CSC 490 Module 1 Part B

Juan Deng
Yu Hau Chen

Due: March 5, 2022

Introduction

In part B, after doing some research on all three suggested directions, we decided to mainly focus on the last direction which is to make the loss function to be more sophisticated. The final goal of our research is how to improve the performance of our model? Then starting from our final goal, there are three main research questions throughout our whole exploring process: How do we represent our data? How do we generate our class detector? How do we train our models and visualize the result? The general research results are as following:

Direction A Research: How do we represent our data?
In our part A, we used a bird's eye view ("BEV") voxel representation which yields a 3-dimensional tensor $O \in \{0,1\}^{D\times H\times W}$. In part B, we did research about the hybrid representation of Point Pillar[1] and the keypoint representation of CenterNet[2]. Point Pillar is used for 3D object detection, but its main point is to use a 2D convolutional neural network layer to make predictions and then decode the predictions to generate the 3D bounding boxes. CenterNet can make the inference process faster since in our models there were many useless detections where CenterNet will generate anchorless predictions based on the centers of the objects.

Direction B Research: How do we generate our class detector?
In our part A, we only generated the detector class for vehicles. However, in real world, the situations in road will be more complex. We are all kind of pedestrians, the moving objects on the roads: motorists, cyclists, or walkers. To create a class for walking pedestrians, a moving model using centroid and bounding box combining the static pose recognition using the height and the width will be helpful[3].

Direction C Research: How do we train our models and visualize the result?
In part A, we used a simple weighted MSE as our loss function and isotropic Gaussian kernel as our create_heatmap function. Since there are too many available loss functions, we started our search from the suggestions on the handout. After doing some researches, we found that focal loss is helpful to reduce the problem of imbalance class and MSE with negative hard mining to decrease false positive numbers. On the other hand, for heatmap related functions, we started our search from the the suggested elliptical heatmaps.

After our research, we think we are more interested in the last directions.

Trivial Direction C: more sophisticated loss functions

Part 1: Focal Loss

Motivation [5 points]

First, we notice that we have implemented a LiDAR-based detector and we have detection class and label class. We want our predicted results to be much closer to real data, thus we used loss function to compute the distance between the algorithm result and the expected result. But our current loss function cannot distinguish the difference between different classes. Focal loss is a pretty useful tool to reduce the class imbalance.

In addition, while doing detections, although TP detections are the targets we want, almost large part of detections are negative and useless. MSE with negative hard mining can help us make our detecting process faster.

After comparing these two techniques, since our model is a one-stage object detector and focal loss has two different equations for one-stage object detector and two-stage object detector, then we decided to implement focal loss.

Techniques [5 points]

First, our two classes are:

Labels		
Detections	TP	FP
	FN	TN

First, to make our calculation more convenient, we will make both predictions and targets to be in the range (0, 1), then we: sigmoid(labels) and sigmoid(detections).

Next, we can compute the binary cross entropy (CE) between predictions and targets using torch.nn.functional.binary_cross_entropy_with_logits.

Then, following the equations in the paper, we can compute the p_t:
p_t = p · targets + (1 - p) · targets

Using p_t, now we can implement our focal loss with a tunable focusing parameter $\gamma \in [0, 5]$. Since the writer mentioned $\gamma = 2$ worked better, then we tried 2 at first:
FL(p_t) = (1 - p_t)^2· CE.

Then, we will decide whether we need to balance our focal loss or not. For default, we set a parameter α to tune our results and its default value is 0.25 as suggested in the paper
If $\alpha \geq 0$, then:
Final loss is (alpha * targets + (1 - alpha) * (1 - targets)) * loss

Evaluation [10 points]

We will compare the predicted heatmap and the overfitted loss result using the Focal Loss function.

Figure 1 showcases the predicted heatmap when we overfit the model with focal loss. Figure 2 shows the results of focal loss changes as during the training process. We generated these two figures using the similar methods in part A. However, we could find that the focal loss are negative and it is hard to describe the results of focal loss in the heatmap.

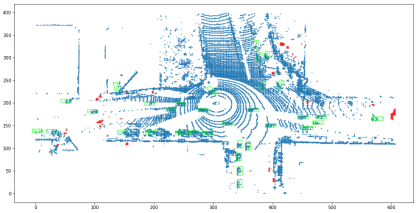
Limitations [5 points]

The main limitations of focal loss is that it does not perfectly support the heatmap presentation for results. Although we actually have some results using focal loss, but it is hard to see these results in heatmap. For several edge cases, focal loss fails to truly reflect whether the model is making better prediction, or indeed predicts the desired ground truth. Zhou [5] has suggested a CenterPoint model combing the focal loss used for the LiDAR-based 3D detection and he suggested the CornerNet model has made an improvement using the Gaissuan heatmap.

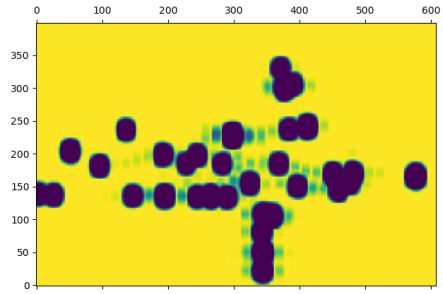
Part 2: Elliptical Heatmap

Motivation [5 points]

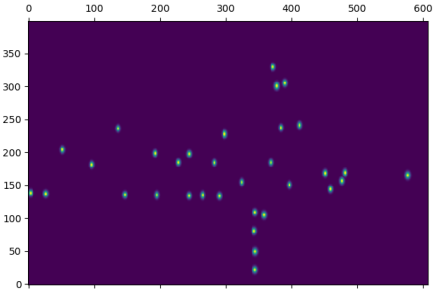
The target detection heatmap $y_{heat}^{(n)} \in \mathcal{R}^{H \times W}$ is used to train the detector to locate the centroid of the vehicles in a frame. Currently, our approach of computing a target detection heatmap is that, given the



(a) Detection



(b) Predicted Heatmap



(c) Target Heatmap

Figure 1: Overfitted Result with Focal Loss

```
100%|          | 500/500 [00:30<00:00, 16.48it/s]
[99/500]: Loss - -16.0181 Heatmap Loss - 0.0499 Offset Loss - -0.4920 Size Loss - -20.2033 Heading Loss - 0.0412
[199/500]: Loss - -21.2089 Heatmap Loss - 0.0279 Offset Loss - -0.5717 Size Loss - -20.3642 Heading Loss - 0.0208
[299/500]: Loss - -23.3626 Heatmap Loss - 0.0185 Offset Loss - -0.6012 Size Loss - -20.4461 Heading Loss - 0.0124
[399/500]: Loss - -24.5291 Heatmap Loss - 0.0127 Offset Loss - -0.6142 Size Loss - -20.5033 Heading Loss - 0.0085
[499/500]: Loss - -25.3726 Heatmap Loss - 0.0079 Offset Loss - -0.6236 Size Loss - -20.5496 Heading Loss - 0.0062
```

(a) Loss Result

Figure 2: Overfitted Result with Focal Loss

centroid of one object $(x_n, y_n) \in \mathcal{R}^2$, $y_{heat}^{(n)}$ is a isotropic Gaussian kernel with center equals (x_n, y_n). The problem of this approach is that, since vehicles have different scales on its length and width, and vehicles are heading in some direction, the isotropic Gaussian Kernel does not utilize this information.

Another approach that can help the detector to utilize information of the vehicles' scale and heading when locating the centroid is to utilize the bounding box of the vehicles and the heading of the vehicle. However, this would introduce additional complexity in the model (we are basically adding additional inputs and connections to the model).

Instead of producing circular heatmaps, we want to produce elliptical heatmaps that provides information of the shape and heading of the vehicles. This way, there is no extra complexity introduced to the model.

Techniques [5 points]

Intuitively, knowing the vehicle's shape helps the model to know how much it should "slide" the bounding boxes to the target along both the x-axis and the y-axis when it is close to the object. Moreover, knowing the heading of the target adjusts the direction that the model should "slide" the bounding boxes.

Our approach is still to construct a BEV heatmap using a Gaussian kernel centered on the centroid $\mu = (x_n, y_n)$, with covariance matrix C and scale σ where

$$y_{heat,i,j}^{(n)} = exp\big(-(X-\mu).T \cdot C^{-1}(X-\mu)/\sigma \big)$$

where $X = (x, y).T$, $C = RT$, $R = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix}$, $T = \begin{bmatrix} x_{size} & 0 \\ 0 & y_{size} \end{bmatrix}$, where θ is the heading of the target, (x_{size}, y_{size}) is the size of the bounding box.

Moreover, heatmap is normalized to have a maximum value of 1, and σ is a constant that we need to tune to obtain a reasonable heatmap.

Evaluation [10 points]

We will compare the predicted heatmap and the overfitted loss result created by three different kernels: isotropic gaussian, anisotropic gaussian, and rotated gaussian to evaluate our methods.

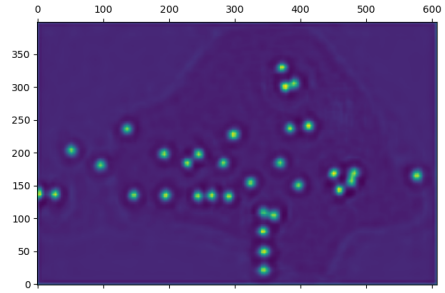
Figure 3 showcases the predicted heatmap when we overfit the model, holding the number of iterations constant (500). We observe that the scaled gaussian provides higher fidelity of the centroids compare to the isotropic gaussian. We want recognized that the rotated gaussian does not perform well, this is due to that we have failed to tune the value of σ correctly. Table 1 showcases the MSE Loss of the overfitted model for every 100 iterations for all different gaussian kernels. We observe that the scaled Gaussian performs better than the Isotropic Gaussian. Since we have failed to tune σ correctly for the rotated gaussian, the loss is significantly large.

Number of Iterations	Isotropic Gaussian	Rotated Gaussian	Scaled Gaussian
100	9.7279	1001.5097	9.1219
200	4.3276	845.7360	3.8150
300	2.9078	726.1313	2.4329
400	2.2069	611.5720	1.7958
500	1.7918	523.5179	1.4158

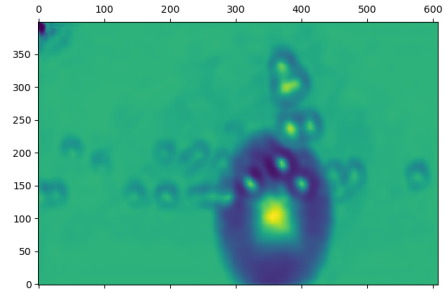
Table 1: MSE Loss by Different Gaussian Kernel

Limitations [5 points]

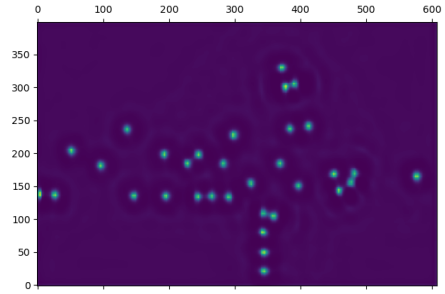
For the approach we choose, we found that it is very difficult to tune σ and obtain a reasonable heatmap. We also want to be careful about the numerical stability of the exponential function. Since we are scaling X with a scaled matrix, it is easy to cause overflow. Huang (2022) [4] states that a joint-optimization loss (JOL) with area normalization and dynamic confidence weighting can improve the performance of the model, this would be a potential path of future work to improve my approach.



(a) Isotropic Gaussian



(b) Rotated Gaussian



(c) Scaled Gaussian

Figure 3: Predicted Heatmap by Different Gaussian Kernel

Reference

1. Lang, Alex H., et al. "Pointpillars: Fast encoders for object detection from point clouds." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019.

2. Duan, Kaiwen, et al. "Centernet: Keypoint triplets for object detection." Proceedings of the IEEE/CVF international conference on computer vision. 2019.

3. Hariyono, Joko, and Kang-Hyun Jo. "Detection of pedestrian crossing road: A study on pedestrian pose recognition." Neurocomputing 234 (2017): 144-153.

4. Huang, Z., Li, W., Xia, X.-G., amp; Tao, R. (2022). A general gaussian heatmap label assignment for arbitrary-oriented object detection. IEEE Transactions on Image Processing, 31, 1895–1910. https://doi.org/10.1109/tip.2022.3148874

5. https://github.com/xingyizhou/CenterNet