BeaverDam: Video Annotation Tool for Computer Vision Training Labels

by

Anting Shen

A thesis submitted in partial satisfaction
of the requirements for the degree of

Master of Science

in

Engineering - Electrical Engineering and Computer Sciences

in the

GRADUATE DIVISION

of the

UNIVERSITY OF CALIFORNIA, BERKELEY

Committee in charge:

Professor Kurt Keutzer, Chair
Professor Only Somewhat Important Guy

Fall 2016

BeaverDam: Video Annotation Tool for Computer Vision Training Labels

by

Anting Shen

Abstract

BeaverDam: Video Annotation Tool for Computer Vision Training Labels

by

Anting Shen

Master of Science in Engineering - Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Kurt Keutzer, Chair

Your abstract should be limited to 350 words. If it is longer, ProQuest will truncate and/or edit it so that it is less than 350 words. This is a very boring dissertation that nobody will probably ever read.

Blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah. Blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah. Blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah. Blah blah blah blah blah blah blah blah blah blah blah blah blah blah. Blah blah blah blah blah blah blah blah blah blah blah blah blah blah. Blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah. Blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah. Blah blah blah blah blah blah blah blah.

Blah blah blah blah blah blah blah. Blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah. Blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah. Blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah. Blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah. Blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah. Blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah. Blah blah blah blah blah blah blah blah blah

blah blah blah blah blah blah. Blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah.

And so it goes...

Professor Kurt Keutzer
Thesis Committee Chair

Contents

Chapter 1

Introduction

Deep learning applications in recent years have come to require rapidly growing amounts of labeled training data. Often, accuracies can be boosted by adding data as much as by spending years on algorithmic development. For example, on the VOC07 benchmark, Faster-RCNN [1] with VGG-16 was able to eliminate 27.5% of errors in the much older R-CNN [2] backed by an equally old neural network architecture (mAP improved from 58.5 to 69.9). However, simply by including additional data from VOC12 and COCO, 29.5% of the remaining error was eliminated (mAP improved from 69.9 to 78.8). Therefore, for real-world application development, data can be cheaper and more effective than scientists. While many existing tools support image classification – it is even built into Amazon Mechanical Turk (MTurk) – and some tools support bounding box labeling in images, few tools exist for frame-by-frame labeling in videos. VATIC [3] stands out as being one of the best, as not only does it make high quality annotations one of its main goals, but also cost and scalability.

My work borrows and improves upon many concepts and results from VATIC's user studies, but I focus on an additional goal that is extremely important in creating datasets for real applications. That goal is researcher happiness. Although VATIC extensively tested its "User Interfaces", I argue in chapter 2 that both the annotators and the experimenters are users, and the interfaces should be smooth for both when creating a tool.

Then, in chapter 3, I discuss my take on VATIC's User Interface principles for the annotator, and improvements upon them.

I also release all related code for BeaverDam, my video labeling platform, on Github.[1]

Related Work

Static image annotators

Vatic, LabelMe, etc

Things other people cite

[1] http://github.com/antingshen/beaverdam

Chapter 2

Experimenter Interface

During our investigation, we had many frustrating experiences with existing research tools in this area. These included installation issues, configuration issues, and dealing with unintuitive and undocumented interfaces. For example, there are dozens of open issues without solutions on the VATIC GitHub, and its installation script fails at multiple points on a brand new Ubuntu box. Due to preciousness of researcher's time, we believe the ability for researchers to test and iterate quickly is just as important as worker speeds when it comes to video annotation. Therefore we have placed improving the experimenter's experience as one of the main goals of this work.

2.1 Interface for researcher

For a basic user creating a crowdsourced video dataset using the default configurations we used, BeaverDam provides a streamlined interface. We provide a setup script that is tested on clean installs of Ubuntu 14.04 and Ubuntu 16.04. In comparison, VATIC was tested on an unspecified Ubuntu installation with exact Apache and MySQL installs, and while the install script claims that it should work on any operating system in theory, installation is difficult in reality. Additionally, our install script configures everything from Nginx and TLS to database config and backups. The user only needs to place their keys and

credentials in the locations specified in our documentation. While a containerization system such as Docker would have also solved VATIC's issues and ensure future compatibility, we felt that the additional complexity is not worth it, as many of our users in the research community are unfamiliar with Docker.

To use BeaverDam after installation, we provide a web interface for researchers to easily add and view videos and jobs. We feel that this is superior to VATIC's command line based approach, as the number of flags needed to specify various configurations was overwhelming. However, to allow experimenters to load large number of videos or perform other tasks programatically, BeaverDam also provides a Python shell interface, backed by Django, to expose every functionality through Python.

Lastly, as BeaverDam is HTML5 video based, no frame extraction step is necessary. H.264 encoded videos will work without preprocessing. However, we do provide scripts to convert these videos to images with matching annotations to feed into machine learning tools if desired.

2.2 Decoupled modules

As the users of BeaverDam will most likely need to modify BeaverDam to fit their needs, we've emphasized modularity in our designs. Since modularity is something VATIC did well with their turkic, vatic, and pyvision splits, we've taken a similar approach. Our infrastructure is split between a crowdsourcing module, an annotation module, and a CV-based tracking module. But we decided to go a step further and make other parts of our platform replaceable as well.

To serve videos, we've provided Nginx to allow VATIC users to continue serving videos efficiently on the same server as their application. But to allow for scalability, users can use AWS S3 or any other CDN, or even a mix, by specifying so when creating jobs. Similarly, our setup pipeline is done through Ansible, a configuration management tool popular in industry. This allows users to deploy on their own servers, or in the cloud.

We also understand that not everyone wants to use Amazon Mechanical Turk. While past research has proven Mturk to be efficient and reliable, companies in industry seeking training data may prefer alternatives such as CrowdFlower, or may even choose to label in-house, or outsource to contractors. BeaverDam is designed with this flexibility in mind as it carries its own authentication and works independently of Mturk. This allows companies with the resources to seek cheaper labor in other countries to use BeaverDam as a platform for their workers and track progress and pay without the need for Amazon Mechanical Turk, which carries a 20% fee.

2.3 Patterns

To facilitate modification of our code, we adopted the Django framework because it sets fairly strict conventions on how a project is structured. Someone familiar with Django will immediate know where to find the files responsible for each function of the backend. As there is a fair bit of frontend logic to video annotation, we've organized the code responsible according to well defined patterns as well. Under Our Pattern Language [**?**], the frontend follows the Model-View-Controller structure, and each of the three components is event-based. While using events to communicate within an application may seem like additional complexity at first, but it enables decoupling and extends the existing events from user interaction.

2.4 Dependencies

The number one problem we encountered during our installations of VATIC was broken dependencies. Its GitHub issues point to problems users are having due to MySQL, cython, ipinfodb, and gcc versioning. BeaverDam addresses this problem in two ways. First, we greatly reduce the number of dependencies. Python 3 and two python packages installed through pip are the only dependencies required to run BeaverDam. Three other dependencies, Nginx, supervisor, and uwsgi, are recommended for efficient deployment. And to avoid

the burdensome installation and configuration of a database that VATIC required, we use sqlite3, which is a Python standard library. In this case, it proves to be enough to handle even a fairly large amount of annotations, and can be swapped out easily if needed. Second, we version each dependency. This ensures that newer versions of dependencies in the future cannot break compatibility. For front-end libraries, we check-in the required files directly instead of using a package manager, avoiding extra complexity.

Another problem we encountered with VATIC's dependencies was that since they are installed and configured system-wide, different users sharing a server would often leave bad state for others, causing hard-to-track bugs. We attempt to solve this by including an idempotent script that verifies all required state, and fixing them if necessary. Additionally, our system is designed to be quickly set up on new VM instances, so one can perform upgrades and fixes by discarding the entire server VM and starting from scratch.

When choosing our dependencies, we strived to use the latest versions of mature technologies. We use Python 3.5 and Django 1.10, the latest versions available at the time. We also use the ES6 version of JavaScript, which includes many features to enhance readability and programmer happiness. While alternatives such as TypeScript provides additional features, ES6 is supported without a transpiler and avoids complexity. Browser compatibility of ES6 was a small issue with workers, but we expect this to improve as browsers fully adopt the standard and workers update to the latest version of their browsers.

2.5 Security

DB Backup

Authentication enables decoupling, web admin

Other standard procedures, HTTPS & HSTS, CSRF, clickjacking

Chapter 3

Annotator Interface

Intro & our method of user studies (not as good as VATIC though)

3.1 Keyframe scheduling & Multiple object annotation

Keyframe viewer when on custom schedule

3.2 Video playback for maintaining identity

Importance of playback

Video loading issue

HTML5 video advantages

3.3 Reducing clicks

Create by default

Object selector

Keyboard shortcuts

3.4 Handling frame exit/enters

Border padding

Allow out-of-border boxes

3.5 Micro vs Macro tasks

Comparison from existing literature

Proposal advocating for micro-tasks in video labeling

Extensible task structure

3.6 Interpolation & Tracking

Chapter 4

Conclusion

Ha! I'm done!

References

[1] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[2] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Computer Vision and Pattern Recognition*, 2014.

[3] C. Vondrick, D. Patterson, and D. Ramanan, "Efficiently scaling up crowdsourced video annotation," *International Journal of Computer Vision*, pp. 1–21, 10.1007/s11263-012-0564-1. [Online]. Available: http://dx.doi.org/10.1007/s11263-012-0564-1