Training YOLO to Detect Doors

Mahrang Saeed
Haoyuan Wang
Xunan Dai
San Jose State University, San Jose, CA 94303

Abstract

An object detector model called YOLO (You Only Look Once), which is used to detect 80 objects in videos/images, is trained to include detection of doors. Main challenges are configuring hardware in order to train model since significant computational power is required to train a neural network, finding/creating a dataset of doors and labeling them to use in training, and training a model. All three of these challenges are time consuming processes. The goal of door detection was achieved on 3 doors in images and one door in video.

1. Introduction

YOLO (You Only Look Once) is a deep-learning object detector model used to detect 80 objects using the COCO dataset. Objects include:

- People
- Bicycles
- Cars and trucks
- Airplanes
- Stop signs and fire hydrants
- Animals, including cats, dogs, birds, horses, cows, and sheep, to name a few
- Kitchen and dining objects, such as wine glasses, cups, forks, knives, spoons, etc.
- ...and more. [1]

YOLO is a single-stage detector, meaning the model's architecture directly predicts the object's location in one stage. [2] This results in a faster prediction of an object's class. [1]

The objective of this paper is to train the YOLO model to detect doors, which is not included in the COCO dataset. This requires a dataset containing a minimum of 300 labeled images of doors in order to train the YOLO model. Training a neural network requires significant computational power. Using a GPU to train a neural network is required. However, doing so requires installation of numerous software packages and installing only the versions that are compatible with one another onto a computer with a powerful GPU. Using a cloud-based GPU, such as that on Google Colab, is possible but training YOLO to include door detection can take over one week.

2. Methodology

A dataset of doors was created and the YOLO model was trained on this additional dataset to include door detection.

2.1. Objectives and Technical Challenges

In order to add doors to the classes of objects detected by YOLO, a door dataset is needed to re-train the YOLO model with the door dataset added to the COCO dataset. The door dataset is downloaded from the MCIndoor 20000 [3]. Training the model requires substantial computational power and is time-consuming. The larger the dataset, the longer it takes for the model to train the neural network.

Ideally, a neural network would be run on a GPU. However, GPU needs to be configured in order to run a neural network algorithm.

In order to train a YOLO model on a GPU, the following is required:

- Windows or Linux
- CMake >= 3.8 for modern CUDA support
- CUDA 10.0
- OpenCV >= 2.4
- cuDNN >= 7.0 for CUDA 10.0
- a GPU with CC >= 3.0
- on Linux GCC or Clang, on Windows MSVC 2015/2017/2019
- Visual Studio version 2017/2019. [4]

The specific versions listed above must be installed. Otherwise, there will be incompatibility issues and errors.

In our team, there are only 2 GPUs. One GPU has been configured for use of TensorFlow, which is only compatible with CUDA 9.0.

2.2. Problem Formulation and Design

In order to detect doors, we must first create/download and label a dataset containing doors. We must then configure a GPU to prepare it for training our YOLO model with the doors dataset. We must convert our dataset into YOLO format, then split it into training and test datasets. We then train the YOLO model with the newly formatted dataset and test the result on images and videos of doors.

3. Implementation

We first attempted to configure on a Windows system to utilize the GPU to train the YOLO model, but the package CUDA 10.0 required in the model does not configure on a Windows system when GPU is utilized.

Then we attempted to configure the second GPU to train YOLO, but ran into several errors trying to do so and were unsuccessful with the configuration. Thus, we resorted to training our door dataset on Google Colab's GPU instead.

We downloaded a dataset containing images of doors that were not labeled. We labeled 60 of the images using BBOx Label Tool. However, the format of the labeled door is not same as the YOLO format. Therefore, we use convert function to transform the data into YOLO format. Then, split the original dataset using process funcion into training dataset and testing dataset. After finish the preparing work, we modify the YOLO configuration file and start training the YOLO model with the addition of this doors dataset on Google Colab's GPU. It took 2 hours to run 1 epoch on the COCO dataset plus the doors dataset. The pre-trained YOLO model only allows for changing the number of epochs to a multiple of 100, so the minimum number of epochs we can run is 100. Thus, it would take over 1 week to train all 60 images on Google Colab.

Due to the length of training time for 60 images of doors, we decided to capture an image of a door with our cell phone and label it and use that single image as our door dataset. We trained the YOLO model using 300 epochs and using only that one image of a door on a Macbook Pro, thus forgoing using a GPU. Training time was 8 hours. The result is shown in Fig. 1 and it was accurate detection of the same door in our dataset.

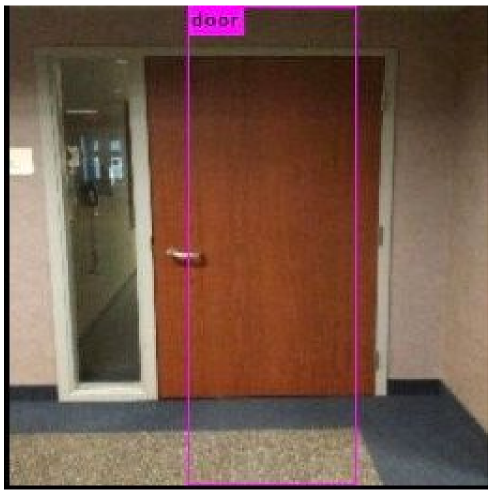

Fig. 1: Model trained with single door. Door detected in image.

Our next attempt at training the YOLO model was to increase the number of images in our door dataset from a single image to 3 images using 100 epochs. Training time was 6 hours on a Macbook Pro.

4. Testing and Verification

After training the YOLO model on our dataset containing 3 images of doors, the result was detection of all 3 doors in images, but with different accuracy. The first door can be detected with the highest accuracy of 42%. The second door can be detected with the highest accuracy of 20%. And the last door can be detected with the highest accuracy of 4%. Since the three doors have different shapes, the training epoch is not enough, and the training dataset is too small. Therefore, the trained model cannot be utilized in other environments.

In the video testing, only detection of one of the doors in a video with 35% accuracy. This is because when the images of the doors were captured, only one of the photo was shot from directly in front of the doors, which is same as our training data. Therefore, only this door can be detected.

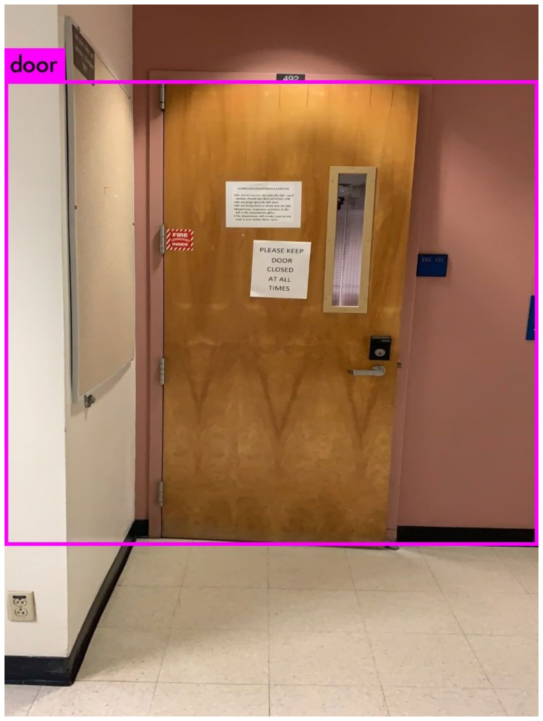

Fig. 2: Model trained with 3 doors. Door #1 detected in image.

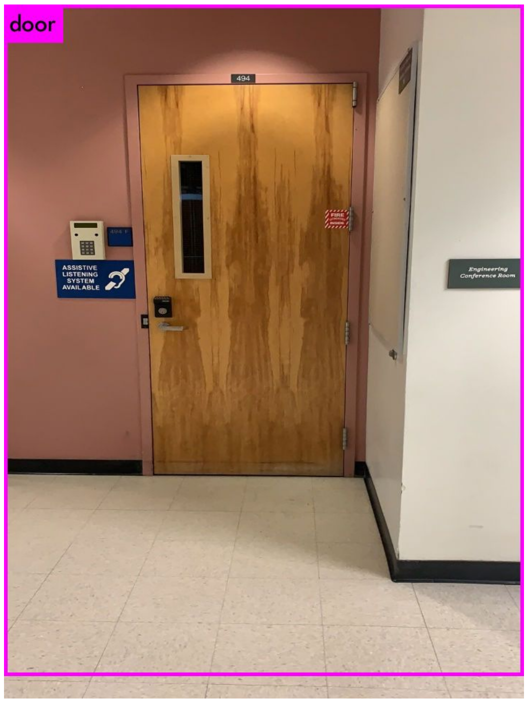
door
494
ASSISTIVE LISTENING SYSTEM AVAILABLE
Engineering Conference Room

Fig. 3: Model trained with 3 doors. Door #2 detected in image.

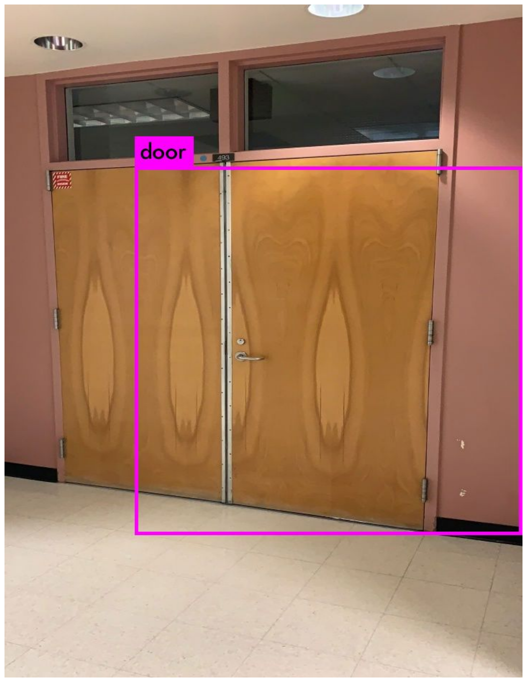

Fig. 4: Model trained with 3 doors. Door #3 detected in image.

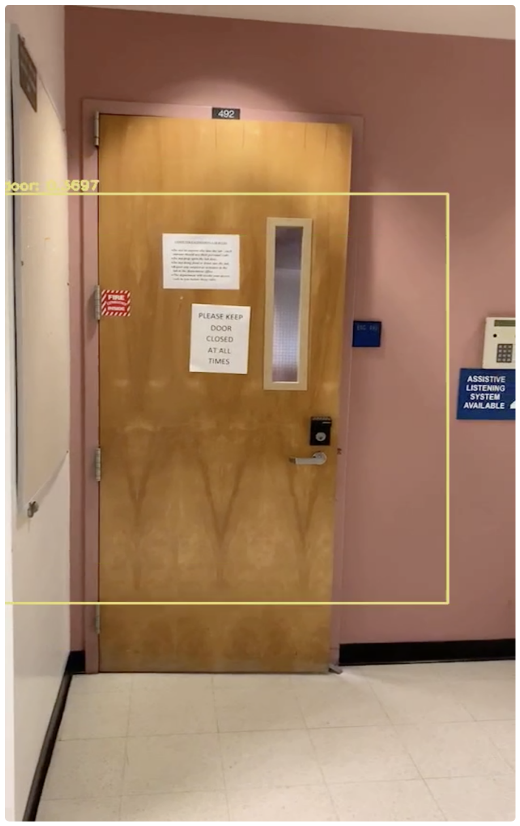
492
PLEASE KEEP DOOR CLOSED AT ALL TIMES
ASSISTIVE LISTENING SYSTEM AVAILABLE

Fig. 5: Model trained with 3 doors. Door #1 detected in video.

However, when the video of the 3 doors was shot, the cameraman was standing directly in front of one of the doors, but shot the other two doors from an angle. The doors shot from an angle were not detected in video.

5. Conclusion

The goal of this project was to detect doors in addition to the 80 objects that YOLO detects. This goal was achieved only for the specific doors used in our small dataset. All doors used in our dataset were detected in images, but only one of those doors was detected in video. A configured GPU is necessary in detecting a wider variety of doors and using a larger dataset. Configuring a GPU and debugging the errors that happen during the configuration process takes 1 week. Labeling a set of images also takes a lot of time.

7. References

[1] A. Rosebrock, “YOLO object detection with OpenCV”, https://www.pyimagesearch.com/2018/11/12/yolo-object-detection-with-opencv/

[2] J. Jordan, “An Overview of Object Detection: One-Stage Methods”, https://www.jeremyjordan.me/object-detection-one-stage/

[3] https://github.com/bircatmcri/MCIndoor20000

[4] “Yolo-v3 and Yolo-v2 for Windows and Linux”, https://github.com/AlexeyAB/darknet