Continuity Report: Revisiting Grocery Recognition using TensorFlow

Samuel C. Hoffman, Dilip Thiagarajan
{sch235, dt372}@cornell.edu

Last Updated: September 15, 2016

1 Introduction

The revolution of convolutional neural networks (ConvNets) in the field of machine learning has spawned applications in a wide array of areas. Some accessibility applications have already begun to harness the potential of ConvNets in computer vision for assisting the visually impaired such as Facebook's Automatic Alternative Text tool for narrating images [4] However, the underlying technology involved in recognizing grocery items to assist visually impaired shoppers has not advanced much beyond the study by Merler, Galleguillos, and Belongie [10]. We aim to revisit this study and apply modern deep convolutional neural networks to solve the problem. We will use the open source library, TensorFlow[1], to implement the deep learning system [1].

We attempt to solve the problem of recognizing objects in the wild using images of the object taken under ideal conditions to train. This challenge has enumerable applications but in this study, we will look specifically at recognizing grocery products sold in Mattin's, a local café on the Cornell campus in Ithaca. However, the results should be easily transferrable to other areas of application. Ideally, training data and testing data would be obtained from the same distribution, however, in real-world applications, it is often more convenient to obtain training data from the web or dedicated datasets. Ultimately, our goal in this study is to achieve real-time recognition and localization with a camera.

The GroZi project[2], from which the original study came, encompasses many applications surrounding this core recognition problem. However, the implementation still relies on color histogram matching, SIFT matching, and boosted Haar-like features and the results left room for improvement.

For further background, please read our original proposal.[3]

[1]https://www.tensorflow.org/
[2]http://grozi.calit2.net/
[3]https://github.com/dthiagarajan/grozi_tf/blob/master/proposal/Proposal.pdf

2 Approach

In this section, we will explain the methods we attempted and motivations for the algorithms we chose.

2.1 TensorFlow

We settled on using TensorFlow as the main library for implementing our deep learning network. In summary, our reasons for choosing TensorFlow are as follows. While it does not offer significant speed or functionality improvements over other popular libraries, TensorFlow does allow for development entirely in Python. This means we can implement networks quickly without learning the semantics of a new language while allowing for easy integration in any workflow (e.g. processing with NumPy or Scikit-learn). Moreover, Tensorflow is designed to work seamlessly with multiple GPUs and speed is on par with other popular libraries. [7]

2.2 Data Collection

In this study, we used the original GroZi-120 dataset for training and testing. The goal of this study is ultimately to recognize groceries at Mattin's, however we haven't collected images of those products yet. We decided our system should be easily applicable to any product set, and thus there was no reason to test at Mattin's until the system worked well enough on the GroZi dataset first.

As we will discuss later, the GroZi-120 dataset was not compiled with deep learning in mind and many products have fewer than 10 images total. After running a few algorithms on the data as is, we attempted to augment the dataset with more canonical images per class. However, this proved difficult since there are only a couple angles at which the products are shot. We briefly explored the idea of collecting data for each item in a controlled environment ourselves, but that process would not scale well and we lacked the resources and time [11] Some manufacturers have also updated the packaging of products in the dataset and so the exact product is no longer produced.

2.3 Deep Learning Models

Implementation details for all algorithms used can be found on our public GitHub repository.[4]

2.3.1 CIFAR-10 Network

To get a sense of the problem and familiarize ourselves with TensorFlow, we started off with a simple model adapted from a network designed to perform on the CIFAR-10 dataset. The model was not pre-trained on any dataset and used only the data from GroZi-120. Furthermore, we narrowed the classification task to two classes: images of Tide laundry detergent or images of a different

[4]https://github.com/dthiagarajan/grozi_tf/

product. The images in the second class were drawn uniformly at random from each of the 119 other classes.

In the training phase of this network, we employed basic data augmentation such as random cropping and scaling. Even with this, the dataset was still small and we opted to mix in some of the cropped video stills from the *"in situ"* set. The rest were reserved for testing and validation. Since doing this shrunk the testing set, we decided to employ data augmentation on that set as well.

The results will be discussed in the next section but in summary, the network overfit the data significantly and ROC curves showed poor performance.

We also attempted to adapt an AlexNet architecture with the same parameters as above with similar results.

2.3.2 Fine-tuning with Inception-v3

In an attempt to improve the performance with such a small training dataset, we decided to fine-tune Google's Inception network which was trained on the ImageNet dataset. Transfer learning has been shown to improve recognition especially on small datasets [3]. We also decided to look at all 120 classes and plot ROC curves in a one-vs-rest fashion.

The results were disappointing again and we realized we needed a new strategy. Adding complexity to the model increased overfitting but less complex models were not accurate enough.

2.3.3 Update GroZi Dataset

The first idea we had was to improve the GroZi dataset, adding new images for each category and updating discontinued products. The plan was to make the dataset more viable for deep learning with hundreds of images per class.

As noted earlier, however, it was difficult to obtain enough data without purchasing the items and capturing images ourselves and therefore this method would not scale well to new items.

2.3.4 Recognition with Few Training Images

At this point, we returned to the original problem of recognizing products from just a few *"in vitro"* images. First, we tried increasing the pre-processing and live data augmentation to mimic new data. This involved augmenting training data with randomized backgrounds and pose variations in addition to generic cropping, rotation, scaling, and photometric transformations [2, 8, 9]. We also investigated leveraging features at different scales (and thus more data per image) like more traditional computer vision techniques [12].

We also investigated a promising study which involved pre-training on rotated objects to learn novel poses with only a few training images [6]. We chose this strategy, along with the data augmentation procedures mentioned, with which to proceed.

3 Results

3.1 CIFAR-10 Network

Our first experiment which trained on the adapted CIFAR network to recognize binary classes (Tide laundry detergent or not) had poor performance. While it achieved an accuracy of around 96% even with overfitting, this was mostly because the testing images so closely resembled the training images. The ROC curve reveals deeper problems and that the confidence for each image was low.

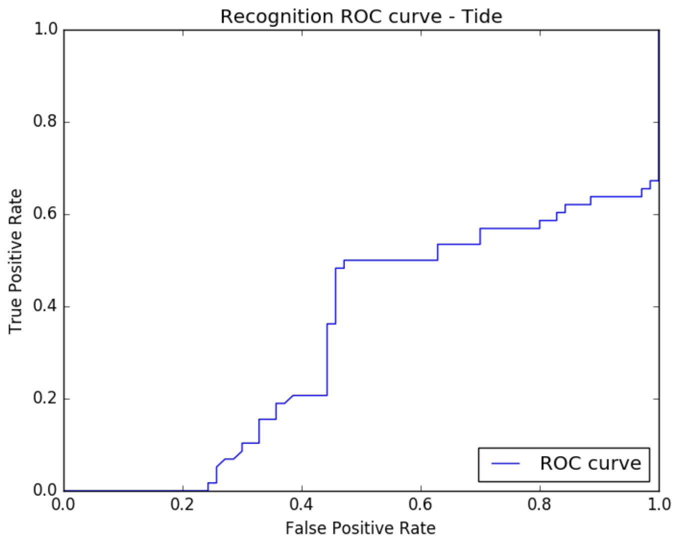

Figure 1: Tide vs not-Tide from CIFAR-10 network with distorted training and testing data

3.2 Fine-tuning with Inception-v3

From the next figure, we can see that the fine-tuning did not significantly improve the ROC curves for the binary classification case.

After generalizing to all 120 classes, the curves still demonstrate underwhelming performance. Most looked like figure 3 below.[5]

4 Discussion

After experimenting with simple networks such as the CIFAR network and retraining the Inception network, we set about writing our own networks to test. TensorFlow makes some tasks extremely easy and others unnecessarily complex. In addition to making development slow, it made the networks difficult to understand as a reader and so we turned to a higher-level API to simplify

[5]For more ROC curves, see https://github.com/dthiagarajan/grozi_tf/blob/master/roc_curves

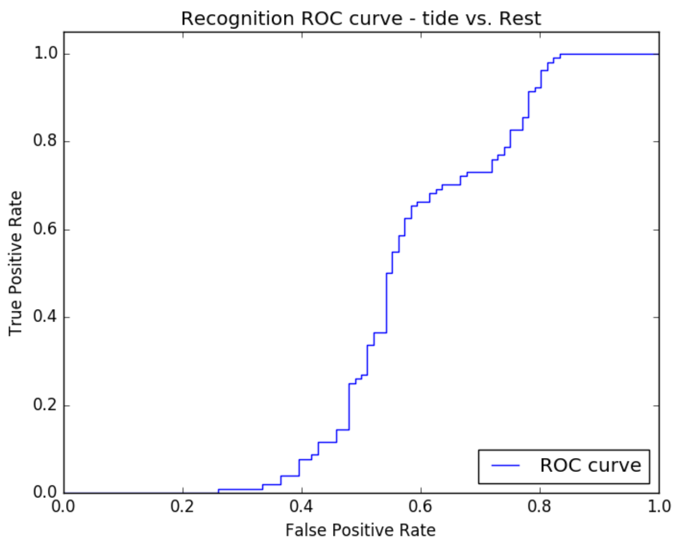

Figure 2: Tide vs not-Tide from fine-tuned Inception network with distorted training and testing data

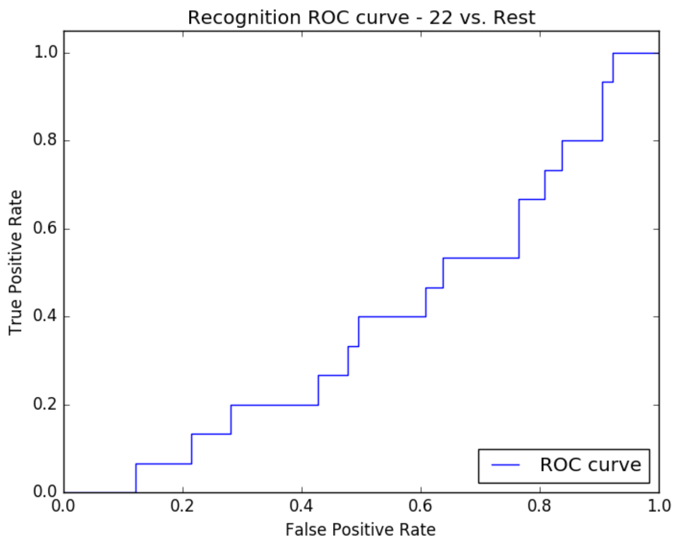

Figure 3: Dentyne Ice gum vs all other products from fine-tuned Inception network, a representative example of the ROC curves obtained from this network

the structure of our code. There were a few options since TensorFlow was open-source and fairly new and at the time of our study, there was no clear popular favorite. We chose TFLearn which seemed to be very functional and extensible.[6]

This ended up causing a few problems when we tried to use our networks for experimentation. While TFLearn allows for the restoration of weights and variables, it uses a format which requires the training to be done using TFLearn also. Since training on large datasets such as ImageNet requires many GPUs or a lot of time, we were limited to using networks for which there were pre-

[6] http://tflearn.org/

trained weights, which at the time was only VGG-16. This in turn caused problems since VGG-16 would use more memory than was available on our AWS instance and would crash.

In the future, we would recommend writing functions to automate common TensorFlow tasks without using a third-party API to make development easier or using a more mature API. Although we did find a tool for converting pre-trained Caffe models to TensorFlow, we were unable to test this tool ourselves.[7] Furthermore, while Amazon Web Services does make training with a GPU easy, the instances are limited to 4GB of video memory and do not always offer large speed improvements.

For continued research, we recommend starting by looking at the Held study [6] and researching other attempts at deep learning with small training data sets. The field of unsupervised learning is also growing rapidly and so more research in that direction seems promising and might yield some insights for small datasets.

[7] https://github.com/ethereon/caffe-tensorflow

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. *CoRR*, abs/1405.3531, 2014.

[3] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. *CoRR*, abs/1310.1531, 2013.

[4] D. G. García, M. Paluri, and S. Wu. Under the hood: Building accessibility tools for the visually imparied at facebook, 2016. Accessed 8-September-2016.

[5] M. George, D. Mircic, G. Sörös, C. Floerkemeier, and F. Mattern. Fine-grained product class recognition for assisted shopping. *CoRR*, abs/1510.04074, 2015.

[6] D. Held, S. Thrun, and S. Savarese. Deep learning for single-view instance recognition. *CoRR*, abs/1507.08286, 2015.

[7] Z. Lipton. Tensorflow is terrific – a sober take on deep learning acceleration, 2015. Accessed 20-June-2016.

[8] I. Masi, A. T. Tran, J. T. Leksut, T. Hassner, and G. G. Medioni. Do we really need to collect millions of faces for effective face recognition? *CoRR*, abs/1603.07057, 2016.

[9] N. McLaughlin, J. M. del Rincón, and P. C. Miller. Data-augmentation for reducing dataset bias in person re-identification. In *AVSS*, pages 1–6. IEEE Computer Society, 2015.

[10] M. Merler, C. Galleguillos, and S. J. Belongie. Recognizing groceries in situ using in vitro training data. In *CVPR*. IEEE Computer Society, 2007.

[11] B. Sapp, A. Saxena, and A. Y. Ng. A fast data collection and augmentation procedure for object recognition. In *AAAI*, pages 1402–1408. AAAI Press, 2008.

[12] Y. Zhang, L. Wang, R. I. Hartley, and H. Li. Handling significant scale difference for object retrieval in a supermarket. In *DICTA*, pages 468–475. IEEE Computer Society, 2009.

A Appendix

Project repository:

https://github.com/dthiagarajan/grozi_tf/

GroZi-120 dataset:

http://grozi.calit2.net/grozi.html