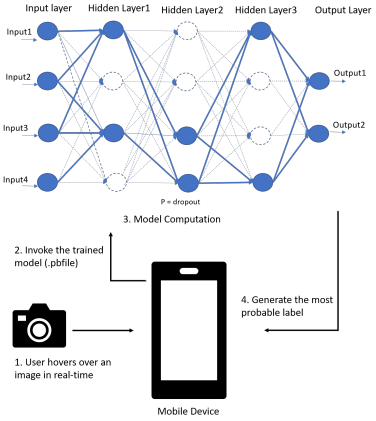

Fig. 1: Device-based Inference - Process Flow

train the mobile device to understand the convolutional neural network structure and 2) to use the weight file which contains the weights at different nodes at different levels in the deep neural network.

A. Model Used

The model is built on VOC data-set [16]. The aim of using this dataset was to recognize objects from a number of visual object classes in real world. The data consist of 20 labeled classes which are given in the following:

Person: person.
Animal: horse, bird, cat, dog, cow, and sheep.
Vehicle: aeroplane, bicycle, boat, bus, car, motorbike, and train.
Indoor: dining table, bottle, chair, sofa, potted plant, tv and monitor.

Hence, the application that we developed on our android mobile phone has trained weights file that can classify only the above 20 categories.

The network in the used model has 19 convolutional layers, 2 fully connected layers and a maxpool layer for reduction of dimensions.

B. Real time detection requirements

To make detection happen in real-time, the darknet with CUDA, darkflow and openCV have to be compiled. Darkflow helps in the conversion of the weights file to a protobuf file which is compatible with mobile devices (JAVA or C++ or Objective-C++). Darkflow does it with the help of cython and openCV. Cython enables the compatibility of C and Python codes where as openCV equipts the app for real-time detection. Once the weights file is converted to a mobile-compatible protobuf file, it is added to the mobile application for successful execution.

C. Setting up the application

The image recognition model on android using tensorflow has been utilized for the research implementation. It performs three functions - Tensorflow Classify, TensorFlow Stylize and Tensorflow Detect. The TensorFlow detect is not activated in the github project. Setting the build-system to cmake, adding the shared object (.so) file and the .jar file will help the Tensorflow Detect ability in the mobile application.

D. Building the .so and .jar file

The .so file is built with the help of bazel. The building of .so file takes large amount of time thus it is favored to create the file on a computer with strong computational resources. The .jar file is also built using bazel. Before the application performs the inference tasks, the compiled library files have to be loaded. The overall procedure and implementation method can be depicted as shown in the (Figure 2)

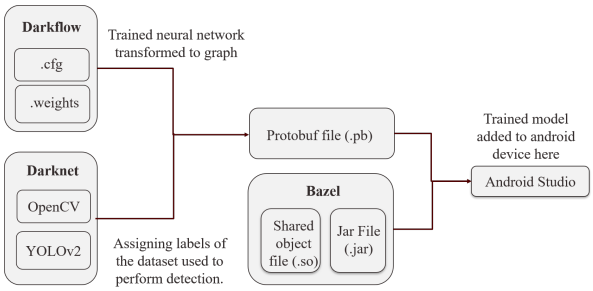

Fig. 2: Flow Chart of the Implementation process

E. Implementation on Android

Once the .so file and .jar file are ready. the .pb file is collected and converted from darkflow and added to the android studio package under the assets section. The addition of these files makes the android application ready to be used in real time. It considers the neural network layers from the protobuf file, predicts and gives output for the new image under consideration in real-time. It classifies the objects in images into 20 categories on which the model is trained.

V. Results

The tensorflow image detection model was developed in the mobile application for the application of real-time object detection using resource-deficient devices like mobile devices. The larger deep learning model was successfully moved to a mobile device which enables the device to inherit the learning obtained after training. Subsequently, the device does not require to undergo any further training and only, testing is performed on the mobile device. The challenge of making the device understand the deep learning layers of neural network from weights file has been addressed. The device detects objects in real-time and it does not require any access to the internet. The accuracy of our implemented model against the baseline references is provided in the Table I.

We have tested the object detection model on our mobile device, and it detects and classifies the objects in real time in fraction of seconds. It predicts the bounding boxes and also provide probability that an object belongs to a particular object category. For the implementation results of object detection, please refer to the Figure 3 and Figure 4.

	mAP	Speed (frames per second)
Fast RCNN	70.1	0.5
Faster RCNN	73.1	7
Fastest DPM	30.5	15
Proposed model on mobile	66.3	21

TABLE I: Speed and Mean Average Precision for different object detection models

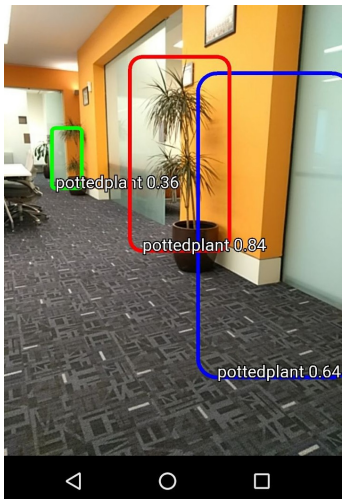

Fig. 3: Real time detection

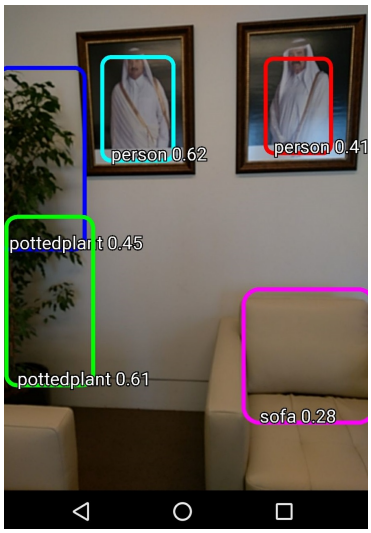

Fig. 4: Real time detection

VI. TECHNICAL CHALLENGES

The generation of protobuf file compatible for deployment on mobile device was equally challenging and time consuming. There were also incompatibilities with the versions of python and openCV. Also, the android studio versions of the SDK and NDK were to be closely looked upon. The main challenge faced was moving a huge neural network into a mobile device and also make the device understand the deep layers of the neural network. Deep learning is usually resource hungry. To develop models which require less resources (for instance, without a GPU) is a great challenge in the field of mobile computing. Also, the models built using NN are complex and it is always very challenging to make a device understand the complex layers, the connections within the layers and the flow structure. Until and unless the knowledge of layers and structure of the network are not transfered to a device, the model cannot be implemented on it. Compressing the huge model to a tiny compressed constant model calls for another space of restrictions and constraints. The research study was able to resolve this issue by reducing the node numbers in the model while maintaining the accuracy. The modes was converted to graphs model to compress it and to deploy it successfully on a mobile device.

VII. VISUALIZATION AND CONCLUSION

Figure 3 and Figure 4 show the bounding boxes of the detected object along with the accuracy of detection. Multi-object detection models run successfully classifying the objects into categories on which the model has been trained. It is quite fast as it uses the state-of-art detection system Yolov2 to perform the detection which has 78.6% mAP at 40-90 frames per second. The application works with few limitations such as it does not classify correctly the objects in motion, the objects in dark or less illuminance, the objects on a TV or computer screen (refer Figure 5). The model deployed on mobile device has also been successfully tested for object detection in videos (Figure 6). And it is clear that the execution is in real time for object detection in images and in videos as well.

VIII. FUTURE WORK

Direction for future research is to improve the detection for objects moving with speed, for objects in low illuminance and detection of objects displayed on monitor. One of the solutions for object detection for images on monitor can be to train the model on images of same objects with different pixel values. It still remains a core challenge in creating accurate machine learning models capable of localizing and identifying multiple objects if they fall under a single grid.

Challenges to be solved in future are: 1) object detection when mobile device moves at higher speed, 2) object detection under low illumination, 3) object detection for images displayed on a screen or monitor (Figure 5).

Tiny YOLO has been used for modeling and it detects up to 20 object categories. In future work, we intend to do the training of the model ourselves so as to include objects from more number of categories. However, it would require time and resources as the training image set should contain millions of images. Computer vision is however honored to be blessed with a gigantic number of labeled data for training. The model should be trained on images of interest to perform detection of desired objects.

REFERENCES

[1] Aleš Ude, Tomohiro Shibata, and Christopher G Atkeson. Real-time visual system for interaction with a humanoid robot. *Robotics and Autonomous Systems*, 37(2-3):115–125, 2001.

[2] GSMA Intelligence. Definitive data and analysis for the mobile industry. https://www.gsmaintelligence.com/.

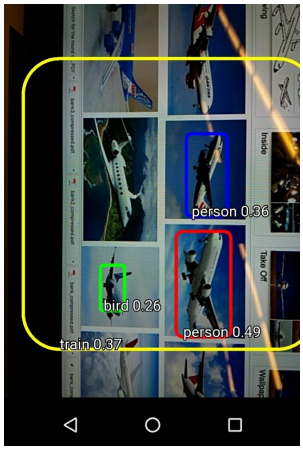

Fig. 5: Detection of objects over images on computer screen

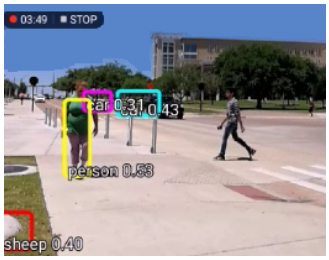

Fig. 6: Detection of objects in video

[3] Anna Bosch, Andrew Zisserman, and Xavier Munoz. Representing shape with a spatial pyramid kernel. In *Proceedings of the 6th ACM international conference on Image and video retrieval*, pages 401–408. ACM, 2007.

[4] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[5] Vittorio Ferrari, Loic Fevrier, Frederic Jurie, and Cordelia Schmid. Groups of adjacent contour segments for object detection. *IEEE transactions on pattern analysis and machine intelligence*, 30(1):36–51, 2008.

[6] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[7] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[9] Chunhui Gu, Joseph J Lim, Pablo Arbeláez, and Jitendra Malik. Recognition using regions. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1030–1037. IEEE, 2009.

[10] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *european conference on computer vision*, pages 346–361. Springer, 2014.

[12] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[13] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[14] Joseph Redmon. Darknet: Open source neural networks in c. http://pjreddie.com/darknet/, 2013–2016.

[15] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[16] Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *International journal of computer vision*, 111(1):98–136, 2015.

[17] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[18] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[19] Ben Kehoe, Akihiro Matsukawa, Sal Candido, James Kuffner, and Ken Goldberg. Cloud-based robot grasping with the google object recognition engine. In *2013 IEEE International Conference on Robotics and Automation*, pages 4263–4270. IEEE, 2013.