Appendix

A Implementation Details

We follow the training procedure of the respective detectors [8, 13, 17] which have open-source code. We describe important implementation details below.

- *Model Architecture.* We adopt the architecture in [9] but make an important modification. The original architecture (for the standard nuScenes benchmark) has six heads designed for ten classes; each head has 64 filters. We first adapted this architecture for LT3D using seven heads designed for 18 classes. We then replace these seven heads with a single head consisting of 512 filters shared by all classes.
- *Training Losses.* We use the sigmoid focal loss (for recognition) [54] and L1 regression loss (for localization) below. Existing works also use the same losses but only with fine labels; we apply the loss to both coarse and fine labels. Concretely, our loss function for CenterPoint is as follows: $L = L_{HM} + \lambda L_{REG}$, where $L_{HM} = \sum_{i=0}^{C} SigmoidFocalLoss(X_i, Y_i)$ and $L_{REG} = |X_{BOX} - Y_{BOX}|$, where X_i and Y_i are the i^{th} class' predicted and ground-truth heat maps, while X_{BOX} and Y_{BOX} are the predicted and ground-truth box attributes. Without our hierarchical loss, C=18. With our hierarchical loss, C=22 (18 fine grained + 3 coarse + 1 object class). λ is set to 0.25. Modifications for other detectors similarly follow.
- *Post-processing.* We use non-maximum suppression (NMS) on detections *within* each class to suppress lower-scoring detections. In contrast, existing works apply NMS on all detections *across* classes, i.e., suppressing detections overlapping other classes' detections (e.g., a pedestrian detection can suppress other pedestrian and traffic cone detections).

B Analysis on Multi-Head Architectures and Class Grouping

Many contemporary networks use a multi-head architecture that groups classes of similar size and shape to facilitate efficient feature sharing. For example, CenterPoint groups `pedestrian` and `traffic-cone` since these objects are both tall and skinny. We study the impact of grouping for both the standard and LT3D problem setups. We define the groups used for this study below. Each group is enclosed in curly braces. Our group-free head includes all classes into a single group.

- Original: {`Car`}, {`Truck, Construction Vehicle`}, {`Bus, Trailer`}, {`Barrier`}, {`Motorcycle, Bicycle`}, {`Pedestrian, Traffic Cone`}
- LT3D: {`Car`}, {`Truck, Construction Vehicle`}, {`Bus, Trailer`}, {`Barrier`}, {`Motorcycle, Bicycle`}, {`Adult, Child, Construction Worker, Police Officer, Traffic Cone`}, {`Pushable Pullable, Debris, Stroller, Personal Mobility, Emergency Vehicle`}

We use the class groups proposed by prior works [9, 13] for the standard benchmark and adapt this grouping for LT3D. Our proposed group-free detector head architecture consistently outperforms grouping-based approaches on both the standard and LT3D benchmarks. We note that sub-optimal grouping strategies (such as those adopted for LT3D) may yield significantly diminished performance, whereas optimized grouping strategies (such as those adopted for the standard setup) have comparable performance to the group-free approach. The group-free approach simplifies architecture design, while also providing competitive performance.

Two insights allow us to train the group-free architecture. First, we make the group-free head proportionally larger to train more classes. The standard grouping setup contains 6 heads, each with 64 convolutional filters. Scaling up to the nearest power of two, our group-free head has 512 convolutional filters. Second, we do not perform between-class NMS. The standard setup performs NMS between classes in each group (e.g., since pedestrians and traffic cones are tall and skinny, the model should only predict that an object is either a traffic cone or a pedestrian). However, performing NMS between classes requires that confidence scores are calibrated, which is not the case. Moreover, for LT3D, score calibration becomes more important for `rare` classes as these classes have lower confidence scores than `common` classes on average, meaning that `common` objects will

Table 3: Our proposed group-free detector head architecture consistently outperforms grouping-based approaches on both the standard and LT3D benchmarks. We note that sub-optimal grouping strategies (such as those adopted for LT3D) may yield significantly diminished performance, whereas optimized grouping strategies (such as those adopted for the standard setup) have comparable performance to the group-free approach. Note, `TC` is `traffic-cone`, `CV` is `construction vehicle`, `MC` is `motorcycle`, `PP` is `pushable-pullable`, `CW` is `construction-worker`, and `PO` is `police-officer`. We highlight classes with `Medium` and `Few` examples per class in blue.

CenterPoint	Multi-Head	Car	Ped.	Barrier	TC	Truck	Bus	Trailer	CV	MC	Bicycle
Original	✓	87.7	87.7	70.7	74.0	63.6	72.7	**45.1**	**26.3**	64.7	47.9
		89.1	**88.4**	**70.8**	**74.3**	**64.8**	**72.9**	42.0	25.7	**65.9**	**53.6**
for LT3D	✓	82.4	—	62.0	60.1	49.4	55.7	28.9	19.7	48.9	33.6
		88.1	—	**72.4**	**72.7**	**62.7**	**70.8**	**40.2**	**24.5**	**62.8**	**48.5**
		Adult	PP	CW	Debris	Child	Stroller	PO	EV	PM	`All`
Original	✓	—	—	—	—	—	—	—	—	—	64.0
		—	—	—	—	—	—	—	—	—	**64.8**
for LT3D	✓	81.2	21.7	14.2	1.1	**0.1**	0.1	**1.3**	0.1	**0.1**	31.2
		86.3	**32.7**	**22.2**	**4.3**	**0.1**	**4.3**	**1.8**	10.3	**0.1**	**39.2**

likely suppress `rare` objects within the same group. Our solution is to only perform within-class NMS, which is standard for 2D detectors [55].

C Ablation on Hierarchical Training and Inference

Classic methods train a hierarchical softmax (in contrast to our simple approach of sigmoid focal loss with both fine and coarse classes), where one multiplies the class probabilities of the hierarchical predictions during training and inference [56]. We implemented such an approach, but found the training did not converge. Interestingly, [56] shows such a hierarchical softmax loss has little impact on long-tailed object detection (in 2D images), which is one reason they have not been historically adopted. Instead, we found better results using the method from [57] (a winning 2D object detection system on the LVIS [40] benchmark) which multiples class probabilities of predictions (e.g. $P_{CAR} = P_{OBJ} * P_{CAR}$) at test-time, even when such predictions are not trained with a hierarchical softmax. We tested three variants and compared it to our approach (which recall, uses only fine-grained class probabilities at inference). Table 4 compares their performance for LT3D.

Table 4: Different variants achieve similar performance. We note that other methods do improve accuracy in the tail by sacrificing performance in the head, suggesting that hybrid approaches that apply different techniques for head-vs-tail classes may further improve accuracy. Unlike [57, 56] which requires a strict label hierarchy, our approach is not limited to a hierarchy.

Method	Hierarchy	`Many`	`Medium`	`Few`	`All`
CenterPoint (w/o Hierarchy) [13]	n/a	76.4	43.1	3.5	39.2
CenterPoint w/ Hierarchy	(a)	**77.1**	45.1	4.3	40.4
	(b)	76.4	45.0	5.3	40.5
	(c)	76.5	**45.2**	5.2	**40.6**
	(d)	74.5	43.5	**5.6**	39.5

(a) Ours (e.g., Finegrain score only)

(b) Object score * Finegrain score ([57], e.g. $P_{CAR} = P_{OBJ} * P_{CAR}$)

(c) Coarse score * Finegrain score (Variant-1 of [57], e.g. $P_{CAR} = P_{VEHICLE} * P_{CAR}$)

(d) Object score * Coarse score * Finegrain score (Variant-2 of [57], e.g. $P_{CAR} = P_{OBJ} * P_{VEHICLE} * P_{CAR}$)

Unlike [57, 56] which require a strict label hierarchy, our approach is not limited to a hierarchy. We find that other hierarchical methods improve accuracy in the tail by sacrificing performance in the head, suggesting that hybrid approaches that apply different techniques for head-vs-tail classes may further improve accuracy.

D Evaluating on Argoverse 2.0

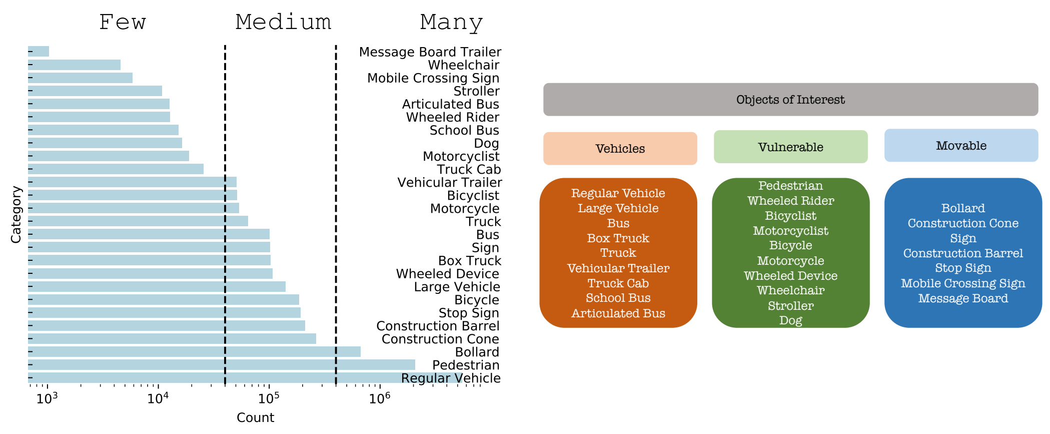

Figure 5: **Left:** According to the histogram of per-class object counts (on the **left**), classes in Argoverse 2.0 (AV2) follow a long tailed distribution. Following [7] and nuScenes (Fig. 1), we report performance for three groups of classes based on their cardinality (split by dotted lines): `Many`, `Medium`, and `Few`. As AV2 does not provide a class hierarchy, we construct one by referring to the nuScenes hierarchy (cf. Fig. 5 on the **right).**

We present results on the large-scale Argoverse 2.0 (AV2) dataset, another long-tailed dataset developed for autonomous vehicle research (Fig. 5 on the **left**). AV2 evaluates on 26 classes, which follow the long-tailed distribution. As AV2 does not provide a semantic hierarchy, we construct one (cf. Fig. 5 on the **right**) by adapting the nuSccenes hierarchy. As show in Table 5, our main conclusions still hold on AV2. Compared to the CenterPoint reported [49], our implementation improves mAP by 14.4 by carefully adopting LiDAR sweep aggregation and class-balanced sampling. Based on our implementation, exploiting hierarchical semantics improves mAP from 44.0 to 47.0. This improves performance for both `common` and `rare` classes alike. Interestingly, FCOS3D performs significantly worse than the LiDAR based detectors, yet multimodal filtering improves mAP to 48.4 mAP. These new results on AV2 are consistent with those on nuScenes, demonstrating the general applicability of our approach.

Table 5: Results (mAP in %) on Argoverse 2.0 (AV2) evaluated at 50 meters. Compared to the CenterPoint reported in [58], our implementation improves mAP by 14.4 by carefully adopting LiDAR sweep aggregation and class-balanced sampling. Based on our implementation, exploiting hierarchical semantics improves mAP from 44.0 to 47.0. Note that AV2 does not provide a semantic hierarchy; we construct one based on the nuScenes hierarchy (Fig. 3). Interestingly, FCOS3D performs significantly worse than the LiDAR based detectors, yet multimodal filtering improves mAP to 48.4 mAP. These new results on AV2 are consistent with those on nuScenes (cf. Table 1), demonstrating the general applicability of our approach.

Method	Multimodal	`Many`	`Medium`	`Few`	`All`
FCOS3D [59] (RGB-only)		27.4	17.0	7.8	14.6
CenterPoint (LiDAR-only) [13] reported by [49]		66.7	32.9	14.1	29.6
CenterPoint (LiDAR-only) [Our Implementation]		77.4	46.9	30.2	44.0
+ Hierarchy		**79.0**	50.3	33.6	47.0
+ Multimodal Filtering	✓	**79.0**	**51.4**	**35.3**	**48.4**