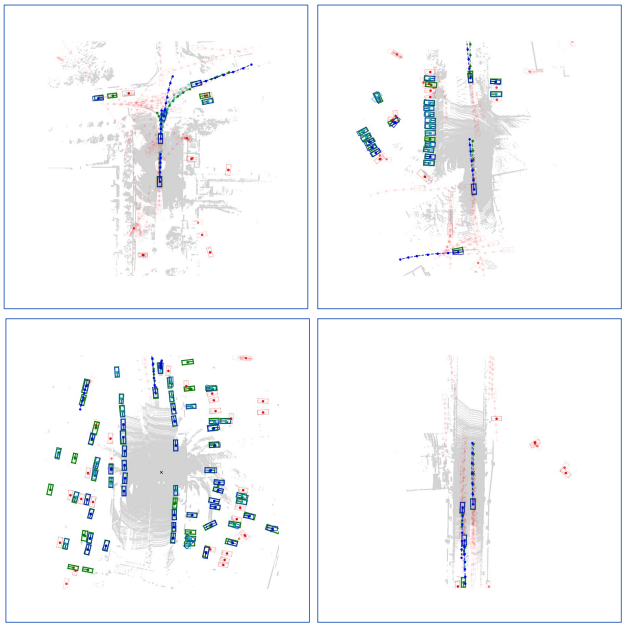

Figure 4. The raw output of *FutureDet* includes many false positive detections and forecasts (shown in **red**). Further post-processing is required to leverage the output of our end-to-end model in further downstream tasks.

A. Examining *FutureDet*'s Predictions

One of the challenges of forecasting from raw sensor data is appropriately handling false positive detections and forecasts. The standard forecasting setup allows us to build models in isolation from other factors. However, the standard assumption of having perfect input trajectories is not feasible in practice as it critically depends on *perfect* object tracks (and by extension perfect detections) as inputs, which are nearly impossible to obtain in practice. As seen in Figure 4, the raw output of *FutureDet* makes it challenging to use in practice. Intuitively, training a model to make "multiple bets" about the position of objects may induce more false positives. Further post-processing is required to leverage the output of our end-to-end model in further downstream tasks.

B. Evaluating Pedestrian Forecasting

In this section, we evaluate pedestrian forecasting on the nuScenes dataset. Forecasting pedestrian movement can be considerably more challenging than car forecasting because pedestrians are more dynamic. Given our 3 second forecasting horizon, pedestrians typically do not move very far from their initial position. As a result, we define tighter match thresholds for pedestrian forecasting. A successful match in the current frame is determined based on the distance from the center, averaged over distance thresholds of $\{0.125, 0.25, 0.5, 1\}m$. A successful match in the final timestep is determined based on the distance from center, averaged over distance thresholds of $\{0.25, 0.5, 1, 2\}m$ respectively. We highlight the results of pedestrian forecasting in Table 3.

We see that *FutureDet* performs the best overall, with 26.9 mAP_f. Looking to Figure 5, it is clear that pedestrian detections are tightly clustered together, making backcasting less effective overall. We also find that many of the predicted multiple-futures are very similar to one another, indicating that the model is not able to model dynamic pedestrian futures. However, *FutureDet* still consistently improves over FaF* by 1% on AP_f metrics.

We train a version of our model with road masks as an additional input channel into the BEV feature representation (after the sparse-voxel backbone). This brings very little change to the results. We hypothesize that adding the map information does not provide additional information. However, further exploration is required to evaluate how to best fuse LiDAR and map information.

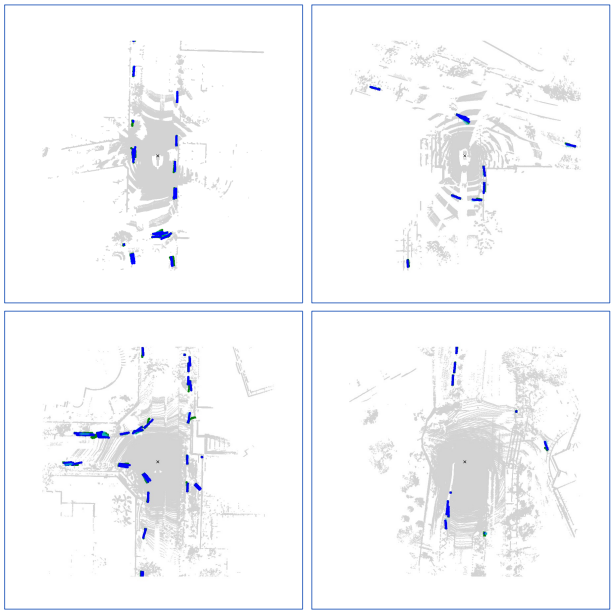

Figure 5. We qualitatively evaluate pedestrian forecasts from *FutureDet* (we denote the ground-truth trajectories with **green** and multiple future predictions with **blue** for the highest confidence forecast and **cyan** for the remaining future predictions). Pedestrian forecasting is more difficult than car forecasting due to the dynamic movement of pedestrians. *FutureDet* struggles to accurately forecast pedestrians because they often travel in crowds. This makes it difficult to accurately detect and forecast individual pedestrian motion. Often, the predicted multiple futures are all linearly moving, and are often similar to each other.

	K=1								K=5							
	$AP^{stat.}$		$AP^{lin.}$		$AP^{non-lin.}$		mAP		$AP^{stat.}$		$AP^{lin..}$		$AP^{non-lin.}$		mAP	
	$AP_{det.}$	AP_f	$AP_{det.}$	AP_f	$AP_{det.}$	AP_f	$mAP_{det.}$	mAP_f	$AP_{det.}$	AP_f	$AP_{det.}$	AP_f	$AP_{det.}$	AP_f	$mAP_{det.}$	mAP_f
Detection + Constant Velocity	**55.1**	33.3	73.5	27.8	96.9	12.4	**75.2**	24.5	**55.1**	33.3	73.5	27.8	96.9	12.4	**75.2**	24.5
Detection + Forecast (*cf*. [37])	53.7	**35.0**	**73.9**	30.8	**97.2**	13.3	74.9	26.4	53.7	35.0	**73.9**	30.8	**97.2**	13.3	74.9	26.4
Trajectron++ [43]	**55.1**	16.4	73.5	7.8	96.9	5.2	**75.2**	9.8	55.1	18.1	73.5	9.0	96.9	6.9	**75.2**	11.3
FutureDet	53.1	33.3	72.4	**32.6**	95.3	**14.7**	73.6	**26.9**	53.1	**35.1**	72.4	**34.0**	95.2	**15.0**	73.6	**28.0**
FutureDet-PointPillars	41.0	20.7	69.1	29.8	93.3	13.3	67.8	21.3	41.0	22.9	69.2	31.0	93.1	13.5	67.7	22.5
FutureDet + Map	52.4	33.0	71.8	32.0	95.3	14.4	73.2	26.5	52.4	34.8	71.8	33.4	95.2	14.8	73.2	27.7

Table 3. Joint pedestrian detection and forecasting evaluation on nuScenes. We adopt top-K evaluation for forecasting and evaluate under two settings of $K = 1$ and $K = 5$ (for forecasting only). We further breakdown the performance of each model by examining the detection AP ($AP_{det.}$) and forecasting AP (AP_f) on static, linear, and non-linearly moving sub-categories. Note that since pedestrians have smaller displacement over a 3 second forecasting horizon, we tighten the match thresholds as described above. *FutureDet* performs the best, improving over FaF* by 0.5 mAP_f. As with car forecasting, FaF* and the constant velocity baseline beat Trajectron++ by 14.4 % and 16.6 % mAP_f respectively. Notably, training with a PointPillars backbone dramatically reduces *FutureDet* performance on all metrics. In addition, we find that using a road mask does not significantly change the performance of *FutureDet*, indicating that the model might already be reasoning about spatial context.

C. *FutureDet* Architecture

In this section, we further describe the implementation details of *FutureDet*. Specifically, we focus on the detector head architecture, and the sampling strategy used to improve nonlinear trajectory forecasting.

Recurrent Features. We re-purpose CenterPoint for our implementation of *FutureDet*. However, CenterPoint is designed to detect objects in the current frame. It uses a shared feature representation for all classes. Although this effectively captures object spatial location, it does not allow for a robust representation of forecasted features. Specifically, since *FutureDet* detects `cars` and `future-cars`, we expect that the features required to detect these temporally offset classes should be different. To this end, we allow the model to learn a shallow network that transforms current features into future features as shown in Figure 6.

Trajectory Sampler. The distribution of static, linear, and non-linear trajectories in the nuScenes dataset is unbalanced. Since most cars are parked, we find that 60% of the trajectories are static. In order to address this data imbalance, we leverage copy-paste augmentation proposed by [57] to oversample linear and nonlinear trajectories during training. Importantly, we ensure that our copy-paste augmentation samples at the trajectory level, instead of at the class level, allowing consistent augmented trajectories across all detection heads (i.e. classes).

D. Computing Motion Subclass AP

Computing subclass average precision is straightforward in principle if both predictions and ground-truth have subclass labels; one can simply treat the sub-class as a class and apply standard precision-recall metrics. In our case,

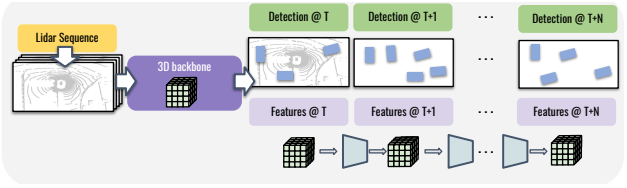

Figure 6. *FutureDet*'s detector head architecture adapts CenterPoint's architecture for the task of forecasting. Importantly, CenterPoint shares one set of features for all classes (i.e. cars, trucks, pedestrians, etc.). Since we adapt the architecture to forecast `cars` and `future-cars`, the single shared feature may not be able to effectively model long-term forecasting. To this end, we allow the model to learn a shallow network that transforms current features into future features.

predictions do not come with a subclass label. Instead, we match predictions to ground-truth at a class-level, and assign the ground-truth sub-class to the true positive matched predictions. However, this will not produce any sub-class labels for false positive predictions (that are unmatched). Instead, the metric evaluation code *derives* sub-class labels for false positive predictions, by applying the same logic used to derive sub-class labels for the ground-truth. We follow this procedure as it is also used to produce small-vs-large sub-class precision-recall metrics for standard detection toolkits [35]. Finally, although we use the language of sub-classes, our formalism can apply to any attributes associated with a detection.

We derive the subclass label as a function of the (ground truth or predicted) trajectory. For each trajectory, we first compute the IoU between bounding boxes at the first and last timestep. If the IoU is greater than 0, this trajectory is considered to be static. Next, using the velocity of the first

timestep bounding box, we apply a constant velocity forecast to the initial position to compute a target box. If the IoU between the last timestep box and target box is greater than 0, this trajectory is considered to be linear. All trajectories that are not classified as static or linear as considered to be non-linear trajectories.

E. Broader Impact

Autonomous agents will play an important role in the automation of tasks that can be considered unsafe (*e.g.*, due to a high number of traffic accidents). Forecasting is at the heart of autonomous vehicle navigation: safe navigation necessitates motion prediction of surrounding agents to ensure driving safety. By leveraging LiDAR sensory data to accomplish this task, we can better understand world geometry and dynamics. Moreover, establishing the proper metrics, particularly considering the performance of moving and static car trajectories, is essential for building safe embodied robotics systems.