Distributed Ensembles for Image Classification

Andrea Burns Gavin Brown Iden Kalemaj

December 9, 2019

1 Links

- Github link: https://github.com/gavinrbrown1/distributed-ensemble-project
- Slice name: Project5

2 Introduction

The general goal of image classification is to determine which of several fixed classes an image belongs to. One common benchmark is the CIFAR-10 dataset [1], which contains 32×32 pixel images of ten different classes: airplanes, automobiles, birds, cats, deer, dogs, frogs, horse, ship, truck. We've included a selection of these images in Figure 1. Image recognition is an important machine learning task, as it is an initial step toward visual scene understanding – a skill needed to interact with and reason about an environment (use cases include robot assistants, autonomous vehicles, etc).

Ensembling is a popular way to improve the performance of a machine learning task; an ensemble is a set of different classifiers that are 'combined' to make a collective, better decision on the task. For a reference see Chapter 14 of Bishop's *Pattern Recognition and Machine Learning* [2].

In this experiment we distribute 4 independent classifiers across different network nodes (servers) to be used as an ensemble, similar to bagging, in which we take a majority vote of their decisions and use that as the final classification result. This distributed system allows

Figure 1: A few images from CIFAR-10. From left to right: a dog, a truck, and an airplane.

us to investigate latency versus performance benefits when introducing additional classifiers to the ensemble. Lastly, we try hand-crafted caching tools that can act as short-cut decisions, to see the impact on end-to-end delay.

A client wishing to receive classification for an image connects to a manager and sends the image to the manager. The manager then sends copies of the image to the classifiers.

3 Experimental methodology

3.1 Caching and Sampling

When choosing how to test the performance of our system, we must specify how we select images to be classified. One method is to sample them uniformly at random from our test set, which matches how machine learning models are usually evaluated. This may not reflect how the system is used in practice: some images may occur far more than others, such as a automated security system that spends much of its time examining the same empty hallways. To model this we also sample from a prototypically "long-tailed" distribution, a simple form of the Zipf distribution discussed in Chapter 16 of [3], where each image i is assigned a weight $w_i \in 1 \ldots, N$ and is sampled with probability $p(i) \propto \frac{1}{w_i}$. Thus some images will keep reappearing while others will be very rare.

4 Results

4.1 Usage instructions

4.2 Analysis results

5 Conclusion

6 Division of labor

Andrea, Iden, and Gavin decided on the project topic and wrote the proposal and report together. All high-level decisions were made together: network topology, communication protocols, experiments, etc. Much of the coding was done in pairs, as well. The focus of specific sections were as follows: Andrea created the code to control caching. Iden wrote the bulk of the socket code that allowed the machines to communicate. Gavin wrote the code to handle the prediction as well as modeling error and delay on the part of the classifier.

References

[1] Krizhevsky, Alex, and Geoffrey Hinton. Learning multiple layers of features from tiny images. Vol. 1. No. 4. Technical report, University of Toronto, 2009.

[2] Bishop, Christopher M. Pattern recognition and machine learning. Springer, 2006.

[3] Mitzenmacher, Michael, and Eli Upfal. Probability and computing: randomization and probabilistic techniques in algorithms and data analysis. Cambridge university press, 2017.