Towards Fair and Comprehensive Comparisons for Image-Based 3D Object Detection

Xinzu Ma\textsuperscript{1,2}, Yongtao Wang\textsuperscript{3}, Yinmin Zhang\textsuperscript{1,2}, Zhiyi Xia\textsuperscript{4}, Yuan Meng\textsuperscript{4,*}
Zhihui Wang\textsuperscript{3,*}, Haojie Li\textsuperscript{3}, Wanli Ouyang\textsuperscript{1}
\textsuperscript{1}Shanghai AI Lab \textsuperscript{2}University of Sydney \textsuperscript{3}Dalian University of Technology \textsuperscript{4}Tsinghua University

\{maxinzhu, zhangyinmin, ouyangwanli\}@pjlab.org.cn
\{yongtaowang@mail., zhwang@, hjli\}@dlut.edu.cn \{yuanmeng, xiazy21\}@mail.tsinghua.edu.cn

Abstract

In this work, we build a modular-designed codebase, formulate strong training recipes, design an error diagnosis toolbox, and discuss current methods for image-based 3D object detection. In particular, different from other highly mature tasks, e.g., 2D object detection, the community of image-based 3D object detection is still evolving, where methods often adopt different training recipes and tricks resulting in unfair evaluations and comparisons. What is worse, these tricks may overwhelm their proposed designs in performance, even leading to wrong conclusions. To address this issue, we build a module-designed codebase and formulate unified training standards for the community. Furthermore, we also design an error diagnosis toolbox to measure the detailed characterization of detection models. Using these tools, we analyze current methods in-depth under varying settings and provide discussions for some open questions, e.g., discrepancies in conclusions on KITTI-3D and nuScenes datasets, which have led to different dominant methods for these datasets. We hope that this work will facilitate future research in image-based 3D object detection. Our codes will be released at https://github.com/OpenGVLab/3dodi.

1. Introduction

As a new and rapidly developing research field, vision-based 3D object detection [49] shows promising potential in autonomous driving and attracts lots of attention from both academia and industry. Thanks to the unremitting efforts of numerous researchers, lots of advanced technologies, such as model designs [3, 46, 29], detection pipelines [95, 83, 27], and challenging datasets [21, 6, 75], are continuously proposed, which significantly promotes the development of this research field.

However, although lots of breakthroughs in detection accuracy and inference speed have been achieved, there are still some critical problems to be solved, especially the standardization of evaluation protocols. Specifically, compared with the encouraging developments at the technique level, the conventional rules in model building, training recipes, and evaluation are not well-defined. As shown in Table 1, existing methods generally adopt different settings, e.g., backbones, training strategies, augmentations, etc., to build and evaluate their models, and the commonly used benchmarks (and most of the papers) only record the final accuracy. This makes the comparison of the detectors unfair and may lead to misleading conclusions.

In this paper, we aim to provide a unified platform for image-based 3D object detectors and standardize the protocols in model building and evaluation. Specifically, similar to the advanced codebases in 2D detection [9, 86], we decompose the image-based 3D detection frameworks into several separate components, e.g. backbones, necks, etc., and provide a unified implementation for current methods. Furthermore, we fully investigate existing algorithms and formulate several efficient training protocols, e.g. training schedules, data augmentation, etc. Our summarized training recipes not only provide a fair environment for evaluation but also significantly improve the performance of current methods, particularly in the KITTI-3D dataset. For example, the methods [51, 46] published two years ago trained with our recipes can achieve better (or similar) performance than the recent works [29, 55], emphasizing the importance of establishing a standard training recipe.

Furthermore, to systematically analyze the bottlenecks and issues in image-based 3D object detection, we urgently require a tool to thoroughly examine detection results. Inspired by TIDE [2], we propose an error diagnosis toolbox, named TIDE3D, to measure the detailed characterization of
2. Related Work

Image-based 3D detection. Image-based 3D detection is a rapidly developing research direction, and there are lots of detectors proposed. Here we review these works based on the taxonomy proposed in [49]: methods based on result-lifting, methods based on feature-lifting, and methods based on data-lifting. Refer to [49] for the detailed and comprehensive literature review.

Result-lifting. The methods in this branch first estimate the 2D projections and other items (e.g. depth, orientation, etc.) of the 3D bounding boxes and then lift the results from the 2D image plane to the 3D world space. In particular, pioneering works [11, 10, 3, 95, 81, 32, 84] introduce the popular 2D detection paradigms, e.g. faster R-CNN [64] and FCOS [77], into this research field, and lots of following works improve these baseline models in several aspects, including backbone designs [3, 29], loss functions [70, 51, 22, 52], depth estimation [46, 93, 71, 80, 35, 37, 56], geometric constraints [60, 41, 39, 8, 44, 28, 62, 31, 61], feature embedding [42, 94], key-point constrains [7, 33, 45], depth augmented learning [87, 18, 15, 23, 14, 78], temporal sequences [4, 79], semi-supervised learning [54, 48, 40, 26], NMS [68, 30], data integration [5], etc.

Feature-lifting. OFT [65] and DSGN [13] are the pioneering works in this research line, which lift the 2D features into 3D features (generally represented by BEV map) and then directly estimates the 3D bounding boxes with the resultting features. CaDDN [63] uses a better transformation strategy [57] and achieves promising results. BEVDet [27] introduces this paradigm into the multi-camera setting, and this BEV pipeline gradually dominates the nuScenes [6] benchmark with effects of lots of follow-up works, such as [38, 36, 34].

Data-lifting. The data-lifting-based methods first estimate dense depth maps for the input images and then lift the pixels into 3D points using the estimated depth and camera parameters. After that, they generally leverage the LiDAR-based 3D detectors to predict the results from these ‘pseudo-LiDAR’ signals [83, 50]. The followers in this group are involved in the following aspects: improving the quality of pseudo-LiDAR [89, 20], focusing on foreground objects [82, 74, 88], end-to-ending training [59], feature representation [47, 53], geometric constraints [85] or confidence refinement [69].

Codebases. At present, there are two public codebases that

<table>
<thead>
<tr>
<th>backbone</th>
<th># epochs</th>
<th>KITTI data aug.</th>
<th>others</th>
<th>nuScenes data aug.</th>
<th>others</th>
</tr>
</thead>
<tbody>
<tr>
<td>GUPNet [46]</td>
<td>DLAS-DLAUp</td>
<td>140</td>
<td>flip, crop</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>FCOS3D [81]</td>
<td>ResNet101-DCN-FPN</td>
<td>-</td>
<td>-</td>
<td>12</td>
<td>flip</td>
</tr>
<tr>
<td>PGD [80]</td>
<td>ResNet101-DCN-FPN</td>
<td>48</td>
<td>flip</td>
<td>TTA</td>
<td>-</td>
</tr>
<tr>
<td>BEVDet [27]</td>
<td>ResNet50-FPN-LSS</td>
<td>-</td>
<td>-</td>
<td>24</td>
<td>flip, BEV aug.</td>
</tr>
</tbody>
</table>

Table 1: Overview of example methods on KITTI-3D and nuScenes benchmarks. The existing methods generally adopt different settings, e.g. backbones, epochs, augmentations, etc., in model training. Besides, KITTI-3D and nuScenes apply different evaluation metrics and are dominated by different detection pipelines. TTA denotes the test-time augmentation.
support vision-based 3D object detection: MMDetection3D [16] and OpenPCDet [76]. However, these two codebases are initially designed for LiDAR-based 3D object detection, and the supported vision-based methods are limited. In particular, MMDetection3D supports four vision-based detectors [66, 44, 81, 80], and OpenPCDet only support CaDDN [63] (also note some methods are developed based on these two codebases although they are not included in the official ones). In contrast, our codebase is designed for vision-based methods, excluding redundant codes and dependencies for LiDAR-based methods.

Error diagnosis for object detection. To obtain the detailed characterization under the overall evaluation metric and show the strengths and weaknesses of the given models, some works [25, 1, 2] exist to analyze the errors in 2D detectors. In particular, [25] divides the errors of false positives into several categories and selects the top N most confident detections for errors analysis. [1] further adds statistics for false negatives. More specifically, it progressively replaces the predicted items with corresponding ground-truth values and uses the ΔAP metric to quantify the importance of each item. [2] points out that this iterative approach incorrectly amplifies the impact of later error types, thus proposing a strategy where each error type is fixed independently. These works contribute to model analysis in the field of 2D detection, and to our best knowledge, there is no previous attempt to provide such a generic toolbox for 3D object detection. In this work, we provide a costumed error diagnosis toolbox to fill this gap.

3. Codebase

The objective of this work is to build a fair platform and provide experimental analysis for image-based 3D detection algorithms. For this purpose, we first build the general codebase which can support this work and facilitate future research. Specifically, we decompose the image-based 3D detection frameworks into different components, e.g. backbones, necks, heads, etc., and we provide common choices for each component, e.g. ResNet [24], DLA [90], or DCN [96] for the backbones. In addition to the CNN modules, we also provide the implementation of other parts, such as augmentations, post-processing, etc. Currently, the proposed codebase supports more than ten methods such as [51, 46, 91, 43, 44, 95, 81, 48, 54, 27, 80, 15] and two commonly used datasets [21, 6]. Our codebase will be publicly available and continuously maintained.

Comparison with official implementation. Here we provide a comparison of our implementation and the official implementation for some representative works. In particular, here we take MonoDLE and GUPNet as examples for KITTI-3D dataset, and FCOS3D as example for nuScenes dataset. Based on the results shown in Table 2, we can find our implementation gets consistent or better results than the official numbers. More details and the results for other models can be found in our public codebase.

<table>
<thead>
<tr>
<th></th>
<th>KITTI</th>
<th>nuScenes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Easy</td>
<td>Mod.</td>
</tr>
<tr>
<td>MonoDLE [51]</td>
<td>17.89</td>
<td>13.87</td>
</tr>
<tr>
<td>MonoDLE - ours</td>
<td>18.37</td>
<td>13.79</td>
</tr>
<tr>
<td>GUPNet [46]</td>
<td>21.48</td>
<td>15.22</td>
</tr>
<tr>
<td>GUPNet - ours</td>
<td>21.66</td>
<td>15.45</td>
</tr>
<tr>
<td>FCOS3D [81]</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>FCOS3D - ours</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 2: Comparison of the implementations on KITTI and nuScenes validation sets. We report the average performance of the last epoch for both the official codes and our implementations over five runs.

4. Diagnosis Toolkit

We propose a general toolkit, named TIDE3D, to diagnose the cause of errors for 3D object detection models. We first briefly review the computing process of mean Average Precision (mAP) [19] and then introduce how to divide the detection errors into fine-grained types. Finally, we present how to weigh these errors and the implementation details. We also recommend readers refer to TIDE [2] for preliminary knowledge.

Review of mAP. As the most commonly used metric for evaluating object detection methods, mAP provides a comprehensive overview of a detector’s performance. Given the predictions and corresponding ground truths, each ground truth is matched to at most one prediction according to a specified metric, e.g. KITTI-3D uses 3D Intersection over Union (IoU) and nuScenes adopts center distance as the metric. If multiple predictions meet the constraint, the ground truth only matches the prediction which has the maximum confidence score. The matched predictions are true positives (TP), and the remaining ones are false positives (FP). After sorting the predictions by descending confidence, the number of true positives and false positives in the subset of predictions with confidence scores greater than c is counted as N_{TP} and N_{FP}. Then, the precision and recall of the predictions subset are obtained by:

\[
P_c = \frac{N_{TP}}{N_{TP} + N_{FP}} \quad R_c = \frac{N_{TP}}{N_{GT}}
\]

where N_{GT} is the number of ground-truth objects. As the confidence threshold c changes, the precision-recall curve is plotted and the Average Precision (AP) is obtained by calculating the area under the curve. Finally, mAP is obtained by averaging the AP values of all categories.

Error definitions. Due to the complexity of calculating
mAP, it prevents researchers from further analyzing the performance, and two detectors with the same mAP value may have different strengths and weaknesses. To address this issue, we segment the detection errors into several separate groups and measure the contribution of each error type. Specifically, based on the taxonomy of 2D detection error proposed in TIDE [2] (including classification error, localization error, both classification and localization error, duplication error, background error, and missing error. See Table 3 for details), we further propose two modifications for 3D object detection, including the sub-error types of localization and the ranking error.

Sub-error types of localization. Since the localization of the 3D bounding box is determined by many factors, we further divide the localization error into three sub-errors to explore the effect of predicted location, dimension, and orientation on inaccurate localization.

Ranking error. To describe each object in the scene, the detection model needs to output the 3D bounding box and its confidence. However, some detections may have higher confidence but lower accuracy. In the process of calculating AP, the detected boxes are sorted by descending confidence and the precision of the high-confidence detections is calculated first. Therefore, the misalignment between confidence and the quality of the box naturally leads to the decline of AP and brings about possible room for improvement.

Weighting the errors. We use \( \Delta AP \) to quantify the impact of each error type. In particular, similar to TIDE [2], we independently fix the errors (called “oracle”, see Table 3 for the details), and compute the change of AP to measure the impact of this error type by:

\[
\Delta AP_o = AP_o - AP,
\]

where \( AP_o \) is the AP after applying the oracle \( o \). In this way, we can capture the effect of each error type on the final metric, and a lower \( \Delta AP \) of the given error type indicates the model performs better at this part.

Due to current datasets generally adopting different evaluation metrics, the detailed implementations of TIDE3D are also slightly different for these datasets [21, 6, 17]. More details of the specific implementations are provided in Appendix A.3.

5. Approach

We first discuss the effects of common training settings on the final accuracy of current methods and then provide efficient and fair training recipes for further work. Furthermore, we also show the state-of-the-art (SOTA) methods can be further promoted in performance with the summarized training recipes and some existing techniques in the KITTI-3D benchmark. Besides, the detailed features of the detection algorithms are evaluated with the proposed TIDE3D, and we also show our diagnosis tool can be used to explore some open issues. Finally, we discuss the conflicts between KITTI-3D and nuScenes datasets.

Training recipes. We observe that current methods generally adopt different training recipes, e.g., the training schedule in KITTI-3D ranges from 60 epochs [44] to 200 epochs [55]. We show that training differences like this greatly affect the performance of the algorithms and may mislead the latecomers to some extent, and it is necessary to formulate standard training recipes. Specifically, in KITTI-3D, we take the 70-epoch schedule as baseline \( (1 \times) \) and also evaluate the models under \( 2 \times \) and \( 3 \times \) schedules. Similarly, the corresponding settings in nuScenes are 12, 24, and 36 epochs. We recommend the \( 2 \times \) schedule because this is the closest one to the common choice of other algorithms, and use other schedulers for further evaluation. Furthermore, we also ablate other choices in model training and provide a general training recipe for image-based 3D detectors. We report, in addition to some basic choices such as random crop or horizontal flip, the photometric distortion and one-cycle learning rate [73] are also highly effective for our task and should be included in the conventional setting.

Promoted baselines in KITTI-3D. We also find, although KITTI-3D is released for a longer time and more methods are proposed on KITTI-3D benchmark, the baseline models in KITTI-3D are still under-tuned compared to these on nuScenes. Here we show current baseline models can be greatly promoted. We report that a two-year-ago model [46] (re-trained with pre-trained backbone, pseudo-labels, and the summarized training recipe) can achieve 17.65 AP\(_{40}\) on the KITTI-3D validation set, surpassing the SOTA by 1.11 points. Even so, we believe this number can be improved with further tuning or other tricks (e.g. test-time augmentation and model-ensemble [81]).

Dataset and metrics. KITTI-3D [21] and nuScenes [6] are the most concerned datasets in the image-based 3D detection field, however, they are dominated by different detection pipelines. We find that they are different in many aspects, and nuScene is generally considered to be more friendly to image-based algorithms. We adopt nuScenes-style evaluation on the KITTI-3D (and vice versa) and provide our discussion based on the cross-metric evaluation. More details are given in Section 6.4.

Detailed analysis with TIDE3D. With the TIDE3D introduced in Section 4, we can: (i) Analyze the detailed characteristics of 3D object detection models, whether it is an image-based model or LiDAR-based model, evaluated with KITTI-3D-style or nuScene-style metrics, and designed for outdoor or indoor scenes (see Appendix A.5 for the analyses for indoor dataset [17] and LiDAR-based models). (ii) Conduct solid ablation studies and explore the action mechanisms for specific designs/modules.
<table>
<thead>
<tr>
<th>Error Type</th>
<th>Definition</th>
<th>Oracle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classification</td>
<td>Incorrect classification and correct localization.</td>
<td>Correct the category classification / location of detection, and delete this detection if it becomes duplicated.</td>
</tr>
<tr>
<td>Localization</td>
<td>Correct classification and incorrect localization.</td>
<td>Delete the inaccurate / duplicate detection.</td>
</tr>
<tr>
<td>BothCls. and Loc.</td>
<td>Incorrect classification and incorrect localization.</td>
<td>Delete the missed ground truth.</td>
</tr>
<tr>
<td>Duplication</td>
<td>Correct classification and correct localization, but the corresponding GT has been matched by another higher-scoring detection.</td>
<td>Sort the detections by descending IoU score and calculate the precision of accurate detections first.</td>
</tr>
<tr>
<td>Background</td>
<td>Detection from the background area.</td>
<td></td>
</tr>
<tr>
<td>Missing</td>
<td>Undetected GT not covered by classification and localization error.</td>
<td></td>
</tr>
<tr>
<td>Ranking</td>
<td>Inconsistency between confidence and localization quality of detections.</td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Definition and oracle of detection errors.

6. Experiments and Analysis

**Dataset and metrics.** We conduct experiments on KITTI-3D and nuScenes datasets, and the detailed introduction for these two datasets is provided in Appendix A.1. Due to access restrictions of testing servers, the evaluation is conducted on validation set, and we further apply cross-dataset metrics for better evaluation, e.g. applying nuScenes-style metrics on the KITTI-3D dataset.

**Baseline models.** In this work, we mainly focus on the baselines based on the popular detection pipeline, including the CenterNet pipeline [46, 29], the BEV pipeline [27], and the FCOS3D pipeline [81, 80].

**Other settings.** To replicate the reported performance and minimize the impact of complex architectures and additional hyper-parameters, we adopt the simplest default configurations provided in the official code for most methods. Specifically, we use basic training schedules, simple architectures, and standard data augmentation techniques as the default settings. Further details beyond to these settings are provided in according sections.

6.1. Training Recipes

**Training schedules.** We first present the effects of training schedules. Specifically, we experiment with three different schedules, ranging from 1× to 3×, for both KITTI-3D and nuScenes. Notably, we plot the training curves of all three schedules in the same figure to facilitate comparison. The AP curves shown in Figure 1 indicate that schedules have a significant impact on the final performance, highlighting the importance of a standard scheduler for a fair comparison. Furthermore, the following observations can be summarized from these curves:

1. Different detection models require varying numbers of epochs as the optimal settings. For example, BEVDet exhibits significant and smooth performance improvements as the number of training epochs increases, while FCOS3D achieves near-optimal performance faster with perturbation after learning rate decay. This discrepancy may stem from the difference in optimizers, iterations, and detection pipelines. Intuitively, required training time can substantially vary across different settings, which may reduce the

![Figure 1: Learning curves on nuScenes (Left) and KITTI-3D (Right) with different detection models. With the 3× scheduler, BEVDet with R50-FPN optimized by AdamW and FCOS3D with R101-DCN-FPN optimized by SGD achieve 28.56 and 31.67 mAP on nuScenes, respectively, while GUPNet with DLA-34 optimized by Adam achieves 15.64 AP_{40}@IoU=0.7 on KITTI-3D (validation set). For better presentation, we present the curves by moving the average with windows size = 5. The original curves are shown in Appendix A.4.](image)
and can work together. Although nuScenes is a large-scale 3D dataset, applying photometric distortion still brings a modest mAP improvement. Besides, we find the effects of one-cycle learning rate on nuScenes baselines are unstable, so we omit the corresponding cells to avoid misleading.

Table 5: The photometric distortion and one-cycle learning rate are also effective training choices.

<table>
<thead>
<tr>
<th>schedule</th>
<th>pre-training</th>
<th>AP$_{40}$@IoU=0.7</th>
<th>AP$_{40}$@IoU=0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>GUPNet [46]</td>
<td>2× ImageNet</td>
<td>21.48 15.22 12.79</td>
<td>59.30 43.83 38.40</td>
</tr>
<tr>
<td>DID-M3D [55]</td>
<td>3× ImageNet</td>
<td>22.98 16.12 14.03</td>
<td>- - -</td>
</tr>
<tr>
<td>DEVIAN [29]</td>
<td>2× ImageNet</td>
<td>24.63 16.54 14.52</td>
<td>61.00 46.00 40.18</td>
</tr>
<tr>
<td>GUPNet</td>
<td>2× None</td>
<td>18.74 13.40 10.62</td>
<td>53.80 40.69 35.73</td>
</tr>
<tr>
<td>GUPNet</td>
<td>2× ImageNet</td>
<td>21.51 15.82 13.30</td>
<td>60.18 44.35 39.27</td>
</tr>
<tr>
<td>GUPNet</td>
<td>2× DD3D</td>
<td>24.25 15.82 13.26</td>
<td>63.16 44.60 40.08</td>
</tr>
<tr>
<td>GUPNet + PL</td>
<td>2× DD3D</td>
<td>24.71 17.25 15.21</td>
<td>63.73 46.56 42.45</td>
</tr>
</tbody>
</table>

Table 4: Promoted baselines on KITTI-3D validation set. We report the 3D object detection performance of GUPNet with various pretrained backbones, evaluated by AP$_{40}$@IoU=0.7/0.5. We also present SOTA methods for reference. Note that both DID-M3D and DEVIAN are built based on the GUPNet baseline, and our modifications are only involved in the training phase, which shows the importance of building fair training recipes again.

- **i.** Increasing the training procedure enhances the stability and precision of the fine-tuned model. Even after completing the 3× training, the trend of mAP/AP$_{40}$ improvement continues to exist. The commonly accepted training schedule (mostly 1× or 2× on nuScenes) may be insufficient to unlock the full potential of a model. Therefore, the standard training procedure remains further discussed.

- **ii.** We observe a significant drop and a recovery at the beginning of the finetuning stage of 1× training in the experiments of BEVDet and GUPNet. With training time increasing from 2× to 3×, this phenomenon alleviates and even disappears. We suppose that longer training time helps the network learn more robust features, which prevents the overfitting brought by the sudden decay of the learning rate. More training steps at a small learning rate also help counter this issue.

- **iv.** Even when the training is about to be completed, we still observe performances are unstable (see Appendix A.4 for the un-smoothed curves). This phenomenon is particularly evident for the KITTI-3D-style metrics. Based on this, the comparison based on KITTI-3D metrics should be carefully conducted.

### Other training recipes

- Due to strict geometric relations between the 2D image plane and the 3D world space, some complicated data augmentations are hard to be applied, and the horizontal flip is the common choice for existing methods. Note some simple geometric augmentations, like center crop and resize, are also commonly used in the KITTI-3D dataset, and see Appendix A.6 for more discussions on such data augmentations. Here we show the photometric distortion and the one-cycle learning rate schedule are two cost-free training tricks to improve the detection accuracy (especially in KITTI-3D) and show their effectiveness with two representative baselines in Table 5. We can find that both of them improve the GUPNet by a significant margin and can work together. Although nuScenes is a large-scale dataset, applying photometric distortion still brings a modest mAP improvement. Besides, we find the effects of one-cycle learning rate on nuScenes baselines are unstable, so we omit the corresponding cells to avoid misleading.

<table>
<thead>
<tr>
<th>baseline</th>
<th>w/ distortion</th>
<th>w/ one-cycle lr</th>
<th>full</th>
</tr>
</thead>
<tbody>
<tr>
<td>GUPNet [AP$_{40}$]</td>
<td>15.26 16.54 15.87 16.83</td>
<td>- -</td>
<td>- -</td>
</tr>
<tr>
<td>BEVDet [mAP]</td>
<td>25.12 25.20 - -</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6: Information of different pre-training models. Ep. denotes the pre-training epochs.

### Leveraging pre-training weights

Leveraging pre-training weights. We show the impact of weight initialization on detection models in Table 4. In particular, we compare three initial weights, including random initialization, ImageNet pretraining, and DD3D pertaining [53]. We can find that: i. Similar to other computer vision tasks, model pre-training provides useful priors to the models and improve the performance significantly. ii. As the data domains and proxy tasks become more similar (see Table 6), the pre-training weights also become more effective. Based on the above observations, designing custom pre-training algorithms, especially in an unsupervised manner, should be a promising research direction for future work.

<table>
<thead>
<tr>
<th>domain</th>
<th>proxy task</th>
<th># img. × # ep.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ImageNet</td>
<td>daily scenes</td>
<td>1.3M × 100</td>
</tr>
<tr>
<td>DD3D</td>
<td>driving scenes</td>
<td>15M × 12.8</td>
</tr>
</tbody>
</table>

Table 6: Information of different pre-training models. Ep. denotes the pre-training epochs.
6.3. TIDE3D Analysis

**Error diagnosis.** Here we present the error diagnosis results of some representative methods, settings, and evaluation metrics in Figure 2. According to these error distributions, we can get the following observations:

i. Similar to [51, 29, 53, 80], we find localization error is the bottleneck of current methods and it is mainly caused by the inaccurate location (center). Even for the LiDAR-based method [67] which estimates the results from the data with accurate spatial information, localization error still ranks first among all error types. Differently, for [67], the dimension and the orientation also have a significant impact on the quality of localization.

ii. We find the misalignment between the confidence and the bounding box’s quality is a serious problem in this field, suggesting that giving more accurate confidence is an effective way to improve detection accuracy. An empirical study of this problem is given in the following part. iii. Under a lower IoU threshold, the KITTI-3D-style metric is still dominated by the localization error. Meanwhile, this issue has been alleviated on nuScenes-style metric. This indicates 3D IoU-based AP is a very sensitive metric, and center distance-based AP is more friendly to image-based methods. More TIDE3D results and analyses, e.g. indoor scenes, are provided in Appendix A.5.

**Validating design choices.** We show the proposed TIDE3D can validate whether a given design choice supports its motivation/claim. For instance, some works, such as [46, 69, 5] proposed their designs to get better confidence scores of detection results. Here we use TIDE3D to validate the design of GUPNet. Particularly, GUPNet models the uncertainty of estimated depth ($\sigma$) and further gets the confidence of depth with this uncertainty by:

$$p_{depth} = e^{|-\sigma|}.$$  

Finally, the 3D confidence ($p_{3D}$) can be obtained by correcting 2D confidence with depth confidence:

$$p_{3D} = p_{depth} \cdot p_{2D}.$$  

Table 7 gives the results of GUPNet equipped with 2D confidence and 3D confidence respectively. We observe that the AP$_{40}$ improved by 2.21 and the ranking error reduces to 10.81 from 15.10, which confirms the effectiveness of this design. Meanwhile, we find the localization error also reduced, which is caused by the high-quality results being improved and corresponding results having a higher priority in AP computing. See Appendix A.7 for more analysis about the ranking error and localization error in TIDE3D.

6.4. Metric Analysis for KITTI-3D and nuScenes

Additionally, we have observed interesting discrepancies in evaluation metrics between KITTI-3D and nuScenes. Specifically, as shown in Table 8, GUPNet and FCOS3D get similar AP of Car category under the nuScenes-style metric.
<table>
<thead>
<tr>
<th>Dataset</th>
<th>Class</th>
<th>KITTI Metric @0.7</th>
<th>KITTI Metric @0.5</th>
<th>KITTI Metric @0.25</th>
<th>nuScenes Metric mAP</th>
<th>nuScenes Metric NDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>GUPNet</td>
<td>KITTI-3D Car</td>
<td>14.85</td>
<td>40.85</td>
<td>62.87</td>
<td>47.57</td>
<td>61.85</td>
</tr>
<tr>
<td>PV-RCNN</td>
<td>KITTI-3D Car</td>
<td>82.41</td>
<td>94.31</td>
<td>94.47</td>
<td>92.74</td>
<td>92.61</td>
</tr>
<tr>
<td>FCOS3D</td>
<td>nuScenes Car</td>
<td>1.83</td>
<td>20.87</td>
<td>50.74</td>
<td>47.96</td>
<td>54.32</td>
</tr>
<tr>
<td>BEVDet</td>
<td>nuScenes All</td>
<td>0.26</td>
<td>6.06</td>
<td>17.57</td>
<td>31.05</td>
<td>40.31</td>
</tr>
<tr>
<td></td>
<td>nuScenes All</td>
<td>0.10</td>
<td>6.55</td>
<td>19.25</td>
<td>30.75</td>
<td>38.22</td>
</tr>
</tbody>
</table>

Table 8: Evaluation Metric. We exchange the metrics between KITTI-3D and nuScenes for the evaluation of 3D object detectors. This provides a standardized way to compare the performance of 3D object detectors across the two datasets. Dataset represents the source of training data for the detector. For detectors trained on KITTI-3D, we only compute translation, scale, and orientation for NDS under nuScenes metric. For KITTI metric, we evaluate the results of cars under the hard strict with varying IoU thresholds.

Figure 3: TIDE3D analyses of BEVDet [27] and FCOS3D [81] for all categories on nuScenes validation set based on the KITTI-style metric with 0.25 IoU threshold. Our results indicate that FCOS3D is impacted by various errors, while BEVDet is mainly dominated by localization errors.

7. Conclusion

In this paper, we aim to standardize the evaluation protocols for image-based 3D object detection, an ever-evolving research field. We build a modular-designed codebase and provide efficient training recipes for this task, leading to significant improvements in the performance of current methods. Besides, we offer an error diagnosis toolbox to measure the detailed characterization of detection algorithms and highlight some open problems in the field. We hope our codebase, training recipes, error diagnosis toolbox, and discussions will foster better and more standardized research practices within the image-based 3D object detection community.

Acknowledgments

This work was supported by the Australian Medical Research Future Fund MRFAI000085, CRC-P Smart Material Recovery Facility (SMRF) – Curby Soft Plastics, CRC-P ARIA - Bionic Visual-Spatial Prosthesis for the Blind, and National Natural Science Foundation of China (NSFC Grants NO. 61976038).
References


[35] Yingyan Li, Yuntao Chen, Jiawei He, and Zhaoxiang Zhang. Densely constrained depth estimator for monocular 3d object detection. In *ECCV*, 2022. 2


[46] Yan Lu, Xinzhu Ma, Lei Yang, Tianzhu Zhang, Yating Liu, Qi Chu, Junjie Yan, and Wanli Ouyang. Geometry uncertainty projection network for monocular 3d object detection. In *ICCV*, 2019. 1, 2, 3, 4, 5, 6, 7, 14


[51] Xinzhu Ma, Yinmin Zhang, Dan Xu, Dongzhan Zhou, Shuai Yi, Haojie Li, and Wanli Ouyang. Delving into localization errors for monocular 3d object detection. In *CVPR*, 2021. 1, 2, 3, 7, 14


[54] Liang Peng, Fei Liu, Zhengyu Yu, Senbo Yan, Dan Deng, Zheng Yang, Haifeng Liu, and Deng Cai. Lidar point cloud guided monocular 3d object detection. In *ECCV*, 2022. 2, 3, 6


[60] Zequn Qin and Xi Li. Monoground: Detecting monocular 3d objects from the ground. In *CVPR*, 2022. 2


[70] Andrea Simonelli, Samuel Rota Bulò, Lorenzo Porzi, Manuel López-Antequera, and Peter Kontschieder. Disen-
Shilei Wen, Errui Ding, Ajin Meng, and Liusheng Huang.

Object detection from monocular images. In com/facebookresearch/detectron2


Zhenbo Xu, Wei Zhang, Xiaqing Ye, Xiao Tan, Wei Yang, Shilei Wen, Errui Ding, Ajin Meng, and Liusheng Huang.


