Exploring Multi-Modal Fusion in Perception for Autonomous Vehicles

Albert Xiao
anxiao2@illinois.org

Abstract

Perception in the scope of autonomous vehicles refers to the ability of a vehicle to perceive and interpret its environment without human intervention, through the utilization of sensors, such as cameras, lidar, and sonar. Using algorithms and deep learning techniques to process the sensor data enables automatic tasks such as self-driving, which has a lot of relevance today.

In this project, I explore data from two of these sensors, namely cameras and lidar data, and assess the performance of different types of fusions of the data for the two sensors. I do so in the scope of 2D object detection, where our task is to detect vehicles, pedestrians, and cyclists in an image, and draw bounding boxes around them. Note that unimodal 2D detection is already a well-explored field, which can produce very accurate results. However, the direction of this project explores whether another weakly related modality can improve performance, using an older model like YOLOv1. To do so, I design 3 models for multi-modal fusion, one for each of early, mid, and late stage fusion. Then, I train these models with all factors and hyperparameters held constant. Finally, I evaluate the results to determine the best fusion method, and if the weak modality helped at all.

Code is available at: https://github.com/AlbyYuggle/498FP-exploring-multimodal-perception-detection

Teaser Video: https://drive.google.com/file/d/1fB2SpDeUDixHhnG9cQyAp9zkVun8NJ_t/view?usp=sharing

1. Introduction

Self-driving has been a long-pursued technology, and definitely one of my interests when initially studying computer science. Furthermore, solving problems with data from multiple modalities has been a hot topic in deep learning research. As such, this project seeks to merge these two concepts using the sub-task of object detection. In terms of multi-modal object detection in 2D, not much work has been done. My contribution is to explore different types of plausible multi-modal fusion for lidar and camera data and conclude whether or not a weak modality such as 3D location or depth benefits the accuracy of 2D object detection on a simpler, cheaper model.

1.1. Motivation

3D object detection is a sub-task of self-driving. In an autonomous vehicle, the AI agent needs to understand the location, movement, and size of relevant objects that may change the decisions made by the autonomous agents. First, this includes other vehicles. The agent needs to first guarantee the safety of its passengers by avoiding collisions with other cars. Second, pedestrians are another key object, for which the agent must avoid colliding with for the safety of the pedestrian. Similarly, cyclists need to be avoided as well, but they follow similar traffic rules as vehicles. Finally, signs and lights need to be detected so that the agent can understand the regulations of a vehicle in the signs' scopes.

3D object detection is quite a bit more involved than 2D object detection, so for the scope of this project, I aim to focus on the more manageable 2D detection. With this motivation in mind I seek to improve the accuracy of 2D detection through the addition and fusion of a depth modality derived from the lidar point cloud scan.

1.2. Problem Definition

Now, I define 2D multi-modal object detection, as follows: Given a set of $\{(I_j, L_j)\}$, where I_j is an image and L_j is a lidar point cloud taken at the same time as the image, we want to output, B_j where $B_j = x_1, y_1, x_2, y_2, c, t$, a list of detection 2D bounding boxes, confidence scores for each box, and corresponding classification labels.

1.3. Approach Outline

Now, I set up the experiments with the three models. First, I establish a baseline unimodal 2D detection model as a control variable. This sets the benchmark evaluations so that I can perform ablation on whether or not the added weak modality was beneficial.

Next I need to design 3 new models for each of the different fusion stages. For this, I define the following types of

fusions:

Early stage/Feature-level fusion: In feature-level fusion, the raw data from each sensor is processed independently to extract relevant features, such as object size, shape, and position. The extracted features from each sensor are then combined to form a unified feature representation, which is used as input to a perception model for object detection and classification.

Late Stage/Decision-level fusion: In decision-level fusion, the perception results from each sensor are combined at the decision level, typically using a voting or averag- ing mechanism. For example, if two sensors detect the same object with different probabilities, decision-level fusion can combine their probabilities to obtain a more confident detection result.

Mid-stage/Hybrid/Deep learning-based fusion: In deep learning-based fusion, the data from each sensor is processed by a separate neural network, and the output of each network is then combined using a fusion network. The fusion network can learn to weight the input from each sensor dynamically based on the context, allowing the model to leverage the complementary strengths of each sensor.

Once a model has been designed, I can begin training and experimentation.

2. Background and Related Work

Now that I have defined the problem and outlined the solution, I conduct a literature review on relevant topics, such as unimodal object detection methods, fusion of lidar and camera, and multi-modal detection/segmentation.

2.1. Research Background

I have some background in Computer Vision and Deep Learning. I don't have too much research experience in computer vision, but I have taken CS 543 Computer Vision, in which I completed a team final project as well, and am taking CS 444 Deep Learning for Computer Vision currently. My research experience includes a machine learning research internship at Argonne National Labs, some computer vision-related development work at AbbVie, and Natural Language Processing experience at Forward Data Lab.

2.2. Unimodal Detection

This section deals with models for 2D object detection from an image.

Fast R-CNN [1] is a model for 2D object detection and recognition. It is an improvement over the earlier R-CNN model, which used a two-stage pipeline for object detection that was slow and computationally expensive due to a forward pass for each proposed region.

Fast R-CNN works by first processing the entire input image with a convolutional neural network (CNN) to extract a feature map, on which region proposals/ROI are projected. Then, it uses a region of interest (ROI) pooling layer to extract a fixed-length feature vector from each ROI, regardless of its size or aspect ratio. This feature vector is then fed into a series of fully connected layers that perform classification and bounding box regression.

Faster R-CNN [5] is a model for 2D object detection and recognition and an extension of the earlier R-CNN and Fast R-CNN models, which improved upon the traditional sliding window approach to object detection by using region proposals to localize objects. Faster R-CNN further improved upon these methods by introducing a Region Proposal Network (RPN) that generates region proposals directly from convolutional features, eliminating the need for external region proposal methods. Faster R-CNN consists of two main components: the RPN, which generates region proposals, and a Fast R-CNN detector that classifies and refines the proposals.

The main idea behind YOLO [4] is to perform object detection and classification in a single forward pass of a neural network, which makes it very fast compared to other object detection methods that use multiple stages or region proposals. YOLO divides the input image into a grid of cells and predicts bounding boxes and class probabilities for each cell. It then uses a custom loss function made up of no object loss, classification loss, regression loss, and containing confidence loss. This model is the primary model I will be editing for my project.

DetNet [3] is a novel backbone, which is specifically designed for object detection tasks by maintaining the spatial resolution and enlarging the receptive field. The main contribution of DetNet builds on the fact that modern object-detection networks are often pre-trained on ImageNet, which is weakly similar to object detection. Then, they are fine-tuned on object detection, which may desire higher spatial resolutions on the deeper layers. As a result, it improves on existing models. This work will be helpful in improving my model accuracy, as I will discuss later.

2.3. Multi-Modal Tasks

FuseNet [2] is one of the earlier uses of multi-modal fusion in 2D image tasks. This network uses RGB-D images with a separate CNN encoder for RGB and Depth images, fused at each layer. Then, the fused model is decoded by upsampling CNN layers to get a semantic segmentation.

A multilevel fusion network for 3D object detection [7] proposes a model that uses multi-modal fusion for 3D object detection from an RGB-D image. To achieve its results, this paper introduces a multi-level fusion network using two copies of a CNN, then fused together after ROI pooling, and finally passed through fully connected layers.

Frustum PointNets is another example of using RGB-D image data to predict 3D bounding boxes in the lidar space.

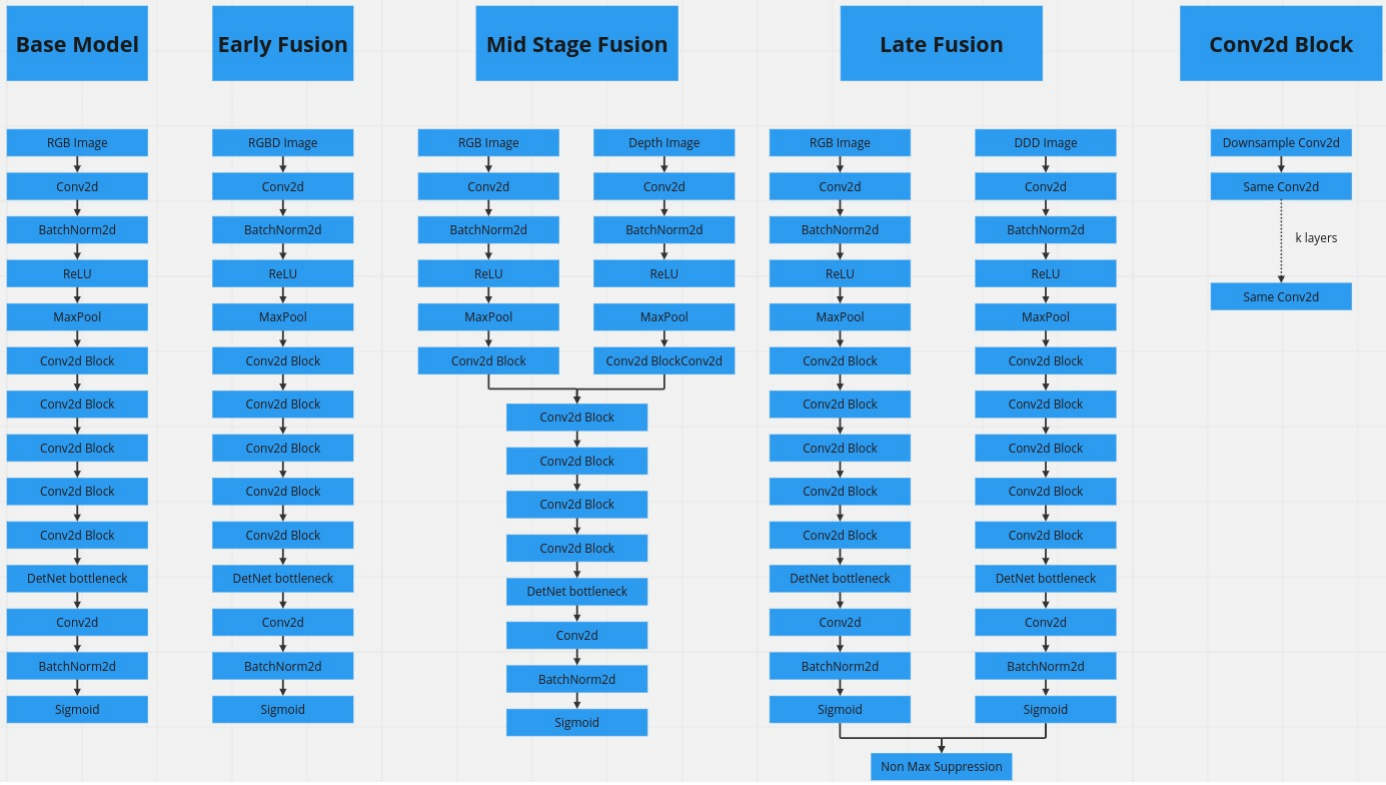

Figure 1. Model Architectures

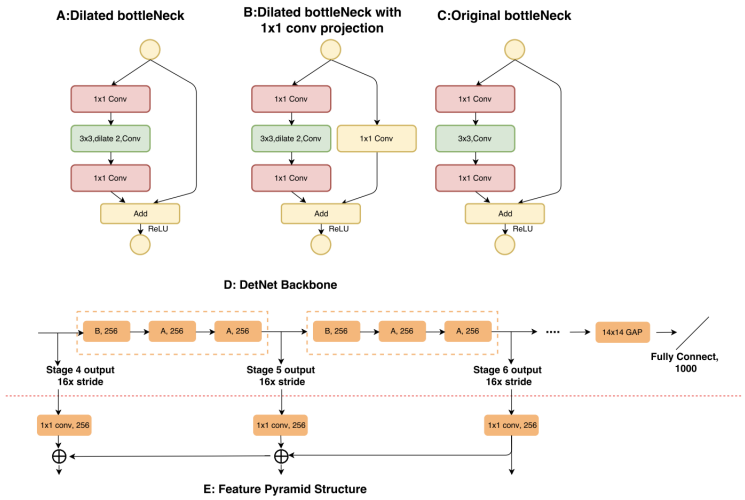

Figure 2. DetNet Block

The main idea is to generate 2D region proposals, then extrude them to a 3D camera frustum, which represents the scale of the region at each depth.

3. Methods

The pipeline for this project can be split into 3 main components: data acquisition, cleaning, and preprocessing; model training; and evaluation.

Figure 3. Example camera data with bounding box labels

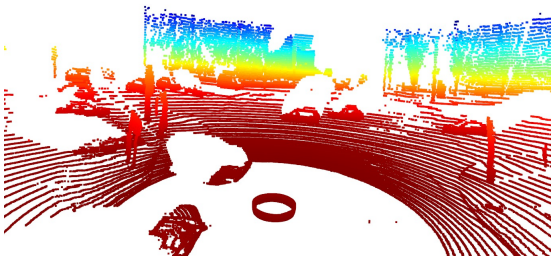

Figure 4. Example lidar point cloud

3.1. Data Description

The dataset used for this project is the Waymo Open Dataset for Perception [6]. This dataset is composed of two main components: camera images from 5 cameras around a car; and lidar point clouds from a short-range and mid-range laser. Each camera and lidar sensor captures a video, which can be broken down into frames. At each frame, we have an image, lidar 3D point cloud 2D bounding boxes for the camera images, and 3D bounding boxes for the lidar point clouds. Each bounding box also has a class label, which is either vehicle, pedestrian, or cyclist. Additionally, the dataset provides lidar to camera projection mappings, which map each 3D point to the two cameras that best capture the point. Finally, calibration parameters, such as intrinsic and extrinsic matrices are provided for both sensor data. Since the data is so large, due to storage restrictions, I use the validation split of the dataset, then split it into my own train/test split as opposed to the entire dataset.

3.2. Preprocessing

Preprocessing the data is done at a per frame per camera level. At each frame, for each camera, I project the lidar data that is marked as captured by the camera onto the camera frame using the extrinsic matrices. To do this, I modified a built-in function that the Waymo Dataset provides to generate an RGB-D image. This gets a sparse depth image, as

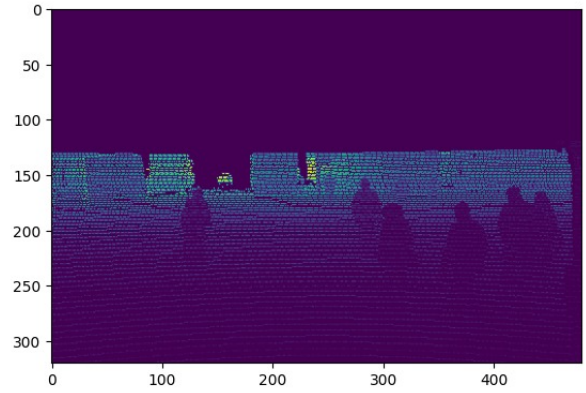

Figure 5. Sparse depth image after lidar projection onto camera frame

lidar cannot fully capture the continuous depth. Then, the RGB-D images are resized to (448, 448), as this is the size that the model I use (will be discussed later) takes in. This generates around 200000 images. However, not all frames have bounding boxes, and these frames are pruned, leaving 160000 data samples (RGB-D image and bounding box labels).

3.3. Baseline Model Design

First, I create a baseline model that takes in RGB image input and predicts 2D object detection bounding boxes. For this, my choices were Faster-RCNN and YOLO. Due to the single-stage nature of YOLO as opposed to the additional region proposal network in Faster-RCNN that could be trickier to the program, I decided to use a YOLO-like. To do this, I reference and modified an off-the-shelf YoloDetNet model.

This model has a few main features that are suitable for my problem. First, it is essentially a fully convolutional network, which allows me to conveniently add a depth channel for early fusion. Furthermore, it allows me to copy the first few layers of the network for the second modality, then fuse them in the middle of the network by stacking the channels. The fact that my modalities are the same input size allows this usage of channels, which makes the model design more sleek.

Another feature of the YOLO-Detnet is the small model size. Due to the lack of computing resources, a model with not too many parameters that can be trained quickly is critical.

Third, the model is initialized with weights from a pretrained resnet50 on ImageNet provided by Pytorch. This is important because it reduces the training time needed due to a weakly related pre-training objective. Furthermore, it helps with the classification task which is a subtask of 2D object detection.

Finally, this model adds an additional improvement with a DetNet in the deep layers of the models. As previously mentioned, the DetNet backbone helps with Object Detection accuracy by preserving spatial resolution in deeper layers, and ensuring we aren't getting false negatives. Furthermore, DetNet helps ensure that tiny objects are not lost from too many downsamples. In our dataset, it's noticeable that many tiny vehicles in the distance are labeled, so DetNet should help in this case.

3.4. Feature Level/Early Stage Fusion

For early stage fusion, because I preprocessed the LiDar data into a depth image, which has the same width and height as the RGB, this allows me to combine the features of both modalities into an RGB-D image. Then, I modify the Yolo-DetNet to take in 4 channels initially and leave the remaining layers unchanged.

3.5. Mid-stage Fusion

For mid-stage fusion, I duplicate the first 5 layers/blocks into two branches, one which takes an RGB image as input, and the other which takes a DDD (3 depth channels) image. To accommodate the extra channels, I halve the out channels of the first five layers to preserve the later layers. Then, I stack the outputs of the two branches of the first five layers to fuse them.

3.6. Late-stage/Decision Level Fusion

For decision-level fusion, I again leverage the fact that the depth image is the same input size as the RGB image. To do this, I train two baseline models, one on RGB data, which is the same as the control, and one on Depth data. The motivation is to generate two sets of bounding boxes, one from each modality, and merge them at the last step. To do this, I merge together the bounding boxes from each modality and perform non-max suppression on the merged bounding boxes. This allows both modalities to contribute to the output boxes, with the NMS just reducing any redundant predictions.

3.7. Loss Function

For the loss function for the models, I implemented YOLO Loss as presented in the YOLO paper. This loss function is the only loss function I used for all four models.

4. Experiments

All that's left is to train and evaluate the models on the same data with the same training parameters. This allows us to isolate the fusion type as the primary factor for any differences in evaluations.

4.1. Experimental Setup

The first part of the experiment is to train all four models. Because of the large number of data points and computational limitations, I train each model on one epoch of training data. I start at a learning rate of 0.001 and decay the learning rate by 10 times after every 3000 batches. I use a batch size of 20 and an SGD optimizer. For the loss function, I tuned the hyperparameters for YOLO loss. This is because on initial training, the model seemed to converge to a local minimum in which it predicts no bounding boxes, as this minimizes the no-object loss. My final hyperparameters were $\lambda_{\text{coord}} = 5$ and $\lambda_{\text{noobj}} = 0.3$.

After training the model, I evaluate it on my test split.

4.2. Evaluation Metrics

The main metric I use is mean Average Precision, which takes into account intersection over union, confidence for predictions, and classification predictions. Unfortunately, none of the models performed very well. All four models seemed to not learn to predict anything other than vehicles. As such, the class label portion of the mean average precision became somewhat meaningless. Furthermore, the prediction boxes were very unconfident. Lastly, the predictions did not have high IoU with the ground truth boxes. As such, to get a more understandable metric I modified the ground truth labels to all be vehicles. The reasoning is that previously, I evaluate that both the bounding boxes and class labels are correct, but since it permanently predicts the vehicle class (aside from a few exceptions). Now, I am simply evaluating if the prediction contains any of the three classes. Additionally, I halved the IoU threshold from 0.5 to 0.25. This threshold essentially determines whether a predicted bounding box and ground truth are detecting the same thing. Finally, I increase the confidence score on each prediction, as almost all predictions have very low confidence, meaning pruning these boxes would essentially mean predicting nothing. Since my predictions are not great, these changes help accept some boxes that are close to being correct or boxes that are correct but have low confidence, so that the mAP is not 0.0 for all models, and can be differentiated (mAP of 0.0 for all models occurs when using unadjusted mAP).

4.3. Comparison

Now, using the defined evaluation metric from above, we can test the results of each of our models. The following table displays the mAP scores for each model.

Here, we notice that mid-stage has the best score, followed by late stage, then early stage, and lastly unimodal. From this, we can conclude that adding a weak modality does in fact improve the evaluation score by quite a large margin. We can also conclude that mid-stage fusion should

Model	mAP (adjusted)
Unimodal	0.01
Early stage	0.09
Mid stage	**0.34**
Late stage	0.18

Table 1. Evaluation Scores

be the preferred fusion method. So we are done, right? Unfortunately not. Just looking at a few examples of output predictions immediately shows that the predictions are terrible, often full of false positives and false negatives.

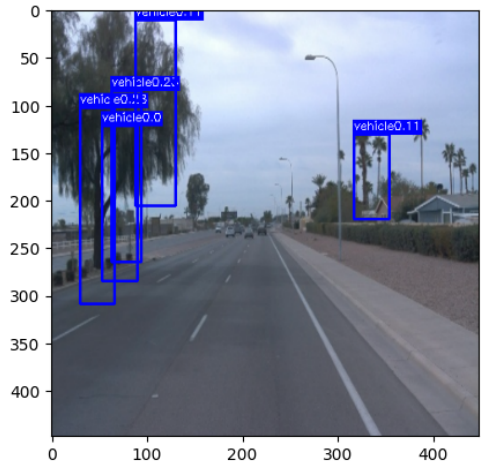

Figure 6. Typical example of poor performance

4.4. Human Evaluation

Figure 6 and 7 help visualize some of the poor predictions resulting from the four different models. In figure 6, it's easy to see a large number of false positives from predicting trees while missing all of the cars in the background. Figure 6 is just one typical example of the many poor predictions. In figure 7, we see slightly more reasonable results, with 2 out of the models having bounding boxes around the white sedan on the left side of the image. However, it is again dominated by false positives. From the images in other test examples, we can notice that the models tend to either predict no objects or a large number of boxes in a small region. Furthermore, the models tend to confuse other objects for vehicles, such as bushes. Finally, we notice that many of the bounding boxes tend to be a similar aspect ratio that is tall and thin. This could be due to the fact that pedestrians and cyclists have tall and thin bounding boxes, and using a tall thin bounding box could become a local minimum, especially since it still has reasonable IoU with a vehicle-shaped box. This is particularly true because in order to fit the image dimensions to the model, they had to be stretched vertically.

4.5. Analysis

There are a few things that could help explain the observations. First of all, obviously, training all of the models on multiple epochs and tuning the loss function, such as adding additional weightings of the YOLO loss components could improve the model. However, due to computational constraints, it was difficult to train the models for too long. Furthermore, I notice that there is some distribution imbalance, as the dataset contains mostly bounding boxes of vehicles, and very rarely bounding boxes of cyclists. One way to improve this is to add weights to our classification loss and give more priority to correctly classifying pedestrians and cyclists. This would likely help circumvent the model collapsing to only predicting one label.

The misclassification of some objects as vehicles is another thing to analyze. This could possibly be a result of the pre-trained weights, as training on ImageNet results in many different categories, whereas our dataset only predicts three. Due to the lack of training time, it could be that either the models did not fully learn the features of the desired classes or that weight remnants from the pre-training objective have not been fully fine-tuned.

Lastly, we notice that our evaluation metrics suggest that mid-stage fusion provides the best results. Interestingly, I originally predicted the mid-stage fusion would have the best results, due to the model having the opportunity to learn from both modalities before combining. Furthermore, other research seems to favor mid-stage fusion. However, this is difficult to interpret. First of all, my mid-stage fusion implementation has the most deviation from resnet-50, meaning it has the least amount of pre-trained weights being loaded into its layers. Furthermore, the results are so poor on all four models that it might just be luck or randomness. Finally, as previously mentioned in the human evaluation, the predicts are not as good as the score entails. As such, although the results claim that mid-stage fusion is the best, it's difficult to trust this conclusion.

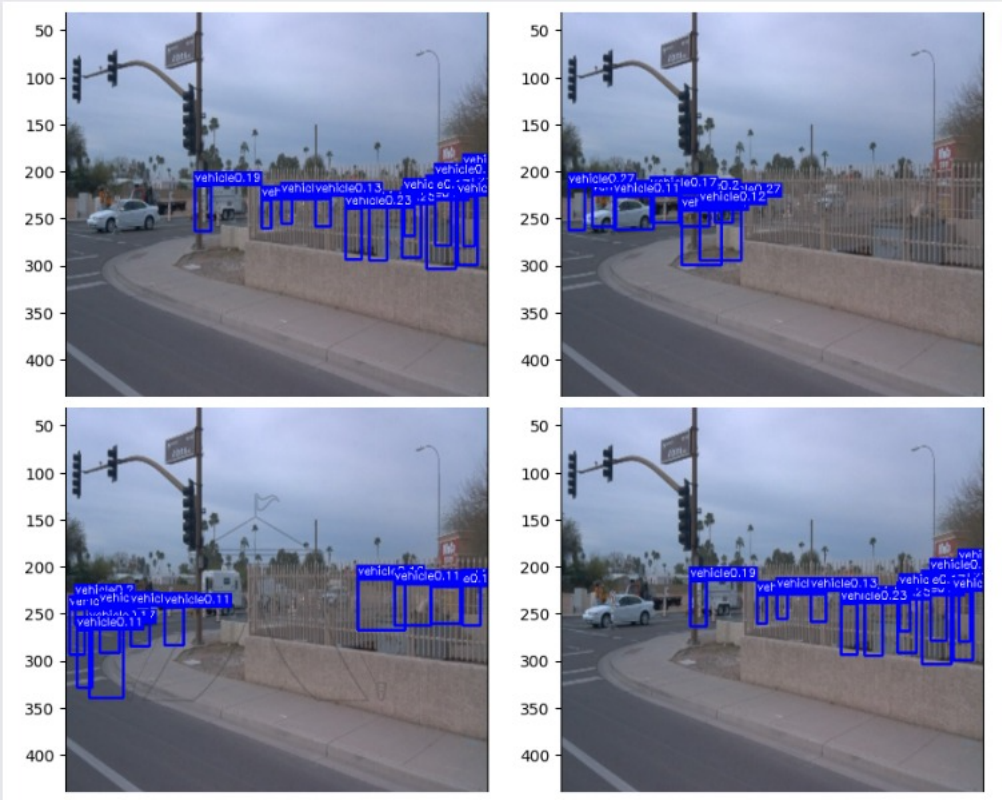

Figure 7. Top left: baseline unimodal, top right: early stage fusion, bottom left: mid stage fusion, bottom right: late stage fusion

5. Conclusion

To wrap things up, I'd like to discuss some overall concluding remarks about the project.

5.1. Discussion

In this project, I explore multi-modal data in the form of lidar and camera data. I also explore different types of multi-modal fusion: early, mid, and late-stage fusion. Although the results were not as good as expected, creating the end-to-end pipeline was a difficult task, and it was nice to see that the results had some resemblance of accuracy. The biggest challenge in my opinion was working with the multi-modal data. First, the data preprocessing was a very involved task, especially as the data was split into different parquet files, and I needed to understand the data parsing and processing tools provided by Waymo with little documentation. Second, I needed to come up with a clever way to preprocess the data such that all three types of fusion would be convenient and at the same time work with a convenient detection model. Despite these challenges, I think this final project was a worthwhile experience. Over the span of the project, I've sharpened my research intuition and process. I've also gained experience in the actual implementation of research ideas, as well as comprehension of Pytorch pipelines.

5.2. Future Work

I think that this area is definitely a very interesting one, and one that is currently fairly popular. Of course, one of the best ways to further experiment is to train the data for many epochs and add some of the hyperparameter/loss tuning previously mentioned. Doing so will help create a more convincing ablation study/conclusion. Some other ideas that could work in the future would be adding some attention mechanisms. Currently, the association of the data is done by creating a depth map from the lidar data. However, I can foresee the possibility of a learned association through attention between the lidar and camera data. Another direction could be trying multi-modal fusion in vision transformers. Finally, extending the scope of the detection from 2D to 3D would be more useful in practice, due to the 3D nature of self-driving.

References

[1] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015. 2

[2] Caner Hazirbas, Lingni Ma, Csaba Domokos, and Daniel Cremers. Fusenet: Incorporating depth into semantic segmentation via fusion-based cnn architecture. In *Computer Vision–ACCV 2016: 13th Asian Conference on Computer Vision, Taipei, Taiwan, November 20-24, 2016, Revised Selected Papers, Part I 13*, pages 213–228. Springer, 2017. 2

[3] Zeming Li, Chao Peng, Gang Yu, Xiangyu Zhang, Yangdong Deng, and Jian Sun. Detnet: A backbone network for object detection. *arXiv preprint arXiv:1804.06215*, 2018. 2

[4] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016. 2

[5] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015. 2

[6] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 4

[7] Chunlong Xia, Ping Wei, Wenwen Wei, and Nanning Zheng. A multilevel fusion network for 3d object detection. *Neurocomputing*, 437:107–117, 2021. 2