Jayse Hall
CS 5600
12/11/25

Final Project Report - Synthesizing Data for YOLO Training

For the final project in CS5600 I worked on making a simple system to aid in the creation of training data for a YOLO model. All code and this writeup can be accessed on github here: https://github.com/Blu3bear/Targetfinder.

Motivation

The motivation behind the TargetFinder project stems from the difficulty of acquiring and annotating datasets for training a YOLO model. The TargetFinder project aims to ease, in some circumstances, the acquisition of a sufficient data set by synthesizing training data and automatically annotating the synthesized data. The TargetFinder project aims to provide a solution from data synthesis up until model training/tuning, as such it consists of two python scripts: one to handle data synthesis, and one to handle model training/tuning.

The TargetGenerator script allows for many configurations of data synthesis for users to be able to synthesize a data set from either fully synthetic data or by taking actual images of targets and backgrounds to compose into a data set. The YOLOTuner python script aims to ease reconfiguration of model training and tuning by providing a simple interface to run a training iteration or run a hyper parameter tuning cycle, with various configurations.

Technical Detail/Method

The bulk of the work spent on the TargetFinder project went into the creation and testing of TargetGenerator.py. This script needed to be able to allow for many different configurations and options for different training scenarios, the main difference being that data might be generated completely synthetically or there might be images that exist to be used in the creation of the training data. For a proof of concept of the TargetFinder system, the only data which the TargetGenerator script is able to generate completely synthetically is images of ukrainian flags lying on grass or gravel. The choice of target was made as the Ukrainian flag can be described quite simply to be procedurally generated and it is composed of colors which are not found commonly in nature. The grass and gravel backgrounds are procedurally generated by applying a noise map to a base color. The script also can check for configured directories containing target/s and background images. Should such files exist, the script will pull from these images as it creates the data set. In the context of the TargetFinder, data synthesized from procedurally generated images is referred to as fully synthetic data, where data synthesized from captured images is referred to as mixed synthetic data. For reference, figure 1 shows a comparison of the two kinds of images generated by the TargetGenerator, on the left is a mosaic showing a sample of 9 fully synthetic training images, and on the right is a similar mosaic of mixed synthetic images. Both images were used to train a model and the model predictions were compared on the same set of images.

Once TargetGenerator was able to load or create target and background images, it was made to generate a specified number of training images. Each individual training image is created by picking random numbers from various distributions. The first value is pulled from a continuous normal distribution to determine how much to scale the target image. Then an integer is pulled from a discrete uniform distribution to determine rotation angle.

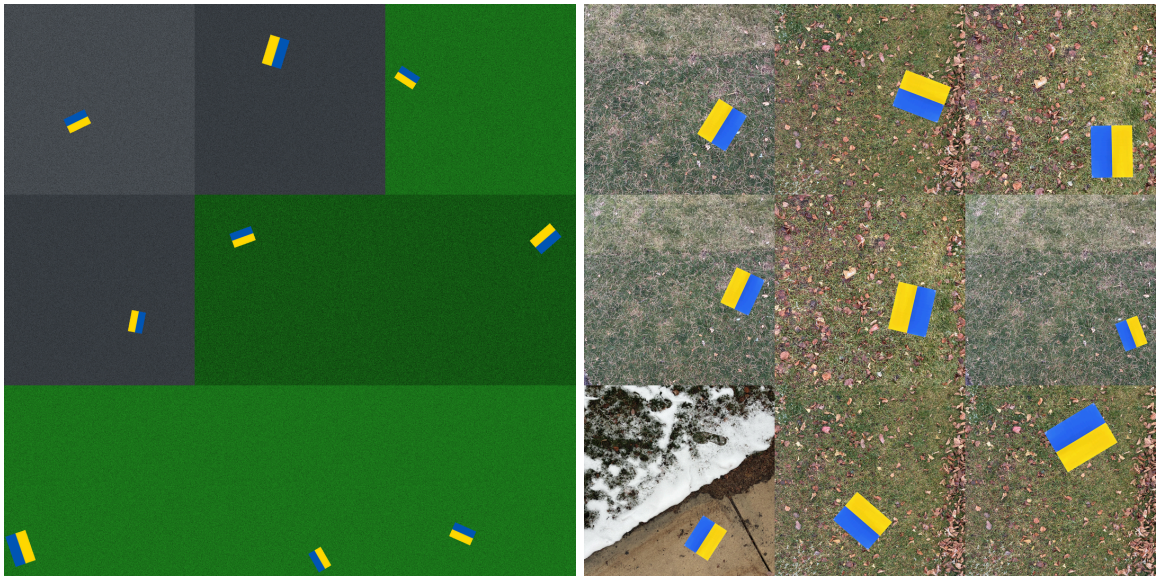

Fig 1: A comparison of fully synthetic data and mixed synthetic data

Finally, a pair of values are pulled from discrete uniform distributions to determine location of the target on the background. Once all of these values are determined, the target can be placed on the background. Originally, TargetGenerator was planned to support data augmentation to further modify the training images to avoid models overfitting on synthetic data, however in the interest of time the augmentation functions were left unfinished.

In order for an image to be a useful part of a training set for YOLO, it must have an associated annotation. Once TargetGenerator composes a training image, the bounding box coordinates must be calculated. Currently only YOLO oriented bounding box(OBB) coordinates are fully supported, however there is a plan to support non oriented bounding box coordinates in the future. The bounding box coordinates are calculated by finding the corner coordinates of the target and then transforming them into the rotated frame based on the rotation that was applied to the target image. Once the coordinates are found then a string can be composed by taking the classification number for the target and appending the coordinates of the bounding box. Finally, the coordinates are normalized to the size of the background image.

Once TargetGenerator has composed and annotated the entire specified set, a YOLO model can be trained or tuned on the data set. To accomplish this, yoloTuner.py was created to simplify the process of training. As the Ultralytics library already provides functions to call for training, tuning, and validation, yoloTuner only needs to set up the model and call the appropriate function. The yoloTunerscript was used to train a yolo11n-obb model on different dataset produced by TargetGenerator.

Results

Due to available time and the cost of training and tuning, only two models were trained. The first, trained on fully synthetic training data, called full-syn.pt can be found in the weights directory in the github repository. The second, trained on a data set of 3000 mixed synthetic training data is called mixed-syn.pt and can also be found in the weights directory in the

repository. Figure 2 shows a capture of resource utilization during training, where images were processed at 640x640 pixels, and batch size was 14.

Fig 2: Computer resource usage during training

During training the models validated on a percentage of the same set as the training data, so models trained and validated on the same synthetic data. However, post training, I had each model make a prediction on each of a set of test images. These test images were acquired by lying a printed Ukrainian flag in the grass and taking overhead photos with an Iphone. Originally the thought was to use a quadcopter to capture an aerial view of the target, but due to timing with the acquisition of the physical target and the weather the quadcopter was not used. Instead photos were taken from atop a tree branch on campus where a tree had blocked snow from covering the grass. Figures 3 and 4 show the result of the different models predictions on one of these test images.

Fig 3: Mixed synthetic model prediction

As there were no annotations for the test images of the physical target, no YOLO validation was run on the test images, but rather a qualitative description of the observed behavior of the models was created. It was interesting to note the difference produced by the two models. The model trained on fully synthetic data was able to identify the target with 0.99 confidence in each training image except in IMG_2640. However, this model often had high confidence false positives. On almost all tree branches present in the test images, and various sections of snow, the model would have greater than 0.50 confidence in the presence of the target. The mixed synthetic model did not have such issues with false positives, and did not show a false negative

in the training images, but in almost all cases it was unable to properly identify the orientation of the target.

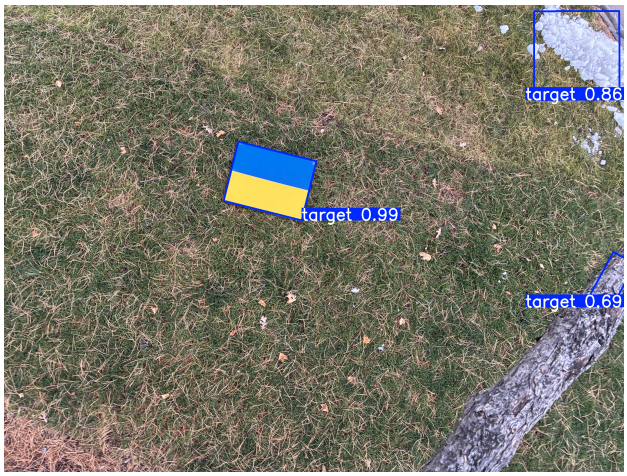

Fig 4: Full synthetic model prediction