Detecting Pedestrians Using Neural Networks

DS340: Kevin Gold

Aditya Chopra, Dhruv Gandhi

Introduction:

The vehicles we drive every day continue to advance technologically, with efforts to increase road safety and to help make driving easier for everyone. Cars today use cameras, LiDAR, radar, and various sensors for use cases such as forward collision and lane departure warnings. In autonomous vehicles, these sensors are used to make driving decisions to optimize safety and efficiency (Arrow).

A key component of cars, and especially autonomous vehicles, is the accurate detection of pedestrians. In populated areas and around schools, unexpected encounters with pedestrians are very likely, increasing the importance of the ability of the vehicle to detect them. As pedestrian fatalities from crashes have consistently increased since 2010, there is clear room for improvement in vehicles' pedestrian detection. ("Pedestrians"). Dash Cams play a crucial role in recording pedestrian interactions and are one method for detection in both self-driving and regular cars. Therefore, training a machine learning model on dashcam images in order to detect pedestrians is a highly relevant and practical project.

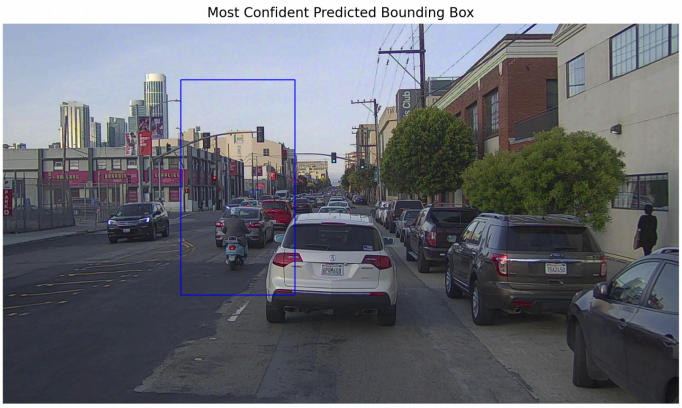
The machine learning model we chose for this task is a neural network. Neural networks are already used by car companies for a variety of purposes. For example, Waymo has fully autonomous vehicles on roads that are powered by neural networks (Hawkins). As we are training on dashcam images, convolutional neural networks (CNNs) are the most applicable to this project as they include a layer that transforms the image into a feature map. For this reason, CNNs generally perform better in tasks dealing with image classification and object detection.

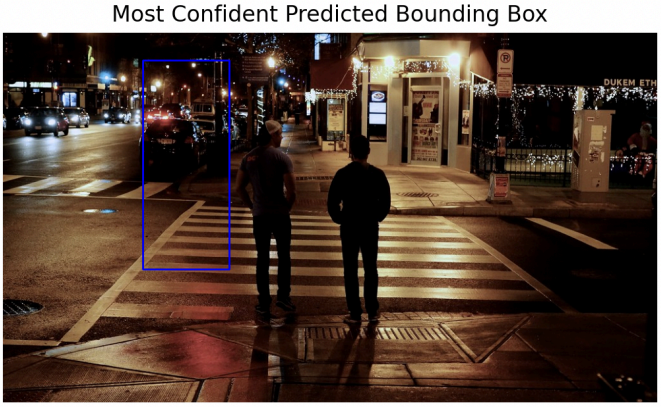
We originally started our project with a focus on child detection in road scenarios to enhance safety in school zones and other populated areas. The goal was to design a neural network capable of distinguishing between children and adults based on such features as height, backpacks, and strollers. However, differentiating children from other pedestrians proved to be highly complex, especially in dashcam images of lower quality and resolution. This led us to broaden our focus toward the overall detection of pedestrians, with the hope that the model would still be able to detect children as pedestrians. This adjustment could allow for a more robust model yet still meet the original intention of improving road safety.

Methodology:

Originally, we planned on training and testing on the PandaSet dataset which was found on Kaggle. This dataset is large, with over 30 GB of images and their annotations. While the images in the dataset contained cuboid and semantic segmentation annotations, we found that the object identification values in the annotations often didn't map with the images. As a result, we then tried to manually annotate the pedestrians in the images using the LabelImg tool. This tool

allowed us to drag bounding boxes to indicate where the pedestrians in the images are. The coordinates of these bounding boxes are saved using the LabelImg tool, where we were then able to save each annotation as an XML file. While this step provided valuable insights into annotation techniques, it presented significant challenges, including difficulty in consistently extracting annotations for training and the time-intensive nature of manual labeling. Therefore, this method was ultimately unsustainable for scaling our dataset. Below are some examples of the incorrect annotations that our original model outputted.





Due to our lack of success with the PandaSet dataset, we had to incorporate a different dataset of images for our project. Luckily, we found the Penn-Fudan Database from Kaggle where the format of the annotations of these images as well as the overall image quality was far better than the PandaSet data. Unfortunately, this dataset was far smaller than the PandaSet data with only 170 images. This many images would not be enough to train a CNN, so more data needed to be added. As a result, we also included the IIT Delhi Campus Pedestrian Dataset (Detection) as well as the CityPersons dataset. After compiling the data from all three sources, we had over 5,000 images we could use to train our model. These additions also introduced more variability in image sizes and quality, which would help our model's ability to generalize across different scenarios.

Before building and implementing the neural network, some preprocessing steps had to be done to the data. To standardize input dimensions, all images were resized to 224 × 224 pixels and normalized to scale pixel values between 0 and 1. Also, because we were pulling data from different sources, we needed to ensure that our data was in the same format for training. Some of our items were in Pascal VOC format, while others were in YOLO with .txt files. We decided that it would be best to convert all our data into YOLO .txt files. This involved extracting the bounding box coordinates from the XML annotations files and then normalizing them relative to

the image size. This ensured that all our data was in a standardized format so that we were not required to create multiple input models that took different formats in order to train.

We also needed to ensure that our images were in a standardized naming convention. This was necessary in order to ensure that our model would be able to understand which file had the annotations for the images. This allowed us to seamlessly integrate all the data into our training pipeline.

Model, Training, and Architecture:

We experimented with many different methods within the CNN in order to increase the accuracy of the box overlap. Firstly, we experimented with the number of convolution layers used, as well as adjusting the kernel size to see what yields better results. We decided to use three convolutional layers after many experimental rounds, where layer 1: captured basic features; layer 2: captured more complex patterns, looking at intermediate features; layer 3: captured objects, looking at more advanced features.

We then looked at kernel sizes including 3 x 3 kernels, as well as 5 x 5 kernels. Through the results of accuracy, we saw that smaller kernels were able to yield a better accuracy, which could be due to the fact that smaller kernels are known to preserve finer spatial details. We also took into account the fact that smaller kernels take less computing power and are able to process quicker.

To try and further refine our model for better results, we also looked at testing different filter counts per layer, starting with 16 to 64. However, we saw that starting from 32 and increasing to 128 slightly increased our accuracy each time.

Additionally, we compared how max pooling and average pooling affected our accuracy and realised that max pooling consistently outperformed average pooling throughout our testing, which is what we decided to integrate within our CNN.

We also made use of dense layers. Testing multiple dense layers lowered our accuracy significantly when tested with different numbers of neurons. However, particularly using one dense layer with 256 neurons slightly increased our accuracy for each test and provided a good representation of the data without causing overfitting.

Lastly, we experimented with many different drop out rates as we realized with some randomly assigned testing data our model was overfitting. This only happened when our model randomly picked the majority images from only one dataset within its testing data. Dropout rates of 0.3, 0.4, and 0.5 were tested. 0.5 yielded the best results especially when the data was overfitting and allowed the model to generalise better on our testing data with the dropout rate added.

We were required to use a loss function as errors varied significantly depending on the size and position of the pedestrian. This was something that we struggled with as the errors varied so much that most loss functions struggled to handle large errors. However, after experimenting with different functions, we used the Smooth L1 Loss function, which allowed us to yield better results in terms of the prediction of the boxes. Aside from testing, the other reason we landed on this function was because it is known for being less sensitive to outliers while still penalizing significant errors. The model helped with the training and improved our predictions (Fengade).

Results:

Overall, the pedestrian detection model performed well in both the training and validation phases, with promising results across the tested metrics. First, the smooth L1 loss on the validation set converged to approximately 0.008. Smooth L1 loss quantifies the accuracy of the model's bounding box coordinates compared to the annotated ones, with smaller values indicating closer alignment to the annotated coordinates. As shown in the figure below, the training loss consistently decreases, while the validation loss initially decreases but then plateaus. This could mean the model is starting to overfit after a certain number of epochs. Overall, minimizing this metric ensures that our model localizes the area of pedestrians well.

Mean Absolute Error (MAE) reflects the average absolute difference between the model's predicted and the actual bounding box coordinates. Our model's MAE was 0.0781, which is quite a low value. This demonstrates that the model's bounding boxes are well-aligned with the

actual ones and reduce the risk of misidentifying objects as pedestrians or missing them altogether.

The metric used to determine whether the model detected a pedestrian at all is called intersection over union (IoU). This metric evaluates the overlap of the actual bounding box and the model's predicted bounding box. This is calculated by dividing the area of overlap by the area of union. For this project, we set the IoU threshold to consider any IoU > 0 as a positive match. This means that any overlap of the predicted bounding box with the actual box was counted as correct. We made this decision as we wanted to prioritize identifying pedestrians even when bounding boxes were not perfectly aligned. Additionally, the low MAE that our model achieved indicates a good alignment of the bounding boxes regardless. Using this rule for IoU, the model achieved an overall accuracy of 84.08% in the validation set, meaning it correctly detected at least one pedestrian in 87.08% of the validation images. This metric is significant because it measures the model's ability to reliably identify pedestrians under diverse conditions, which is essential for real-world applications where false negatives could lead to safety hazards.

The model also demonstrated efficient runtime during evaluation, processing validation images in approximately 27 ms per step. A fast runtime is important for the potential deployment of the model in a real vehicle, where quick decision-making is required.

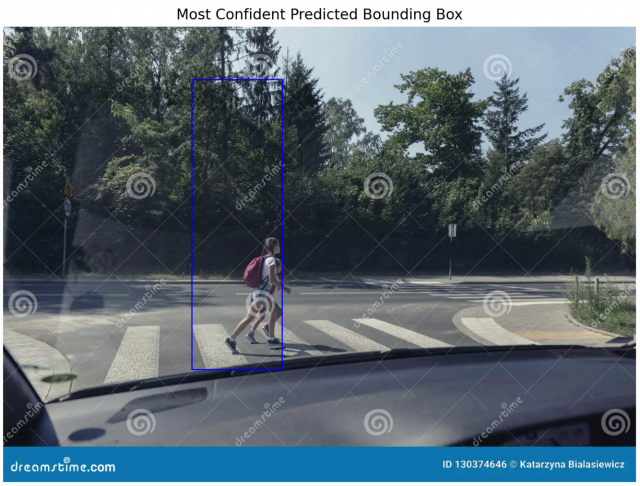
Conclusions:

In this project, we were able to create a robust neural network model that is capable of identifying pedestrians in diverse scenarios and images. The project demonstrated the effectiveness of convolutional neural networks (CNNs) and transfer learning for pedestrian detection, highlighting the importance of preprocessing, high-quality data, and scalable training methods.

One key takeaway from this project is the sensitivity of neural networks to the data they are trained on. As many of the pedestrians in the original PandaSet images were blurry and distorted due to the fishbowl lens effect, our model often just identified the blurriest part of the testing images as pedestrians. This was especially the case as we were only able to manually annotate a small number of images from this dataset, which was not enough data for the model to effectively learn. However, when we added more images of higher quality from different datasets, our model had far more success in identifying pedestrians.

As our original plan for this project was to build a model that could detect children, we were curious to see if there were any images of children in our testing dataset and whether the model was able to detect them successfully. As shown in the image of our model's output below, the child crossing the street was detected and identified as a pedestrian in our model. This confirmed

that our model is generalizable enough to detect people of different sizes. We were pleased with this result as it shows that our project did not completely sacrifice our original goal of optimizing children's safety in populated areas such as school zones.



In terms of real-world applications of this project, it is difficult to say whether our model could be used in a car to actually detect pedestrians. As vehicles use Lidar, radar, and many other motion sensors in conjunction with cameras, a model that only detects pedestrians through images may be antiquated. While incorporating sensor features into our model would be ideal and make for a more interesting project, this was not feasible with our resources. Additionally, while the model was efficient with this data, we are unsure if it would successfully process thousands of images per second in a moving vehicle. However, with relatively good intersection over union and recall scores, the model performed well within the scope of this project.

Works Cited

Arrow Electronics. “Sensor Technologies in the Modern Vehicle.” *Arrow.com*, Arrow.com, 3 Jan. 2023, www.arrow.com/en/research-and-events/articles/sensor-technologies-in-the-modern-vehicle.

Citypersons Object Detection Dataset (v11, 2022-12-10 9:01pm) by Citypersons conversion. “Citypersons Object Detection Dataset (V11, 2022-12-10 9:01pm) by Citypersons Conversion.” *Roboflow*, 2022, universe.roboflow.com/citypersons-conversion/citypersons-woqjq/dataset/11. Accessed 10 Dec. 2024.

Fengade, Somesh. “Understanding L1 and SmoothL1Loss - Som - Medium.” *Medium*, 20 Aug. 2023, someshfengde.medium.com/understanding-l1-and-smoothl1loss-f5af0f801c71. Accessed 10 Dec. 2024.

Garg, Aryan. “IIT Delhi Campus Pedestrian Dataset (Detection).” *Kaggle.com*, 2023, www.kaggle.com/datasets/aryangarg01/iit-delhi-campus-pedestrian-dataset-detection. Accessed 10 Dec. 2024.

Hawkins, Andrew J. “Inside the Lab Where Waymo Is Building the Brains for Its Driverless Cars.” *The Verge*, 9 May 2018, www.theverge.com/2018/5/9/17307156/google-waymo-driverless-cars-deep-learning-neural-net-interview.

“Pedestrians - Data Details.” *Injury Facts*, injuryfacts.nsc.org/motor-vehicle/road-users/pedestrians/data-details/.

psv. “Penn-Fudan Database.” *Kaggle.com*, 2020, www.kaggle.com/datasets/psvishnu/pennfudan-database-for-pedestrian-detection-zip/data , https://doi.org/1033175/1740524/06097e53e866f5d1c258ffde08e80bcd. Accessed 10 Dec. 2024.

Tensor Girl. “PandaSet Dataset.” *Kaggle.com*, 2021, www.kaggle.com/datasets/usharengaraju/pandaset-dataset?resource=download-directory, https://doi.org/1073535/1806921/a1f7d5f2ef9a0291d5ef3d2d2502abcc. Accessed 10 Dec. 2024.