MOBILE OBJECT DETECTION ALGHORITHMS USING TENSAFLOW

Najmiddinov Shakhozdbek Shukhrat ugli
Odiljonov Umidjon Odil ugli
Tashkent University of Information Technologies named after Muhammad al-Kharezmy

ABSTRACT. In recent years, deep learning has advanced, enabling us to build sophisticated machine learning models for object detection in photos regardless of the properties of the objects to be recognized. This advancement has made it possible for engineers to replace current heuristics-based systems with machine learning models that perform better. In this study, we assess the effectiveness of utilizing deep learning models as feature extractors for existing algorithms or end-to-end systems for object detection in real-time video feeds on mobile devices in terms of object detection performance and inference delay. In compared to existing techniques, our results demonstrate a considerable improvement in object detection performance when transfer learning is applied to neural networks that have been optimized for mobile application.

KEYWORDS: machine learning, heuristics-based systems, end-to-end systems, neural networks

INTRODUCTION

Over the past few years, Deep Learning (DL) has advanced to the point that we are now able to build sophisticated machine learning (ML) models for object detection in photos, regardless of the properties of the objects to be recognized. Due to this advancement, engineers may now replace current heuristics-based systems in favor of more effective ML models [1]. The industry must adopt more complex technology to live up to expectations as people use their mobile phones more frequently and demand increasingly sophisticated performance [2] from their mobile applications. Use of ML algorithms for object detection may be one such adaption. Tensaflow is a program that is frequently used for Machine Learning jobs. As it offers an interface to represent popular Machine Learning techniques and executable code of the models, it is widely used. Models produced in Tensaflow can be transferred to heterogeneous systems with devices ranging from mobile phones to distributed servers with little to no modification. Tensaflow is

used internally by the corporation for Machine Learning applications and was developed and is maintained by Google.

BACKGROUND

Using a recognition algorithm on each sub-window of the original image, object detection is the process of identifying items on an image[3], ranging from one to many kinds of objects. For instance, using object detection to find faces, pedestrians, or cars in an image. When a picture is divided into many sub-windows, the process is shown using the YOLO network, which is used to recognize objects in the image. Localizing the items in the image is necessary for object detection but not for categorization. Google designed Tensaflow to be able to function on a variety of platforms, including mobile devices. This was brought on by the difficulties associated with transferring data back and forth between devices and data centers when computations could be completed locally on the device. By eliminating the need for network round-trip delays for Machine Learning computations, Tensaflow mobile allowed developers to design interactive applications. Due to the high computational cost of Machine Learning jobs, model optimisation is employed to enhance performance. Because the intended latency for mobile applications is minimal, the minimum hardware requirements for Tensaflow machine in terms of Random-Access Memory (RAM) size and CPU speed are low, and the main bottleneck is the computation speed.

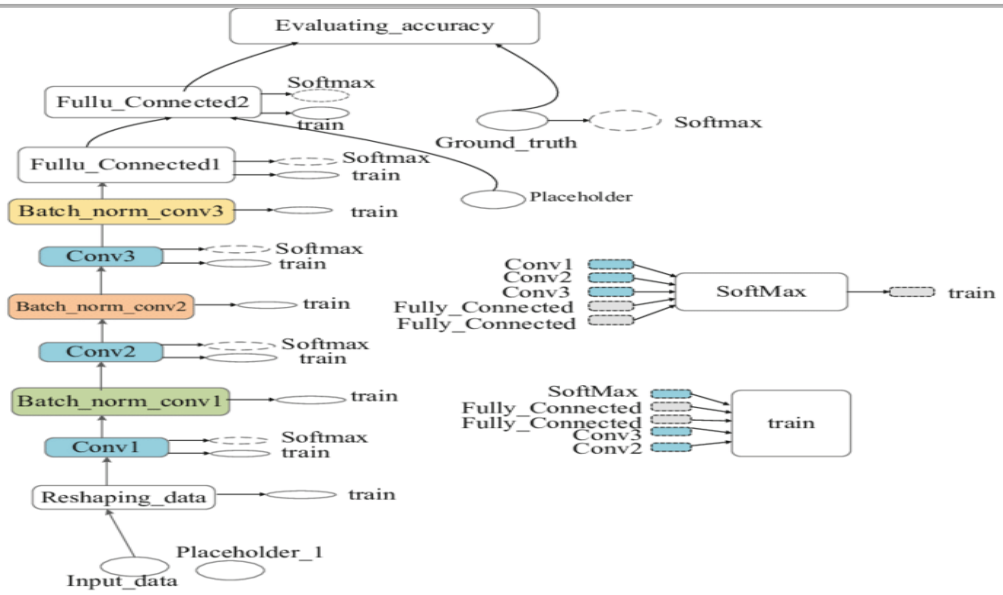


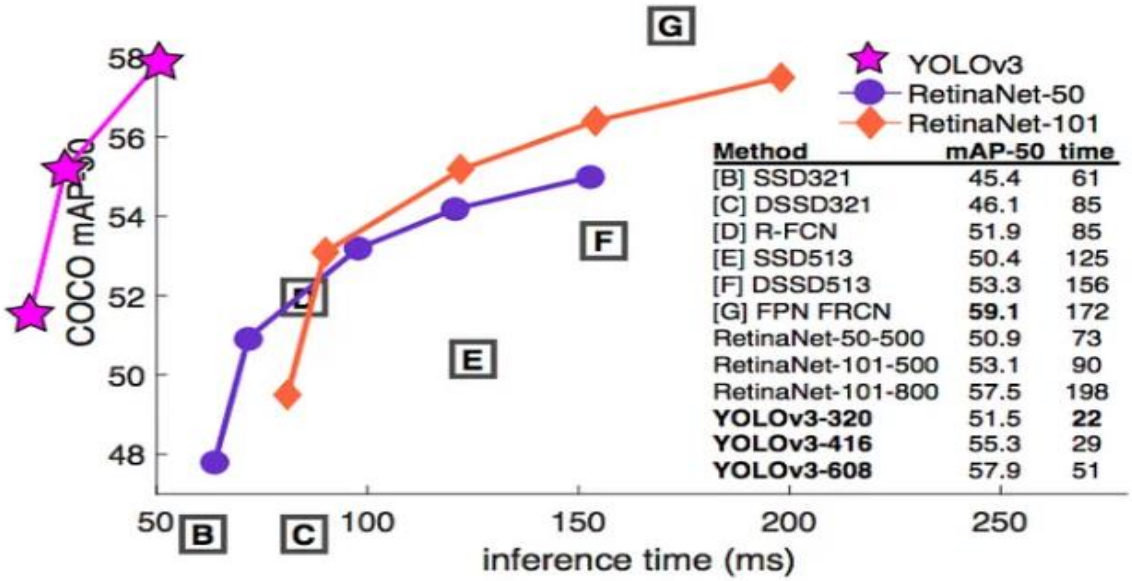
Figure 1. Computational graph for CNN generated using TensorFlow

METHODS AND EXPERIMENTS

The method used in the experiments as well as the decision to use a certain approach are explained in this section. The studies were designed to evaluate the effectiveness of the object detection algorithms and their applicability in this computationally constrained setting. The pre-processed photos of Post-it R notes in Bontouch's dataset can be used to compare the deep learning model to the current heuristics-based technique. There are only a few hundred entries in this dataset, and additional data is needed for training. The command-line Python program instagram-scraper was used to collect more pictures of Post-it R notes. The program enables effective image scraping from user profiles and hashtags on Instagram, one of the biggest social media networks with 233 million active members that focuses on photo-sharing. Networks like SSD[4], R-CNN[5], and YOLO[6] were assessed as the detection frameworks for the CNN to be extended and fine-tuned, and MobileNet, Inception, and ResNet were evaluated as the base networks because Deep CNNs require intense training.

Among these cutting-edge models, choosing the basis model for the object identification algorithms strongly depends on the trade-off between speed and performance. The speed and accuracy of the various models, as well as the mAP between the networks, differ significantly, as shown in Figure 2. Tiny YOLO is faster than object detection models created with SSD, but performs worse in terms

of detection, as is also the case between Faster R-CNN and SSD, where SSD is the faster but worse networking technology.



Method	mAP-50	time
[B] SSD321	45.4	61
[C] DSSD321	46.1	85
[D] R-FCN	51.9	85
[E] SSD513	50.4	125
[F] DSSD513	53.3	156
[G] FPN FRCN	**59.1**	172
RetinaNet-50-500	50.9	73
RetinaNet-101-500	53.1	90
RetinaNet-101-800	57.5	198
YOLOv3-320	51.5	**22**
YOLOv3-416	55.3	29
YOLOv3-608	57.9	51

Figure 2. YOLOv3 Performance on the COCO dataset

The FPS at which a mobile device can do computations in a mobile environment is essential for the mobile experience. The device would struggle to run the application at more than 1 FPS with an overly complicated and computationally expensive network. Professional developers at Bontouch claim that a smooth mobile experience necessitates a minimum of 2 FPS, which restricted the base network to either SSD or YOLO because Faster R-CNN performs poorly on mobile devices. However, evaluations and assessments of all the described architectures were conducted. Darkflow, an open-source toolkit, was used to train the Tiny YOLO network since it made it simple to transfer learning from the base model to our particular model by loading the pre-trained model and removing the final two layers. Additionally, Darkflow offers an API that allows users to export their bounding box predictions in JSON format for use in object detection. Additionally, darkflow made it possible for us to export the darknet model to tensorflow for deployment on mobile devices, which was necessary to assess the model's viability in terms of FPS on those devices.

RESULTS

The Tiny YOLO V2, SSD MobileNet V1, SSD MobileNet V2, SSD Inception V2, Faster RCNN Inception V2. Machine Learning models were trained and assessed in terms of mAP performance in order to discover the model that performed the best for the task of detecting Post-it R notes. Table 1, where all models were trained on the Post-it R training data and tested on the test data, shows the performance of the trained models.

MODEL	Base model train data	mAP
Tiny YOLO V2	VOC 2007+2012	87.57 %
SSD MOBILE NET 1	COCO trainval	91.16 %
SSD MOBILE NET 2	COCO trainval	91.90 %
SSD inception	COCO trainval	96.82 %
FASTER RCNN Inception V2	COCO trainval	96.69 %

Table 1. mAP performance among machine learning models

The trade-off between model performance and latency relies on the object detection job at hand when selecting an appropriate ML model for object recognition on mobile devices. More accurate and complicated models, like the Faster RCNN Inception model, are appropriate in environments where latency is less critical, such as environments where inference jobs are not done in real-time.

SUMMARY

This article outlines a method for creating a Post-it R note object detection CNN that can be used with mobile devices. We successfully trained and developed a number of models that may be utilized for real-time note identification in an Android application using a range of base models and object detection frameworks. The network could be trained on a tiny quantity of data us Transfer Learning and yet achieve good mAP scores. In terms of inference time and accuracy, ML models

are suitable for use on mobile devices and can deliver a heuristic-based corner-detection technique with bounding boxes of high recall. More images from settings typical to the end use case could improve the performance of the less complex models when detecting multiple notes, as these models may have been constrained by the quality of the training data, which could increase performance in order to further develop the work presented in this report.

REFERENCES

1. Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. “CNN Features off-the-shelf: an Astounding Baseline for Recognition”.
2. Dimensional Research. FAILING TO MEET MOBILE APP USER EXPECTATIONS – A MOBILE APP USER SURVEY.
3. Richard Szeliski. Computer vision algorithms and applications. London; New York: Springer, 2011, pp. 10–17.
4. Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, and Scott E. Reed. “SSD: Single Shot MultiBox Detector.”
5. Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. “Rich feature hierarchies for accurate object detection and semantic segmentation”.
6. Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. “You Only Look Once: Unified, Real-Time Object Detection”.