YouTube Videos Object Detector: Can you spot the teddy bear?

04 - Team RGB

Ridha Alkhabaz
University of Illinois Urbana-Champaign
ridhama2@illinois.edu

German Bautista
University of Illinois Urbana-Champaign
gbauti5@illinois.edu

***Abstract*—In this project, we built a framework to detect desired objects in YouTube videos. Our first experiment was focused on detecting an iconic teddy bear from a famous YouTuber's videos called KSI. Afterwards, we attempted to use or partially build a network architecture that would learn features associated with objects that were annotated in a crowdsource fashion. Finally, we produced a Google Colab notebook where people can build a YOLOv4 network to detect any objects of interests. Although we did not achieve the accuracy we desired for video files, our project and attempts yielded interesting results for object detection in YouTube videos.**

***Index Terms*—YouTube, Object Detection, Video Annotation.**

I. Introduction

We were interested in building a deep learning framework to detect objects in YouTube videos. We used videos from the popular YouTube channel *KSI* to build a binary classifier that could detect teddy bears.

We were interested in this problem because it has a wide array of applications. For example, many social media platforms are struggling to automate the process of flagging controversial or sensitive material [1]. Hence, a fast object detector can be used on YouTube videos to ease this process. Moreover, detecting dangerous or hazardous objects in public settings is a great national security matter. Thus, having a scalable framework to detect objects of interest in videos might be a good security procedure in airports, banks, and other public or private settings.

Our framework takes as input a YouTube video link, then transforms the video to display labels of desired items. In our case, we wanted our object detector to locate the presence of a teddy bear in the *KSI* video.

II. Related Work

The computer vision research community has been investigating multiple ways to detect objects in videos. We will briefly go over five different papers where their solution took our attention. As mentioned in our deep learning course, You Only Look Once (YOLO) has been a very successful network architecture to detect objects in benchmark data sets, like the Common Objects in Context (COCO) data set [2], [3]. Furthermore, we used Bochkovskiy et al's new object detector implementation in our project.

A. Seq-NMS:

We used an idea similar to the one used in [4] and we cropped the image of the teddy bear in the image frame to help the model learn better.

B. Multi-Label Extension:

There are some problems with only using one label to classify images as outlined in the paper [5]. One is that in video data, a variety of actions can happen simultaneously so one label may not accurately represent the whole frame. On the other hand, more specific labels such as "car running" can eliminate the "semantic relationship" between two instances e.g. a human running and a car running are two quite different actions, but they both are based on the idea of a continuous ongoing process. Understanding semantic nuances is easy for humans, but it is not trivial for machines. Thus, keeping this "semantic relationship" can help machines learn

how to do so. To bypass the stated concerns, the researchers in the paper mentioned offered a multi-label extension to the Moments in Time data set. Similarly, we tried adding a null class to see if our accuracy in video files increases.

C. Video Object Detection:

Two methods that are commonly used in video object detection are sparse feature propagation and dense feature aggregation. Sparse feature propagation is based on the idea that video frames that are close together usually have objects with similar features. We can take advantage of this and speed up the feature network's learning time and lower overall computational cost. However, this can decrease accuracy if we assume too much similarity between video frames. On the other hand, dense feature aggregation is based on the idea that the features of objects in blurry images or images with noise can be better learned by using the features of nearby frames thus improving accuracy but possibly increasing computational cost. Three techniques were proposed to combine the benefits of both methods while reducing their negative consequences; in particular one is spatially-adaptive partial feature updating. [6]

D. Object Proposal

Deep convolutional neural networks (CNN) perform well in object tracking. However, in video object detection, using only still-image object detectors lead to 37.4% mean average precision in some experiments. Using object proposals lead to 45.3% mean average precision. [7]

III. Data

A. Dataset Creation and Preprocessing:

We used the platform Roboflow to create a set of images from downloaded YouTube videos with 720p resolution [6]. Through the platform we created two data sets. The first data set contained three seconds of videos when the teddy bear appears. This data set was used to created a tuned YOLOv4 architecture. We noticed major video data sets usually contain videos of 3 to 5 seconds. Hence, we wanted to match this standard. For testing purposes we created a second data set with a minute of recording where the teddy bear appears in most of the frames.

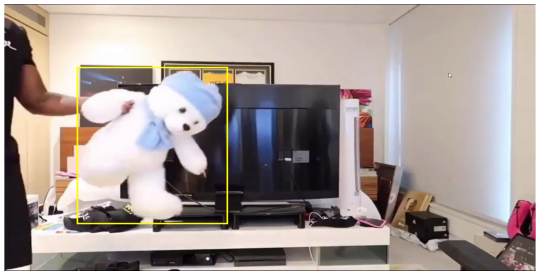

Fig. 1. The teddy bear used in the classification.

Fig. 2. The image used in our validation and testing process.

From each video, we created sixty frames per second and manually annotated frames where the teddy bear appeared. See Figure 1. Afterwards, we resized our inputs to be 416 X 416. We also isolated the teddy bear in order to decrease the noise coming from surroundings in the video frame. See Figure 2.

For the data set [1] used to train YOLOv4, we greyscaled our images in the training data set. This was to force our network to learn geometric characteristics of the teddy bear rather than solely recognizing the contrast between the white and blue colors of the teddy bear. See Figure 1. In order to increase the size of our training set, we randomly rotated our images by fifteen degrees in either direction and zoomed in or zoomed out some frames. To tune YOLOv4, we split the data set into eighty eight percent training, nine percent validation, and three percent testing sets. Our larger dataset was reserved to further test our detectors.

IV. Methods

A. Model Set Up:

In general, object detectors have a backbone, neck, and head. See Figure 6 in Appendix. Back-

[1] the API key for our data set is RZryTInMjR8Idq7lopCC, we hid it here for security reasons.

bones are networks that are large enough to extract features, ours has more than 150 convolutional layers. In our case, this is a convolutional neural network (CNN) called CSPDarknet53. This convolutional network uses a CSPNet strategy to partition the feature map of the base layer into two parts. Then, it integrates these partitions through a cross-stage hierarchy. CSPDarknet53 uses 27.6 million parameters to extract features from photos. The neck collects feature maps from different stages. The head, which is YOLOv3 in our case, performs dense prediction, i.e. a large dense network. Hence, we have more convolutional layers and additional neck objects to extract more features and potentially have higher accuracy. Thus, we have a much more sophisticated representation of desired objects, e.g the teddy bear [2].

Our 416 by 416 images get transformed by many convolutional networks. The representation from these convolutions gets aggregated by the neck's methods in order to capture a very sophisticated representation of our labels. Here we use mean average precision (mAP) as our loss. Mean average precision measures how often the shared area of the ground truth label with the prediction label, divided by the union of these two areas, is above a certain threshold. See Figure 3. We used 0.5 as our threshold and took the mean of the average precision across our validation data set, see Figure 7 in Appendix. We achieved 93.75 percent accuracy on our image test data set. Our model showed robustness since it did not get affected by the orientation or resolution of the images. See Figures 4 & 5.

We trained two models. The first one uses YOLOv4 and has one class of objects (Binary model). The second one also uses YOLOv4 and has two classes (Multi-class model): one of which is a null class and we explain its importance in the discussion section.

B. Transfer Learning:

Since this model was clearly very large to train from scratch, we used pretrained weights from Bochkovskiy et al's paper [3]. There, Bochkovskiy et al trained YOLOv4 on 80 classes of images, one which is teddy bears. Using these weights, we trained on images of the teddy bear and changed the number of classes to one in the Binary model, or two in the Multi-class model.

C. Training Techniques:

Due to prediction issues, we had to increase the number of null objects in the training data sets. Moreover, we had to lower accuracy without our mentioned data augmentation techniques. Thus, we deployed these techniques to increase accuracy. Moreover, we used various levels of null object counts in our training data. However, the existence of more null objects did not have a significant effect on our network accuracy.

V. Results and Discussion

A. Why YOLOv4?

YOLOv4 has proven its robustness and accuracy across various benchmark data sets [3]. Moreover, since YOLOv4 uses milliseconds to predict on frames, so, it is favorable for its expedience. It also is a favorable network of choice among researchers in detection and localization research. YOLOv4 worked fairly in our experience.

B. Why two models?

When we achieved high accuracy in image localization, we wanted to deploy our networks against YouTube videos. However, when the object of interest, the teddy bear, did not appear, the network selected the YouTuber, KSI, instead of the teddy bear. We had two solutions for this problem. First, we added more null objects to our training data sets, which increased the accuracy in image object detection. However, it made the network classify a large portion of the video frame as a teddy bear when deployed against video data if the teddy bear did not appear. Our other solution was to train a model with an additional null class of objects. This solution was suggested because it did not require additional annotation and we thought it might increase accuracy among frames that did not have any teddy bear. However, in addition to lower accuracy as seen in figure 8 in Appendix, we have not solved the issue of ignoring frames with no object. One potential reason for the low accuracy in the image detection was the existence of the null class because the network detect never detects the null class.

[2]Our code can be found at https://github.com/ridhaalkhabaz/YTVidObjectDectector

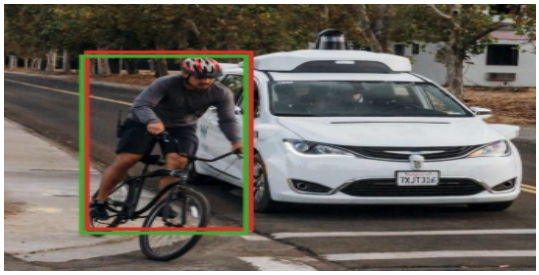

Fig. 3. Green is the ground truth and red is the prediction, from [8]

Fig. 4. One prediction from our YOLO network.

Fig. 5. Another prediction from our YOLO network.

VI. Conclusions

This paper was concerned on building a framework that could detect an object of interest in YouTube videos. Our experiments were done on a teddy bear that appears in KSI's YouTube videos. We could detect its appearance using a YOLOv4 network that was tuned specifically for this object. Although we achieved a high accuracy of 93.75% in image detection, we get an undesireable result in the video files. We propose a number of ways to solve these issues in the following paragraph. We would like to mention a couple of really interesting things we learned throughout the project. We learned how to apply transfer learning techniques by using trained network weights and training the network on a specific class. We also learned how to use Roboflow to apply common data preprocessing and data augmentation techniques, and the tedious nature of image labeling manually. We also learned how to tackle bugs and update our approach to the project.

In regards to future work, our model framework can be trained to detect other common objects in videos and generalize our object detector. However, to solve the issue of detecting null objects, we may need to increase the size of the training data set. Moreover, we might use already trained models to create multiple classes from targeted videos. Using these results, the annotation process decreases in workload because annotators will only need to assign labels to self-supervised created masks.

Contributions

We collaborated using a driver-Navigator programming style. Here, Ridha Alkhabaz wrote most of the code and preliminary technical details and results while German Bautista annotated the data set and contributed to the brainstorming of different methods and where to apply them. Then, we swapped roles and reviewed each others' work. Ridha and German both gave their `import numpy as np  np.Inf` percent effort to make this project awesome.

References

[1] K. Cooke, "Facebook developing artificial intelligence to flag offensive live videos," Dec 2016. [Online]. Available: https://www.reuters.com/article/us-facebook-ai-video/facebook-developing-artificial-intelligence-to-flag-offensive\-live-videos-idUSKBN13Q52M

[2] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," 2016.

[3] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," 2020.

[4] W. Han, P. Khorrami, T. L. Paine, P. Ramachandran, M. Babaeizadeh, H. Shi, J. Li, S. Yan, and T. S. Huang, "Seq-nms for video object detection," 2016.

[5] M. Monfort, K. Ramakrishnan, D. Gutfreund, and A. Oliva, "A large scale multi-label action dataset for video understanding," *2018 Conference on Cognitive Computational Neuroscience*, 2018.

[6] roboflow inc., "Give your software the power to see objects in images and video." [Online]. Available: https://roboflow.com/

[7] K. Kang, W. Ouyang, H. Li, and X. Wang, "Object detection from video tubelets with convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[8] S. Yohanandan, "Map (mean average precision) might confuse you!" Jun 2020.

[Online]. Available: https://towardsdatascience.com/map-mean-average-precision-might-confuse-you-5956f1bfa9e2

APPENDIX

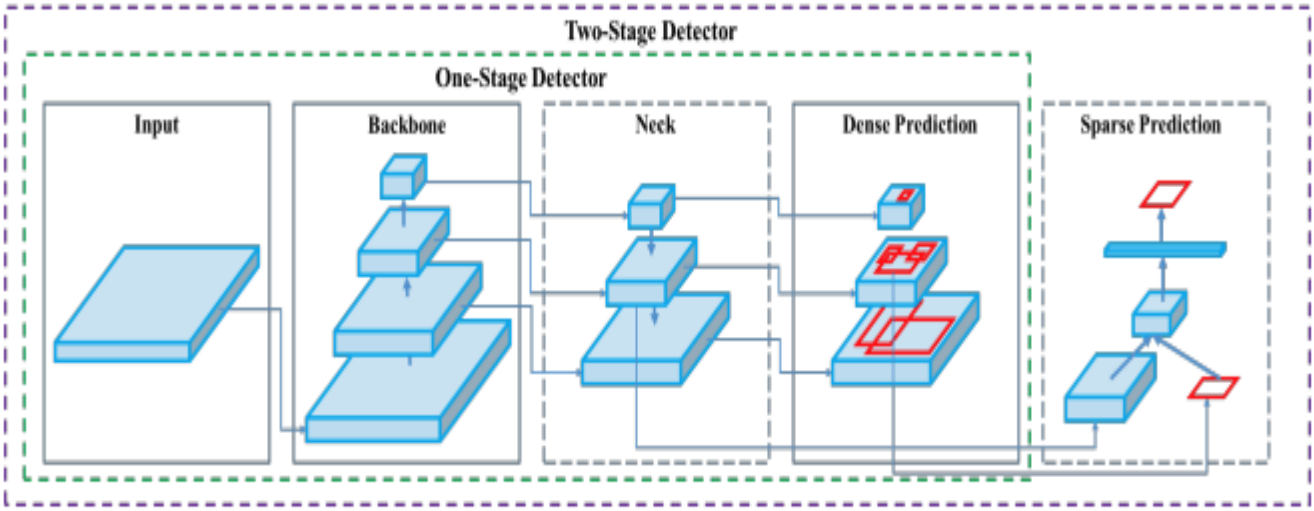

Fig. 6. Object detector overview from YOLOv4 paper.

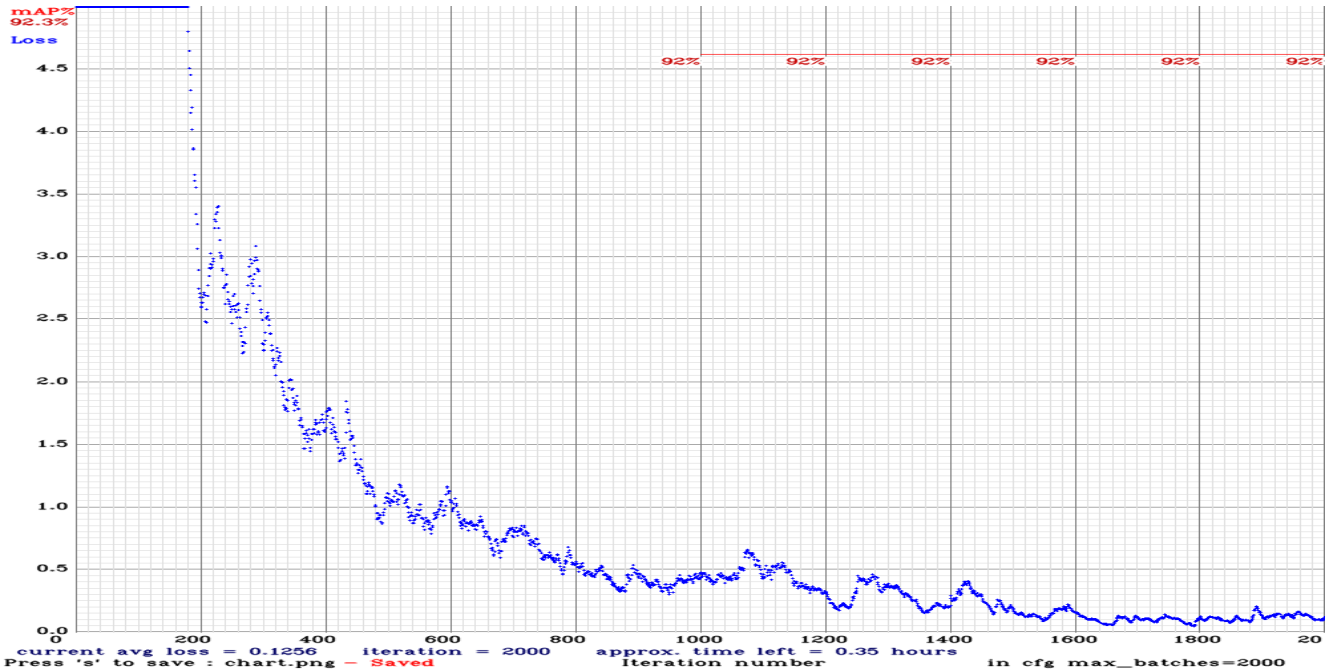

Fig. 7. Our loss graph for training the Binary model.

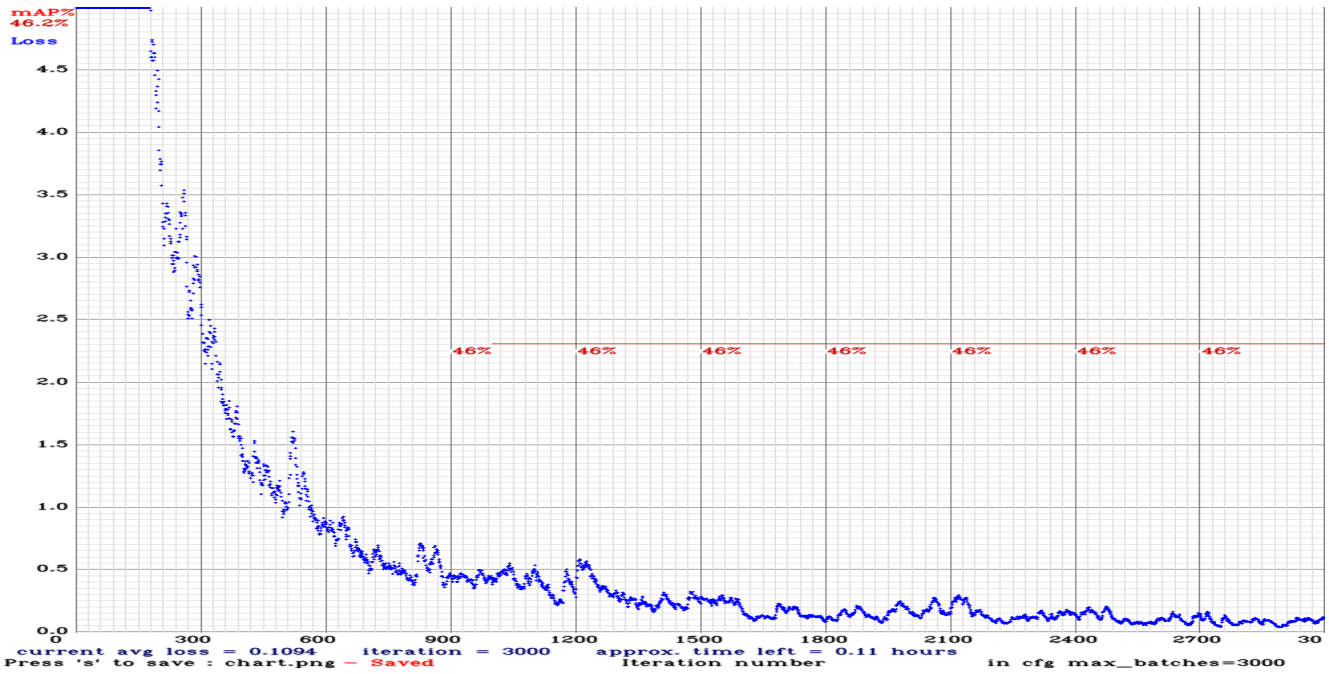

Fig. 8. Our loss graph for training the Multi-model.