Class Prototypical Loss for Enhanced Feature Separation in 3D Object Detection

Marc Perez[1,2,†], Antonio Agudo[1], Gijs Dubbelman[3] and Pavol Jancura[3]

***Abstract*— We present a novel loss to increase the class separation of learned features for 3D object detection from lidar point clouds. To correctly classify objects, learned object-level feature distributions of each class need to be distinct. Therefore, we hypothesize that if we make the feature distributions of the classes more separated, then the overall performance of the object detector will improve. To this end, we calculate class prototypes as the mean and covariance of the feature vectors extracted from the annotated objects of each class. Then, we exploit these prototypes with a novel class prototypical loss, defined as the Mahalanobis distance from the feature vector of annotated objects to the corresponding class prototype. This auxiliary loss is then integrated with other object detection losses to improve the object-level feature separation between classes and the overall performance of the detector. We show results applying this loss to the NuScenes dataset where we get improvements of +3.85% and +1.76% mAP for 1 and 10 frames, respectively, compared to the baseline Centerpoint detector, while keeping the same inference computational cost.**

I. Introduction

In autonomous driving, a vehicle requires a reliable understanding of the environment and the objects in the scene to be able to drive safely. While there has been a lot of advancement in camera-based 3D object detection, RGB cameras do not measure depth directly but instead rely on stereo, depth priors or learning-based models to obtain the 3D coordinates of the objects. In contrast, lidars capture a point cloud of 360-degree 3D measurements, which can be exploited to accurately estimate the object position. However, a major disadvantage of lidars compared to cameras is that they tend to have a lower resolution and cannot capture rich semantic information from colour. Therefore, it is harder to differentiate similar classes (such as bicycles and motorcycles) in point clouds than in images. In this work, we present a method to improve the capability of an object detector for point clouds to discriminate between similar classes. We achieve that by increasing the feature separation per class when training a model as seen in Fig. 1.

Recent 3D object detectors [1]–[5] use a neural network to extract a Birds-Eye-View (BEV) feature map [6] and then apply a detection head on top of that representation to determine the location, orientation, and object class. As a consequence, the performance of the detector relies on the quality of the features contained in the BEV map. To study the features extracted by modern 3D object detectors in terms of class discrimination, we train a Centerpoint [1] baseline on NuScenes [7] and analyze the distribution of feature vectors extracted from the BEV feature map for ground truth objects on the validation set. We show a t-SNE plot where feature distributions are distinct for each class in Fig. 2a.

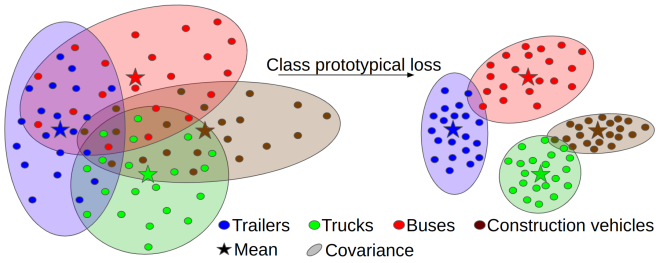


Fig. 1: **Class prototypical loss. Left:** Some feature distributions from an object detector are displayed along with their mean and covariance per class. **Right:** Using a class prototypical loss the covariance is reduced, making classes more separable, and enhancing the performance of the object detector.

To understand how the feature separation originates from class-level supervision, we also train a class-agnostic model. We follow the previous procedure but change the head so it can only detect a single *object* class, and we also modify the annotations of the dataset so that all classes are converted to this general *object* class. Then, we can train a detector capable of detecting all objects without class supervision. Interestingly, the feature vectors for these objects are still separated by class as seen in Fig. 2b, therefore the feature separation is not entirely reliant on class-level supervision and the intrinsic differences in the data generate distinct feature distributions.

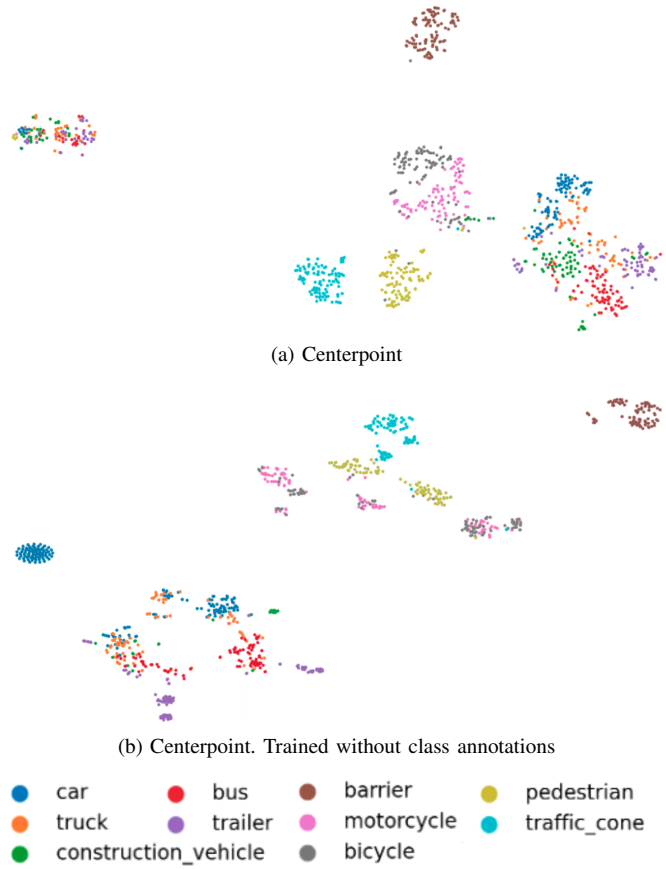
Based on these insights, and motivated by previous works that show that a classifier can improve its performance by increasing feature separation [8]–[11], we formulate the following hypotheses:

Hypothesis 1: In a 3D object detector, an increase in the separation per class of object-level features can lead to improved performance.

Hypothesis 2: Using the mean and covariance of object-level features as class prototypes, we can apply a *class prototypical loss* based on a Mahalanobis distance from annotated objects to the class prototypes, increasing the feature separation per class.

The hypothesized effect of the class prototypical loss is conceptually visualized in Fig. 1, where we see how the

[1]Institut de Robòtica i Informàtica Industrial, CSIC-UPC, Barcelona, Spain.
[2]Applus+ IDIADA, Barcelona, Spain, Email: marc.perez@idiada.com.
[3]Technical University of Eindhoven (TU/e), Eindhoven, Netherlands.
[†] Part of this work was done while M. Perez was at TU/e.



(a) Centerpoint

(b) Centerpoint. Trained without class annotations

Fig. 2: **Analysis of feature vector distributions for objects on the validation set**. t-SNE plots with the color-coded points by class. (a) Corresponds to the Centerpoint [1] baseline, (b) we remove the class from the model and the annotations. Notice how learned features are separated by class, even when the model has no class supervision.

distributions of the object-level features can overlap for similar classes such as trailers, buses, trucks, and construction vehicles. Then, by using the class prototypical loss we propose, the covariance of the resulting distribution can be reduced, making the features for objects of different classes better separated.

As a consequence, the performance of the object detector in terms of mean average precision (mAP) in NuScenes [7], both for the standard case of aggregating ten 360-degree lidar frames (+1.76%) and also using just one frame (+3.85%), is improved. Furthermore, the performance gets especially better for challenging cases where classes are more similar to each other (such as bus, construction vehicle, truck and trailer or the classes motorcycle and bicycle). We also analyze the relative separation between feature distributions of different classes when training with and without the class prototypical loss and see that our proposed loss improves the relative separation between classes and prevents the collapse of feature distributions of similar classes. We observe that increased feature separation correlates with improved mAP for similar classes.

II. Related Work

A. 3D Object Detection

The goal of 3D object detection is to detect a 3D rotated bounding box for each object in the input point cloud. In the case of autonomous driving, the pitch and roll angles can be ignored as the driving direction is determined by a yaw angle or heading. Most recent 3D object detectors voxelize the input point cloud into a 3D [4], [12] or 2D [2], [13]–[16] grid, then apply 3D or 2D convolutions to encode it –usually into a bird's-eye view feature map [1]–[3], [15]–[18]– and finally use an anchor- [2], [3], [17], [18] or center-based [1], [19] head to regress the bounding box. Li *et al.* [13] used a 2D fully convolutional network to detect 3D objects on range images (the point cloud projected to an image where the pixel values are the depth of the points). Vote3Deep [12] used 3D convolutions to encode the point cloud, leveraging a voting scheme on non-zero features to make it more efficient on sparse data, and regularization to encourage sparsity in the intermediate feature maps. 3D sparse convolutions were refined for semantic segmentation [20] and then applied to object detection in Voxelnet [4], where they employed a Pointnet [21] to encode each voxel. Moreover, [3] improved the efficiency of 3D sparse convolutions and proposed a new loss for angle regression and new data augmentation strategies. To avoid 3D convolutions, Pixor [16] maps the point cloud to a 2D bird's-eye-view representation and then applies faster 2D convolutions to get object predictions at each pixel. Voxels can also be represented as pillars [2], compressing the information along the height of the 3D grid to create a 2D BEV feature map. These pillars can be enhanced by adding the features from the perspective [14] or cylindrical [15] views. PV-RCNN [18] and PV-RCNN++ [22] combine point- and voxel-level features. As an alternative to anchor-based heads, Votenet [19] groups and aggregates votes for object centres to detect the centres of the bounding boxes and then regress the other attributes. Another alternative to detect the centres of the objects directly is to predict a 2D heat map of object centres in BEV as in Centerpoint [1], they then regress the other attributes and use additional point information on a second stage to refine the predictions. Voxelnext [23] removes the need for anchor-based or centre-based heads and instead uses a fully 3D sparse convolutional network. FocalFormer3D [5] uses a multi-stage approach and detects the false negatives at each stage to focus on them in the next one. MDRNet [24] improves the 3D-to-2D compression of the BEV map by focusing on the valuable points of the objects. LargeKernel3D [25] proposes to use larger kernels for the 3D sparse convolutions copying parameters in neighbouring areas, and LinK [26] modifies them to adapt the position of the parameters to input data. IA-SSD [27] proposes a downsampling strategy that keeps more foreground points belonging to objects of interest. Following their success in other areas, transformers have started to be applied to the task of 3D object detection [28]–[31]. Finally, some works have adopted a multimodal approach to overcome the limitations of lidars [32]–[34]

B. Feature separation

Recent work [35]–[37] has shown that under some conditions the features on the last layer of a neural network classifier converge to a state of *neural collapse* with the following properties:

- *Intra-class distance*: As training progresses, the within-class variation of the features becomes smaller as they collapse to their class means.
- *Inter-class distance*: After centering by the global mean, the class means converge to having equal length and equal angles between any pair. Therefore, they create a simplex equiangular tight frame.
- *Convergence to self-duality*: The class means are aligned with the last-layer linear classifiers.
- *Simplification to nearest class centre*: The classification decision is equivalent to selecting the class with the closest mean in standard Euclidean distance.

This phenomenon is present for sufficiently large neural networks trained with a cross-entropy loss [35], a mean squared error loss [36], label smoothing or focal loss [37]. Prior to the discovery of this phenomenon, some works [8]–[11] already showed that using a loss that explicitly aims to reduce the intra-class feature distance and increase the inter-class one leads to improved classification performance. In addition to that, this idea has also been explored for open-set recognition [38] and few-shot object detection [39].

Class prototypes can also be exploited to improve the feature separation by setting the prototypes a priori to be maximally separated on the surface of a hypersphere [40]. They can be also estimated from data, usually as the mean of features for each class, and then guided with a loss to be maximally separated [41]. Class prototypes are also used in the context of open-world object detection [42], [43] and domain adaptation [44], [45]. Most of these works focus on image data, but some include promising experiments on point cloud data as well [11], [45].

We propose a novel definition of class prototypes that includes the covariance of the features. We exploit them with novel auxiliary losses, based on the Mahalanobis distance, to increase the feature separation and performance in 3D object detection. Previously the Mahalanobis distance has been used mainly for metric learning [46], [47], anomaly detection [48], [49], face recognition [50], and uncertainty estimation [51].

III. Method

A. Centerpoint

We use a two-stage Centerpoint [1] as our baseline model. First, the model voxelizes the input point cloud and then applies Voxelnet [4] to encode the features using 3D convolutions, obtaining a $\mathbf{M} \in \mathbb{R}^{W \times L \times F}$ BEV feature map, where W, L and F represents width, length and number of channels, respectively. $\mathbf{M}$ is used by the *centre heatmap head* to generate C heatmaps $\mathbf{Y}_c \in \mathbb{R}^{W \times L}$, one for each of the C classes. This head is trained with a focal loss $\mathcal{L}_{hm}$ to produce spikes at the centres of objects.

Additionally, a sub-voxel location refinement $\mathbf{o} \in \mathbb{R}^2$, height-above-ground $h_g \in \mathbb{R}$, 3D size $\mathbf{s} \in \mathbb{R}^3$, yaw rotation angle $(\sin(\alpha), \cos(\alpha)) \in [-1, 1] \times [-1, 1]$, and velocity $\mathbf{v} \in \mathbb{R}^2$ are regressed densely, at the ground truth object's centre location, with individual heads trained with $\mathcal{L}_o$, $\mathcal{L}_{hg}$, $\mathcal{L}_s$, $\mathcal{L}_\alpha$ and $\mathcal{L}_v$ losses, respectively. These losses are computed as the Mean Absolute Error (MAE), and they are aggregated in the regression loss $\mathcal{L}_{reg} = w_o\mathcal{L}_o + w_{hg}\mathcal{L}_{hg} + w_s\mathcal{L}_s + w_\alpha\mathcal{L}_\alpha + w_v\mathcal{L}_v$ with configurable weight coefficients w_o, w_{hg}, w_s, w_α, $w_v \in [0, 1]$. Finally, $\mathcal{L}_{reg}$ is weighted by the coefficient $w_{reg} \in [0, 1]$ and combined with the heatmap loss $\mathcal{L}_{hm}$, obtaining the baseline loss of the first stage $\mathcal{L}_{B1}$ as:

$$\mathcal{L}_{B1} = \mathcal{L}_{hm} + w_{reg}\mathcal{L}_{reg}. \tag{1}$$

After that, the model extracts from the BEV feature map $\mathbf{M}$ one point-feature from the 3D centre of each face of the predicted bounding box. The five point-features are concatenated to create the feature vector $\mathbf{f}_{pred} \in \mathbb{R}^{330}$ for each prediction. These feature vectors are then passed through a multilayer perceptron. In this stage the model predicts a class-agnostic confidence score $\hat{I}$ and box refinement for each detection. The class-agnostic confidence score $\hat{I}$ is trained with a binary cross-entropy loss $\mathcal{L}_{score}$ such that:

$$\mathcal{L}_{score} = -I \log(\hat{I}) - (1 - I) \log(1 - \hat{I}), \tag{2}$$

targeting a score $I = \min(1, \max(0, 2IoU - 0.5))$ guided by the box's 3D Intersection over Union (IoU) with the corresponding ground truth bounding box IoU.

For box refinement, the model is trained with a regression loss $\mathcal{L}_{r\hat{e}g}$ composed of MAE losses $\mathcal{L}_{\hat{o}}$, $\mathcal{L}_{\hat{hg}}$, $\mathcal{L}_{\hat{s}}$, $\mathcal{L}_{\hat{\alpha}}$ and $\mathcal{L}_{\hat{v}}$ using the same weight coefficients as in the first stage.

The second stage loss is then:

$$\mathcal{L}_{B2} = \mathcal{L}_{score} + \mathcal{L}_{r\hat{e}g}, \tag{3}$$

the final confidence score $\hat{Q}_t$ of object t is the geometric average of the scores of the first stage $\hat{Y}_t$ and second one $\hat{I}_t$ as $\hat{Q}_t = \sqrt{\hat{Y}_t \cdot \hat{I}_t}$, where $\hat{Y}_t = \max_{0 \le c \le C} \mathbf{Y}_{c,t}$ is the maximum value at the center of object t of the heatmaps.

B. Class prototypical loss for 3D Object Detection

Following [1] we use a training strategy with two stages, as shown in Fig. 3. In the first one, we train the Centerpoint [1] model with the same object detection losses, with the addition of a 1D batch normalization layer after the extraction of feature vectors $\mathbf{f}_{gt} \in \mathbb{R}^{330}$ for the ground truth objects, to generate a normalized feature vector $\hat{\mathbf{f}}_{gt} \in \mathbb{R}^{330}$.

Once the first stage is trained, we extract the feature vectors $\hat{\mathbf{f}}_{gt} \in \mathbb{R}^{330}$ for all the objects of the ground truth in the training set. We only consider objects that have a minimum number of points (60 in our experiments) so that they will be representative of the class. Then we compute the class prototypes $\mathbf{P}_c$ as the mean $\bar{\mathbf{f}}_c$ and the covariance $\mathbf{S}_c$ of the feature vectors for each class c and store them.

In the second training stage, in addition to the loss exploited in [1] we consider a class prototypical loss as:

$$\mathcal{L}_{CP} = \frac{1}{N} \sum_x \sqrt{(\hat{\mathbf{f}}_x - \bar{\mathbf{f}}_c)^\intercal \mathbf{S}_c^{-1} (\hat{\mathbf{f}}_x - \bar{\mathbf{f}}_c)}, \tag{4}$$

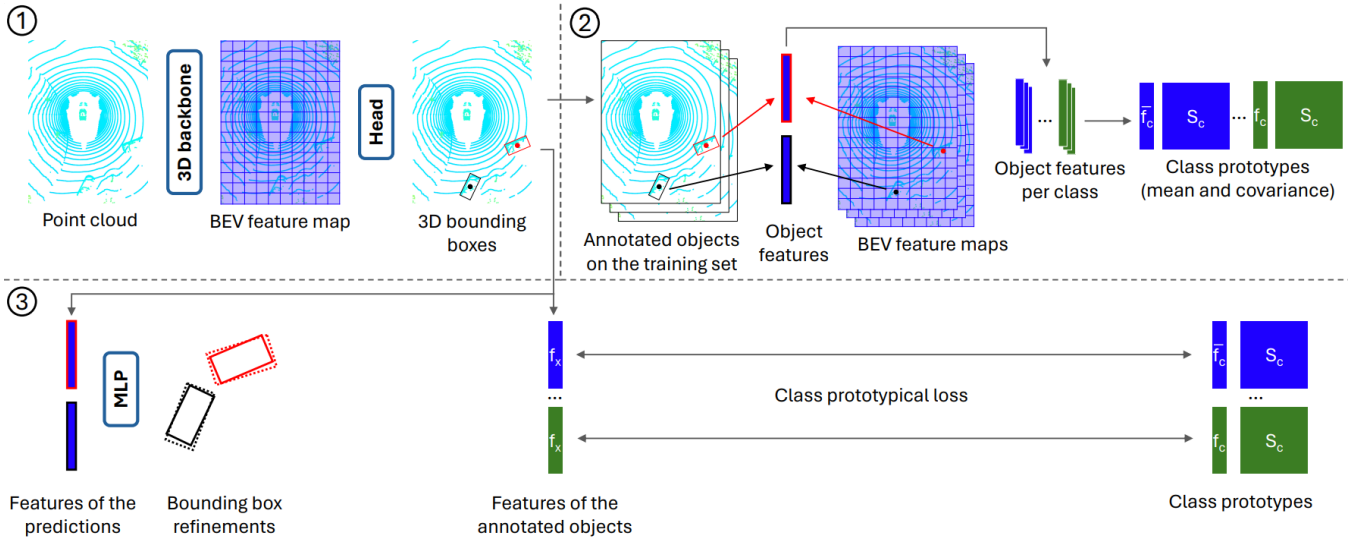


Fig. 3: **Training stages with the class protypical loss.** 1) On a first stage we train a 3D backbone and head to generate 3D bounding boxes with the detection losses from the baseline. 2) We calculate the mean and covariance of the feature vectors for each class taking annotated objects from the training set with a minimum of 60 points. 3) On a second stage, we use an MLP to get bounding box refinements following Centerpoint, and add the class prototypical loss as the Mahalanobis distance of the feature vectors extracted for all annotated objects to the class prototypes of the corresponding classes.

that is defined as the average of the Mahalanobis distance from the feature vectors $\hat{\mathbf{f}}_x$ of the N ground truth objects x of class c to its class prototype $\mathbf{P}_c = \{\bar{\mathbf{f}}_c, \mathbf{S}_c\}$.

The final loss $\mathcal{L}$ is expressed as a combination of $\mathcal{L}_{B2}$ and the class prototypical loss as:

$$\mathcal{L} = \mathcal{L}_{B2} + w_{cp}\mathcal{L}_{CP}, \tag{5}$$

where w_{cp} is a weight coefficient.

We hypothesize that this loss will make the model generate BEV feature maps with better class separation. Moreover, if the class prototypes are fixed after the first training stage, the class prototypical loss acts as a regularization mechanism keeping the feature distributions stable and preventing overfitting.

C. Updating class prototypes after every epoch

Since the backbone is not frozen during the training of the second stage, the feature distribution of each class shifts during training. We discussed the potential benefits of having fixed prototypes, but updating the prototypes after each epoch could lead to an even more distinct feature distribution that could boost the performance of the object detector even further. There are two challenges that we need to consider when updating the prototypes after each epoch, collapse of covariances and collapse of means.

Since our proposed loss moves the feature distribution of each class closer to the mean of that class, this reduces the covariance. If we update the covariance after each epoch, it will be smaller every time. This will then lead to a higher value of the class prototypical loss since it is scaled by the inverse of the covariance matrix. A higher $\mathcal{L}_{CP}$ then leads to a further reduction in the covariance and after some iterations, the covariance could collapse to a very small value making the $\mathcal{L}_{CP}$ loss dominate too much over the other losses. To prevent the covariances from collapsing we propose to use a normalized inverse of the covariance matrix $\hat{\mathbf{S}}_c$, that is normalized by the mean of the absolute value of its elements. With this normalization, we keep the useful information on the shape of the distribution while keeping the scale under control, preventing collapse.

As for the evolution of the means of the feature distributions, since the $\mathcal{L}_{CP}$ loss pulls features close to the mean, a trivial setting that would lead to a low $\mathcal{L}_{CP}$ loss is all the means being very close with small covariances. To prevent the evolution of prototypes leading to a collapse of the means, we propose to add an inter-class prototypical loss $\mathcal{L}_{ICP}$, that pushes the features of each class away from the prototypes of the other classes. We define this loss as the average, over the N ground truth objects x and the $C-1$ classes different from its class c, of the inverse of the Mahalanobis distance from the feature vectors $\mathbf{f}_{\hat{x}}$ to the $C-1$ class prototypes $\mathbf{P}_{c_i} = \{\bar{\mathbf{f}}_{c_i}, \mathbf{S}_{ci}\}$:

$$\mathcal{L}_{ICP} = \frac{\sum_x \sum_{c_i \neq c} \left(\sqrt{(\hat{\mathbf{f}}_x - \bar{\mathbf{f}}_{c_i})^\top \mathbf{S}_{c_i}^{-1} (\hat{\mathbf{f}}_x - \bar{\mathbf{f}}_{c_i})} \right)^{-1}}{N(C-1)} \tag{6}$$

The weight of the inter-class $\mathcal{L}_{ICP}$ loss in relation to the intra-class $\mathcal{L}_{CP}$ loss is defined by a parameter $w_{icp} \in [0, 1]$. Finally, the final loss $\mathcal{L}_U$ can be expressed as:

$$\mathcal{L}_U = \mathcal{L}_{B2} + w_{cp}(w_{icp}\mathcal{L}_{CP} + (1 - w_{icp})\mathcal{L}_{ICP}). \tag{7}$$

In our experiments, we analyze the effect of our proposed losses and the contribution of each component.

1 lidar frame	NDS ↑	mAP ↑	car	truck	bus	trailer	const.	pede.	moto.	bicy.	cone	barrier
Baseline [1]	48.67	42.71	75.14	36.27	54.49	29.75	5.93	67.31	33.78	15.25	49.80	59.39
Baseline + $\mathcal{L}_{CP}$ $w_{cp} = 0.01$	**51.30**	**46.56**	**77.24**	37.73	55.21	**32.34**	**13.33**	70.95	**44.63**	21.26	53.04	59.88
Baseline + $\mathcal{L}_{CP}$ $w_{cp} = 0.10$	50.66	46.45	76.37	37.56	**58.18**	30.92	12.20	70.48	44.62	20.37	**53.57**	60.21
Baseline + $\mathcal{L}_{CP}$/$\mathcal{L}_{ICP}$, $w_{cp} = 0.01$, UP	50.01	45.75	75.59	**38.34**	55.72	30.23	11.43	**71.06**	40.83	**22.47**	51.18	60.59
Baseline + $\mathcal{L}_{CP}$/$\mathcal{L}_{ICP}$, $w_{cp} = 0.01$, UP	49.67	45.48	76.34	36.79	54.93	31.19	11.39	68.94	42.63	20.04	51.49	**61.11**

10 lidar frames	NDS ↑	mAP ↑	car	truck	bus	trailer	const.	pede.	moto.	bicy.	cone	barrier
Baseline [1]	70.73	67.04	**87.77**	63.64	71.81	39.79	23.87	**89.79**	75.15	64.68	80.34	**73.58**
Baseline + $\mathcal{L}_{CP}$, $w_{cp} = 0.01$	71.13	67.50	86.91	63.25	73.44	41.14	27.07	89.47	77.12	66.97	79.66	70.01
Baseline + $\mathcal{L}_{CP}$, $w_{cp} = 0.10$	71.36	67.95	86.99	64.19	72.65	41.98	26.81	89.22	77.93	67.79	79.87	72.02
Baseline + $\mathcal{L}_{CP}$/$\mathcal{L}_{ICP}$, $w_{cp} = 0.01$, UP	71.46	67.94	87.38	63.68	73.55	**43.37**	25.76	89.46	76.59	68.41	**80.37**	70.81
Baseline + $\mathcal{L}_{CP}$/$\mathcal{L}_{ICP}$, $w_{cp} = 0.1$, UP	**72.04**	**68.80**	87.33	**64.39**	**74.14**	43.28	**27.33**	89.57	**79.60**	**70.99**	79.97	71.41

TABLE I: **Validation metrics on NuScenes [7]**. The table reports NDS, average mAP and mAP per class for different configurations and methods. UP refers to updating prototypes.

IV. Experiments

A. Experimental Setup

We perform most of our experiments on NuScenes [7], which contains 700, 150, and 150 driving scenes for training, validation, and testing, respectively. Each sequence has an approximate duration of 20 seconds, and the lidar operates at a frequency of 20 frames per second. The dataset provides calibrated vehicle pose information for each lidar frame, with box annotations available every ten frames (0.5 seconds), because of this way of annotating many works provide results aggregating ten frames. The lidar sensor has 32 beams, generating approximately 30,000 points per frame. Annotations cover 10 classes: car, truck, bus, trailer, construction vehicle, pedestrian, motorcycle, bicycle, cone and barrier. We test our method under different conditions by considering one and ten frames. For the 10-frame case, we use Centerpoint [1] with Voxelnet [4] as the backbone, two stages and virtual points [32]. Following the same configuration, we use a range of $[-54, 54]$m in X and Y axis, and a range of $[-5.0, 3.0]$m in the Z axis to voxelize the point cloud. The resolution is set to 0.075m for the X and Y axis, and 0.2m for the Z one. For the 1-frame case, we use the same model without additional virtual points and with the voxelization strategy used in ST3D [52], [53] and other domain adaptation works. A range of $[-75.2, 75.2]$m in X and Y axis, and a $[-2.0, 4.0]$m in the Z after moving the ground plane to 0, in NuScenes [7] that corresponds to adding 1.8 to the Z component of every point. The resolution is 0.1m for the X and Y axis, and 0.15m for the Z one. Following [1], the weight coefficients for the first stage are set to $w_{reg} = 0.25$, $w_o = w_{hg} = w_s = w_v = 1$, $w_\alpha = 0.2$ in all our experiments on Nuscenes [7]. We use the 3D detection metrics defined in [7]: a mean average precision (mAP) and the NuScenes detection score (NDS).

To test the distribution of feature vectors across different datasets, we also use the Waymo Open Dataset [54], which contains 798 training sequences and 202 validation ones for classes vehicle, pedestrian and bicycle. The point clouds are captured with a 64-beam lidar, which produce about 180k lidar points every 0.1 seconds, a significantly higher resolution than in NuScenes [7]. We report 3D object detection results only on the NuScenes dataset since our method focuses on increasing the discrimination of similar classes and the three Waymo classes are very distinct.

B. 3D Object Detection with the class prototypical loss

We first evaluate our method for 3D object detection on NuScenes [7] with 1 and 10 lidar frames, fixing $w_{cp} = \{0.01, 0.10\}$, $w_{icp} = 0.5$, and comparing our solution with Centerpoint [1]. We show results in terms of NDS, average mAP, and mAP per class in Table I. As it can be observed, using the class prototypical losses consistently increases both NDS and average mAP in all cases.

When 1 frame is used, the point clouds are quite sparse because NuScenes [7] was recorded with a low-resolution lidar. In this case, it is hard for the model to learn useful feature representations for objects with a low number of points, and the additional supervisory signal from the $\mathcal{L}_{CP}$/$\mathcal{L}_{ICP}$ losses leads to an improvement in mAP across all classes, especially for the classes with lower mAP. For the case with 10 frames, the point clouds are denser and therefore, using the $\mathcal{L}_{CP}$/$\mathcal{L}_{ICP}$ losses increases the mAP of the classes that have other similar classes within the dataset, i.e., large vehicles (truck, bus, trailer and construction vehicle) and 2-wheeled vehicles (bicycle and motorcycle). These classes have close feature representations when the model is trained without the $\mathcal{L}_{CP}$/$\mathcal{L}_{ICP}$ losses, and those losses help to make them more distinct, as we will analyze later.

We analyze the confidence of the detections, both true positives and false ones, for our method (68.80 mAP) and the baseline (67.04 mAP) by plotting histograms for each class in Fig. 4. In the figure, we can see that our method produces fewer false positives, and they have a lower confidence score compared to the baseline, especially for the classes with other similar classes in the dataset. The lower number of false positives can also be seen in the detections displayed in Fig. 5.

C. Analysis of feature distribution

We hypothesize that there is an underlying structure in the feature distributions of the objects based on their class, and that we can exploit this structure to boost the performance of the object detector by adding a loss to make the BEV feature map more consistent with this structure. To test our initial hypothesis, we use a Centerpoint [1] model trained on NuScenes [7] without our proposed loss, to extract feature vectors $\mathbf{f}_x \in \mathbb{R}^{330}$ from the BEV feature map corresponding to annotated objects x on the validation set. After the

feature extraction, we add a 1D batch normalization layer that outputs normalized features $\hat{\mathbf{f}}_x \in \mathbb{R}^{330}$ to facilitate the visualization with t-SNE [55]. The t-SNE plot of the features $\hat{\mathbf{f}}_x$, shown in Fig. 2a, revealed a structure with objects from the same class closer than objects from different classes.

Additionally, we modified the data loader so that all the objects would be considered from the same class during training and then did the t-SNE projection using the actual class to colour the points, as seen in Fig. 2b. We can see that the different classes are still separated, although similar classes are more mixed than in the case of the detector trained with class information. This means that the BEV feature map represents different classes distinctively even when trained without class information. This could be used to discover different object classes from a class-agnostic general object detector based on a clustering of the feature vectors.

To see how consistent these representations are across datasets, we train a model on the Waymo [54] dataset and extract feature vectors from the predicted objects when applying the model to both Waymo [54] and NuScenes [7] datasets. By using the t-SNE projection we can see in Fig. 6 that there is still a structure even when the underlying point clouds have a different number of beams and resolution. Based on these findings, we hypothesize that we could also use the proposed loss for unsupervised domain adaptation in 3D object detection, but this study is left for future work.

D. Feature separation with the class prototypical loss

To analyze the effect of our proposed loss on the feature separation, for each class c_k we get the feature vectors $\{\hat{\mathbf{f}}_{ck,i} \mid i \in [1, N]\}$ of N=2000 ground truth objects in the validation set and calculate the distance $d_{ck,cl}$ between two classes c_k and c_l as the mean of the pair-wise Euclidean distance between all the pairs of feature vectors from these classes as:

$$d_{ck,cl} = \sum_{i=1}^{N} \sum_{j=1}^{N} N^{-2} \|\hat{\mathbf{f}}_{ck,i} - \hat{\mathbf{f}}_{cl,j}\|_2. \tag{8}$$

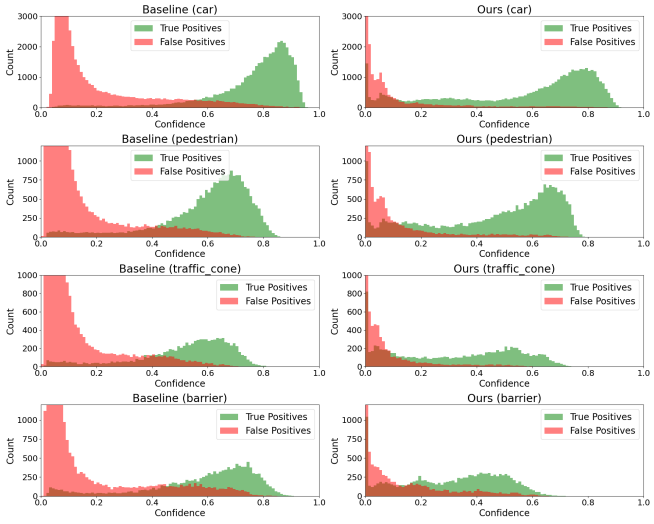
We can then calculate normalized distances $\hat{d}_{ck,cl}$ dividing $d_{ck,cl}$ by the average of the distance between the objects of the same class $\{d_{ck,ck}, d_{cl,cl}\}$:

$$\hat{d}_{ck,cl} = \frac{2 \cdot d_{ck,cl}}{d_{ck,ck} + d_{cl,cl}}. \tag{9}$$

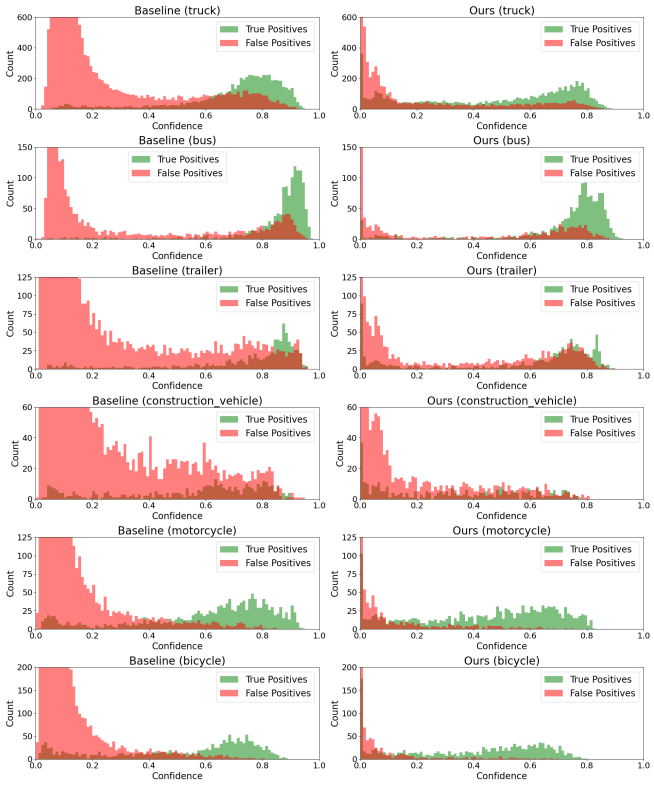
Note that the normalized distance between a class and itself is 1, i.e., $\hat{d}_{ck,ck} = 1, \forall k$.

We can now construct a matrix $\mathbf{M}$ of relative feature distances between classes by assigning $\hat{d}_{ck,cl}$ to the element in the k-th row and l-th column. This creates a symmetric matrix with diagonal values equal to 1. Higher values on the matrix represent a higher inter-class separation of the features, relative to the intra-class separation of the features.

Figure 7 shows this matrix as a heat map, for a model trained with and without our proposed class prototypical losses, where we can see that the feature separation increases when the loss we propose is used.



(a) Distinct classes



(b) Similar classes

Fig. 4: **Confidence histograms for true and false positives**. Compared to the baseline [1], our method reduces the number of false positives, and lowers their confidence scores, especially for similar classes.

In Table II, we show the average feature distance for all classes, for the similar classes and for the distinct ones on NuScenes [7], where we see that the feature separation increases when using our loss in all configurations. This effect is especially seen in similar classes since they have an average distance of 1.1, very close to the distance to the same class which is 1. Using our proposed loss, this

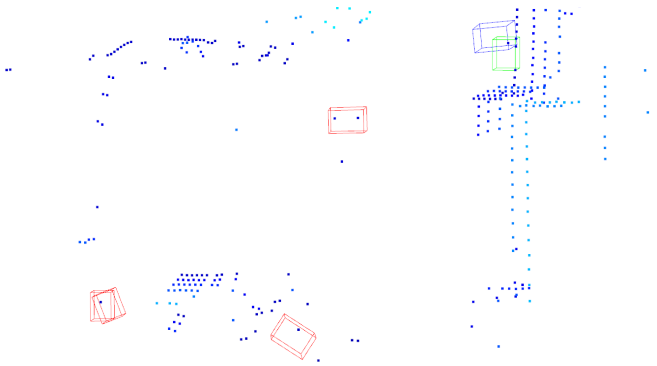


Fig. 5: **Detected bicycles and motorcycles**. The point cloud is represented by blue squares, with lighter values indicating points higher above the ground. The annotations in green, the baseline [1] predictions in red, and ours in blue. We detect the bicycle while avoiding some false positives.

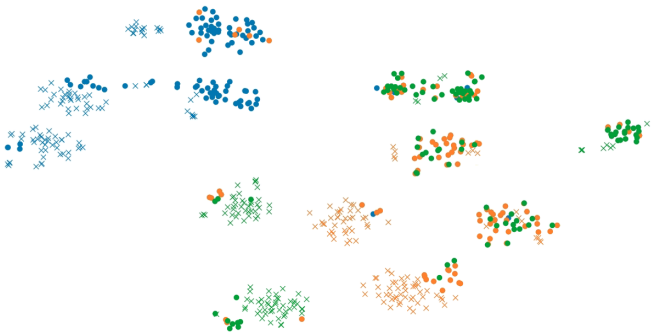


Fig. 6: **t-SNE plot of feature vectors across datasets.** Waymo [54] (x) and Nuscenes [7] (·) feature vectors for a model trained on Waymo [54], color-coded by class. Notice how there is some consistent structure between datasets.

distance increases to at least 1.25. Therefore, the feature representations for similar classes are more distinct.

E. Ablation study

We have trained the model with different configurations to assess the contribution of each module, and the results are reported in Table II. There, we can see that the best object detection performance comes from activating all the modules and that the feature separation improves when using the class prototypical loss in all configurations with respect to the baseline without our proposed loss. Interestingly, the best feature separation is achieved when the covariance normalization and inter-class prototypical loss are not used.

V. Conclusion

We analyzed the feature vectors extracted per object from the BEV feature map of a 3D object detector and showed a structure with the feature distribution of each class separated. Moreover, we showed that feature distributions are distinct even when the object detector has not been provided with class-level supervision.

This motivated us to propose a novel class prototypical loss for 3D object detection based on the Mahalanobis

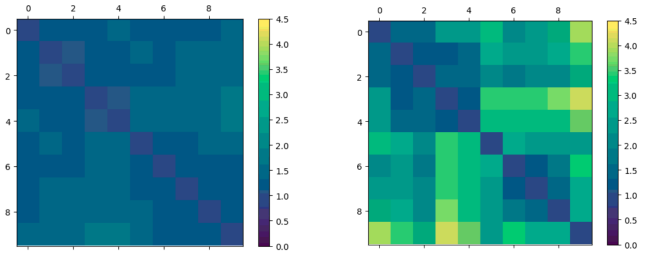


Fig. 7: **Relative feature distance between classes**. **Left:** Baseline [1]. **Right:** Baseline with $\mathcal{L}_{CP}$/$\mathcal{L}_{ICP}$ loss. Notice how using our class prototypical loss significantly improves the feature separation across all classes.

Method				Detection metrics ↑		Feature separation ↑		
$\mathcal{L}_{CP}$	CN	$\mathcal{L}_{ICP}$	UP	NDS	mAP	All	Sim.	Dist.
				70.73	67.04	1.311	1.100	1.391
✓				71.36	67.95	2.336	1.315	2.700
✓	✓			71.85	68.82	2.134	1.261	2.444
✓		✓		71.79	68.23	2.145	1.244	2.461
✓			✓	71.66	68.31	**2.898**	**1.387**	**3.426**
✓	✓	✓		71.99	68.45	2.001	1.255	2.270
✓	✓		✓	71.32	67.90	2.504	1.321	2.920
✓		✓	✓	71.45	68.03	2.784	1.372	3.278
✓	✓	✓	✓	**72.04**	**68.80**	1.993	1.252	2.260

TABLE II: **Ablation study.** Object detection results and feature separation for all, similar (Sim.) and distinct (Dist.) classes. The components studied are the class prototypical loss ($\mathcal{L}_{CP}$), the normalization of the covariance matrix (CN), the inter-class prototypical loss ($\mathcal{L}_{ICP}$) and the update of the prototypes after each epoch (UP). The best object detection results are obtained with our method when all the components are considered.

distance of the feature vectors to the class prototype of their corresponding annotated class, and we showed how it can be used as an auxiliary loss to improve the performance of the object detector as well as to increase the feature separation between classes. The simplicity of our approach makes it easy to apply to most 3D object detectors, as the only requirement is that a BEV feature map can be extracted. Furthermore, we analyzed the feature distribution of objects from NuScenes and Waymo datasets, observing some consistent structure of the distributions per class across datasets. In future work, it would be interesting to see whether the class prototypical loss could be used as an auxiliary loss for domain adaptation, by using the loss to shift the feature distribution for each class on the target dataset closer to the feature distribution on the source dataset. Moreover, the class prototypical loss proposed could be extended to other modalities such as images, with minor modifications.

VI. Acknowledgments

This work has been supported by the project GRAVATAR PID2023-151184OB-I00 funded by MCIU/AEI/10.13039/501100011033 and by ERDF, UE and by the Government of Catalonia under 2020 DI 105.

REFERENCES

[1] T. Yin, X. Zhou, and P. Krahenbuhl, "Center-based 3D object detection and tracking," in *CVPR*, 2021.

[2] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *CVPR*, 2019.

[3] Y. Yan, Y. Mao, and B. Li, "Second: Sparsely embedded convolutional detection," *Sensors*, vol. 18, no. 10, p. 3337, 2018.

[4] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3D object detection," in *CVPR*, 2018.

[5] Y. Chen, Z. Yu, Y. Chen, S. Lan, A. Anandkumar, J. Jia, and J. M. Alvarez, "Focalformer3D: Focusing on hard instance for 3D object detection," in *CVPR*, 2023.

[6] H. Li, C. Sima, J. Dai, W. Wang, L. Lu, H. Wang, J. Zeng, Z. Li, J. Yang, H. Deng, *et al.*, "Delving into the devils of bird's-eye-view perception: A review, evaluation and recipe," *PAMI*, 2023.

[7] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "Nuscenes: A multimodal dataset for autonomous driving," in *CVPR*, 2020.

[8] W. Liu, Y. Wen, Z. Yu, and M. Yang, "Large-margin softmax loss for convolutional neural networks," *arXiv preprint arXiv:1612.02295*, 2016.

[9] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song, "Sphereface: Deep hypersphere embedding for face recognition," in *CVPR*, 2017.

[10] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," in *CVPR*, 2019.

[11] R. Lin, W. Liu, Z. Liu, C. Feng, Z. Yu, J. M. Rehg, L. Xiong, and L. Song, "Regularizing neural networks via minimizing hyperspherical energy," in *CVPR*, 2020.

[12] M. Engelcke, D. Rao, D. Z. Wang, C. H. Tong, and I. Posner, "Vote3deep: Fast object detection in 3D point clouds using efficient convolutional neural networks," in *ICRA*, 2017.

[13] B. Li, T. Zhang, and T. Xia, "Vehicle detection from 3D lidar using fully convolutional network," *arXiv preprint arXiv:1608.07916*, 2016.

[14] Y. Zhou, P. Sun, Y. Zhang, D. Anguelov, J. Gao, T. Ouyang, J. Guo, J. Ngiam, and V. Vasudevan, "End-to-end multi-view fusion for 3D object detection in lidar point clouds," in *CoRL*, 2020.

[15] Y. Wang, A. Fathi, A. Kundu, D. A. Ross, C. Pantofaru, T. Funkhouser, and J. Solomon, "Pillar-based object detection for autonomous driving," in *ECCV*, 2020.

[16] B. Yang, W. Luo, and R. Urtasun, "Pixor: Real-time 3D object detection from point clouds," in *CVPR*, 2018.

[17] B. Zhu, Z. Jiang, X. Zhou, Z. Li, and G. Yu, "Class-balanced grouping and sampling for point cloud 3D object detection," *arXiv preprint arXiv:1908.09492*, 2019.

[18] S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang, and H. Li, "Pv-rcnn: Point-voxel feature set abstraction for 3D object detection," in *CVPR*, 2020.

[19] C. R. Qi, O. Litany, K. He, and L. J. Guibas, "Deep hough voting for 3D object detection in point clouds," in *ICCV*, 2019.

[20] B. Graham, M. Engelcke, and L. Van Der Maaten, "3D semantic segmentation with submanifold sparse convolutional networks," in *CVPR*, 2018.

[21] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *CVPR*, 2017.

[22] S. Shi, L. Jiang, J. Deng, Z. Wang, C. Guo, J. Shi, X. Wang, and H. Li, "PV-RCNN++: Point-voxel feature set abstraction with local vector representation for 3D object detection," *IJCV*, vol. 131, no. 2, pp. 531–551, 2023.

[23] Y. Chen, J. Liu, X. Zhang, X. Qi, and J. Jia, "Voxelnext: Fully sparse voxelnet for 3D object detection and tracking," in *CVPR*, 2023.

[24] D. Huang, Y. Chen, Y. Ding, J. Liao, J. Liu, K. Wu, Q. Nie, Y. Liu, C. Wang, and Z. Li, "Rethinking dimensionality reduction in grid-based 3D object detection," *arXiv preprint arXiv:2209.09464*, 2022.

[25] Y. Chen, J. Liu, X. Zhang, X. Qi, and J. Jia, "Largekernel3d: Scaling up kernels in 3D sparse CNNs," in *CVPR*, 2023.

[26] T. Lu, X. Ding, H. Liu, G. Wu, and L. Wang, "Link: Linear kernel for lidar-based 3D perception," in *CVPR*, 2023.

[27] Y. Zhang, Q. Hu, G. Xu, Y. Ma, J. Wan, and Y. Guo, "Not all points are equal: Learning highly efficient point-based detectors for 3D lidar point clouds," in *CVPR*, 2022.

[28] H. Wang, C. Shi, S. Shi, M. Lei, S. Wang, D. He, B. Schiele, and L. Wang, "DSVT: dynamic sparse voxel transformer with rotated sets," in *CVPR*, 2023.

[29] D. Zhang, Z. Zheng, H. Niu, X. Wang, and X. Liu, "Fully sparse transformer 3D detector for lidar point cloud," *TGRS*, 2023.

[30] J. Mao, Y. Xue, M. Niu, H. Bai, J. Feng, X. Liang, H. Xu, and C. Xu, "Voxel transformer for 3D object detection," in *ICCV*, 2021.

[31] P. Sun, M. Tan, W. Wang, C. Liu, F. Xia, Z. Leng, and D. Anguelov, "Swformer: Sparse window transformer for 3D object detection in point clouds," in *ECCV*, 2022.

[32] T. Yin, X. Zhou, and P. Krähenbühl, "Multimodal virtual point 3D detection," *NeurIPS*, 2021.

[33] N. Benbarka, J. Schröder, and A. Zell, "Score refinement for confidence-based 3D multi-object tracking," in *IROS*, 2021.

[34] M. Perez and A. Agudo, "Robust multimodal and multi-object tracking for autonomous driving applications," in *ICAR*, 2023.

[35] V. Papyan, X. Han, and D. L. Donoho, "Prevalence of neural collapse during the terminal phase of deep learning training," *PNAS*, vol. 117, no. 40, pp. 24 652–24 663, 2020.

[36] X. Han, V. Papyan, and D. L. Donoho, "Neural collapse under MSE loss: Proximity to and dynamics on the central path," *arXiv preprint arXiv:2106.02073*, 2021.

[37] J. Zhou, C. You, X. Li, K. Liu, S. Liu, Q. Qu, and Z. Zhu, "Are all losses created equal: A neural collapse perspective," *NeurIPS*, 2022.

[38] J. Park, H. Park, E. Jeong, and A. B. J. Teoh, "Understanding open-set recognition by jacobian norm and inter-class separation," *PR*, vol. 145, p. 109942, 2024.

[39] J. Ma, Y. Niu, J. Xu, S. Huang, G. Han, and S.-F. Chang, "Digeo: Discriminative geometry-aware learning for generalized few-shot object detection," in *CVPR*, 2023.

[40] P. Mettes, E. Van der Pol, and C. Snoek, "Hyperspherical prototype networks," *NeurIPS*, vol. 32, 2019.

[41] B. Li, B. Yang, C. Liu, F. Liu, R. Ji, and Q. Ye, "Beyond max-margin: Class margin equilibrium for few-shot object detection," in *CVPR*, 2021.

[42] K. Joseph, S. Khan, F. S. Khan, and V. N. Balasubramanian, "Towards open world object detection," in *CVPR*, 2021.

[43] J. Yu, L. Ma, Z. Li, Y. Peng, and S. Xie, "Open-world object detection via discriminative class prototype learning," *arXiv preprint arXiv:2302.11757*, 2023.

[44] Y. Zhang, Z. Wang, and Y. Mao, "RPN prototype alignment for domain adaptive object detector," in *CVPR*, 2021.

[45] D. Hegde and V. M. Patel, "Attentive prototypes for source-free unsupervised domain adaptive 3D object detection," in *WACV*, 2024.

[46] A. Bar-Hillel, T. Hertz, N. Shental, D. Weinshall, and G. Ridgeway, "Learning a mahalanobis metric from equivalence constraints." *JMLR*, vol. 6, no. 6, 2005.

[47] S. Xiang, F. Nie, and C. Zhang, "Learning a mahalanobis distance metric for data clustering and classification," *PR*, vol. 41, no. 12, pp. 3600–3612, 2008.

[48] R. Kamoi and K. Kobayashi, "Why is the mahalanobis distance effective for anomaly detection?" *arXiv preprint arXiv:2003.00402*, 2020.

[49] T. Denouden, R. Salay, K. Czarnecki, V. Abdelzad, B. Phan, and S. Vernekar, "Improving reconstruction autoencoder out-of-distribution detection with mahalanobis distance," *arXiv preprint arXiv:1812.02765*, 2018.

[50] P. M. Roth, M. Hirzer, M. Köstinger, C. Beleznai, and H. Bischof, "Mahalanobis distance learning for person re-identification," *Person re-identification*, pp. 247–267, 2014.

[51] A. Venkataramanan, A. Benbihi, M. Laviale, and C. Pradalier, "Gaussian latent representations for uncertainty estimation using mahalanobis distance in deep classifiers," in *ICCV*, 2023.

[52] J. Yang, S. Shi, Z. Wang, H. Li, and X. Qi, "ST3D: self-training for unsupervised domain adaptation on 3D object detection," in *CVPR*, 2021.

[53] J. Yang, S. Shi, Z. Wang, H. Li, and X. Qi, "ST3D++: Denoised self-training for unsupervised domain adaptation on 3D object detection," *PAMI*, vol. 45, no. 5, pp. 6354–6371, 2022.

[54] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, *et al.*, "Scalability in perception for autonomous driving: Waymo open dataset," in *CVPR*, 2020.

[55] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE," *JMLR*, vol. 9, no. 11, 2008.