Object Detection in Videos

Akash Gaonkar
University of Colorado
akga1284@colorado.edu

Jacob Munoz
University of Colorado
jamu0075@colorado.edu

Avery Olson
University of Colorado
avol4799@colorado.edu

1 Problem Statement/motivation

The contents of a video are clear to any human that watches, but is it possible for a computer to understand what it is seeing? Can an algorithm learn to recognize objects in a video and alert humans of a potential threat? These are the questions we hope to answer by training and tweaking an algorithm that can recognize objects in each frame of a video. If a computer could identify threats reliably it could help prevent violent events before they occur. While security agencies (such as those at the airport) do their best, people make mistakes and may miss a harmful weapon while watching a busy terminal. Having a computer to watch alongside people can confirm a human's suspicion or alert them of an unknown threat.

2 Literature survey

In March of 2017, Google announced its Video Intelligence API. With this, developers can make applications for searching for specific objects in videos and tagging scene changes. It automatically detects objects in videos, which before this, only could still images could be used for object detection. It can also recognize words and characters of several languages, detect major landmarks and logos, and has a component called safe search, which filters inappropriate content. The safe search feature does not give the percentage but says very unlikely, unlikely, possible, likely or very likely instead. The Video Intelligence API uses Google Cloud Storage to hold the images and videos. To the right of the video/image it will give you the percentage probability an object is in it.

Similar to this is a website called Clarifai, which does that same with the percentage likelihood of an object in the image/video, filtering inappropriate content, and having the ability to tag and search the objects found.

3 Proposed Work

To begin, we plan on replicating what has been done so far, and if we can, make it faster. First we need to pre-process the data so that we can work with it, and then the three main things to do are identifying objects, searching objects, and filtering inappropriate content.

3.1 Data Preprocessing

3.1.1 *Scaling*. The first step is to rescale the images to be of consistent form. It is important for all the images to be lined up exactly so the algorithm can compare images pixel by pixel without having to worry about varying image sizes.
3.1.2 *Outliers*. The next step is to clean the data by removing infrequent classes (image with objects). If an object only appears a few times within millions of images it is best to remove said image to improve the learning algorithm. Leaving outliers in the training

data may actually decreases the overall performance of the algorithm.

3.2 Algorithm Training
Having acquired a sufficient corpus of labeled images, we will partition it into training, validation, and test datasets. The training set is for teaching our classifier to properly predict bounding boxes and object labels, while the validation set is for tweaking classifier parameters and the test set is for comparing classification algorithms if we have time.

We have several options for our project's object-detector. Aside from the built-in classifiers described later (in our Tools section), we can choose between various ML techniques and systems. Deep learning is the current champion at object detection, but even within that category there are several algorithms. We're particularly intrigued by YOLOv2 (You only look once), which can process up to 45 frames per second under the right conditions, fast enough to detect objects in real time.

3.3 Evaluation
After running the algorithm through the training dataset it is important to check for reliability. One example is to ensure the algorithm can recognize a chair from multiple angles or lighting conditions. After the initial training has proved successful can we push the algorithm to recognize more obscure objects such as a man holding knife.

4 Data set
Open Images - 9 million imges - https://github.com/openimages/dataset
The data contains URLs to images that have been annotated with image-level labels and bounding boxes spanning thousands of classes. All of the image-level labels are automatically generated by a computer vision model similar to the Google Cloud Vision API mentioned above. This data is split into three sets, the training set, the validation set, and the test set. The labels in the validation and test set are all human-verified image-level labels (done by Google annotators) as well as some in the training set. And the bounding boxes have also been manually drawn by Google annotators for all the human-verified image-level labels.

ImageNet - 14 million images - http://image-net.org/index
This data contains images for nouns, organized into a tree-like structure to become more specific leading the images for the more specific category.

5 Evaluation Methods
We will test videos in our version and compare the results to the Google Cloud Vision API and the Clarifai website. This will tell us how close or far we are from the current standard and if we can get more accurate results than these we know we have made it better.

6 Tools
We plan to use Python3 as the main language for our project, and Numpy will be used to help us properly format our images. From this launching point, we have various toolsets depending on how we choose to focus our project.

The simplest way to get object detection running would be to hook up an existing tool, such as Google Cloud Vision API, Tensorflow Object Detection API, or perhaps Darknet, a C program that would make YOLO available from a command line interface.

In this case, we'd want a toolset useful for building something on top of object detection, so we might use a Python/Flask backend serving pages with a Polymer 2 or Angular 4 frontend, maybe with npm and bower for dependency management.

Alternatively, we could focus on training our own instance of one of the neural net algorithms, creating each layer and running the images through our model. This would involve us much more deeply with the actual construction of an object detection tool, and so we'd be more inclined to use pure Tensorflow (as opposed to the apis it makes available). This would make us rely much less heavily on the web toolkit above, since we'd be creating a more generic detector rather than an application.

7 Milestones

We would like to try and pursue an agile-ish methodology with our project, and so have made detailed plans for near timepoints, but would like to be more flexible after that.

Starting 3/11, there are approximately 7 weeks before the end of the project. We will break the project up into three phases. Phases 1 and 2 will focus on getting object detection working by both pre-processing our data and training our algorithm. We expect to iterate over the two phases several times before arriving at something stable, so this process should take 3.5-4 weeks (milestone on 4/2). The remaining 3 weeks, phase 3, are reserved for taking a next step to our tool, be it finding an application or focusing even more on streamlining our detection algorithm.

Phase 1's first tasks are to 1) find a way to loop through our dataset, since we can't just download every image, 2) properly shape each image to have the same dimensions, 3) test existing object detection tools and see if we can get a working system prior to integrating it with our dataset.

8 Summary of peer review session

The preparation for the peer review forced us to focus on how we wanted to use object recognition. The positive feedback showed interest in our questions and motivated us to find meaningful answers. Moving forward we intend to continue with our original plan of a learning algorithm to attempt to identify potential threats within video.

9 Citations

https://techcrunch.com/2017/03/08/googles-new-machine-learning-api-recognizes-objects-in-videos/

https://cloud.google.com/blog/big-data/2016/09/experience-googles-machine-learning-on-your-own-images-voice-and-text

https://cloud.google.com/blog/big-data/2016/08/filtering-inappropriate-content-with-the-cloud-vision-api

https://research.googleblog.com/2017/06/supercharge-your-computer-vision-models.html

https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Redmon_You_Only_Look_CVPR_2016_paper.pdf

https://pjreddie.com/darknet/

https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-i-hello-world

https://www.polymer-project.org/about