Object Detection and Its Implementation on Android Devices

Zhongjie Li
Stanford University
450 Serra Mall, Stanford, CA 94305
jay2015@stanford.edu

Rao Zhang
Stanford University
450 Serra Mall, Stanford, CA 94305
zhangrao@stanford.edu

Abstract

Object detection is a very important task for different applications including autonomous driving, face detection, video surveillance, etc. CNN based algorithm could be a great solution for object detection with high accuracy. Besides, most current deep learning applications are running on servers or desktop computers. Considering there are a lot of mobile computing devices available, we implemented the CNN based object detection algorithm on Android devices. The model architecture is based on SqueezeNet and further improved to find bounding boxes for recognized objects. The total model size is around 8 MB and makes it run fast, especially on mobile devices.

1. Introduction

Deep learning based object detection has been very successful in recent years. Especially the CNN (convolutional neural network) model has significantly improved the recognition accuracy on large dataset. For the ImageNet benchmark data set, the CNN based model has been dominating the leaderboard since it's introduced by Krizhevsky in 2012 for the first time.

While CNN based model can achieve higher accuracies, they have following disadvantages:

- **High computation cost.** The CNN based model are usually very deep and each layer takes a lot of computation.
- **Large memory demand.** The CNN based model has a lot of parameters that usually take hundreds of Megabytes of memory space.
- **Low efficiency.** Most CNN based model are not designed for efficiency.

As mobile computing devices are very popular and comparatively powerful, people want to embrace the benefits of CNN with their mobile devices. However, to enable their mobile application, new CNN architectures need to be developed to overcome the above issues. Also, most deep learning frameworks have provided interface for mobile platforms, including iOS and Android. In this paper, we developed a CNN based model and then implemented it with Tensorflow and Android.

Our model is trained with KITTI benchmark. The KITTI data-set has over 100 GBytes of well-labeled data for object detection purpose. After training, our model is able to detect objects in view of camera on the Android device.

The rest of this paper is organized in the following order. Section 2 lists the related work of CNN architectures as well as CNN for object detection, discusses the state-of-the-art progress in CNN model compression. In Section 3, our model based on SqueezeDet is represented and elaborated. Section 4 introduces details of the KITTI data-set and the features to be used for our model. In Section 5, we conduct experiments with our proposed model and analyze the results from the experiments. Section 6 concludes our work and our future work is stated in Section 7.

2. Related Work

In this section, we talk about CNN related work in object detection and the trend towards smaller CNN models.

2.1. CNN Architectures

Convolutional Neural Network (CNN) usually stands for the neural network which contains one or more convolutional neural layers. Each neural layer can be regarded as a combination of several spatial filters. These filters are used for extracting features from pictures. Some well-known filters are Histogram of Oriented Gradients (HOG) and color histograms, etc. A typical input for an convolutional layer is a 3-dimensional grid. They are height (H), width (W) and channels (C). Here each channel represents a filter in the convolutional layer. The input of first layer usually has a shape of (H, W, 3), where 3 stands for the RGB channels for the raw pictures.

CNN became popular in visual recognition field when it is introduced by LeCun *et al.* for handwritten zip code recognition [10] in the late 90s. In their work, they used (5, 5, C)-size filters. Later work proved that smaller filters have multiple advantages, such as less parameters and reducing the size of network activations. In a VGG network [15] proposed by Karen Simonyan *et al.*, (3, 3, C)-size filters are extensively used, while the networks such as Network-in-Networ [12] and GoogLeNet [17] widely adopt (1, 1, C)-size filters, the possibly smallest filters and used for compressing volume of the networks.

With the networks go deep, the filter size design gradually become a problem that almost all the CNN practitioners have to face. Hence, several schemes for network modularization are proposed. Such modules usually include multiple convolutional layers with different filter sizes and these layers are combined together by stack or concatenation. In a GoogLeNet architecture, such as [17, 18], (1, 1, C)-size, (3, 3, C)-size and (5, 5, C)-size are usually combined together to form an "Inception" module and even with filter size of (1, 3, C) or (3, 1, C).

In addition to modularizing the network, communication and connections across multiple layers also improve the performance of the network. This seems to be a similar idea with Long Short Term Memory (LSTM) or Gated Recurrent Unit (GRU) architecture in Recurrent Neural Network (RNN). Residual Network (ResNet) [7] and Highway Network [16] adopted such ideas to allow connections to skip multiple layers. These "bypass" connections can effectively send back the gradients through multiple layers without any blocking in a backward propagation pass when necessary.

2.2. CNN for Object Detection

With the advancement of accuracy in image classification, the research for object detection also developed in a fast speed. Before 2013, feature extraction techniques such as [1], which proposed an combined application of HoG and SVM can achieve a high accuracy on the PASCAL data-set [3]. In 2013, a fundamental revolution occurred in this field, which was caused by the introduction of Region-based Convolutional Neural Networks (R-CNN), proposed by Girshick and Ross. R-CNN firstly proposes possible regions for residing objects, then makes use of CNN to classify objects in these regions. However, these two independent operations require high computation and make it time-consuming. An modification of R-CNN is made by Girshick and Ross, which is called fast R-CNN [5]. This architecture integrate the two independent tasks into one multi-task loss function, which accelerates the computation of proposals and classification. Later, a more integrated version of R-CNN, namely the faster R-CNN [14] was proposed by Ren *et al.*, which achieves more than 10x faster than the original R-CNN. A recent proposal, R-FCN [11] with a fully convolutional layer as the final parameterized layer further shortens the computation time used for region proposals.

R-CNN can be regarded as a cornerstone for the development of CNN for object detection. A large amount of work is based on this architecture and achieves great accuracy. However, a recent work shows that CNN based object detection can be even faster. YOLO (You Only Look Once) [13] is such an architecture integrating region proposition and object classification into one single stage, which significantly contributes to simplification of the pipeline of object detection, as well as reduction of the total computation time.

2.3. Toward Smaller Models

With CNN goes deeper, more parameters need to be stored, which makes the model larger and larger. Deeper CNN and larger modules usually achieve a higher accuracy, but people wonder whether a small model can reach a similar accuracy as a large model. In this sub-section, we talk about several popular model compression techniques aiming to reduce the size of CNN models.

As we know, singular value decomposition (SVD) is widely used to reduce matrix dimensionality. It is also introduced to pre-trained CNN models [2] to reduce model size. Another approach reported is Network Pruning [6], proposed by Han *et al.*, which prunes the parameters below a certain threshold to construct a sparse CNN. Recently, Han *et al.* have further improved their approach and proposed a new approach, Deep Compression, together with their hardware design to accelerate the computation of CNN models. A recent research called SqueezeNet [8] even reveals that a complex CNN model as AlexNet [9] accuracy can be compressed to smaller than 0.5 Mbytes.

Here are two examples of model compression. The famous ImageNet winner VGG-19 model stores more than 500 Mbytes parameters, which achieves a top-5 accuracy of about 87% on ImageNet, while the equally famous ImageNet winner GoogLeNet-v1 only contains about 50 Mbytes parameters, achieving the same accuracy as VGG-19. The well-known AlexNet [9] model with a size of more than 200 Mbytes parameters, achieves about 80% top-5 accuracy on ImageNet image classification challenge, while the SqueezeNet [8] model with a much smaller size, about 4.8 Mbytes parameters, can also achieve that accuracy. We can anticipate that there is much room left for compressing these CNN models, to better fit them to portable devices.

3. Methods

As for the baseline, we implemented SqueezeDet model in an Android device. The SqueezeDet model is a fully convolutional neural network for object detection. It's based on SqueezeNet architecture that extract feature maps from image with CNN. Then another convolutional layer is used to find bounding boxes and class probabilities. The model has

Figure 1. Example of A Picture in Data-set

the benefit of small model size, good energy efficiency and goo accuracy due to the fact that it's fully convolutional and only contains a single forward pass.

For the implementation of CNN model in Android device, we used the interface provided by "Tensorflow Android Camera Demo". However, the interface is not well documented and it's designed for the specific application. The basic idea is to save the tensorflow graph and variables into a file after training the model. Then load the graph into Android device and evaluate operators' output with new input for the trained graph and variables in the Android application.

4. Data-set and Features

The data-set we use is The KITTI Vision Benchmark Suite[4], which is made for academic use in the area of autonomous driving. For our target, we use the object detection data-set, which contains 7481 training images and 7518 test images. Total 80256 objects are labeled for this data-set and the 3 classes used for evaluation are cars, pedestrians and cyclists. The pictures in this data-set are fully color PNG files.

No pre-processing is done for the data-set, but data augmentation such as flipping and random cropping will be applied to increase the robustness of the network. Batch normalization will also be added to ameliorate the network initialization. Figure 1 is an example picture in the KITTI data-set.

5. Experiments

The experiments include two parts. The first part is to examine the predicting accuracy of our CNN model against KITTI test data set. And the second part is to test the Android application in real world scenarios. We expect to compare mean average precision (mAP), model size and detecting speed between different models. We will primarily focus on reducing model size and increasing detecting speed without losing much accuracy.

6. Conclusion

As a baseline, we have a running Android app that runs our CNN model trained by Tensorflow offline. The model size is 8 MegaBytes and the testing accuracy is 76.7%.

7. Future Work

Next step is to improve the Android application stability and functionality. In addition, we will improve the CNN model to further reduce the model size and improve the accuracy.

References

[1] N. Dalal and B. 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.

[2] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in Neural Information Processing Systems*, pages 1269–1277, 2014.

[3] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[4] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3354–3361. IEEE, 2012.

[5] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1440–1448, 2015.

[6] S. Han, J. Pool, J. Tran, and W. Dally. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems*, pages 1135–1143, 2015.

[7] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[8] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[10] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[11] Y. Li, K. He, J. Sun, et al. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in Neural Information Processing Systems*, pages 379–387, 2016.

[12] M. Lin, Q. Chen, and S. Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[13] 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*, pages 779–788, 2016.

[14] R. S. Ren, K. He, and R. Faster. Towards real-time object detection with region proposal networks, arxiv preprint. *arXiv preprint arXiv:1506.01497*.

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

[16] R. K. Srivastava, K. Greff, and J. Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

[17] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[18] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.