Our Big, Fun, Hand-Driven Driving Game

Hamza Mir, Jack Schumann, Stephen Shamaiengar
[hm6ex, js9pa, sas7dd]

Abstract

Object detection is a well studied problem in the field of computer vision. In this paper, we attempt to apply existing techniques to the EgoHands dataset to create a hand tracking model for video game driving controls. This specific domain of hand tracking has been popular in recent years due to the rising interest in controller-less virtual reality systems. In this paper, we show that even with a limited dataset and limited compute power, surprisingly accurate results can be achieved.

1. Introduction

In order to use hand tracking data for a driving game, the hand tracking model should be able to reliably identify the approximate centers of two hands. With this data, the game can approximate the angle of the hands relative to each other and use the result as input for steering the car. As a result, the problem can be boiled down to identifying bounding boxes for multiple hands in an image. By achieving this, the centers of the bounding boxes can be used as approximate centers for the hands. An important note is that a small amount of error or imprecision is acceptable as mildly unwieldy controls may actually make gameplay more fun.

Historically, this kind of object detection has been accomplished with SIFT feature detectors [4] or deep learning techniques. We opt for a deep-learning approach as SIFT is patented and computationally heavy.

Our motivation for solving this problem is related to the rise of virtual and augmented reality in areas such as industry and gaming. The ability to track hands using a simple computer camera could, in addition to making gaming more fun and interactive, could be used in navigating, manipulating, and interacting with a computer, as well as with objects in computer programs.

2. Related Work

The past several decades have spawned various techniques for achieving object detection on images. One such technique, called the scale-invariant feature transform (SIFT), is an algorithm published in 1999 to detect features of objects within images. One of the hallmarks of this method is that it is invariant to changes in size and orientation. This means that SIFT is able to extract the same keypoints in different images, even if one is scaled, rotated, or translated relative to another [4]. The Viola-Jones object detection framework is yet another object detection technique proposed in 2001, whose primary focus is face detection. As part of its implementation, Viola-Jones extracts Haar-like features (i.e. rectangular patterns of groups of pixels) [5].

Deep learning methods also exist for this task. Among the current state-of-the-art deep learning methods to perform object detection is Faster R-CNN. Faster R-CNN generates several region proposals from input images and passes them through the Convolutional Neural Network (CNN) involving convolutional and max-pooling layers. At the end of the network are several linear layers that pipe into a softmax classifier. Faster R-CNN offers performance improvements over prior deep learning models (such as R-CNN and Fast R-CNN) by computing region proposals using a CNN and merging it with the rest of the Faster-RCNN architecture [2]. This model is capable of performing real-time object detection with a reasonable amount of computing resources.

With regards to hand-detection specifically, we found an attempt to perform this task using the Single Shot Detector technique, authored by Victor Dibia. His attempt successfully detected hands in images as well as real time (21 frames per second) in videos, providing feedback by drawing bounding boxes around detected hands in frames passed into the model. Dibia used mean average precision (mAP) as a performance metric, and was able to achieve an mAP of 0.96@0.50 IoU out of 1 [7].

3. Methods

To reduce the complexity of this problem we do not focus on image segmentation, and therefore opt for a region-based network without a mask output. This leaves us with a Faster R-CNN architecture. Due to our limited computational power and relatively small 4,800 image dataset, we use a pretrained ResNet50 backbone (trained on COCO)

from PyTorch's torchvision [1]. Unfortunately, the EgoHands dataset was created for gesture recognition and segmentation [3], so we must do some work to transform the dataset's annotations into a usable format.

Fortunately, Victor Dibia produced a script to transform the EgoHands MATLAB annotations into a CSV file that can be parsed by Google's Tensorflow [7]. From this intermediate step, we write our own script to convert the CSV into another CSV that contains one annotation per line, making it easy for the `pandas` framework to serve to PyTorch.

Once the modified dataset is set up, we finetune the model by training with a stochastic gradient descent (SGD) optimizer with momentum (0.9) and weight decay (0.0005) and a scheduled learning rate of 0.005 decreasing by a factor of 10 every three epochs. During training, we calculate the average intersection over union (IoU) on the validation set as an accuracy measure after each epoch. In order to calculate this, we calculated all pairwise IoU values between the ground truth bounding boxes and the predicted bounding boxes for each validation image. For each true bounding box, we took the greatest IoU value (that of the 'best' corresponding predicted bounding box), and then averaged those across all true bounding boxes to get the mean IoU for a validation image. Then, we averaged the mean IoU values of all validation images to get a mean IoU measurement after each epoch.

After training, the model is used to provide input for a simple open-source driving game built in PyGame [6]. The model returns bounding boxes (and confidence levels) of detected hands, so we take the best two bounding boxes in a frame captured by the computer's camera, compute the angle of the line connecting their centers, and use this result to simulate direction and magnitude of steering input.

4. Results

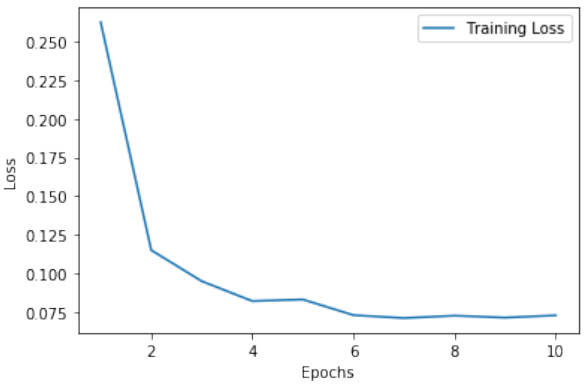

Figure 1. Training loss over 10 epochs on the Faster R-CNN model with ResNet50 backbone, using the EgoHands dataset.

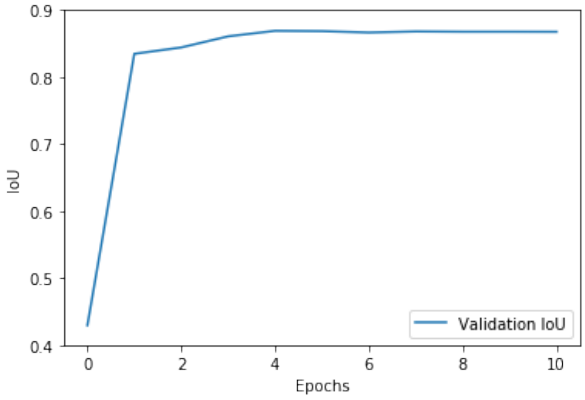

Figure 2. Mean Intersection over Union of bounding boxes across validation set during training on EgoHands dataset.

Figures 1 and 2 show performance of the model over 10 epochs of training on the EgoHands dataset. The loss curve in the former is included to show training progress, but the IoU curve in the latter is more indicative of final performance of the model. After 10 epochs of training, the model reached a mean IoU of 0.867. Roughly, this means that across validation images, our trained model produces bounding boxes whose areas overlap on average 86.7% with the true bounding boxes. We deemed this value sufficient for our use in steering angle calculation, especially since further accuracy on a larger model would have a greater toll on performance of the driving game.

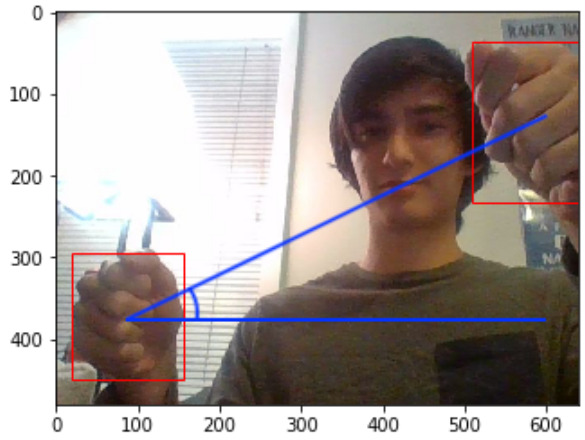

Figure 3. Sample webcam frame demonstrating hand detection and steering angle calculation (which would be reflected across vertical axis to account for camera orientation). Bounding boxes are drawn based on inference of trained model. Angle is drawn for illustrative purposes.

Figure 3 demonstrates the use of hand detection as input to the driving game, as described in Methods. In the simple 2D game that we modified, the magnitude of the angle between the hands on a given frame is used to scale a factor by

which the user's car moves to the left or to the right in the next frame. We tested the game with this modification on a MacBook Pro running only a CPU. Without a GPU, inference of the trained model took up to 10 seconds, but we did observe the car changing directions as expected every 10 seconds. Unfortunately, we did not have access to a physical machine with a CUDA-compatible GPU. Running on Google Colab with such a GPU, we timed the model's inference of a single validation image at 0.08 seconds, so we expect that running the modified game on a GPU-enabled machine would give an acceptable input rate of 10 frames per second.

There are several areas for potential future improvement in our work. Although we believe that our model's inference speed would be sufficient when running on a machine with a GPU, there are several other object detection models that may offer similar bounding box accuracy at greater speed; namely, the Single Shot Detector and the You Only Look Once systems. Training one of these models for hand detection may give the driving game a better input rate. Other areas of improvement concern the dataset we used. At only 4,800 images and 15,053 labeled hands, EgoHands is smaller than other datasets for object detection like PASCAL VOC and CIFAR [3]. The images come from still frames of egocentric (first-person perspective) videos of one or two people doing a few manual activities, such as chess, Jenga, and jigsaw puzzles. The people pictured seem to exclusively have lighter skin and appear in relatively well-lit environments. For these reasons, we expect that our model may perform worse for users with darker skin or who are in poorly lit environments. In a small test of this hypothesis, we verified that for some images containing people with darker skin, bounding boxes were inaccurate or not predicted at all. This shortcoming should be corrected before this model is used in a production environment, by training it on labeled data featuring a greater diversity of people.

5. Conclusion

In this project, we finetuned a pretrained Faster R-CNN model for hand detection. We deployed the model within a 2D driving game to enable steering input based on the position of hands captured by the computer's camera every 10 seconds. We learned that doing this determination at a sufficient speed for real time gameplay would require a GPU, and that a larger quantity and variety of features in training data (i.e. skin tone and lighting of hands) is needed for a model to perform equally well for all potential users.

References

[1] Torchvision object detection finetuning tutorial. `https://pytorch.org/tutorials/intermediate/torchvision_tutorial.html`.

[2] Faster r-cnn: Towards real-time object detection with region proposal networks. 2015.

[3] S. Bambach, S. Lee, D. J. Crandall, and C. Yu. Lending a hand: Detecting hands and recognizing activities in complex egocentric interactions. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[4] L. D.G. Object recognition from local scale-invariant features. 1999.

[5] C. Kaiqi. Study of viola-jones real time face detector. 2016.

[6] J. Kent. pygame-car: A simple vehicle dodging game created with python. *GitHub repository*, 2017.

[7] D. Victor. Handtrack: A library for prototyping real-time hand trackinginterfaces using convolutional neural networks. *GitHub repository*, 2017.