Hyperboard: Tracking productivity in edge-based image discovery with Eureka

Kai Wen Wang 1 Ziqiang Feng 1 Shilpa George 1 Mahadev Satyanarayanan 1

Abstract
Eureka is an edge-based interactive distributed system that improves the productivity of humans when labelling data. While it was shown to improve productivity over brute-force approaches by up to 100x on Deer, Taj Mahal, Fire Hydrant, it does not provide any signals to the user on whether Eureka is used correctly. In particular, the number of cloudlets is chosen and models could get worse overtime, dropping increasing number of false negatives and decreasing productivity. Therefore, it is critical to track productivity to diagnose these issues when Eureka is deployed in the real world. In this project, we built Hyperboard, which transparently tracks statistics (e.g. productivity) at fine granularity and provides other diagnostic features such as branching from previous sessions and replaying entire sessions. Hyperboard is incorporated into and is started automatically with Eureka. Our experiments illustrate the uses and usefulness of Hyperboard.

1. Introduction
Most of the recent success in machine learning has been from the realm of supervised learning, where it is necessary to have a large labeled dataset. However, getting quality labels is often laborious and expensive. Also, there is a large quantity of unlabeled data, which would be nice to put to good use. Eureka (see Figure 1), an edge-based image discovery application for labeling positive images, addresses both of these problems (Feng et al., 2018).

Previously, tracking productivity and other statistics during a session in Eureka was laborious, error-prone and distracting from the main task of labeling images. These problems would be solved if there could be an automatic way to track and visualize statistics. In this research project (my 15-591 Independent Study in Fall 2019), we built Hyperboard, a tracking and visualization tool for Eureka that gives both a high-level summary (see Figure 2) and live statistics plots for each session. In addition, Hyperboard enables restarting from a previous checkpoint and replaying the images from previous sessions in the order that they were shown. This allows for reanalysis of past sessions’ images and labels, which is particularly useful for mission-critical datasets (e.g. x-ray images containing some disease) where multiple expert-opinions are needed.

Hyperboard is now transparently incorporated into Eureka, meaning that there is no additional work during sessions. Hyperboard starts running automatically behind the scenes when a Eureka search begins. The only added step is to supply an optional command line argument specifying a path for storing the logs, which Hyperboard will read. If not given, Hyperboard will be disabled. Once Hyperboard is enabled, the user can access the visualizations at localhost:5000 in a web browser.

In summary, we built Hyperboard, a tracking and visualiza-
Hyperboard: Tracking productivity in edge-based image discovery with Eureka

2. Preliminaries

2.1. Eureka

In this section, we describe specifics of how Eureka searches and sessions operate. The task Eureka addresses is when human labelers (a.k.a. experts) search through a large pool of unlabeled data (e.g. YFCC100M dataset) for specific (often rare) objects. Eureka improves labeling productivity by using early discard with filters on edge cloudlets, which are placed as close to the data source as possible for low latency and high bandwidth processing. Experts will only be shown images that the filters could not confidently classify as negative. In Note that false negatives, though should be rare, will be dropped unnoticed and it is not possible to calculate recall since we do not know the ground truth. This means that filter quality is a significant factor into the performance boost gained from using Eureka.

Eureka searches are broken down into sessions, where each new session should ideally have better filters since the user can pick and train the filters using domain knowledge and labeled data from all previous sessions. Therefore, the common pattern is to start off with simple filters in the initial session and to train more complex and precise filters in later sessions. Having a more precise filter often leads to higher productivity since only very relevant images make their way to the user. Searching through a dataset with 100M images for an object with 0.05% base rate was a task that seemed impossibly laborious before, but now could be done in less than half an hour using Eureka.

2.2. Productivity

We define productivity of a session as

\[ \text{productivity} = \frac{\text{new}_\text{hits}}{\text{time}_\text{elapsed(min)}} \]

where new_hits is the number of positive labels given in a session and time_elapsed(min) is the length of the session in minutes (Feng et al., 2018).

Without Eureka, the task of searching a rare object in a large pool of images has low productivity because there are very few new_hits – most images are not the desired object and the expert wastes ample time on filtering negatives. Eureka aims to increase new_hits by providing precise filters, so the images shown to the user can be mostly true positives. However, there are many ways that the productivity can decrease even when Eureka is used. Some examples are:

- Computational Bottleneck: if the number of cloudlets in use is low and/or the pass rate of the filters is low, then the user would be constantly waiting for new images to come. This can be solved by adding more edge nodes.
- Model Degradation: if the precision gets worse, then the user will be presented with more junk images. If recall gets worse, then the number of true positives (assuming the distribution of objects in the datasets is constant) will decrease, leading to fewer new_hits. This can often be caused by lower quality data from the last session or a badly retrained model. This can only be solved with better filters.
- Distributional Shift: this is similar to the second point, except instead of the model degrading, the distribution of streaming input data has changed. Thus, the previously accurate model is no longer appropriate. In this case, the user should start from simple models and progressively build up to adapt to the new domain.

For the first case, the number of cloudlets is an important but difficult hyper-parameter to determine. It is generally not easy to find the right number, especially without guidance from Eureka. It is hard to detect the second case at train-time since we cannot calculate the true recall of a filter. Finally, it is impossible to predict whether a distributional shift in the input data and the affect it will have on our filters. However, the core quantity we care about is productivity, so if the user has a regular and precise measurement of productivity, it is easy to be alerted. Tracking and visualizing other statistics, such as precision and pass rate, can be useful for debugging.

2.3. Other visualizers

In many applications involving user interaction, having a transparent visualization framework can significantly improve usability. Using tools without these frameworks would be akin to driving a car without a dashboard. There are plenty of such tools, some more involved to use than others, such as Jupyter Notebook, Apache Zeppelin and Tensorboard. The most relevant to our work is Tensorboard, which is widely used for Tensorflow and PyTorch users to help track and debug training and performance of deep learning models. We took some inspiration from their design, in terms of usability and tracking methodologies, such as how logs are stored and how users start-up Hyperboard.

3. Method

Figure 3 shows the high-level design of Hyperboard and how it interacts with Eureka. The implementation of Hyperboard consists of two main components:

1. modifying Eureka, in particular Hyperfind and Open-Diamond, to log useful statistics.
2. writing a web app to read statistics and serve to the browser

Since the task of logging and reading from a log is not highly speed sensitive, we chose the simplistic approach to store logs to the file system. Currently we track several statistics: items processed, items shown, new hits, pass rate, precision, and productivity.

We extended Hyperfind with a Hyperboard logger class to log necessary information at different times of a session:

- **Session endpoints**: We track the start and end times of the session. At session’s start, we also export predicates to enable branching and restarting previous sessions. At session’s end, we also record the final statistics at the end of a session.

- **Image arrival**: Since statistics are displayed in image-level granularity, we log statistics, image attributes and thumbnail, every time an image arrives to the user.

- **User feedback**: We track an append-only log of user feedbacks, containing time, image ID and the feedback itself. Although this can take more space since ultimately the last feedback for an image is the final decision, we decided to track all user interactions since it gives us a precise history of events for the search.

In addition to live statistics, Hyperboard enables restarting from previous sessions. This is possible because we store the predicate metadata at the beginning of each session, and retrieving them is as simple as clicking a button in Hyperboard’s front-end. This metadata can be directly imported into Hyperfind to start a session with an identical predicate structure.

Also, Hyperboard provides session replays, which shows image thumbnails of a session at the same order that they were shown to the user. The labels which the user gave to each thumbnail is also visible (Positive as green checkmark and Negative as red cross).

Hyperboard’s web app is implemented using Flask and its visualizations are written in JavaScript with D3.

### 4. Experiments

We searched on YFCC100M augmented with Oxford Flowers for lotus flowers, which had an overall base rate of 0.05%. We conducted two searches, first with eight cloudlets and again with only one cloudlet. For fairness of comparison, both searches used the same training images for the DNN + JIT SVM filter. It should be noted that these images were obtained from the eight-cloudlet search, since it is a lot faster than the one-cloudlet search to gather a reasonable amount of images (at least 20) in each session. We started the search with six images of lotuses and six negative images from Google Images. Our results for this search are summarized in Figure 2.

![Figure 4. Precision plots during searches for Lotus.](image)

![Figure 5. Productivity plots during searches for Lotus.](image)

Figure 4 and 5 show comparisons of precision and productivity of these two searches. As we expect, the precision plots of the filters are very comparable, since the filters are...
trained on the same set of images. In both cases, the precision grew, starting at $\approx 1\%$ and ending at $\approx 10\%$, which is indicative of less junk for the user to go through in later sessions.

However, in the productivity plots, we can clearly see that not only is the eight-cloudlets case initially $\approx 4 \times$ better than the one-cloudlet case, the eight-cloudlets productivity gets progressively higher as the model’s precision improves. However, the one-cloudlet experiment is *computationally bottlenecked* and the productivity remains low – the user is constantly bored, waiting for new images to come in. Using a tool like Hyperboard, the user can easily compare the productivity plots between sessions and search for a good number of cloudlets that achieves a sufficient but not excessive amount of computation.

5. Reflection

As part of the Independent Study in Computer Science (15-591) course, this research project was definitely a rewarding experience overall: working first-hand with Eureka and working with my mentors Edmond, Shilpa and Professor Satya, I was exposed to the principles of distributed systems research that prioritized robust, performant, clean and generally production-level code. Having designed and built Hyperboard, now a part of Eureka, I worked closely with edge-computing, witnessed several of its practical applications and thought and worked on addressing inherent trade-offs such as compute vs. attention vs. cost. In particular, witnessing how having the right setup in Eureka can achieve a 100x boost in productivity from the brute-force approach is very exciting. In the process, I also gained other useful skills such as learning to make and deploy a visualization web app. Most importantly, this research project has reaffirmed my interests in distributed systems and its importance to the world now and especially in the near future.

There is one thing that I would have done differently if I were to do this again: get user feedback early. When designing a tool to simplify the lives, I realized it is actually important to get their feedback, and this takes time. Throughout the project, I was my own guinea pig by using Hyperboard but I quickly got used to the user interface. As a result, the user interface is probably not optimal and certainly does not look as nice as Tensorboard’s. Also, user feedback could have inspired supporting other useful functionalities unknown to me right now.

This research project was especially timely since I was reflecting and focusing my research interests for choosing and applying to Ph.D. programs. While I really enjoyed writing robust systems code for user-facing applications, I also enjoyed designing state-of-the-art machine learning algorithms and understanding what makes them tick to give provable guarantees. Having witnessed both research fields, I am confident that we live in an exciting time for fruitful collaboration between machine learning and distributed systems, which could give the 10x-100x performance boost that some elusive technologies, such as self-driving cars or multitask learning, need to become a reality.
References