Pedestrian Detection Project

Bach Le, Junnan Shimizu, Tsugunobu Miyake

Introduction

Today, the pedestrian detection model has become essential in many domains and industries, one prominent example being autonomous and semi-autonomous vehicles to prevent accidents, increase safety, and make smart choices. Many modern non-autonomous vehicles are even implementing pedestrian detection systems to increase safety, such as automatically activating the brake when a pedestrian is detected [1]. Furthermore, pedestrian detection can have other applications such as in security systems. For example, security camera footage can be analyzed using a pedestrian detection model to detect suspicious people. The model can also be utilized in public spaces, such as train stations and airports, to determine how crowded it is [2] in addition to extra security. With many useful applications, for our final project in our Deep Learning course at AIT-Budapest, we decided to develop this model ourselves.

Previous Solutions

There are several approaches to developing a solution to the pedestrian detection problem. For this project, we decided to analyze high-resolution images. An important note to high-level pedestrian detection implemented in modern vehicles can utilize radar sensors, Lidar systems, or a combination of these with cameras to increase reliability in a larger range of scenarios such as dark environments or strenuous weather conditions [1].

R-CNN (Region-based Convolutional Neural Network), a popular object detection architecture, is one potential approach. R-CNN uses selective search algorithms to detect possible candidates for the objects, then applies a regular convolution neural network (CNN) to those candidates and classifies them. The biggest issue with this approach is that R-CNN is slow to train and slow to make predictions [3]. Other methods such as Fast R-CNN, a model that only applies one CNN for all candidate regions [4], and Faster R-CNN, a model that trains both the algorithm to find object candidates through a region proposal network (RPN) and the CNN to detect objects, were potential solutions [5].

Another popular option to consider for this problem is the YOLO (You Only Look Once) object detection model. The original architecture was produced in 2015 and has been incrementally improved since. The main feature of YOLO, unlike R-CNN which proposes regions and classifies the regions, is that it treats the object detection problem as a regression problem after separating images into multiple sub-sections. This enables end-to-end training in object detection [6]. Unfortunately, each approach has the downside of potentially low detection accuracy in very crowded spaces.

Lastly, some other architectures, such as Localized Semantic Feature Mixers were proposed to address this problem [7].

Dataset

This project uses TJU-DHD datasets to train and evaluate the pedestrian detection model. In particular, we use the ped-traffic dataset of TJU-DHD dataset, a collection of images taken from a driving car. There are 13,858 training images and 2,136 validation images, and each image contains 1 to approximately 20 annotated pedestrians. Furthermore, all the images have a 1624 by 1200 pixel resolution, and only pedestrians were annotated in the dataset. We selected this dataset because the license provided free access and contained sufficient high-resolution images for the training and validation of our model. In addition, the dataset contained a diverse environment which can help with a larger number of cases, making our model more versatile. According to the published paper, the images were taken over more than a year, which allows diverse weather conditions, seasons, and times of the day to be reflected in the image. TJU-DHD datasets also contain a ped-campus dataset which contains images taken by smartphones on university campuses, which can later be used to cross-validate our model [8].

Proposed Method

Initially, we planned to take the regression approach, which uses the VGG16 CNN model to train a dense neural network that directly predicts the number of pedestrians from the picture. We take samples from the original training data to achieve a more uniformly distributed number of pedestrians per picture to train the model, as the dataset contained many images of only one to five pedestrians. We have also further fine-tuned the model and tweaked hyperparameters. While the performance of the model did increase quite significantly after manipulating the data, we decided that the model's overall performance is still not good enough. The mean squared error did not decrease below 5.

Because of this, we decided to use the YOLO (You Only Look Once) model, as described in the Previous Solutions sections. We decided to use YOLO v8, the most recent iteration of YOLO released in 2023 by Ultralytics [9]. Specifically, we decided to use the TensorFlow-Keras implementation of the YOLOv8 because we were more familiar with Keras and TensorFlow, and they can take advantage of Bach's dedicated GPU when training the model. To create our detection model, we followed the official Keras tutorial [10].

We used the pre-trained backbone, "yolo_v8_xs_backbone_coco" to train the model, which was trained with the Microsoft COCO dataset. We also augmented images to generate a diverse dataset. The pre-trained model accepts 640 x 640-pixel images, so the augmenter randomly flipped, scaled, cropped, and resized it into 640 x 640-pixel samples [10]. For the validation and test datasets, the augmenter simply resizes the image to 640 x 640 pixels.

Finally, when we predicted the pedestrian using our developed model, the non-max suppression was used to combine duplicate pedestrian detections.

Evaluation Method

We used CIoU (Complete Intersection over Union) to evaluate the developed YOLO v8 architecture. CIoU is an extended version of IoU (Intersection over Union), which measures how much the predicted bounding box intersects with the actual bounding box. CIoU takes the image proportion and difference between the center coordinates of the bounding boxes into account to evaluate the bounding box prediction more accurately [11]. We monitored validation loss and performed early stopping to avoid overfitting our model.

We also compared the VGG16-based model with the YOLO-based model by comparing the mean squared error and accuracy of predicting the number of pedestrians. For the YOLO-based model, this is done by counting the number of predicted bounding boxes and actual bounding boxes.

Results and Discussion

After the training stage, the YoloV8 XS model achieved an accuracy of 16.47% and a Mean Squared Error (MSE) of 7.09 on the test dataset. Additionally, the box_loss was measured at 2.04. In comparison, the VGG16-based model demonstrated better performance, achieving an accuracy of 25.1% and a lower MSE of 5.37. This is contradictory to what we were expecting when moving from the VGG16 model to the YoloV8 counterpart.

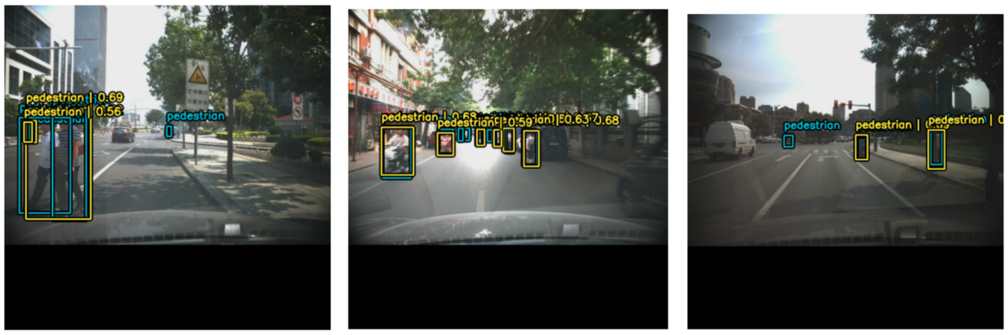

Figure: Sample Predictions by YOLO v8 based model.

It is important to note that all of the bounding boxes predicted by the YOLOv8-based model had a confidence of less than 60%. Due to time and resource constraints, we used the smallest YOLO backbone. However, if we can allocate more computational resources and train the model for a longer time, the YOLO v8 model may predict a much better result. In addition, some annotated bounding boxes in the datasets were extremely small or overlapped with another bounding box. We can further investigate to see if there are better architectures than YOLO v8 to address this issue.

Unfortunately, we could not implement all of our ideas by the deadline due to technical problems. One further research idea can be to cross-validate the model using the ped-campus dataset, provided by TJU-DHD datasets. We can test a model trained with the ped-traffic dataset with the ped-campus dataset, and vice versa to evaluate if our model would work in a new environment.

References

1. Car ADAS. (2021, September 22). *Understanding ADAS: Pedestrian Detection.* https://caradas.com/understanding-adas-pedestrian-detection/
2. Paul, M., Haque, S. M., & Chakraborty, S. (2013). Human detection in surveillance videos and its applications - A Review. *EURASIP Journal on Advances in Signal Processing*, *2013*(1). https://doi.org/10.1186/1687-6180-2013-176
3. Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. *2014 IEEE Conference on Computer Vision and Pattern Recognition*. https://doi.org/10.1109/cvpr.2014.81
4. Girshick, R. (2015). Fast R-CNN. *2015 IEEE International Conference on Computer Vision (ICCV)*. https://doi.org/10.1109/iccv.2015.169
5. Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(6), 1137–1149. https://doi.org/10.1109/tpami.2016.2577031
6. Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. https://doi.org/10.1109/cvpr.2016.91
7. Khan, A. H., Nawaz, M. S., & Dengel, A. (2023). Localized semantic feature mixers for efficient pedestrian detection in autonomous driving. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. https://doi.org/10.1109/cvpr52729.2023.00530
8. Pang, Y., Cao, J., Li, Y., Xie, J., Sun, H., & Gong, J. (2021). Tju-DHD: A diverse high-resolution dataset for Object Detection. *IEEE Transactions on Image Processing*, *30*, 207–219. https://doi.org/10.1109/tip.2020.3034487
9. Ultralytics. (2024, April 17). *Home*. Ultralytics YOLOv8 Docs. https://docs.ultralytics.com/
10. lukewood, Stenbit, I., & Patel, T. (2023, August 10). *Keras Documentation: Object Detection with KerasCV*. Keras. https://keras.io/guides/keras_cv/object_detection_keras_cv/
11. Keras Team. (n.d.). *Keras Documentation: CIoU Loss*. Keras. https://keras.io/api/keras_cv/losses/ciou_loss/