The Dissertation Committee for Xingyi Zhou
certifies that this is the approved version of the following dissertation:

Towards unified object recognition in the wild

Committee:

Dr. Philipp Krähenbühl, Supervisor

Dr. Raymond Mooney

Dr. Yuke Zhu

Dr. Dava Ramanan

Towards unified object recognition in the wild

by

Xingyi Zhou

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

May 2022

Dedicated to my parents, Zhou and Jiang.

Acknowledgments

First and foremost, I would like to thank my advisor Philipp Krähenbühl, for his advice and enormous support on both my research and career.

I would also like to thank my first supervisor and initial committee member Qixing Huang, for his support and guidance in my research.

I thank my committee members Raymond Mooney, Yuke Zhu, Deva Ramanan, for their feedback and strengthening of my thesis.

I thank my mentors during my internships: Vladlen Koltun, Ishan Misra, Rohit Girdhar, Tyler Zhu, for their guidance and inspiration.

I thank my labmates, intern buddies, and friends: Chao-yuan Wu, Dian Chen, Brady Zhou, Yue Zhao, Jeffery Zhang, Vicent Cho, Tianwei Yin, Tianjian Meng, Jierui Lin, Xun Huang, Raymond Yeh, Hexiang Hu, Zhuang Liu, John Lambert, Wei Dong, Kai Zhang, Bowen Cheng, Jiacheng Xu, Jiacheng Zhuo, for their accompany, support, and encouragement.

I thank UT Provost's Graduate Excellence Fellowship and Facebook PhD fellowship for their financial support and recognition of my research.

I thank my lovely girlfriend, Jiarui Gao, for her love, support, and sharing time every day. Finally, I am deeply grateful to my parents, Yunlin Zhou and Ahong Jiang, for their unconditional love, support, and understanding.

Towards unified object recognition in the wild

Publication No. _____

Xingyi Zhou, Ph.D.
The University of Texas at Austin, 2022

Supervisor: Dr. Philipp Krähenbühl

Large-scale well-curated datasets are the fuel of computer vision. However, most datasets only focus on one single domain with a specific task and a fixed label set. Computer vision models trained on a single dataset only apply to a subset of applications in the real world.

The goal of my research is to remove the artificial barriers of datasets and make object recognition generalize in the wild. There should be one single computer vision model, not a zoo of dataset-specific models. The model should be trained on a diverse set of datasets and should be able to recognize objects from different data sources in all domains.

Towards this goal, my thesis focuses on three aspects: a point-based **object representation** that unified multiple vision tasks, a unified framework that detects and **tracks objects through time**, and a **unified vocabulary** between detection and classification annotations.

First, we propose to represent an object using the simplest-possible representation — a point. All object properties, like its size, pose, depth, and velocities, are attributes of the points and are inferred from the point features. We developed a point-based object detector, CenterNet, using standard keypoint detection techniques. We extend this point-based detector to many vision tasks by just adding task-specific regression outputs. The point-based representation achieves the state-of-the-art-level performance and runs fast with a unified framework.

Second, we show the point-based representation also simplifies linking objects through time. We extend our point-based detector into a local tracker by regression the inter-frame motion of each object. The resulting point-based tracker is efficient, accurate, robust, and unified under different domains, tasks, and framerates. Going further, we develop a tracker that associates and classifies objects from the whole video clip. The global association uses a transformer that looks at all objects in a long temporal window, and directly produces trajectories.

Finally, we study how to extend the vocabulary of our recognition system. We explored two directions: 1) merging multiple object detection datasets in different vocabularies and domains with an automatic label-space unification algorithm; 2) introducing additional classification annotations with a much large vocabulary, i.e., twenty-thousand classes. The resulting unified detector has a broad vocabulary, is more robust to changes in the visual domain, and generalizes readily to new unseen environments and taxonomies.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction and Overview

Detecting and recognizing objects is the fundamental task in computer vision. In the past decade, we have witnessed great progress in object recognition. For example, object detection on COCO [109] went from 21.9 mAP [153] to 62.4 mAP [235]; Human pose estimation achieved more than 90 percent accuracy [1] and already yields successful real-world applications [15]; 3D detection on Waymo Open Dataset [178] reached human-level performance [143] within one year of its release. It seems that a deep network can solve any task if there is enough training data. Are we done? Can we just move to annotating more data for computer vision? Maybe, but collecting and maintaining a dataset of the size of our visual world with annotations to any task anybody will ever care about is simply too expensive.

Currently, most vision systems only solve one single task on one single domain in isolation. 3D bounding box detection only applies to traffic or indoor environments [13, 55, 178]. Keypoint estimation only reasons about human [1] or rigid objects [21]. An object detector trained on COCO [109] can only detect the 80 annotated classes. While researchers carefully curated datasets that mimic all aspects of human perception, they all exist in different domains,

data sources, and annotation instructions. Can we apply 3D bounding box detection to any object categories? How can a detector understand that dogs in one dataset, are labeled as animals in another, or hundreds of fine-grained categories in the next? How can it combine annotations for the same object from different datasets? There is no single data source to tie all information together. Even if there was a single data source, it would be outdated as soon as new tasks, imaging modalities, or visual representations emerge.

These problems are even more pronounced as computer vision moves from still images to videos [19, 32]. There is less annotated video data, fewer tasks, and a much higher dimensional input space to the vision system. How can a system learn to reason about tasks across time and space, while the majority of annotations and supervision exist just in still images? How can a unified representation jointly reason about images and videos, and selectively use the temporal signal?

The goal of my research is to remove the artificial barriers of datasets and make computer vision work in the wild. There should be one single *computer vision* model, not a zoo of dataset-specific models. One single model should infer all properties of all objects over a wide range of testing domains. For example, we want a system to learn about object categorization from one dataset (e.g. MSCOCO [109]), 3D layout from another (e.g. nuScenes [13] or SUN-RGBD [175]), and human pose from a third (e.g. CMU Panoptic [74]), and so on (Figure 8.1). The model should be trained on a diverse set of datasets and should not be limited to a single dataset or a single domain.

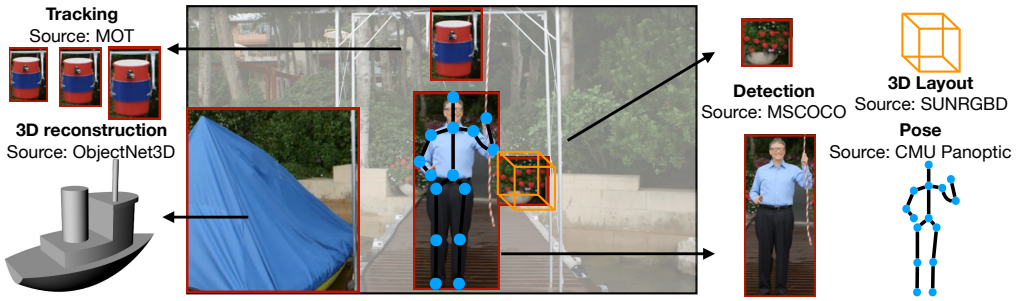


Figure 1.1: We want a unified object recognition system that infers all properties of all objects in a scene. The training data for this system comes from numerous different sources (datasets). Combining tasks and data sources into a unified model is one of the main challenges for the proposed research.

Towards this goal, my research focuses on three parts: a unified point-based objects representation (Chapter 4: CenterNet), a unified framework that detects and tracks object through time (Chapter 5: CenterTrack, Chapter 6: GTR), and a unified vocabulary betweem detection and classification annotations (Chapter 7: UniDet, Chapter 8: Detic).

First, we studied the basic representation of an object. We propose CenterNet [258] that detects a single center point using standard keypoint detection. This single point serves as a representation of an object. All other object properties, e.g. size of the bounding box, pose, 3D orientation and location, and velocity can be considered attributes of that point and can be learned efficiently with regression using the feature in the center. In our experiments, we learn the bounding box width and height for object detection, learn the center-out offsets to each human joint for human pose estimation, and learn the depth, object orientation, and 3D size for 3D bounding box de-

tection. It achieved comparable results with sophisticated multi-stage methods in all the tasks with the point-based representation, while being fast, simple, and unified. Chapter 4 discusses more details, with extensions to two-stage detectors [256] and Lidar-based 3D detection [231].

Second, we show the point-based representation also simplifies linking objects through time. In Chapter 5, we study a simple solution to extend our point-based detector into a tracker in a short temporal window, by regression the inter-frame motion of each object. We show our point-based representation significantly simplifies tracking in two aspects. First, it simplifies tracking-conditioned detection. If each object in past frames is represented by a single point, a constellation of objects can be represented by a heatmap of points. This serves as the state information of history tracks, and can be fed into the detector directly to condition the detector on the previous tracking results. Second, point-based tracking simplifies object association across time. A simple displacement prediction, akin to sparse optical flow, allows objects in different frames to be linked. The displacement prediction is yet another attribute of the center point, and is learned end-to-end with the detector by regression. Results on 3 tracking benchmarks show our tracker performs both faster and more accurate than traditional multi-stage trackers. More details and results are discussed in Chapter 5. This work was published in ECCV 2020 [255].

CenterTrack models videos as pairs of two consecutive frames. While this is effective for short-term tracking, it often fails challenging cases like

long-term occlusion due to the lack of global information. In our recent work, Global Tracking Transformers (GTR) [259], we propose to use a transformer architecture to perform object tracking in a long temporal window. Transformers [17, 38, 200] have recently been shown to be extremely efficient on various vision tasks, and are good candidates for unified object recognition in both images and videos. We follow the two-stage detector [256] setup to produce object proposals individually for each frame, and use the transformer to process all proposals from all frames. The transformer takes all-frames proposals and trajectory queries as input, and produces the global associations among the objects for each query. Object classification is done by considering all object features over the tube, instead of a single frame. We show global tracking framework significantly improves the local pair-wise counterpart, and the tracking accuracy increases consistently when using more frames as the global context. This work was published in CVPR 2022 [260] and achieves the state-of-the-art performance on the challenge TAO [32] dataset.

Like any other detector and tracker, our works above are trained on their corresponding single dataset for one task. This is an issue if we ever want our detection system to generalize to the real world, since there is no dataset of the entire real world. In our project UniDet (Chapter 7), we study how to train an object detector on multiple datasets. There are two challenges. First, different datasets usually require different training recipes, and directly training them together yields sub-optimal results [91, 207]. Second, different datasets annotate objects in their own annotation protocol, and it

is unclear how the labels in one dataset correspond to another. We propose a multi-dataset training framework and show training on multiple datasets is as effective as training multiple models each on individual datasets. We also learn to link up different vocabularies by leveraging inter-categories relations from their visual similarity, rather than relying on the semantics of the class name. We formulate the inter-dataset class relation as a combinatorial problem, and solve it efficiently using linear programming. Our framework outperforms the best human experts in linking up categories, while being fully automatic. This work was published in CVPR 2022 [257] and won the ECCV 2020 Robust Vision Challenge.

UniDet expands object detection vocabulary from a few hundred to more than a thousand. However, detection datasets are inherently smaller than classification datasets in both the overall size and vocabularies, due to the difficulties in annotations. Image classification, on the other hand, reasons about much larger vocabulary in wider domains. To further expand detection vocabulary even to infinite, we propose *Detic*, which simply trains the classifiers of a detector on image classification data and thus expands the vocabulary of detectors to tens of thousands of concepts. Unlike prior work, Detic does not need complex assignment schemes to assign image labels to boxes based on model predictions, making it much easier to implement and compatible with a range of detection architectures and backbones. For the first time, we train a detector with all the twenty-one-thousand classes of the ImageNet dataset and show that it generalizes to new datasets without finetuning.

In summary, to achieve the goal of a unified computer vision model for all object recognition tasks, this proposal introduces our works in three directions: 1): we propose a point-based representation for all objects and design a unified point-based detector (Chapter 4) for multiple object recognition tasks; 2) we propose a unified still-image detector and tracking framework (Chapter 5), and extend it into long-term tracking (Chapter 6). 3) we propose an automatic way to unify the label spaces of multiple different detection datasets (Chapter 7), and further combine detection datasets and classification datasets for open-vocabulary object detection.

In the following chapters, We first review related works on unified object recognition in Chapter 2. Then we introduce common notations and existing technics will use in Chapter 3. From Chapter 4 to Chapter 8, we present the methods and results outlined above. Finally, we summarize my thesis and highlight future directions in Chapter 9.

Chapter 2

Related Work

In this chapter, We give a brief literature review of the related fields, including object detection(Section 2.1.1), keypoint detection (Section 2.1.3), open-vocabulary object recognition (Section 2.3), and object tracking (Section 2.2).

2.1 Object recognition

Recognizing objects in a natural scene is the core task of computer vision. The most fundamental object recognition task is arguably image classification [34, 65]. However image classification requires the input images to be object-centric, which is not the best fit for natural scene-centric images. In this section, we mostly focus on reviewing literature that takes a scene image as input. We start from object detection [109].

2.1.1 Object detection

Object detection requires localizing object bounding boxes and classifying the object inside the box. One of the first successful deep object detectors, RCNN [59], enumerates object location from a large set of region candidates [199], crops them, and classifies each using an image classification

network [87]. Fast-RCNN [58] crops image features instead, to save computation. Both methods rely on slow low-level region proposal methods [199, 271].

Object detection with implicit anchors. Faster RCNN [153] is arguably the first complete deep-learning-based object detection system. It generates region proposals within the detection network. It samples fixed-shape bounding boxes (anchors) around a low-resolution image grid and classifies each into "foreground or not". An anchor is labeled foreground with a $>$0.7 overlap with any ground truth object, background with a $<$0.3 overlap, or ignored otherwise. Each generated region proposal is again classified [58]. Changing the proposal classifier to a multi-class classification forms the basis of one-stage detectors. Several improvements to one-stage detectors include anchor shape priors [150, 151], different feature resolution [112], and loss re-weighting among different samples [108].

Our center-based object representation presented Chapter 4 is closely related to anchor-based one-stage approaches [108, 112, 149]. A center point can be seen as a single shape-agnostic anchor However, there are a few important differences. First, our CenterNet assigns the "anchor" based solely on location, not box overlap [58]. We have no manual thresholds [58] for foreground and background classification. Second, we only have one positive "anchor" per object, and hence do not need Non-Maximum Suppression (NMS) [11]. We simply extract local peaks in the keypoint heatmap [16, 130]. Third, CenterNet uses a larger output resolution (output stride of 4) compared to traditional

object detectors [63, 65] (output stride of 16). This partially eliminates the need for multiple anchors [169].

Keypoint-based object detection. CornerNet [93] detects two bounding box corners as keypoints, while ExtremeNet [261] detects the top-, left-, bottom-, right-most, and center points of all objects, and combines them using a geometric grouping approach. Duan et al. [39] combine the geometric grouping of ExtremeNet with corner estimation of CornerNet. All these methods build on the same robust keypoint estimation network as our CenterNet. However, they all require a combinatorial grouping stage after keypoint detection, which significantly slows down each algorithm.

2.1.2 3D detection

3D bounding box estimation powers autonomous driving [55]. Deep3Dbox [128] uses a slow-RCNN [59] style framework, by first detecting 2D objects [153] and then feeding each object into a 3D estimation network. 3D RCNN [88] adds an additional head to Faster-RCNN [153] followed by a 3D projection. Deep Manta [20] uses a coarse-to-fine Faster-RCNN [153] trained on many tasks. Recently, MonoDIS [168] trains a single stage RetinaNet [108] with a differentiable Huber loss on 3D bounding box corners. Our method is similar to a one-stage version of Deep3Dbox [128] or 3DRCNN [88]. As such, CenterNet (Chapter 4) is much simpler and faster than competing methods.

2.1.3 Keypoint estimation

Keypoint estimation, especially human joint estimation [16, 131, 192, 194, 253] and rigid object keypoint estimation [213, 252], is a widely studied problem in computer vision. In the simplest case, a 2D/3D keypoint can be represented by a 2/3-dimension vector and learned by supervised regression. Toshev et al. [194] first trained a deep neural network for 2D human pose regression and Li et al. [98] extended this approach to 3D. Starting from Tompson et al. [193], the heatmap representation has dominated the 2D keypoint estimation community and has achieved great success in both 2D human pose estimation [131, 208, 228] and *single category* man-made object keypoint detection [212, 213]. Recently, the heatmap representation has been generalized in various different directions. Cao et al. [16] and Newell et al. [130] extended the single peak heatmap (for single keypoint detection) to a multi-peak heatmap where each peak is one instance of a *specific type* of keypoint, enabling bottom-up, multi-person pose estimation. Pavlakos et al. [138] lifted the 2D pixel heatmap to a 3D voxel heatmap, resulting in an end-to-end 3D human pose estimation system. Tulsiani et al. [197] and Pavlakos et al. [137] stacked keypoint heatmaps from different object categories together for multi-category object keypoint estimation. Despite good performance gained by these approaches, they share a common limitation: each heatmap is only trained for a *specific keypoint type* from a specific object.

All these works study semantic keypoints, where the keypoints are predefined and usually share common appearances. In Chapter 4 we show key-

point estimation also works for non-semantic keypoints, i.e., the object center point. This extends the concepts of keypoint detection and we will show how this helps object detection.

Heatmap-conditioned keypoint estimation. Keypoints are not only a good output modality, but also can be a format of input [18, 51, 127]. Auto-context [196] feeds the mask prediction back into the network. Iterative-Error-Feedback (IEF) [18] takes another step by rendering predicted keypoint coordinates into heatmaps. PoseFix [127] generates heatmaps that simulate test errors for human pose refinement.

In Chapter 5, we show this inspires using keypoints for local tracking. A rendered heatmap of prior keypoints [18, 51, 127, 196] is especially appealing in tracking for two reasons. First, the information in the previous frame is freely available and does not slow down the detector. Second, conditional tracking can reason about occluded objects that may no longer be visible in the current frame. The tracker can simply learn to keep those detections from the prior frame around.

2.2 Object Tracking

Tracking links objects across time. In this section, we mostly focus on multi-object tracking.

2.2.1 Tracking-by-detection.

Most modern trackers [8, 44, 95, 158, 163, 188, 211, 220, 265] follow the tracking-by-detection paradigm. An off-the-shelf object detector [48, 152, 153, 224] first finds all objects in each individual frame. Tracking is then a problem of bounding box association. SORT [8] tracks bounding boxes using a Kalman filter and associates each bounding box with its highest overlapping detection in the current frame using bipartite matching. DeepSORT [211] augments the overlap-based association cost in SORT with appearance features from a deep network. More recent approaches focus on increasing the robustness of object association. Tang et al. [188] leverage person-reidentification features and human pose features. Xu et al. [220] take advantage of the spatial locations over time. BeyondPixel [163] uses 3D shape information to track vehicles.

These methods have two drawbacks. First, the data association discards image appearance features [8] or requires a computationally expensive feature extractor [50, 163, 188, 220]. Second, detection is separated from tracking. In our approach in Chapter 5, association is almost free. Association is *learned* jointly with detection. Also, our detector takes the previous tracking results as an input, and can learn to recover missing or occluded objects from this additional cue.

2.2.2 Joint detection and tracking.

A recent trend in multi-object tracking is to convert existing detectors into trackers and combine both tasks in the same framework. Feichtenhofer et

al. [47] use a siamese network with the current and past frame as input and predict inter-frame offsets between bounding boxes. Integrated detection [248] uses tracked bounding boxes as additional region proposals to enhance detection, followed by bipartite-matching-based bounding-box association. Tracktor [7] removes the box association by directly propagating identities of region proposals using bounding box regression. In video object detection, Kang et al. [77, 78] feed stacked consecutive frames into the network and do detection for a whole video segment. And Zhu et al. [270] use flow to warp intermediate features from previous frames to accelerate inference.

Our method in Chapter 5 belongs to this category. The difference is that all of these works adopt the FasterRCNN framework [153], where the tracked boxes are used as region proposals. This assumes that bounding boxes have a large overlap between frames, which is not true in low-framerate regimes. As a consequence, Tracktor [7] requires a motion model [26, 41] for low-framerate sequences. Our approach instead provides the tracked predictions as an additional point-based heatmap input to the network. The network is then able to reason about and match objects anywhere in its receptive field even if the boxes have no overlap at all.

2.2.3 Global tracking.

Traditional trackers first detect objects offline, and consider object association across all frames as a combinatorial optimization problem. Our CenterTrack introduced in Chapter 5 formulates tracking as a min-cost max-flow

problem over a graph, where nodes are detections and edges are valid associations. MPN [12] simplifies the graph construction and proposes a neural solver that performs the graph optimization. LPC [30] additionally considers a classification module on the graph. Lif_T [188] incorporates person ReID and pose features in the graph optimization. These methods are still based on pairwise associations and use combinatorial optimization to select globally consistent pairwise assignments. Our method in Chapter 6 directly outputs consistent long-term trajectories without combinatorial optimization. This is done by a single forward pass in a relatively shallow network.

2.2.4 Transformers in tracking.

Trackformer [125] augments DETR [17] with additional object queries from existing tracks, and propagates track IDs as in Tracktor [7]. TransTrack [179] uses features from historical tracks as queries, but associates objects based on updated bounding box locations. MOTR [238] follows the DETR [17] structure and iteratively propagates and updates track queries to associate object identities. MO3TR [267] additionally uses a temporal attention module to update the status of each track over a temporal window, and feeds updated track features as queries in DETR. The common idea behind these works is to use the object query mechanism in DETR [17] to extend existing tracks frame-by-frame. We use transformers in a different way. Our transformer in Chapter 6 uses queries to generate entire trajectories at once. Our queries do not generate new boxes, but group already-detected boxes into global trajectories.

2.3 Open-vocabulary object recognition

Most object recognition works discussed above only focus on one single dataset or a single domain. Here we review literature of object recognition in a broader perspective, including recognizing objects from multiple datasets/ domains (Section 2.3.1) and recognition on a new testing label set (Section 2.3.2).

2.3.1 Computer vision on multiple datasets

In recent years, training on multiple diverse datasets has emerged as an effective tool to improve model robustness for depth estimation [148] and stereo matching [225]. In these domains, unifying the output space involves modeling different camera transformations and depth ambiguities. In contrast, for recognition, dataset unification involves merging different *semantic* concepts. MSeg [91] manually unified the taxonomies of 7 semantic segmentation datasets and used Amazon Mechanical Turk to resolve inconsistent annotations between datasets. In contrast, In refcptunidet we propose to learn a label space from visual data automatically, without any manual effort.

Wang et al. [207] train a universal object detector on multiple datasets, and gain robustness by joining diverse sources of supervision. This is similar to our split classifier baseline, while they work on small datasets and didn't model the training differences between different datasets. Both Wang et al. [207] and MSeg [91] observe a performance drop in a single unified model. With our dedicated training framework, this is not the case: our unified model performs

as well as single-dataset models on the training datasets. Also, these multi-headed models produce a dataset-specific prediction for each input image. When evaluated in-domain, they require knowledge of the test domain. When evaluated out-of-domain, they produce multiple outputs for a single concept. This limits their generality and usability. Our approach in Chapter 7, on the other hand, unifies visual concepts in a single label space and yields a single consistent model that does not require knowledge of the test domain and can be deployed cleanly in new domains.

Zhao et al. [250] trains a universal detector on multiple datasets: COCO [109], Pascal VOC [42], and SUN-RGBD [175], with under 100 classes in total. They manually merge the taxonomies and then train with cross-dataset pseudo-labels generated by dataset-specific models. The pseudo-label idea is complementary to our work. Our unified label space learning removes the manual labor, and works on a much larger scale: we unify COCO, Objects365, and OpenImages, with more complex label spaces and 900+ classes.

YOLO9000 [150] combines detection and classification datasets to expand the detection vocabulary. LVIS [61] extents COCO annotations to >1000 classes in a federated way. Our approach of fusing multiple annotated datasets is complementary and can be operationalized with no manual effort to unify disparate object detection datasets.

2.3.2 Zero-shot classification and detection

zero-shot object detection, or also named **Open-vocabulary object detection**, aims to detect objects outside of the training vocabulary. The basic solution [5] is to replace the last classification layer with language embeddings (e.g., GloVe [141]) of the class names. Rahman *et al.* [146] and Li *et al.* [105] improve the classifier embedding using external text information. OVR-CNN [236] pretrains the detector on image-text pairs. ViLD [60], OpenSeg [57] and langSeg [97] upgrade the language embedding to CLIP [145]. ViLD further distills region features from CLIP image features. Our framework in Chapter 8 use CLIP [145] classifier as well, but do not use distillation. Instead, we use additional image-labeled data for co-training.

2.3.3 Weakly-supervised object detection

Weakly-supervised object detection (WSOD) trains object detector using image-level labels. Many works use only image-level labels without any box supervision [101,165,166,201,227]. WSDDN [9] and OIRC [188] use a subnetwork to predict per-proposal weighting and sum up proposal scores into a single image scores. PCL [187] first clusters proposals and then assign image labels at the cluster level. CASD [70] further introduces feature-level attention and self-distillation. As no bounding box supervision is used in training, these methods rely on low-level region proposal techniques [2, 199], which leads to reduced localization quality.

Another line of WSOD work uses bounding box supervision together

with image labels, known as **semi-supervised WSOD** [36, 45, 102, 113, 198, 223, 251]. YOLO9000 [150] mixes detection data and classification data in the same mini-batch, and assigns classification labels to anchors with the highest predicted scores. DLWL [147] combines self-training and clustering-based WSOD [187], and again assigns image labels to max-scored proposals. MosaicOS [239] handles domain differences between detection and image datasets by mosaic augmentation [10] and proposed a three-stage self-training and finetuning framework. In segmentation, Pinheiro *et al.* [142] use a log-sum-exponential function to aggregate pixels scores into a global classification. Our work in Chapter 8 belongs to semi-supervised WSOD. Unlike prior work, we use a simple image-supervised loss. Besides image labels, researchers have also studied complementary methods for weak localization supervision like points [23] or scribles [154].

2.3.4 Large-vocabulary object detection

Large-vocabulary object detection [61, 150, 170, 226] requires detecting 1000+ classes. Many existing works focus on handling the long-tail problem [22, 49, 104, 133, 215, 244]. Repeat factor sampling (RFS) [61] oversamples classes with fewer annotations. Equalization losses [183, 184] and SeeSaw loss [203] reweights the per-class loss by balancing the gradients [183] or number of samples [203]. Federated Loss [256] subsamples classes per-iteration to mimic the federated annotation [61]. Yang *et al.* [226] detects 11K classes with a label hierarchy. Our method in Chapter 8 builds on these advances, and we tackle the problem from a different aspect: using additional image-labeled

data.

Proposal Network Generalization. ViLD [60] reports that region proposal networks have certain generalization abilities for new classes by default. Dave *et al.* [33] shows segmentation and localization generalizes across classes. Kim *et al.* [82] further improves proposal generalization with a localization quality estimator. In our experiments, we found proposals to generalize well enough (see § 8.3.1), as also observed in ViLD [60]. Further improvements to RPNs [60, 82, 85, 124] can hopefully lead to better results.

2.3.5 Language supervision for object detection

is a recent popular research topic. VirTex [35] and OVR-CNN [236] pretrain backbones on language tasks and show benefits for detection. Cap2Det [230] learns a mapping from sentences to image labels in the detector's vocabulary then applies WSOD [9]. MDETR [76] performs instance-level detection by training on paired object-caption annotations. Our work in Chapter 8 use language data [162] in a similar way as Cap2Det, but simpler: we extract image labels from captions using a naive text-match.

Chapter 3

Preliminaries

In this chapter, we first formally formulate the tasks we want to solve. Then we introduce existing technics of these tasks that we will build on.

3.1 Object representation

Let $I \in \mathbb{R}^{H \times W \times 3}$ be an image. The goal of object detection is to identify and localize all objects from a set of predefined classes C. I.e., predicting a set of objects $Y = \{b_i, c_i, s_i\}_i^{i=N}$, where $b_i \in \mathbb{R}^4$ is the bounding box coordinate, $c_i \in [1, \ldots, C]$ is the class, and $s_i \in [0, 1]$ is the confidence score.

3.1.1 Bounding Box

An object detector takes the image I as input and produces the set of objects Y as its output. Modern detectors transform predefined locations (one-stage detectors) [189], object proposals (two-stage detectors) [153], or object queries (end-to-end detectors) [17] p_i into box predictions b_i and an associated class score vector $\mathbf{s_i} \in \mathbb{R}^{\mathbf{C}}$. The final class label c_i and the detection score s_i are usually taken as the argmax of the class-score vector $\mathbf{s_i}$.

One-stage detectors [108, 151, 189] jointly predict the object loca-

tion and likelihood score in a single network. Let $L_{i,c} = 1$ indicate a positive detection for object candidate i and class c, and let $L_{i,c} = 0$ indicate background. Most one-stage detectors [108, 189] then parametrize the class likelihood as a Bernoulli distribution using an independent sigmoid per class: $s_i(c) = P(L_{i,c} = 1) = \sigma(w_c^\top \vec{f_i})$, where $f_i \in \mathbb{R}^C$ is a feature produced by the backbone and w_c is a class-specific weight vector. During training, this probabilistic interpretation allows one-stage detectors to simply maximize the log-likelihood $\log(P(L_{i,c}))$ or the focal loss [108] of ground-truth annotations. One-stage detectors differ from each other in the definition of positive $\hat{L}_{i,c} = 1$ and negative $\hat{L}_{i,c} = 0$ samples. Some use anchor overlap [83, 108, 242], others use locations [189]. All optimize log-likelihood and use the class probability to score boxes. All directly regress to bounding box coordinates.

Two stage detectors [14, 153] first extract potential object locations, called object proposals, using an objectness measure $P(O_i)$. They then extract features for each potential object, classify them into $\mathcal{C}$ classes or background $P(C_i|O_i = 1)$ with $C_i \in \mathcal{C} \cup \{bg\}$, and refine the object location. Each stage is supervised independently. In the first stage, a Region Proposal Network (RPN) learns to classify annotated objects b_i as foreground and other boxes as background. This is commonly done through a binary classifier trained with a log-likelihood objective. However, an RPN defines background regions very conservatively. Any prediction that overlaps an annotated object 30% or more may be considered foreground. This label definition favors recall over precision and accurate likelihood estimation. Many partial objects receive

a large proposal score. In the second stage, a softmax classifier learns to classify each proposal into one of the foreground classes or background. The classifier uses a log-likelihood objective, with foreground labels consisting of annotated objects and background labels coming from high-scoring first-stage proposals without annotated objects close-by. During training, this categorical distribution is implicitly conditioned on positive detections of the first stage, as it is only trained and evaluated on them.

3.1.2 Points

Keypoint estimation [16,24,63,130,137,218,254], commonly uses a fully convolutional encoder-decoder network to predict a multi-channel heatmap for each type of keypoint (e.g., one heatmap for human head, another heatmap for human wrist). I.e., they take the image $I \in \mathbb{R}^{H \times W \times 3}$ as input and produce a heatmap $\mathbf{Y} \in \mathbb{R}^{\mathbf{H'} \times \mathbf{W'} \times \mathbf{C}}$, where the heatmap resolution (H', W') is usually downsampled from the input resolution (H, W). The network is trained in a fully supervised manner, with either a L2 loss to a rendered Gaussian map [16, 24, 130, 218] or with a per-pixel logistic regression loss [93, 135, 136]. State-of-the-art keypoint estimation networks, e.g. 104-layer HourglassNetwork [93, 131], are trained in a fully convolutional manner. The training is guided by a multi-peak Gaussian heatmap $Y_{\cdot c} \in (0, 1)^{H \times W}$, where each keypoint defines the mean of a Gaussian Kernel. The standard deviation is either fixed, or set proportional to the object size [93]. The Gaussian heatmap serves as the regression target in the L2 loss case and as the weight map to reduce penalty near the positive location in the logistic regression case [93].

CornerNet for object detection CornerNet [93] first uses keypoint estimation with an HourglassNetwork [131] as an object detector. They predict two sets of heatmaps for the opposing corners of the box. In order to balance the positive and negative detections they use a modified focal loss [108] for training:

$$L_{det} = -\frac{1}{N}\sum_{i=1}^{H}\sum_{j=1}^{W}\begin{cases}(1-\hat{Y}_{ij})^{\alpha}\log(\hat{Y}_{ij}) & \text{if } Y_{ij}=1\\ (1-Y_{ij})^{\beta}(\hat{Y}_{ij})^{\alpha}\log(1-\hat{Y}_{ij}) & \text{o.w.}\end{cases} \tag{3.1}$$

α and β are hyper-parameters and fixed to $\alpha = 2$ and $\beta = 4$ during training [93].

For sub-pixel accuracy of extreme points, Law et al. additionally regress to category-agnostic keypoint offset ($\Delta^{(a)}$) for each corner. This regression recovers part of the information lost in the down-sampling of the hourglass network. The offset map is trained with smooth L1 Loss [58] (SL_1) on each ground truth extreme point location:

$$L_{off} = \frac{1}{N}\sum_{k=1}^{N} SL_1(\Delta^{(a)}, \vec{x}/n - \lfloor \vec{x}/n \rfloor), \tag{3.2}$$

where n is the down-sampling factor ($n = 4$ for HourglassNet), $\vec{x}$ is the coordinate of the keypoint.

3.2 Object recognition through time

Let $I^1, I^2, \ldots, I^T$ be a series of images. The goal of a tracker is to find trajectories $\vec{\tau}_1, \vec{\tau}_2, \ldots, \vec{\tau}_K$ of all objects over time. Each trajectory $\vec{\tau}_k = [\tau_k^1, \ldots, \tau_k^T]$ describes a tube of object locations $\tau_k^t \in \mathbb{R}^4 \cup \{\emptyset\}$ through time t. $\tau_k^t = \emptyset$ indicates that the object k cannot be located in frame t. The tracker

may optionally predict the object class score s_k [32] for each trajectory, usually as the average class of its per-frame slices.

Tracking by detection decomposes the tracking problem into per-frame detection and inter-frame object association. Object detection first finds N_t candidate objects $b_1^t, b_2^t, \ldots$ as bounding boxes $b_i^t \in \mathbb{R}^4$ in each frame I^t. Association then links existing tracks $\vec{\tau}_k$ to current detected objects using an object indicator $\alpha_k^t \in \{\emptyset, 1, 2, \ldots, N_t\}$ at each frame t :

$$\tau_k^t = \begin{cases} \emptyset & \text{if } \alpha_k^t = \emptyset \\ b_{\alpha_k^t}^t & \text{otherwise} \end{cases}$$

Most prior works define the association greedily through pairwise matches between objects in adjacent or nearby frames [7, 8, 247, 255], or rely on offline combinatorial optimization for global association [12, 52, 241]. In Chapter 6, we show how to perform joint detection and global association within a single forward pass through a network. The network learns global tracking within a video clip of 32 frames in an end-to-end fashion.

3.3 Multi-dataset detection

All the frameworks discussed above use a fixed label set C. Many existing works on object detection focus on the COCO dataset [109], which contains balanced annotations for 80 common object classes. Training an object detector on a single follows a simple recipe: Minimize a loss ℓ, usually box-level log-likelihood, over an sampled image $\hat{I}$ and its corresponding anno-

tated bounding boxes annotations $\hat{B}$ from the dataset $\mathcal{D}$:

$$\min_{\Theta} \mathbb{E}_{(\hat{I},\hat{B})\sim\mathcal{D}} \left[\ell(\mathcal{M}(\hat{I};\Theta), \hat{B}) \right]. \tag{3.3}$$

Here, $\hat{B}$ contains class-specific box annotations. The loss ℓ operates on sets of outputs and annotations, and matches them using an overlap criterion.

Let's now consider training a detector on multiple datasets $\mathcal{D}_1, \mathcal{D}_2, \ldots$, each with their own label space $C_1, C_2, \ldots$. Each label space C_i is usually different from each other, often with partial overlaps for common classes. A natural way to train on multiple datasets is to simply combine all annotations of all datasets into a much larger dataset $\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2 \cup \ldots$, and merge their label spaces $C = C_1 \cup C_2 \cup \ldots$. Labels that repeat across datasets are merged. We then optimize the same loss with more data:

$$\min_{\Theta} \mathbb{E}_{(\hat{I},\hat{B})\sim\mathcal{D}_1\cup\mathcal{D}_2\cup\ldots} \left[\ell(\mathcal{M}(\hat{I};\Theta), \hat{B}) \right]. \tag{3.4}$$

This has shown promise on smaller, evenly distributed datasets [42, 207, 216]. It has the advantage that shared classes between the datasets train on a larger set of annotations. However, modern large-scale detection datasets feature more natural class distributions that are imbalanced. Objects365 [160] contains 5× more images than COCO and OpenImages [89] is 18× larger than COCO. While the top 20% of classes in Objects365 and OpenImages contain 19× and 20× more images than COCO, respectively, the bottom 20% are actually smaller than COCO classes. This imbalance in class distributions and dataset sizes all but guarantees that a simple concatenation of datasets will

not work. In fact, not even the same loss (3.3) works for all datasets. Most successful Objects365 models [54] employ class-aware sampling [164]. OpenImages models treat rare classes differently [184] and model the hierarchy of classes in the loss [140].

This suggests that training a detector M_k on a dataset D_k requires a dataset-specific loss ℓ_k:

$$\min_{\Theta} \mathbb{E}_{(\hat{I},\hat{B})\sim\mathcal{D}_k} \left[\ell_k(\mathcal{M}_k(\hat{I};\Theta), \hat{B}) \right]. \tag{3.5}$$

No single loss generalizes to all datasets. In the Chapter 7, we present a different view of multi-dataset training and train a model on all datasets.

3.4 Open-vocabulary object detection

Contrastive Language-Image Pretraining (CLIP) [145] learns image and text features for zero-shot classification. It trains on a large image-text pair dataset $\{(I,t)\}_i$, and enforces the corresponding image and text features of the same pair to be similar. Specifically, for a batch of B image-text pairs $\{(I,t)_i\}_{i=1}^B$, CLIP first extracts image features $F = \mathcal{V}(\{I_i\}_{i=1}^B)$ and text features $G = \mathcal{L}(\{t_i\}_{i=1}^B)$, where $\mathcal{V}$ and $\mathcal{L}$ are the image encoder and text encoder, respectively. $F \in \mathbb{R}^{B\times D}$, $G \in \mathbb{R}^{B\times D}$. CLIP then apply an infoNCE loss:

$$L = \frac{1}{2}\sum_{i=1}^{B}(CE((FG^\top)_{\cdot i}, i) + CE((FG^\top)_{i\cdot}, i))$$

where CE is the cross-entropy loss $CE(s,c) = -\log \text{softmax}(s)_c$. When testing on a classification dataset with vocabulary $\mathcal{C}^{test}$, CLIP [145] compose

a text set of $T = \{\text{a photo of } t_i\}_{t_i \in C^{test}}$, and get its text features $G = \mathcal{L}(T)$ using the trained text encoder. G is used as the classifier during testing.

Open-vocabulary object detection allows the vocabulary C to be different between training and testing. Simply replacing the classification weights $\mathbf{W}$ with fixed language embeddings of class names converts a traditional detector to an open-vocabulary detector [5]. The region features are trained to match the fixed language embeddings. We follow Gu *et al.* [60] to use the CLIP embeddings [145] described above as the classification weights. In theory, this open-vocabulary detector based on CLIP embeddings can detect any object class. However, in practice, it yields unsatisfying results as shown in Figure 8.1. Our method uses image-level supervision to improve detection including in the open-vocabulary setting.

Chapter 4

CenterNet: Unified object representation

[1] Current object detectors represent each object through an axis-aligned bounding box that tightly encompasses the object [58, 59, 108, 149, 153]. They then reduce object detection to image classification of an extensive number of potential bounding boxes. For each bounding box, the classifier determines if the image content is a specific object or background. One-stage detectors [108, 149] slide a complex arrangement of possible bounding boxes, called anchors, over the image and classify them directly without specifying the box content. Two-stage detectors [58,59,153] recompute image-features for each potential box, then classify those features. Post-processing, namely non-maxima suppression, then removes duplicate detections for the same instance by computing bounding box overlap. This post-processing is hard to differentiate and train [66], hence most current detectors are not end-to-end trainable. Nonetheless, over the past five years [59], this idea has achieved good empirical success [29, 63, 69, 73, 103, 111, 169, 171, 219, 264, 268]. Sliding window-based

[1] The main body of this chapter is based on our technical report: Objects as Points, Xingyi Zhou, Dequan Wang, Philipp Krähenbühl, in arXiv 1904.07850.
The results in Section 4.4 are based on our follow-up technical report: Probabilistic two-stage detection, Xingyi Zhou, Vladlen Koltun, Philipp Krähenbühl, in arXiv 2103.07461.
Code is released at https://github.com/xingyizhou/CenterNet and https://github.com/xingyizhou/CenterNet2

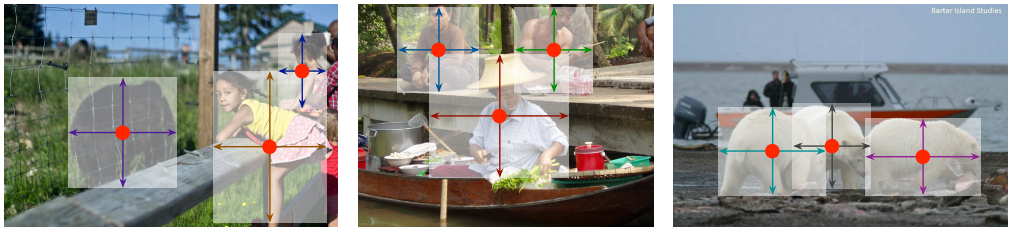


Figure 4.1: We model an object as the center point of its bounding box. The bounding box size and other object properties are inferred from the keypoint feature at the center. Best viewed in color.

detectors are however a bit inefficient, as they need to enumerate all possible object locations and dimensions on a large input resolution.

In this chapter, we provide a much simpler and more effective alternative. We represent objects by a single point at their bounding box center (see Figure 4.1). Other properties, such as object size, dimension, 3D extent, orientation, and pose are then regressed directly from image features at the center location. Object detection is then a standard keypoint estimation problem [15, 130, 254]. We simply feed the input image to a fully convolutional network [119, 131] that generates a heatmap. Peaks in this heatmap correspond to object centers. Image features at each peak predict the objects bounding box height and width. The model trains using standard dense supervised learning [130, 254]. Inference is a single network forward-pass, without non-maximal suppression for post-processing.

Our method is general and can be extended to other tasks with minor effort. We provide experiments on 3D object detection [13, 55] and multi-person human pose estimation [16], by predicting additional outputs at each

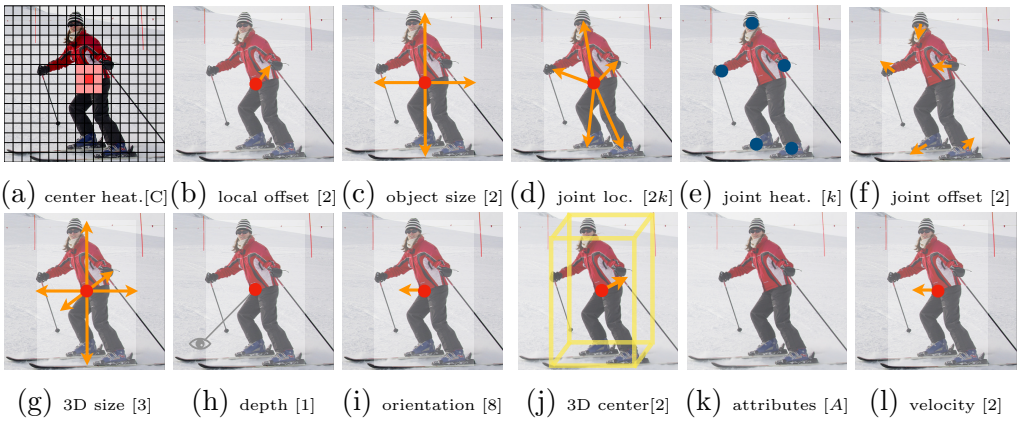


Figure 4.2: Outputs of our network for different tasks: *top left* for object detection, *top right* for pose estimation, *bottom* for 3D object detection. All modalities are produced from a common backbone, with a different 3×3 and 1×1 output convolutions separated by a ReLU. The number in brackets indicates the output channels. See section 4.1 for details.

center point (Figure 4.2). For 3D bounding box estimation, we regress to the object absolute depth, 3D bounding box dimensions, object orientation [128], and the offset from the 2D center to the projected 3D center (due to perspective projection). For human pose estimation, we consider the 2D joint locations as offsets from the center and directly regress to them at the center point location.

The simplicity of our method, CenterNet, allows it to run at a very high speed (Figure 4.3). With a simple Resnet-18 and up-convolutional layers [218], our network runs at 142 FPS with 28.1% COCO bounding box AP. With a carefully designed keypoint detection network, DLA-34 [233], our network achieves 37.4% COCO AP at 52 FPS. Equipped with the state-of-the-art keypoint estimation network, Hourglass-104 [93, 131], and multi-scale

testing, our network achieves 45.1% COCO AP at 1.4 FPS. On 3D bounding box estimation, CenterNet achieves 33.8 mAP on the nuScenes dataset [13], outperforming the state-of-the-art monocular 3D detector [168] and LiDAR based baselines [92]. On human pose estimation, we perform competitively with state-of-the-art at a high inference speed.

4.1 Objects as Points

Let $(x_1^{(k)}, y_1^{(k)}, x_2^{(k)}, y_2^{(k)})$ be the bounding box of object k with category c_k. Its center point is lies at $p_k = (\frac{x_1^{(k)}+x_2^{(k)}}{2}, \frac{y_1^{(k)}+y_2^{(k)}}{2})$. We use our keypoint estimator $\hat{Y}$ to predict all center points. In addition, we regress to the object size $s_k = (x_2^{(k)} - x_1^{(k)}, y_2^{(k)} - y_1^{(k)})$ for each object k. To limit the computational

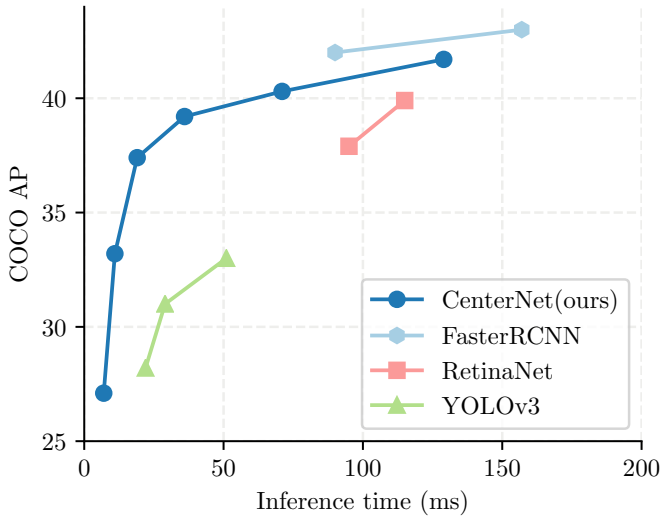


Figure 4.3: Speed-accuracy trade-off on COCO validation for real-time detectors. The proposed CenterNet performs competitively with a range of state-of-the-art algorithms especially on high speed regimes.

burden, we use a single size prediction $\hat{S} \in \mathbb{R}^{\frac{W}{R} \times \frac{H}{R} \times 2}$ for all object categories. We use an L1 loss at the center point similar to Objective 3.2 in CornerNet [93]:

$$L_{size} = \frac{1}{N} \sum_{k=1}^{N} \left| \hat{S}_{p_k} - s_k \right|. \tag{4.1}$$

We do not normalize the scale and directly use the raw pixel coordinates. We instead scale the loss by a constant λ_{size}. The overall training objective is

$$L_{det} = L_k + \lambda_{size} L_{size} + \lambda_{off} L_{off}. \tag{4.2}$$

We set $\lambda_{size} = 0.1$ and $\lambda_{off} = 1$ in all our experiments unless specified otherwise. We use a single network to predict the keypoints $\hat{Y}$, offset $\hat{O}$, and size $\hat{S}$. The network predicts a total of $C + 4$ outputs at each location. All outputs share a common fully-convolutional backbone network. For each modality, the features of the backbone are then passed through a separate 3×3 convolution, ReLU and another 1×1 convolution. Figure 4.2 shows an overview of the network output. Section 4.2 contains additional architectural details.

From points to bounding boxes At inference time, we first extract the peaks in the heatmap for each category independently. We detect all responses whose value is greater or equal to its 8-connected neighbors and keep the top 100 peaks. Let $\hat{\mathcal{P}}_c$ be the set of n detected center points $\hat{\mathcal{P}} = \{(\hat{x}_i, \hat{y}_i)\}_{i=1}^{n}$ of class c. Each keypoint location is given by an integer coordinates $(\hat{x}_i, \hat{y}_i)$. We use the keypoint values $\hat{Y}_{x_i y_i c}$ as a measure of its detection confidence, and

produce a bounding box at location

$$(\hat{x}_i + \delta\hat{x}_i - \hat{w}_i/2, \quad \hat{y}_i + \delta\hat{y}_i - \hat{h}_i/2,$$
$$\hat{x}_i + \delta\hat{x}_i + \hat{w}_i/2, \quad \hat{y}_i + \delta\hat{y}_i + \hat{h}_i/2),$$

where $(\delta\hat{x}_i, \delta\hat{y}_i) = \hat{O}_{\hat{x}_i,\hat{y}_i}$ is the offset prediction and $(\hat{w}_i, \hat{h}_i) = \hat{S}_{\hat{x}_i,\hat{y}_i}$ is the size prediction. All outputs are produced directly from the keypoint estimation without the need for IoU-based non-maxima suppression (NMS) or other post-processing. The peak keypoint extraction serves as a sufficient NMS alternative and can be implemented efficiently on device using a 3×3 max pooling operation.

4.1.1 3D detection

3D detection estimates a three-dimensional bounding box per objects and requires three additional attributes per center point: depth, 3D dimension, and orientation. We add a separate head for each of them. The depth d is a single scalar per center point. However, depth is difficult to regress to directly. We instead use the output transformation of Hu et al. [67] and use the inverse sigmoid transform $d = 1/\sigma(\hat{d}) - 1$. We compute the depth as an additional output channel $\hat{D} \in [0, 1]^{\frac{W}{R} \times \frac{H}{R}}$ of our keypoint estimator. The 3D dimensions of an object are three scalars. We directly predict to their absolute values in meters. All outputs use two convolutional layers separated by a ReLU, and are trained with an L1 loss. Orientation is a single scalar by default. However, it can be hard to regress to. We follow Mousavian et al. [128] and represent the orientation as two bins with in-bin regression.

Due to perspective foreshortening, the 3D bounding box center is not identical to the 2D bounding box center. We thus add one additional head to regress to the offset from the 2D center point to the projected 3D center [2]. In addition, the nuScenes dataset [13] requires predicting objects' attributes (e.g., moving or parking of vehicles) and velocity. We add these regression heads accordingly. We train the attributes head with independent logistic regression and train the velocity head with L1 loss. Figure 4.2 gives an overview of the output heads.

4.1.2 Human pose estimation

Human pose estimation aims to estimate k 2D human joint locations for every human instance in the image ($k = 17$ for COCO). We considered the pose as a $k \times 2$-dimensional property of the center point, and parametrize each keypoint by an offset to the center point. We directly regress to the joint offsets (in pixels) $\hat{J} \in \mathbb{R}^{\frac{W}{R} \times \frac{H}{R} \times k \times 2}$ with an L1 loss. We ignore invisible keypoints by masking the loss. This results in a regression-based one-stage multi-person human pose estimator similar to the slow-RCNN version counterparts Toshev et al. [194] and Sun et al. [181].

To refine the keypoints, we further estimate k human joint heatmaps $\hat{\Phi} \in \mathbb{R}^{\frac{W}{R} \times \frac{H}{R} \times k}$ using standard bottom-up multi-human pose estimation [16, 130, 135]. We train the human joint heatmap with focal loss and local pixel offset analogous to the center detection discussed in Section 3.1.2.

[2] We observe this offset is essential for a good performance in 3D detection.

We then snap our initial predictions to the closest detected keypoint on this heatmap. Here, our center offset acts as a grouping cue, to assign individual keypoint detections to their closest person instance. Specifically, let $(\hat{x}, \hat{y})$ be a detected center point. We first regress to all joint locations $l_j = (\hat{x}, \hat{y}) + \hat{J}_{\hat{x}\hat{y}j}$ for $j \in 1 \dots k$. We also extract all keypoint locations $L_j = \{\tilde{l}_{ji}\}_{i=1}^{n_j}$ with a confidence > 0.1 for each joint type j from the corresponding heatmap $\hat{\Phi}_{\cdot\cdot j}$. We then assign each regressed location l_j to its closest detected keypoint $\arg\min_{l \in L_j} (l - l_j)^2$ considering only joint detections within the bounding box of the detected object.

4.2 Implementation details

We experiment with 4 architectures: ResNet-18, ResNet-101 [218], DLA-34 [233], and Hourglass-104 [93]. We modify both ResNets and DLA-34 using deformable convolution layers [268] and use the Hourglass network as is.

Hourglass The stacked Hourglass Network [93, 131] downsamples the input by $4\times$, followed by two sequential hourglass modules. Each hourglass module is a symmetric 5-layer down- and up-convolutional network with skip connections. This network is quite large, but generally yields the best keypoint estimation performance.

ResNet Xiao et al. [218] augment a standard residual network [65] with three up-convolutional networks to allow for a higher-resolution output (out-

put stride 4). We first change the channels of the three upsampling layers to $256, 128, 64$, respectively, to save computation. We then add one 3×3 deformable convolutional layer before each up-convolution with channel $256, 128, 64$, respectively. The up-convolutional kernels are initialized as bilinear interpolation.

DLA Deep Layer Aggregation (DLA) [233] is an image classification network with hierarchical skip connections. We utilize the fully convolutional upsampling version of DLA for dense prediction, which uses iterative deep aggregation to increase feature map resolution symmetrically. We augment the skip connections with deformable convolution [268] from lower layers to the output. Specifically, we replace the original convolution with 3×3 deformable convolution at every upsampling layer.

We add one 3×3 convolutional layer with 256 channel before each output head. A final 1×1 convolution then produces the desired output.

Training We train on an input resolution of 512×512. This yields an output resolution of 128×128 for all the models. We use random flip, random scaling (between 0.6 to 1.3), cropping, and color jittering as data augmentation, and use Adam [84] to optimize the overall objective. For the residual networks and DLA-34, we train with a batch-size of 128 (on 8 GPUs) and learning rate 5e-4 for 140 epochs, with learning rate dropped $10\times$ at 90 and 120 epochs, respectively (following [218]). For Hourglass-104, we follow ExtremeNet [261] and

use batch-size 29 (on 5 GPUs, with master GPU batch-size 4) and learning rate 2.5e-4 for 50 epochs with 10× learning rate dropped at the 40 epoch. For detection, we fine-tune the Hourglass-104 from ExtremeNet [261] to save computation. The down-sampling layers of Resnet-101 and DLA-34 are initialized with ImageNet pretrain and the up-sampling layers are randomly initialized. Resnet-101 and DLA-34 train in 2 days on 8 TITAN-V GPUs, while Hourglass-104 requires 10 days.

Inference Following CornerNet [94], we use three levels of test augmentations: no augmentation, flip augmentation, and flip and multi-scale (0.5, 0.75, 1, 1.25, 1.5). For flip, we average the network outputs before decoding bounding boxes. For multi-scale, we use NMS to merge results. These augmentations yield different speed-accuracy trade-off, as is shown next.

4.3 Experiments

We evaluate our object detection performance on the MS COCO dataset [109], which contains 118k training images (train2017), 5k validation images (val2017) and 20k hold-out testing images (test-dev). We report average precision over all IOU thresholds (AP), and AP at IOU thresholds 0.5(AP_{50}).

4.3.1 Object detection

Table 4.1 shows our results on COCO validation with different backbones and testing options, while Figure 4.3 compares CenterNet with other

	AP			AP_{50}			AP_{75}			Time (ms)		
	N.A.	F	MS	N.A.	F	MS	N.A.	F	MS	N.A.	F	MS
HG-104	**40.3**	**42.2**	**45.1**	**59.1**	**61.1**	**63.5**	**44.0**	**46.0**	**49.3**	71	129	672
DLA-34	37.4	39.2	41.7	55.1	57.0	60.1	40.8	42.7	44.9	19	36	248
ResNet-101	34.6	36.2	39.3	53.0	54.8	58.5	36.9	38.7	42.0	22	40	259
ResNet-18	28.1	30.0	33.2	44.9	47.5	51.5	29.6	31.6	35.1	**7**	**14**	**81**

Table 4.1: Speed / accuracy trade off for different networks on COCO validation set. We show results without test augmentation (N.A.), flip testing (F), and multi-scale augmentation (MS).

real-time detectors. The running time is tested on our local machine, with Intel Core i7-8086K CPU, Titan Xp GPU, Pytorch 0.4.1, CUDA 9.0, and CUDNN 7.1. We download code and pre-trained models[3][4] to test run time for each model on the same machine.

Hourglass-104 achieves the best accuracy at a relatively good speed, with a 42.2% AP in 7.8 FPS. On this backbone, CenterNet outperforms CornerNet [93] (40.6% AP in 4.1 FPS) and ExtremeNet [261](40.3% AP in 3.1 FPS) in both speed and accuracy. Our run time improvement comes from fewer output heads and a simpler box decoding scheme. The concurrent CenterNet from Duan et al. [39] ((44.9% AP in 2.9 FPS)) performs slightly better than us for its more accurate localization based on local evidence (corner points). However it runs twice slower, due to the need for a complex center pooling and grouping.

Using the standard backbone ResNet-101, we achieve a decent AP of

[3] https://github.com/facebookresearch/detectron2
[4] https://github.com/pjreddie/darknet

	Backbone	FPS	AP	AP_{50}	AP_S	AP_L
YOLOv3 [151]	DarkNet-53	20	33.0	57.9	18.3	41.9
CornerNet-S [94]	Hourglass-54	33	34.4	-	13.7	47.4
CenterNet-DLA	DLA-34	28	39.2	57.1	19.9	51.4
MaskRCNN [63]	ResNeXt-152	3.5	49.3*/ 51.4*	-	-	-
SNIPER [171]	DPN-98	*2.5*	46.1	67.0	29.6	58.1
PANet [111]	ResNeXt-101	-	47.4	67.2	30.1	60.0
TridentNet [103]	ResNet-101	0.7	48.4	69.7	31.8	60.3
RetinaNet [108]	ResNeXt-101	5.4	40.8	61.1	24.1	51.2
FCOS [189]	ResNeXt-101	*8.9*	44.7	64.1	27.6	55.6
AlignDet [25]	ResNeXt-101	*5.6*	44.1	64.7	26.9	54.7
RefineDet [243]	ResNet-101	-	36.4/ 41.8	57.5/ 62.9	16.6/ 25.6	51.4/ 54.1
CornerNet [93]	Hourglass-104	4.1	40.5/ 42.1	56.5/ 57.8	19.4/ 20.8	53.9/ 56.7
ExtremeNet [261]	Hourglass-104	3.1	39.8/ 43.2	54.6/ 59.6	20.3/ 24.1	53.1/ 57.1
FSAF [264]	ResNeXt-101	*2.7*	42.9/ 44.6	63.8/ 65.2	26.6 / 29.7	52.7/ 54.6
RepPoint [229]	ResNet101	8.9	45.0/ 46.5	66.1/ 67.4	26.6/ 30.3	57.5/ 57.1
Duan et al. [39]	Hourglass-104	*2.9*	44.9/ 47.0	62.4/ 64.5	25.6/ 28.9	57.4/ 58.9
CenterNet-HG	Hourglass-104	7.8	42.1/ 45.1	61.1/ 63.9	24.1/ 26.6	52.8/ 57.7

Table 4.2: State-of-the-art comparison on COCO test-dev. Top: detectors with FPS $\geq$ 20; bottom: detectors with FPS $<$ 20. We show single-scale / multi-scale testing performance if applicable. Frame-per-second (FPS) were measured on the same machine whenever possible. Italic FPS highlight the cases, where the performance measure was copied from the original publication. A dash indicates methods for which neither code and models, nor public timings were available. *The MaskRCNN [63] results are on validation set [216].

34.6% in 45 FPS. This is already better and faster than the popular detector YOLOv3 [151]. Our fastest ResNet-18 model also achieves a respectable performance of 28.1% COCO AP at 142 FPS.

DLA-34 gives the best speed/accuracy trade-off. It runs at 52FPS with 37.4%AP. This is more than twice as fast as YOLOv3 [151] and 4.4%AP more

accurate. With flip testing, our model is still faster than YOLOv3 [151] and achieves accuracy of 39.2% AP.

State-of-the-art comparison We compare CenterNet to other state-of-the-art detectors on COCO test-dev in Table 4.2. CenterNet-DLA achieves 39.2% AP in 28 FPS with fliptest, outperform other real time detectors [94, 151] by a large margin. CenterNet-HG performs competitively to concurrent works [86, 93, 189, 229, 261, 264] at an equal or higher frame rate. Sophisticated two-stage detectors [103, 111, 171, 268] are more accurate, but also slower. Even though we did not explicitly handle the multi-scale object problem [107], there is no significant difference between CenterNet and other detectors for the error distribution among different object sizes. CenterNet behaves like a regular detector, but runs faster.

4.3.1.1 Additional experiments

Center point collision. In unlucky circumstances, two different objects might share the same center, if they perfectly align. In this scenario, CenterNet would only detect one of them. We start by studying how often this happens in practice and put it in relation to missing detections of competing methods.

In the COCO training set, there are 614 pairs of objects that collide onto the same center point at stride 4. There are 860001 objects in total, hence CenterNet is unable to predict $< 0.1\%$ of objects due to collisions in center points. This is much less than slow- or fast-RCNN miss due to imperfect

region proposals [199] ($\sim 2\%$), and fewer than anchor-based methods miss due to insufficient anchor placement [153] (20.0% for Faster-RCNN with 15 anchors at 0.5 IOU threshold). In addition, 715 pairs of objects have bounding box IoU > 0.7 and would be assigned to two anchors, hence a center-based assignment causes fewer collisions.

NMS. To verify that IoU-based NMS is not needed for CenterNet, we ran it as a post-processing step on our predictions. For DLA-34 (flip-test), the AP improves from 39.2% to 39.7%. For Hourglass-104, the AP stays at 42.2%. Given the minor impact, we do not use it.

4.3.2 3D detection

We perform 3D bounding box estimation experiments on the newly released nuScenes dataset [13].

nuScenes [13] is a large scale traffic dataset with accurate 3D annotations for 10 categories. In nuScenes, annotations are given per timestamp, which contains 6 synchronized cameras covering 360° with slight overlap. It contains 28k training frames (168k images), 6k validation frames (36k images), and 6k testing frames (36k images). The major evaluation metric is the mean average precision (mAP) defined on the object center distance on the bird eye view, over 4 distance threshold (0.5m, 1m, 2m, 4m) and averaged across all categories. The nuScenes evaluation is done per frame (with 6 cameras). We split the 6 cameras of the same frame as different samples for training. During

	Modality	mAP ↑	mATE ↓	mASE ↓	mAOE ↓	mAVE ↓	mAAE ↓	NDS ↑
PointPillars [92]	Lidar	30.5	**0.52**	0.29	**0.50**	**0.32**	0.37	**45.3**
Mappilary [168]	Camera	30.4	0.74	0.26	0.55	1.	**0.13**	38.4
Ours	Camera	**33.8**	0.66	**0.26**	0.63	1.	0.14	40.1

Table 4.3: Results on nuScenes test set. We show 3D bounding box mAP, mean translation error (mATE), mean size error (mASE), mean orientation error (mAOE), mean velocity error (mATE), mean attributes error (mAAE), and their weighted (with weight 5 on mAP and 1 on others) average NDS.

testing, we run detection on each individual image and merge them with a center-distance based NMS.

The original image resolution for nuScenes is 1600×900. We resize and crop the image to 800×448 for both training and testing. The training converges in 140 epochs, with learning rate dropped at the 90 and 120 epoch, respectively. Training takes about 3 days on 8 Titan-V GPUs. We use the DLA-34 backbone and set the loss weight for depth, orientation, dimension, and 2D-to-3D center offset to 1. All other hyper-parameters are the same as the detection experiments.

The results are shown in Table 4.3. Note that it is infeasible to predict velocity from still images. Both Mappilary [168] and us have the maximal average velocity error (AVE). Our method gives the best mAP compared to the state-of-the-art Camera-based method Mappilary [168] and Lidar-based baselines PointPillars [92].

4.3.3 Pose estimation

Finally, we evaluate CenterNet on human pose estimation in the MS COCO dataset [109]. We evaluate keypoint AP, which is similar to bounding box AP but replaces the bounding box IoU with object keypoint similarity. We test and compare with other methods on COCO test-dev.

We experiment with DLA-34 and Hourglass-104. The Hourglass model is fine-tuned from center point detection. DLA-34 converges in 320 epochs (about 3 days on 8GPUs) and Hourglass-104 converges in 150 epochs (8 days on 5 GPUs). All additional loss weights are set to 1. All other hyper-parameters are the same as object detection.

The results are shown in Table 4.4. Direct regression to keypoints

	AP^{kp}	AP^{kp}_{50}	AP^{kp}_{75}	AP^{kp}_{M}	AP^{kp}_{L}
CMU-Pose [16]	61.8	84.9	67.5	58.0	70.4
Mask-RCNN [63]	63.1	87.3	68.7	57.8	71.4
PersonLab [135]	66.5	85.5	71.3	62.3	70.0
CenterNet-DLA-reg	56.6	83.7	61.4	49.8	68.0
CenterNet-HG-reg	55.0	83.5	59.7	49.4	64.0
CenterNet-DLA	60.4	85.2	66.2	55.6	69.9
CenterNet-HG	63.0	86.8	69.6	58.9	70.4
Pose-AE [130]	62.8	84.6	69.2	57.5	70.6
SPM [132]	66.9	88.5	72.9	62.6	73.1
HRNet [177]	75.5	92.5	83.3	71.9	81.5
CenterNet-DLA+refine	72.8	90.9	80.4	69.3	79.2

Table 4.4: Keypoint detection on COCO test-dev for direct center-out offset (-reg) and matching to closest points (-jd) The results show COCO keypoint AP. Higher is better. Top: top-down methods. Bottom: bottom-up methods or methods with bottom-up refinement.

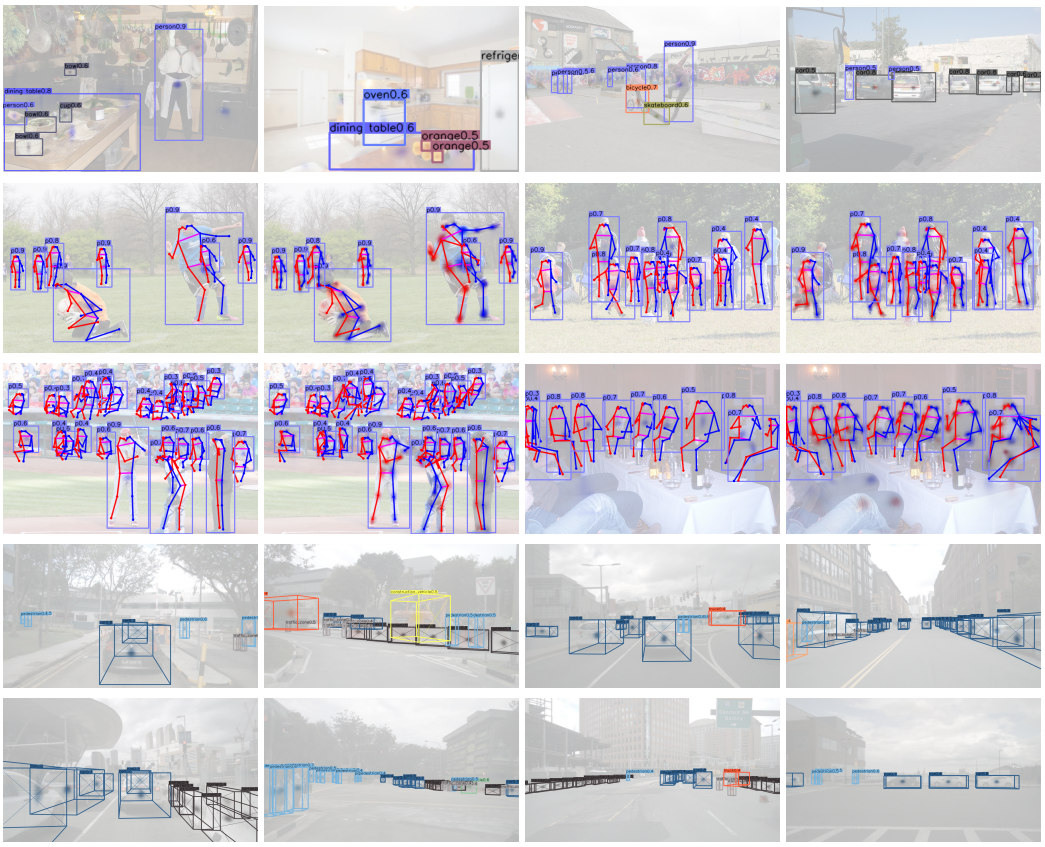


Figure 4.4: Qualitative results. All images were picked thematically without considering our algorithms performance. *First row:* object detection on COCO validation. *Second and third row:* Human pose estimation on COCO validation. For each pair, we show the results of center offset regression (left) and heatmap matching (right). *fourth and fifth row:* 3D bounding box estimation on nuScenes. We show projected bounding box on the image. The center heatmap and 3D boxes are shown overlaid on the original image.

performs reasonably, but not at state-of-the-art. Projecting our output to the closest joint detection improves the results throughout, and performs competitively with state-of-the-art multi-person pose estimators [16, 63, 130, 135]. Following the convention [130, 132], we provide the results after refined by

	Backbone	Stride	Input/Augmentation	Optimizer/epoch	mAP
CenterNet	DLA	4	512×512 rand. crop	Adam 230	37.4
CenterNet*	DLA	4	short512 multi-scale	Adam 24	37.1
	DLA	FPN	short512 multi-scale	SGD 12	35.2
	Res50	FPN	short512 multi-scale	SGD 12	36.1
	Res50	FPN	short800 multi-scale	SGD 12	**40.2**
FCOS [189]	Res50	FPN	short800 multi-scale	SGD 12	39.2
ATSS [242]	Res50	FPN	short800 multi-scale	SGD 12	39.7
FasterRCNN [153]	Res50	FPN	short800 multi-scale	SGD 12	37.9
RetinaNet [108]	Res50	FPN	short800 multi-scale	SGD 12	36.5
CascadeRCNN [14]	Res50	FPN	short800 multi-scale	SGD 12	41.6

Table 4.5: Experiments of interpolating CenterNet and FPN-based detectors [108, 153, 189] on COCO validation set. Top block: the original CenterNet in the main paper. Middle block: CenterNets with details matching FCOS [189] in different settings. Bottom block: popular FPN-based detectors in the standard setting.

single-person pose estimator [177] with our DLA backbone for reference. This verifies that CenterNet is general, easy to adapt to a new task.

Figure 4.4 shows qualitative examples on all tasks. In our followup work CenterPoint [231], we further show our point-based representation achieves the state-of-the-art performance on Lidar-base 3D detection on Waymo [178] and nuScenes [13] datasets.

4.4 CenterNet2

CenterNet was first released in 2019 and had achieved the best speed-accuracy trade-off in COCO as shown in Figure 4.3. However, it was soon exceeded by concurrent object detectors [186, 189, 202] equipped with stronger

backbones [53, 186, 240] and better training recipes [202]. One drawback of CenterNet is it relies on heavy keypoint-estimation backbones [131], which lack multi-scale designs like FPN [107]. In our followup work, we upgrade CenterNet with modern detection training recipes, including multi-scale outputs (FPN), better backbone, larger input size, and different optimizer and training schedule. As shown in Table 4.5, the upgraded CenterNet (CenterNet*) achieved better performance comparing to concurrent one-stage detectors [189, 242]. However, it still underperforms state-of-the-art multi-stage detector CascadeRCNN [14].

In our followup work Probablistic Two-stage Detection [256], we further upgrade CenterNet with multi-stage. We investigated the training objectives of standard two-stage detectors, and found the combinition of region proposal network and the second-stage classification does not follow a probabilistic interpretation. Our interpretation motivates us to use a strong one-stage detector (e.g. CenterNet) to replace the original RPN, and multiple the proposal score with the second-stage classification score as the final detection score.

Table 4.6 compares one- and two-stage detectors to corresponding probabilistic two-stage detectors designed via our framework. The first block of the table shows the performance of the original reference two-stage detectors, FasterRCNN and CascadeRCNN. The following blocks show the performance of four one-stage detectors and the corresponding probabilistic two-stage detectors, obtained when using the respective one-stage detector as the first stage in a probabilistic two-stage framework. For each one-stage detector, we show

	mAP	T_{first}	T_{tot}
FasterRCNN-RPN (original)	37.9	46ms	55ms
CascadeRCNN-RPN (original)	41.6	48ms	78ms
RetinaNet [108]	37.4	82ms	82ms
FasterRCNN-RetinaNet	40.4	60ms	63ms
CascadeRCNN-RetinaNet	42.6	61ms	69ms
GFL [100]	40.2	51ms	51ms
FasterRCNN-GFL	41.7	46ms	50ms
CascadeRCNN-GFL	42.7	46ms	57ms
ATSS [242]	39.7	56ms	56ms
FasterRCNN-ATSS	41.5	47ms	50ms
CascadeRCNN-ATSS	42.7	47ms	57ms
CenterNet*	40.2	51ms	51ms
FasterRCNN-CenterNet	41.5	46ms	50ms
CascadeRCNN-CenterNet	**42.9**	47ms	57ms

Table 4.6: Performance and runtime of a number of two-stage detectors, one-stage detectors, and corresponding probabilistic two-stage detectors (our approach). Results on COCO validation. Top block: two-stage FasterRCNN and CascadeRCNN detectors. Other blocks: Four one-stage detectors, each with two corresponding probabilistic two-stage detectors, one based on FasterRCNN and one based on CascadeRCNN. For each detector, we list its first-stage runtime (T_{first}) and total runtime (T_{tot}). All results are reported using standard Res50-1x with multi-scale training.

two versions of probabilistic two-stage models, one based on FasterRCNN and one based on CascadeRCNN.

All probabilistic two-stage detectors outperform their one-stage and two-stage precursors. Each probabilistic two-stage FasterRCNN model improves upon its one-stage precursor by 1 to 2 percentage points in mAP, and outperforms the original two-stage FasterRCNN by up to 3 percentage points

	Backbone	Epochs	mAP	Runtime
FCOS-RT	DLA-BiFPN-P3	48	42.1	21ms
CenterNet2	DLA-BiFPN-P3	48	43.7	25ms
CenterNet	DLA	230	37.6	18ms
YOLOV4	CSPDarknet-53	300	43.5	30ms
EfficientDet	EfficientNet-B2	500	43.5	23ms*
EfficientDet	EfficientNet-B3	500	46.8	37ms*
CenterNet2	DLA-BiFPN-P3	288	45.6	25ms
CenterNet2	DLA-BiFPN-P5	288	49.2	30ms

Table 4.7: Performance of real-time object detectors on COCO validation. Top: we compare CenterNet2 to realtime-FCOS under exactly the same setting. Bottom: we compare to detectors with different backbones and training schedules. *The runtime of EfficientDet is taken from the original paper [186] as the official model is not available. Other runtimes are measured on the same machine.

in mAP. More interestingly, each two-stage probabilistic FasterRCNN is *faster* than its one-stage precursor due to the leaner head design. A number of probabilistic two-stage FasterRCNN models are faster than the original two-stage FasterRCNN, due to more efficient FPN levels (P3-P7 vs. P2-P6) and because the probabilistic detectors use fewer proposals (256 vs. 1K). We observe similar trends with the CascadeRCNN models. Specifically, the CascadeRCNN-CenterNet design performs best among these probabilistic two-stage models, and we refer this model as CenterNet2.

Real-time models. Table 4.7 compares CenterNet2 models with small input size and light backbones to other real-time detectors. CenterNet2 outperforms realtime-FCOS [190] by 1.6 mAP with the same backbone and training

schedule, and is only 4 ms slower. Using the same FCOS-based backbone with longer training schedules [10, 186], it improves upon the original CenterNet [258] by 7.7 mAP, and comfortably outperforms the popular YOLOv4 [10] and EfficientDet-B2 [186] detectors with 45.6 mAP at 40 fps. Using a slightly different FPN structure and combining with self-training [272], CenterNet2 gets 49.2 mAP at 33 fps. While most existing real-time detectors are one-stage, here we show that two-stage detectors can be as fast as one-stage designs, while delivering higher accuracy.

State-of-the-art comparison. Table 4.8 compares our large models to state-of-the-art detectors on COCO test-dev. Using a "standard" large backbone ResNeXt101-DCN, CenterNet2 achieves 50.2 mAP, outperforming all existing models with the same backbone, both one- and two-stage. Note that CenterNet2 outperforms the corresponding CascadeRCNN model with the same backbone by 1.4 percentage points in mAP. This again highlights the benefits of a probabilistic treatment of two-stage detection.

To push the state-of-the-art of object detection, we further switch to a stronger backbone Res2Net [53] with BiFPN [186], a larger input resolution (1280×1280 in training and 1560×1560 in testing) with heavy crop augmentation (ratio 0.1 to 2) [186], and a longer schedule ($8\times$) with self-training [272] on COCO unlabeled images. Our final model achieves 56.4 mAP with a single model, outperforming all published numbers in the literature.

	Backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
CornerNet [93]	Hourglass-104	40.6	56.4	43.2	19.1	42.8	54.3
CenterNet [258]	Hourglass-104	42.1	61.1	45.9	24.1	45.5	52.8
Duan et al. [39]	Hourglass-104	44.9	62.4	48.1	25.6	47.4	57.4
RepPoint [229]	ResNet101-DCN	45.0	66.1	49.0	26.6	48.6	57.5
MAL [80]	ResNeXt-101	45.9	65.4	49.7	27.8	49.1	57.8
FreeAnchor [246]	ResNeXt-101	46.0	65.6	49.8	27.8	49.5	57.7
CentripetalNet [37]	Hourglass-104	46.1	63.1	49.7	25.3	48.7	59.2
FCOS [189]	ResNeXt-101-DCN	46.6	65.9	50.8	28.6	49.1	58.6
TridentNet [103]	ResNet-101-DCN	46.8	67.6	51.5	28.0	51.2	60.5
CPN [40]	Hourglass-104	47.0	65.0	51.0	26.5	50.2	60.7
SAPD [263]	ResNeXt-101-DCN	47.4	67.4	51.1	28.1	50.3	61.5
ATSS [242]	ResNeXt-101-DCN	47.7	66.6	52.1	29.3	50.8	59.7
BorderDet [229]	ResNeXt-101-DCN	48.0	67.1	52.1	29.4	50.7	60.5
GFL [100]	ResNeXt-101-DCN	48.2	67.4	52.6	29.2	51.7	60.2
PAA [83]	ResNeXt-101-DCN	49.0	67.8	53.3	30.2	52.8	62.2
TSD [174]	ResNeXt-101-DCN	49.4	69.6	54.4	**32.7**	52.5	61.0
RepPointv2 [229]	ResNeXt-101-DCN	49.4	68.9	53.4	30.3	52.1	62.3
AutoAssign [262]	ResNeXt-101-DCN	49.5	68.7	54.0	29.9	52.6	62.0
Deformable DETR [269]	ResNeXt-101-DCN	50.1	**69.7**	54.6	30.6	52.8	**65.6**
CascadeRCNN [14]	ResNeXt-101-DCN	48.8	67.7	52.9	29.7	51.8	61.8
CenterNet*	ResNeXt-101-DCN	49.1	67.8	53.3	30.2	52.4	62.0
CenterNet2 (ours)	ResNeXt-101-DCN	**50.2**	68.0	**55.0**	31.2	**53.5**	63.6
CRCNN-ResNeSt [240]	ResNeSt-200	49.1	67.8	53.2	31.6	52.6	62.8
GFLV2 [99]	Res2Net-101-DCN	50.6	69.0	55.3	31.3	54.3	63.5
DetectRS [144]	ResNeXt-101-DCN-RFP	53.3	71.6	58.5	33.9	56.5	66.9
EfficientDet-D7x [186]	EfficientNet-D7x-BiFPN	55.1	73.4	59.9	-	-	-
ScaledYOLOv4 [202]	CSPDarkNet-P7	55.4	73.3	60.7	38.1	59.5	67.4
CenterNet2 (ours)	Res2Net-101-DCN-BiFPN	**56.4**	**74.0**	**61.6**	**38.7**	**59.7**	**68.6**

Table 4.8: Comparison to the state of the art on COCO test-dev. We list object detection accuracy with single-scale testing. We retrained our baselines, CascadeRCNN (ResNeXt-101-DCN) and CenterNet*, under comparable settings. Other results are taken from the original publications. Top: detectors with comparable backbones (ResNeXt-101-DCN) and training schedules (2x). Bottom: detectors with their best-fit backbones, input size, and schedules.

Summary In this chapter, we present a new representation of objects: as points. Our CenterNet object detector builds on successful keypoint estimation networks, finds object centers, and regresses to their size. The algorithm is simple, fast, accurate, and end-to-end differentiable without NMS post-processing. The idea is general and has broad applications beyond two-dimensional detection. CenterNet can estimate a range of additional object properties, such as pose, 3D orientation, depth and extent, in one single forward pass. Our initial experiments are encouraging and open up a new direction for object recognition and related tasks.

Chapter 5

CenterTrack: Unified recognition and tracking

[1]CenterNet works on still images. However, our world is dynamic. To link objects across time, we next look into the task of object tracking. In this chapter, we show how to augment our point-based detector (Chapter 4) into a tracker for unified object recognition and tracking.

In early computer vision, tracking was commonly phrased as following interest points through space and time [167, 191]. Early trackers were simple, fast, and reasonably robust. However, they were liable to fail in the absence of strong low-level cues such as corners and intensity peaks. With the advent of high-performing object detection models [48, 153], a powerful alternative emerged: tracking-by-detection (or more precisely, tracking-after-detection) [8, 188, 220]. These models rely on a given accurate recognition to identify objects and then link them up through time in a separate stage. Tracking-by-detection leverages the power of deep-learning-based object detectors and is currently the dominant tracking paradigm. Yet the best-performing object trackers are not without drawbacks. Many rely on slow and complex

[1]The work in this chapter was originally published as: Tracking Objects as Points, Xingyi Zhou, Vladlen Koltun, Philipp Krähenbühl, in ECCV 2020. Code is available at https://github.com/xingyizhou/CenterTrack.

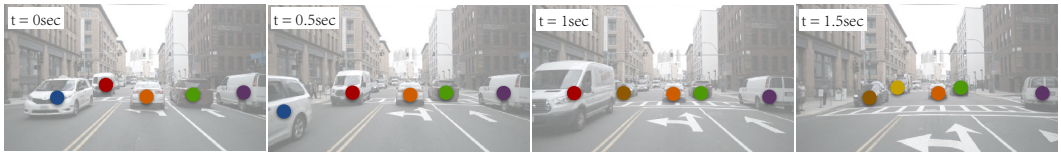


Figure 5.1: We track objects by tracking their centers. We learn a 2D offset between two adjacent frames and associate them based on center distance.

association strategies to link detected boxes through time [67, 188, 217, 220]. Recent work on *simultaneous detection and tracking* [7, 47] has made progress in alleviating some of this complexity. Here, we show how combining ideas from point-based tracking and simultaneous detection and tracking further simplifies tracking.

We present a point-based framework for joint detection and tracking, referred to as CenterTrack. Each object is represented by a single point at the center of its bounding box. This center point is then tracked through time (Figure 5.1). Specifically, we adopt the recent CenterNet detector to localize object centers [258]. We condition the detector on two consecutive frames, as well as a heatmap of prior tracklets, represented as points. We train the detector to also output an offset vector from the current object center to its center in the previous frame. We learn this offset as an attribute of the center point at little additional computational cost. A greedy matching, based solely on the distance between this predicted offset and the detected center point in the previous frame, suffices for object association. The tracker is end-to-end trainable and differentiable.

Tracking objects as points simplifies two key components of the track-

ing pipeline. First, it simplifies tracking-conditioned detection. If each object in past frames is represented by a single point, a constellation of objects can be represented by a heatmap of points [18]. Our tracking-conditioned detector directly ingests this heatmap and reasons about all objects jointly when associating them across frames. Second, point-based tracking simplifies object association across time. A simple displacement prediction, akin to sparse optical flow, allows objects in different frames to be linked. This displacement prediction is conditioned on prior detections. It learns to jointly detect objects in the current frame and associate them to prior detections.

While the overall idea is simple, subtle details matter in making this work. Tracked objects in consecutive frames are highly correlated. With the previous-frame heatmap given as input, CenterTrack could easily learn to repeat the predictions from the preceding frame, and thus refuse to track without incurring a large training error. We prevent this through an aggressive data-augmentation scheme during training. In fact, our data augmentation is aggressive enough for the model to learn to track objects from *static images*. That is, CenterTrack can be successfully trained on static image datasets (with "hallucinated" motion), with no real video input.

CenterTrack is purely local. It only associates objects in adjacent frames, without reinitializing lost long-range tracks. It trades the ability to reconnect long-range tracks for simplicity, speed, and high accuracy in the local regime. Our experiments indicate that this trade-off is well worth it. CenterTrack outperforms complex tracking-by-detection strategies on the MOT [126]

and KITTI [55] tracking benchmarks. We further apply the approach to monocular 3D object tracking on the nuScenes dataset [13]. Our monocular tracker achieves 28.3% AMOTA@0.2, outperforming the monocular baseline by a factor of 3, while running at 22 FPS. It can be trained on labelled video sequences, if available, or on static images with data augmentation.

5.1 Approach

We approach tracking from a local perspective. When an object leaves the frame or is occluded and reappears, it is assigned a new identity. We thus treat tracking as the problem of propagating detection identities across *consecutive* frames, without re-establishing associations across temporal gaps.

At time t, we are given an image of the current frame $I^{(t)} \in \mathbb{R}^{W \times H \times 3}$ and the previous frame $I^{(t-1)} \in \mathbb{R}^{W \times H \times 3}$, as well as the tracked objects in the previous frame $T^{(t-1)} = \{b_0^{(t-1)}, b_1^{(t-1)}, \ldots\}_i$. Each object $b = (\vec{p}, \vec{s}, w, id)$ is described by its center location $\vec{p} \in \mathbb{R}^2$, size $\vec{s} \in \mathbb{R}^2$, detection confidence $w \in [0, 1]$, and unique identity $id \in \mathbb{I}$. Our aim is to detect and track objects $T^{(t)} = \{b_0^{(t)}, b_1^{(t)}, \ldots\}$ in the current frame t, and assign objects that appear in both frames a consistent id.

There are two main challenges here. The first is finding all objects in every frame – including occluded ones. The second challenge is associating these objects through time. We address both via a single deep network, trained end-to-end. Section 5.1.1 describes a tracking-conditioned detector that leverages tracked detections from the previous frame to improve detection in the

Figure 5.2: Illustration of our framework. The network takes the current frame, the previous frame, and a heatmap rendered from tracked object centers as inputs, and produces a center detection heatmap for the current frame, the bounding box size map, and an offset map. At test time, object sizes and offsets are extracted from peaks in the heatmap.

current frame. Section 5.1.2 then presents a simple offset prediction scheme that is able to link detections through time. Finally, Sections 5.1.3 and 5.1.4 show how to train this detector from video or static image data.

5.1.1 Tracking-conditioned detection

As an object detector, CenterNet already infers most of the required information for tracking: object locations $\hat{\vec{p}}$, their size $\hat{\vec{s}} = \hat{S}_{\hat{\vec{p}}}$, and a confidence measure $\hat{w} = \hat{Y}_{\hat{\vec{p}}}$. However, it is unable to find objects that are not directly visible in the current frame, and the detected objects may not be temporally coherent. One natural way to increase temporal coherence is to provide the detector with additional image inputs from past frames. In CenterTrack, we provide the detection network with two frames as input: the current frame $I^{(t)}$ and the prior frame $I^{(t-1)}$. This allows the network to estimate the change in the scene and potentially recover occluded objects at time t from visual evidence at time $t-1$.

CenterTrack also takes prior detections $\{\vec{p}_0^{(t-1)}, \vec{p}_1^{(t-1)}, \ldots\}$ as additional

input. How should these detections be represented in a form that is easily provided to a network? The point-based nature of our tracklets is helpful here. Since each detected object is represented by a single point, we can conveniently render all detections in a class-agnostic single-channel heatmap $H^{(t-1)} = \mathcal{R}(\{\vec{p}_0^{(t-1)}, \vec{p}_1^{(t-1)}, \ldots\})$, using the same Gaussian render function as in the training of point-based detectors. To reduce the propagation of false positive detections, we only render objects with a confidence score greater than a threshold τ. The architecture of CenterTrack is essentially identical to CenterNet, with four additional input channels. (See Figure 8.2.)

Tracking-conditioned detection provides a temporally coherent set of detected objects. However, it does not link these detections across time. In the next section, we show how to add one additional output to point-based detection to track objects through space and time.

5.1.2 Association through offsets

To associate detections through time, CenterTrack predicts a 2D displacement as two additional output channels $\hat{D}^{(t)} \in \mathbb{R}^{\frac{W}{R} \times \frac{H}{R} \times 2}$. For each detected object at location $\hat{\vec{p}}^{(t)}$, the displacement $\hat{\vec{d}}^{(t)} = \hat{D}^{(t)}_{\hat{\vec{p}}^{(t)}}$ captures the difference in location of the object in the current frame $\hat{\vec{p}}^{(t)}$ and the previous frame $\hat{\vec{p}}^{(t-1)}$: $\hat{\vec{d}}^{(t)} = \hat{\vec{p}}^{(t)} - \hat{\vec{p}}^{(t-1)}$. We learn this displacement using the same regression objective as size or location refinement:

$$L_{off} = \frac{1}{N} \sum_{i=1}^{N} \left| \hat{D}_{\vec{p}_i^{(t)}} - (\vec{p}_i^{(t-1)} - \vec{p}_i^{(t)}) \right|, \tag{5.1}$$

where $\bar{p}_i^{(t-1)}$ and $\bar{p}_i^{(t)}$ are tracked ground-truth objects. Figure 8.2 shows an example of this offset prediction.

With a sufficiently good offset prediction, a simple greedy matching algorithm can associate objects across time. For each detection at position $\hat{p}$, we greedily associate it with the closest unmatched prior detection at position $\hat{p} - \hat{D}_{\hat{p}}$, in descending order of confidence $\hat{w}$. If there is no unmatched prior detection within a radius κ, we spawn a new tracklet. We define κ as the geometric mean of the width and height of the predicted bounding box for each tracklet. The simplicity of this greedy matching algorithm again highlights the advantages of tracking objects as points. A simple displacement prediction is sufficient to link objects across time. There is no need for a complicated distance metric or graph matching.

5.1.3 Training on video data

CenterTrack is first and foremost an object detector, and trained as such. The architectural changed from CenterNet to CenterTrack are minor: four additional input channels and two output channels. This allows us to fine-tune CenterTrack directly from a pretrained CenterNet detector [258]. We copy all weights related to the current detection pipeline. All weights corresponding to additional inputs or outputs are initialized randomly. We follow the CenterNet training protocol and train all predictions as multi-task learning. We use the same training objective with the addition of offset regression L_{off}.

The main challenge in training CenterTrack comes in producing a realistic tracklet heatmap $H^{(t-1)}$. At inference time, this tracklet heatmap can contain an arbitrary number of missing tracklets, wrongly localized objects, or even false positives. These errors are not present in ground-truth tracklets $\{\vec{p}_0^{(t-1)}, \vec{p}_1^{(t-1)}, \ldots\}$ provided during training. We instead simulate this test-time error during training. Specifically, we simulate three types of error. First, we locally jitter each tracklet $\vec{p}^{(t-1)}$ from the prior frame by adding Gaussian noise to each center. That is, we render $p'_i = (x_i + r \times \lambda_{jt} \times w_i, y_i + r \times \lambda_{jt} \times h_i)$, where r is sampled from a Gaussian distribution. We use $\lambda_{jt} = 0.05$ in all experiments. Second, we randomly add false positives near ground-truth object locations by rendering a spurious noisy peak p'_i with probability λ_{fp}. Third, we simulate false negatives by randomly removing detections with probability λ_{fn}. λ_{fp} and λ_{fn} are set according to the statistics of our baseline model. These three augmentations are sufficient to train a robust tracking-conditioned object detector.

In practice, $I^{(t-1)}$ does not need to be the immediately preceding frame from time $t-1$. It can be a different frame from the same video sequence. In our experiments, we randomly sample frames near t to avoid overfitting to the framerate. Specifically, we sample from all frames k where $|k-t| < M_f$, where $M_f = 3$ is a hyperparameter.

5.1.4 Training on static image data

Without labeled video data, CenterTrack does not have access to a prior frame $I^{(t-1)}$ or tracked detections $\{\vec{p}_0^{(t-1)}, \vec{p}_1^{(t-1)}, \ldots\}$. However, we can simulate tracking on standard detection benchmarks, given only single images $I^{(t)}$ and detections $\{\vec{p}_0^{(t)}, \vec{p}_1^{(t)}, \ldots\}$. The idea is simple: we simulate the previous frame by randomly scaling and translating the current frame. As our experiments will demonstrate, this is surprisingly effective.

5.1.5 End-to-end 3D object tracking

To perform monocular 3D tracking, we adopt the monocular 3D detection form of CenterNet [258]. Specifically, we train output heads to predict object depth, rotation (encoded as an 8-dimensional vector [67]), and 3D extent. Since the projection of the center of the 3D bounding box may not align with the center of the object's 2D bounding box (due to perspective projection), we also predict a 2D-to-3D center offset. Further details are provided in the supplement.

5.2 Experiments

We evaluate 2D multi-object tracking on the MOT17 [126] and KITTI [55] tracking benchmarks. We also evaluate monocular 3D tracking on the nuScenes dataset [13].

5.2.1 Datasets and evaluation metrics

MOT. MOT17 contains 7 training sequences and 7 test sequences [126], The videos were captured by stationary cameras mounted in high-density scenes with heavy occlusion. Only pedestrians are annotated and evaluated. The video framerate is 25-30 FPS. The MOT dataset does not provide an official validation split. For ablation experiments, we split each training sequence into two halves, and use the first half frames for training and the second for validation. Our main results are reported on the test set.

KITTI. The KITTI tracking benchmark consists of 21 training sequences and 29 test sequences [55]. They are collected by a camera mounted on a car moving through traffic. The dataset provides 2D bounding box annotations for cars, pedestrians, and cyclists, but only cars are evaluated. Videos are captured at 10 FPS and contain large inter-frame motions. KITTI does not provide detections, and all entries use private detection. We again split all training sequences into halves for training and validation.

nuScenes. nuScenes is a newly released large-scale driving dataset with 7 object classes annotated for tracking [13]. It contains 700 training sequences, 150 validation sequences, and 150 test sequences. Each sequence contains roughly 40 frames at 2 FPS with 6 slightly overlapping images in a panoramic 360° view, resulting in 168k training, 36k validation, and 36k test images. The videos are sampled at 12 FPS, but frames are only annotated and evaluated

at 2 FPS. All baselines and CenterTrack only use keyframes for training and evaluation. Due to the low framerate, the inter-frame motion is significant.

Evaluation metrics. We use the official evaluation metrics in each dataset. The common metric is multi-object tracking accuracy [96, 176]: $MOTA = 1 - \frac{\sum_t (FP_t + FN_t + IDSW_t)}{\sum_t GT_t}$, where GT_t, FP_t, FN_t, and $IDSW_t$ are the number of ground-truth bounding boxes, false positives, false negatives, and identity switches in frame t, respectively. MOTA does not rank tracklets according to confidence and is sensitive to the task-dependent output threshold θ [209]. The thresholds we use are listed in Section 5.2.2. The interplay between output threshold and true positive criteria matters. For 2D tracking [55, 126], > 0.5 bounding box IoU is a the true positive. For 3D tracking [13], bounding box center distance $< 2m$ on the ground plane is the criterion for a true positive. When objects are successfully detected, but not tracked, they are identified as an identity switch (IDSW). The IDF1 metric measures the minimal cost change from predicted ids to the correct ids. In our ablation studies, we report false positve rate (FP) $\frac{\sum_t FP_t}{\sum_t GT_t}$, false negative rate (FN) $\frac{\sum_t FN_t}{\sum_t GT_t}$, and identity switches (IDSW) $\frac{\sum_t IDSW_t}{\sum_t GT_t}$ separately. In comparisons with other methods, we report the absolute numbers following the dataset convention [55,126]. We also report the Most Tracked ratio (MT) for the ratio of most tracked ($> 80\%$ time) objects and Most Lost ratio (ML) for most lost ($< 20\%$ time) objects [176].

nuScenes adopts a more robust metric, AMOTA, which is a weighted

average of MOTA across different output thresholds. Specifically,

$$AMOTA = \frac{1}{n-1} \sum_{r \in \{\frac{1}{n-1}, \frac{2}{n-1}, \cdots, 1\}} MOTA_r$$
$$MOTA_r = max(0, 1 - \alpha \frac{IDSW_r + FP_r + FN_r - (1-r) \times P}{r \times P})$$

where r is a fixed recall threshold, $P = \sum_t GT_t$ is the total number of annotated objects among all frames, and $FP_r = \sum_t FP_{r,t}$ is the total number of false positive samples only considering the top confident samples that achieve the recall threshold r. The hyperparameters $n = 40$ and $\alpha = 0.2$ (AMOTA@0.2), or $\alpha = 1$ (AMOTA@1) are set by the benchmark organizers. The overall AMOTA is the average AMOTA among all 7 categories.

5.2.2 Implementation details

Our implementation is based on CenterNet [258]. We use DLA [233] as the network backbone, optimized with Adam [84] with learning rate $1.25e-4$ and batchsize 32. Data augmentations include random horizontal flipping, random resized cropping, and color jittering. For all experiments, we train the networks for 70 epochs. The learning rate is dropped by a factor of 10 at the 60th epoch. We test the runtime on a machine with an Intel Core i7-8086K CPU and a Titan Xp GPU. The runtimes depend on the number of objects for rendering and the input resolution in each dataset.

The MOT dataset [126] annotates each pedestrian as an amodal bounding box. That is, the bounding box always covers the whole body even when part of the object is out of the frame. In contrast, CenterNet [258] requires

the center of each inferred bounding box to be within the frame. To handle this, we separately predict the visible and amodal bounding boxes [189]. We follow prior works [157, 173, 188, 232, 248] to pretrain on external data. We train our network on the CrowdHuman [161] dataset, using the static image training described in Section 5.1.4.

The default input resolution for MOT images is 1920×1080. We resize and pad the images to 960×544. We use random false positive ratio $\lambda_{fp} = 0.1$ and random false negative ratio $\lambda_{fn} = 0.4$. We only output tracklets that have a confidence of $\theta = 0.4$ or higher, and set the heatmap rendering threshold to $\tau = 0.5$.

For KITTI [55], we keep the original input resolution 1280×384 in training and testing. The hyperparameters are set at $\lambda_{fp} = 0.1$ and $\lambda_{fn} = 0.2$, with output threshold $\theta = 0.4$ and rendering threshold $\tau = 0.4$. We fine-tune our KITTI model from a nuScenes tracking model.

For nuScenes [13], we use input resolution 800×448. We set $\lambda_{fp} = 0.1$ and $\lambda_{fn} = 0.4$, and use output threshold $\theta = 0.1$ and rendering threshold $\tau = 0.1$. We first train our nuScenes model for 140 epochs for just 3D detection [258] and then fine-tune for 70 epochs for 3D tracking. Note that nuScenes evaluation is done per 360 panorama, not per image. We naively fuse all outputs from the 6 cameras together, without handling duplicate detections at the intersection of views [168].

Track rebirth. Following common practice [7, 248], we keep unmatched tracks "inactive" until they remain undetected for K consecutive frames. Inactive tracks can be matched to detections and regain their ID, but not appear in the prior heatmap or output. The tracker stays online. Rebirth only matters for the MOT test set, where we use $K = 32$. For all other experiments, we found rebirth not to be required ($K = 0$).

5.2.3 Public detection

The MOT17 challenge only supports public detection. That is, participants are asked to use the provided detections. Public detection is meant to test a tracker's ability to associate objects, irrespective of its ability to detect objects. Our method operates in the private detection mode by default. For the MOT challenge we created a public-detection version of CenterTrack that uses the externally provided (public) detections and is thus fairly compared to other participants in the challenge. This shows that the advantages of CenterTrack are not due to the accuracy of the detections but are due to the tracking framework itself.

Note that refining and rescoring the given bounding boxes is allowed and is commonly used by participants in the challenge [7, 81, 118]. Following Tracktor [7], we keep the bounding boxes that are close to an existing bounding box in the previous frame. We only initialize a new trajectory if it is near a public detection. All bounding boxes in our results are either near a public detection in the current frame or near a tracked box in the previous frame.

	Time(ms)	MOTA ↑	IDF1 ↑	MT ↑	ML ↓	FP ↓	FN ↓	IDSW ↓
Tracktor17 [7]	666+D	53.5	52.3	19.5	36.6	12201	248047	2072
LSST17 [50]	666+D	54.7	**62.3**	20.4	40.1	26091	228434	**1243**
Tracktor v2 [7]	666+D	56.5	55.1	21.1	35.3	**8866**	235449	3763
GMOT	167+D	55.4	57.9	22.7	34.7	20608	229511	1403
Ours (Public)	**57+D**	**61.5**	59.6	**26.4**	**31.9**	14076	**200672**	2583
Ours (Private)	57	67.8	64.7	34.6	24.6	18498	160332	3039

Table 5.1: Evaluation on the MOT17 test sets (top: public detection; bottom: private detection). We compare to published entries on the leaderboard. The runtime is calculated from the HZ column on the leaderboard. +D means detection time, which is usually $>$ 100ms [153].

We use this public-detection configuration of CenterTrack for MOT17 test set evaluation and use the private-detection setting in our ablation studies.

5.2.4 Main results

All three datasets – MOT17 [126], KITTI [55], and nuScenes [13] – host test servers with hidden annotations and leaderboards. We compare to all published results on these leaderboards. The numbers were accessed on Mar. 5th, 2020. We retrain CenterTrack on the full training set with the same hyperparameters in the ablation experiments.

Table 5.1 lists the results on the MOT17 challenge. We use our public configuration in Section 5.2.3 and do not pretrain on CrowdHuman [161]. CenterTrack significantly outperforms the prior state of the art even when restricted to the public-detection configuration. For example CenterTrack improves MOTA by 5 points (an 8.6% relative improvement) over Tracktor v2 [7].

The public detection setting ensures that all methods build on the same

	Time(ms)	MOTA ↑	MOTP ↑	MT ↑	ML ↓	IDSW ↓	FRAG ↓
AB3D [209]	4+D	83.84	85.24	66.92	11.38	**9**	**224**
BeyondPixel [163]	300+D	84.24	85.73	73.23	2.77	468	944
3DT [67]	30+D	84.52	85.64	73.38	2.77	377	847
mmMOT [245]	10+D	84.77	85.21	73.23	2.77	284	753
MOTSFusion [121]	440+D	84.83	85.21	3.08	2.77	275	759
MASS [79]	10+D	85.04	**85.53**	74.31	2.77	301	744
Ours	82	**89.44**	85.05	**82.31**	**2.31**	116	334

Table 5.2: Evaluation on the KITTI test set. We compare to all published entries on the leaderboard. Runtimes are from the leaderboard. +D means detection time.

underlying detector. Our gains come from two sources. Firstly, the heatmap input makes our tracker better preserve tracklets from the previous frame, which results in a much lower rate of false negatives. And second, our simple learned offset is effective. (See Section 5.2.6 for more analysis.) For reference, we also included a private detection version, where CenterTrack simultaneously detects and tracks objects (Table 5.1, bottom). It further improves the MOTA to 67.3%, and runs at 17 FPS end-to-end (including detection).

For IDF1 and id-switch, our local model is not as strong as offline methods such as LSST17 [50], but is better than other online methods [7]. We believe that there is an exciting avenue for future work in combining local trackers (such as our work) with stronger offline long-range models (such as SORT [8], LMP [188], and other ReID-based trackers [220, 232]).

On KITTI [55], we submitted our best-performing model with flip testing [258]. The model runs at 82ms and yields 89.44% MOTA, outperforming all published work (Table 5.2). Note that our model without flip testing runs

	Time(ms)	AMOTA@0.2 ↑	AMOTA@1 ↑	AMOTP ↓
Mapillary [168]+AB3D [209]	-	6.9	1.8	1.8
Ours	45	**27.8**	**4.6**	**1.5**

Table 5.3: Evaluation on the nuScenes test set. We compare to the official monocular 3D tracking baseline, which applies a state-of-the-art 3D tracker [209]. We list the average AMOTA@0.2, AMOTA@1, and AMOTP over all 7 categories.

at 45ms with 88.7% MOTA on the validation set (vs. 89.63% with flip testing on the validation set). We avoid submitting to the test server multiple times following their test policy. The results again indicate that CenterTrack performs competitively with more complex methods.

On nuScenes [13], our monocular tracking method achieves an AMOTA@0.2 of 28.3% and an AMOTA@1 of 4.6%, outperforming the monocular baseline [168, 209] by a large margin. There are two main reasons. Firstly, we use a stronger and faster 3D detector [258] More importantly, as shown in Table 5.6, the Kalman-filter-based 3D tracking baseline relies on hand-crafted motion rules [209], which are less effective in low-framerate regimes. Our method learns object motion from data and is much more stable at low framerates.

5.2.5 Ablation studies

We first ablate our two main technical contributions: tracking-conditioned detection (Section 5.1.1) and offset prediction (Section 5.1.2) on all three datasets. Specifically, we compare our full framework with three baselines.

	MOT17				KITTI				nuScenes	
	MOTA↑	FP↓	FN↓	IDSW↓	MOTA↑	FP↓	FN↓	IDSW↓	AMOTA@0.2↑	AMOTA@1↑
detection only	63.6	3.5%	30.3%	2.5 %	84.3	4.3%	9.8%	1.5%	18.1	3.4
w/o offset	65.8	**4.5%**	**28.4%**	1.3%	87.1	**5.4%**	**5.8%**	1.6%	17.8	3.6
w/o heatmap	63.9	3.5%	30.3%	2.3%	85.4	4.3%	9.8%	0.4%	26.5	5.9
Ours	**66.1**	**4.5%**	**28.4%**	**1.0%**	**88.7**	**5.4%**	**5.8%**	**0.1%**	**28.3**	**6.8**

Table 5.4: Ablation study on MOT17, KITTI, and nuScenes. All results are on validation sets (Section 5.2.1). For each dataset, we report the corresponding official metrics. ↑ indicates that higher is better, ↓ indicates that lower is better.

Detection only runs a CenterNet detector at each individual frame and associates their identity only based on 2D center distance. This model does not use video data, but still uses two input images.

Without offset uses just tracking-conditioned prediction with a predicted offset of zero. Every object is again associated to its closest object in the previous frame.

Without heatmap predicts the center offset between frames and uses the updated center distance as the association metric, but the prior heatmap is not provided. The offset-based greedy association is used.

Table 5.4 shows the results. On all datasets, our full CenterTrack model performs significantly better than the baselines. Tracking-conditioned detection yields $\sim 2\%$ MOTA improvement on MOT and $\sim 3\%$ MOTA improvement on KITTI, with or without offset prediction. It produces more false positives

but fewer false negatives. This is because with the heatmap prior, the network tends to predict more objects around the previous peaks, which are sometimes misleading. The merits of the heatmap outweigh the limitations and improve MOTA overall. Using the prior heatmap also significantly reduces IDSW on both datasets, indicating that the heatmap stabilizes detection.

Tracking offset prediction gives a huge boost on nuScenes and reduces IDSW consistently in MOT and KITTI. The effectiveness of the tracking offset appears to be related to the video framerate. When the framerate is high, motion between frames is small, and a zero offset is often a reasonable starting point for association. When framerate is low, as in the nuScenes dataset, motion between frames is large and static object association is considerably less effective. Our offset prediction scheme helps deal with such large inter-frame motion. Next, we ablate other components on MOT17.

Training with noisy heatmap. The 2nd row in Table 5.5 shows the importance of injecting noise into heatmaps during training (Section 5.1.3). Without noise injection, the model fails to generalize and yields dramatically lower accuracy. In particular, this model has a large false negative rate. One reason is that in the first frame, the input heatmap is empty. This model had a hard time discovering new objects that were not indicated in the prior heatmap.

Training on static images. We train a version of our model on static images only, as described in Section 5.1.4. The results are shown in Table 5.5

	MOTA ↑	IDF1 ↑	MT ↑	ML ↓	FP ↓	FN ↓	IDSW ↓
Ours	66.1	64.2	41.3	21.2	4.5%	28.4%	1.0%
w.o. noisy hm	34.4	46.2	26.3	42.2	7.3%	57.4%	0.9%
Static image	66.1	65.4	41.6	19.2	5.4%	27.5%	1.0%
w. Hungarian	66.1	61.0	40.7	20.9	4.5%	28.3%	1.0%
w. rebirth	66.2	69.4	39.5	22.1	3.9%	29.5%	0.4%

Table 5.5: Additional experiments on the MOT17 validation set. From top to bottom: our model, our model trained without simulating heatmap noise, our model trained on static images only, our model with Hungarian matching, and our model with track rebirth.

(3rd row, 'Static image'). As reported in this table, training on static images gives the same performance as training on videos on the MOT dataset. Separately, we observed that training on static images is less effective on nuScenes, where framerate is low.

Matching algorithm. We use a simple greedy matching algorithm based on the detection score, while most other trackers use the Hungarian algorithm. We show the performance of CenterTrack with Hungarian matching in the 4th row of Table 5.5. It does not improve performance. We choose greedy matching for simplicity.

Track rebirth. We show CenterTrack with track rebirth (K=32) in the last row of Table 5.5. While the MOTA performance keeps similar, it significantly increases IDF1 and reduces ID switch. We use this setting for our MOT test set submission. For other datasets and evaluation metrics no rebirth was required ($K = 0$).

	MOT17				KITTI				nuScenes	
	MOTA↑	FP↓	FN↓	IDSW↓	MOTA↑	FP↓	FN↓	IDSW↓	AMOTA@0.2↑	AMOTA@1↑
no motion	65.8	**4.5%**	**28.4%**	1.3%	87.1	**5.4%**	**5.8%**	1.6%	17.8	3.6
Kalman filter	**66.1**	**4.5%**	**28.4%**	**1.0%**	87.9	**5.4%**	**5.8%**	0.9%	18.3	3.8
optical flow	**66.1**	**4.5%**	**28.4%**	**1.0%**	88.4	**5.4%**	**5.8%**	0.4%	26.6	6.2
ours	**66.1**	**4.5%**	**28.4%**	**1.0%**	**88.7**	**5.4%**	**5.8%**	**0.1%**	**28.3**	**6.8**

Table 5.6: Comparing different motion models on MOT17, KITTI, and nuScenes. All results are on validation sets (Section 5.2.1). All experiments on the same dataset are from the same model.

5.2.6 Comparison to alternative motion models

Our offset prediction is able to estimate object motion, but also performs a simple association, as current objects are linked to prior detections, which CenterTrack receives as one of its inputs. To verify the effectiveness of our learned association, we replace our offset prediction with three alternative motion models:

No motion. We set the offset to zeros. It is copied from Table 5.4 for reference only.

Kalman filter. The Kalman filter predicts each object's future state through an explicit motion model estimated from its history. It is the most widely used motion model in traditional real-time trackers [8, 209, 211]. We use the popular public implementation from SORT [8] [2].

[2] https://github.com/abewley/sort

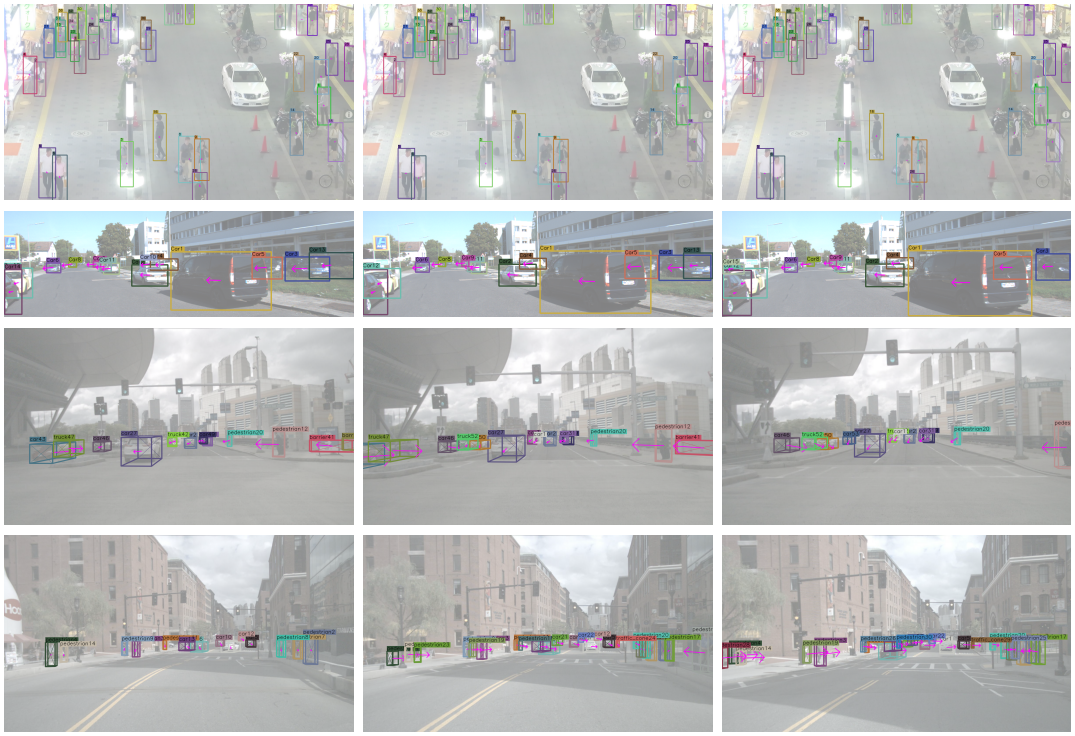


Figure 5.3: Qualitative results on MOT (1st row), KITTI (2nd row), and nuScenes (3rd and 4th rows). Each row shows three consecutive frames. We show the predicted tracking offset in arrow. Tracks are coded by color. Best viewed on the screen.

Optical flow. As an alternative motion model, we use FlowNet2 [71]. The model was trained to estimate dense pixel motion for all objects in a scene. We run the strongest officially released FlowNet2 model ($\sim$ 150ms / image pair), and replace our learned offset with the predicted optical flow at each predicted object center [3].

The results are shown in Table 5.6. All models use the exact same de-

[3] https://github.com/NVIDIA/flownet2-pytorch

tector. On the high-framerate MOT17 dataset, any motion model suffices, and even no motion model at all performs competitively. On KITTI and nuScenes, where the intra-frame motions are non-trivial, the hand-crafted motion rule of the Kalman filter performs significantly worse, and even the performance of optical flow degrades. This emphasizes that our offset model does more than just motion estimation. CenterTrack is conditioned on prior detections and can learn to snap offset predictions to exactly those prior detections. Our training procedure strongly encourages this through heavy data augmentation.

Summary We presented an end-to-end simultaneous object detection and tracking framework. Our method takes two frames and a prior heatmap as input, and produces detection and tracking offsets for the current frame. Our tracker is purely local and associates objects greedily through time. It runs online (no knowledge of future frames) and in real time, and sets a new state of the art on the challenging MOT17, KITTI, and nuScenes 3D tracking benchmarks.

Chapter 6

GTR: Unified global detection and tracking

[1] In the previous chapter, we approach tracking by linking objects in adjacent frames. While this is effective for short-term tracking, it often fails challenging cases like long-term occlusion due to the lack of global information. Also, object recognition is still done per frame. In this chapter, we propose a global tracker, Global Tracking Transformers (GTR), that does object classification after tracking. This way, object recognition is done globally in a unified way across time.

In recent years, *tracking-by-detection* has emerged as the dominant tracking paradigm, powered by advances in deep learning and object detection. Tracking-by-detection reduces tracking to two steps: detection and association. First, an object detector independently finds potential objects in each frame of the video stream. Second, an association step links detections through time. Greedy trackers [7,8,207,210,220,247] primarily consider pairwise associations. They maintain a status of each trajectory based on location and/or identity features, and associate current-frame detections with each trajectory based

[1]The work in this chapter was originally published as: Global Tracking Transformers, Xingyi Zhou, Tianwei Yin, Vladlen Koltun, Philipp Krähenbühl, in CVPR 2022. The code is available at https://github.com/xingyizhou/GTR.

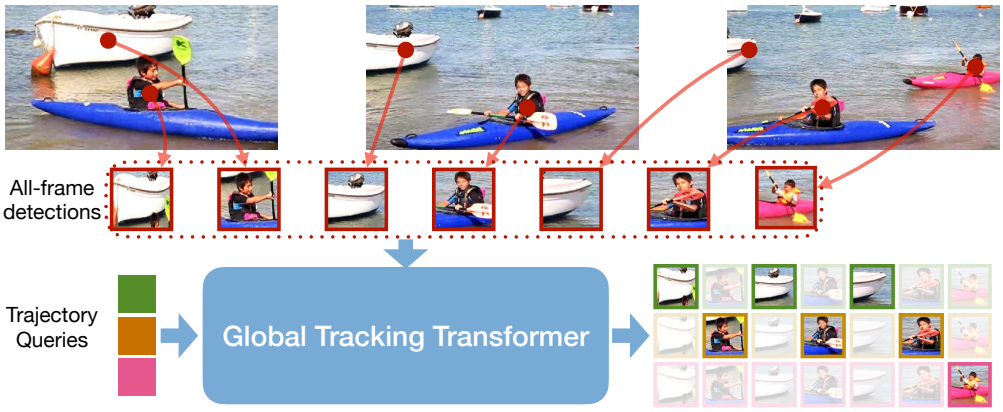


Figure 6.1: Overview of our joint detection and tracking framework. An object detector first independently detects objects in all frames. Object features are concatenated and fed into the encoder of our global tracking transformer (GTR). The GTR additionally takes trajectory queries as decoder input, and produces association scores between each query and object. The association matrix links objects for each query. During testing, the trajectory queries are object features in the last frame.

on its last visible status. This pairwise association is efficient, but lacks an explicit model of trajectories as a whole, and sometimes struggles with heavy occlusion or strong appearance change. Global trackers [6, 12, 188, 234, 241] run offline graph-based combinatorial optimization over pairwise associations. They can resolve inconsistently grouped detections and are more robust, but can be slow and are usually detached from the detector.

In this chapter, we show how to represent global tracking as a few layers in a deep network (Figure 6.1). Our network directly outputs trajectories and thus sidesteps both pairwise association and graph-based optimization. We show that two-stage detectors [63, 153] can be augmented by transformer layers to turn them into joint detectors and trackers. Our global tracking

transformer (GTR) encodes detections from multiple consecutive frames, and uses *trajectory queries* to group them into trajectories. The queries are detection features from a single frame after non-maximum suppression, and are transformed by the GTR into trajectories. Each trajectory query produces a single global trajectory by assigning to it a detection from each frame using a softmax distribution. The outputs of our model are thus detections and their associations through time. During training, we explicitly supervise the output of our global tracking transformer using ground-truth trajectories and their image-level bounding boxes. During inference, we run the GTR in a sliding window manner with a moderate temporal size of 32 frames, and link trajectories between windows online. The model is end-to-end differentiable within the temporal window.

Our framework is motivated by the recent success of transformer models [200] in computer vision in general [38, 115, 195, 249] and in object detection in particular [17, 206]. The cross-attention structure between queries and encoder features mines similarities between objects and naturally fits the association objective in multi-object tracking. We perform cross-attention between trajectory queries and object features, and explicitly supervise it to produce a query-to-detections assignment. Each assignment directly corresponds to a global trajectory. Unlike transformer-based detectors [17, 125, 179, 206] that learn queries as fixed parameters, our queries come from existing detections and adapt with the image content. Furthermore, our transformer operates on detected objects rather than raw pixels [17]. This enables us to take full

advantage of well-developed two-stage object detectors [63, 256].

Our framework is end-to-end trainable, and easily integrates with state-of-the-art object detectors. On the challenging large-scale TAO dataset, our model reaches 20.1 tracking mAP on the test set, significantly outperforming published work, which achieved 12.4 tracking mAP [32, 134]. On the MOT17 [126] benchmark, our entry achieves competitive 74.1 MOTA and 59.0 HOTA, outperforming concurrent transformer-based trackers [125, 222, 238], and outperforming most association-based trackers in either HOTA or MOTA. Code will be released under a permissive license upon acceptance.

6.1 Approach

Out global tracking transformer (GTR) associates objects in a probabilistic and differentiable manner. It links proposals p_i^t in each frame I^t to a set of trajectory queries q_k. Each trajectory query q_k produces an object association score vector $g \in \mathbb{R}^N$ over proposals from all frames. This association score vector then yields a per-frame proposal-level association $\alpha_k^t \in \{\emptyset, 1, \ldots, N_t\}$, where $\alpha_k^t = \emptyset$ indicates no association and N_t is the number of detected proposals in frame I^t. The combination of associations then produces a trajectory $\vec{\tau}_k$. Figure 6.1 provides an overview. The association step is differentiable and can be jointly trained with the underlying object detector.

6.1.1 Tracking transformers

Let $p_1^t, \ldots, p_{N_t}^t$ be a set of high-confidence object proposals for image I^t. Let $B^t = \{b_1^t, \ldots, b_{N_t}^t\}$ be their corresponding bounding boxes. Let $f_i^t \in \mathbb{R}^D$ be the D-dimentional features extracted from boxes b_i^t. For convenience let $F^t = \{f_1^t, \ldots, f_{N_t}^t\}$ be the set of all detection features of image I^t, and $F = F^1 \cup \ldots \cup F^T$ be the set of all features through time. The collection of all proposal features $F \in \mathbb{R}^{N \times D}$ is the input to our tracking transformer, where $N = \sum_t^T N_t$ is the total number of detections in all frames. The tracking transformer takes features F and a trajectory query $q_k \in \mathbb{R}^D$, and produces a trajectory-specific association score $g(q_k, F) \in \mathbb{R}^N$.

Formally, let $g_i^t(q_k, F) \in \mathbb{R}$ be the score of the i-th object in the t-th frame. A special output token $g_\emptyset^t(q_k, F) = 0$ indicates no association at time t. The tracking transformer then predicts a distribution of associations over all objects i in frame I^t for each trajectory k. We model this as an independent softmax activation for each time-step t:

$$P_A(\alpha^t = i | q_k, F) = \frac{\exp\left(g_i^t(q_k, F)\right)}{\sum_{j \in \{\emptyset, 1, \ldots N_t\}} \exp\left(g_j^t(q_k, F)\right)}. \tag{6.1}$$

Since a detector produces a single bounding box b_i^t for each proposal p_i^t, there is a one-to-one mapping between the association distribution P_A and a distribution P_t over bounding boxes for trajectory k at time t: $P_t(b|q_k, F) = \sum_{i=1}^{N_t} 1_{[b=b_i^t]} P_A(\alpha^t = i | q_k, F)$, where the indicator $1_{[\cdot]}$ assigns an output bounding box to each associated query. In practice, a detector's non-maximum suppression (NMS) ensures that there is also a unique mapping from P_t back

to P_A. The distribution over bounding boxes in turn leads to a distribution over entire trajectories $P_T(\vec{\tau}|q_k, F) = \prod_{t=1}^{T} P_t(\tau^t|q_k, F)$.

During training, we maximize the log-likelihood of the ground-truth trajectories. During inference, we use the likelihood to produce long-term tracks in an online manner.

6.1.2 Training

Given a set of ground-truth trajectories $\hat{\vec{\tau}}_1, \ldots, \hat{\vec{\tau}}_K$, our goal is to learn a tracking transformer that estimates P_A, and implicitly the trajectory distribution P_T. We jointly train the tracking transformer with detection by treating the transformer as an RoI head in a two-stage detector [153]. At each training iteration, we first obtain high-confidence proposals $b_1^t, \ldots, b_{N_t}^t$ and their corresponding features F_t after non-maximum suppression. We then maximize $\log P_T(\vec{\tau}|q_k, F)$ for each ground-truth trajectory $\vec{\tau}$. This is equivalent to maximizing $\log P_A(\alpha^t|q_k, F)$ after assigning $\vec{\tau}$ to a set of proposals. We follow object detection and use a simple intersection-over-union (IoU) assignment rule:

$$\hat{\alpha}_k^t = \begin{cases} \emptyset & \text{if } \tau_k^t = \emptyset \text{ or } \max_i IoU(b_i^t, \tau_k^t) \text{ ¡ } 0.5 \\ \text{argmax}_i IoU(b_i^t, \tau_k^t) & \text{otherwise} \end{cases}. \tag{6.2}$$

We use this assignment to both train the bounding box regression of the underlying two-stage detector, and our assignment likelihood P_A. However, this assignment likelihood further depends on the trajectory query q_k, which we define next.

Trajectory queries are key to our formulation. Each query q_k generates a trajectory. In prior work [17], object queries were learned as network parameters and fixed during inference. This makes queries image-agnostic and requires a near-exhaustive enumeration of them [117, 182, 269]. For objects this is feasible [17], as anchors [108] or proposals [180] showed. Trajectories, however, live in the exponentially larger space of potential moving objects than simple boxes, and hence require many more queries to cover that space. Furthermore, tracking datasets feature many fewer annotated instances, and learned trajectories easily overfit and remember the dataset.

We instead use object features f_i^t of the proposals p_i^t as the object queries. Specifically, let $\hat{\alpha}_k$ be the matched proposals for a ground-truth trajectory $\vec{\tau}_k$ according to Equation (6.2). Any feature $\{f^1_{\hat{\alpha}^1_k}, f^2_{\hat{\alpha}^2_k}, \ldots\}$ can serve as the trajectory query for trajectory $\vec{\tau}_k$. In practice, we use all object features F as queries and train the transformer T times for a sequence length of T. Any unmatched features f_i^t are used as background queries and supervised to produce $\emptyset$ for all frames. We allow multiple queries to produce the same trajectory, and do not require a one-to-one match [17]. During inference, we only use object features from one single frame as the queries to avoid duplicate outputs. All object features within a frame are different (after NMS) and hence produce different trajectories.

Training objective. The overall training objective combines the assignment in Equation (6.2) and trajectory queries to maximize the log-likelihood of each

trajectory under its assigned queries. For each trajectory τ_k we optimize the log-likelihood of its assignments $\hat{\alpha}_k$:

$$\ell_{asso}(F, \vec{\hat{\tau}}_k) = - \sum_{s \in \{1 \dots T | \hat{\alpha}_k^s \neq \emptyset\}} \sum_{t=1}^{T} \log P_A(\hat{\alpha}_k^t | F_{\hat{\alpha}_k^s}^s, F). \tag{6.3}$$

For any unassociated features, we produce empty trajectories:

$$\ell_{bg}(F) = - \sum_{s=1}^{T} \sum_{j : \nexists \hat{\alpha}_k^s = j} \sum_{t=1}^{T} \log P_A(\alpha^t = \emptyset | F_j^s, F). \tag{6.4}$$

The final loss simply combines the two terms:

$$L_{asso}(F, \{\vec{\hat{\tau}}_1, \dots, \vec{\hat{\tau}}_K\}) = \ell_{bg}(F) + \sum_{\vec{\hat{\tau}}_k} \ell_{asso}(F, \vec{\hat{\tau}}_k) \tag{6.5}$$

We train L_{asso} jointly with standard detection losses [256], including classification and bounding-box regression in both the proposal network and RoI heads.

6.1.3 Online Inference

During inference, we process the video stream online in a sliding-window manner with window size $T = 32$ and stride 1. For each individual frame t, we feed the image to the network before the tracking transformer and obtain N_t bounding boxes B^t and object features F^t. We keep a temporal history buffer of T frames, i.e., $B = \{B^{t-T+1}, \cdots, B^t\}$ and $F = \{F^{t-T+1}, \cdots, F^t\}$, and run the tracking transformer for each sliding window. We use object proposals from the current frame t as trajectory queries $q_k = F_k^t$ to produce N_t trajectories.

For the first frame, we initialize all detections as trajectories. For any subsequent frame, we link current predicted trajectories to existing tracks using the average assignment likelihood P_A as a distance metric. Since the current trajectories share up to $T-1$ boxes and features with past trajectories, the overlap can be quite large. We use a Hungarian algorithm to ensure that the mapping from current long-term trajectories to existing tracks is unique. If the average association score with any prior trajectory is lower than a threshold θ, we start a new track. Otherwise we append the underlying current detection (query) that generates the trajectory to the matched existing track.

6.1.4 Network architecture

The global tracking transformer takes a stack of object features $F \in \mathbb{R}^{N \times D}$ as the encoder input, a matrix of queries $Q \in \mathbb{R}^{M \times D}$ as the decoder input, and produces an association matrix $G \in \mathbb{R}^{M \times N}$ between queries and objects. The detailed structure of the tracking transformer is shown in Figure 6.2 (left). It follows DETR [17] but only uses a one-layer encoder and a one-layer decoder. Empirically, we observe that self-attention for queries and Layer Normalization [4] were not required. See Section 6.2.5 for an ablation. The resulting network structure is lightweight, with 10 linear layers in total. It runs in a fraction of the runtime of the backbone detector, even for hundreds of queries.

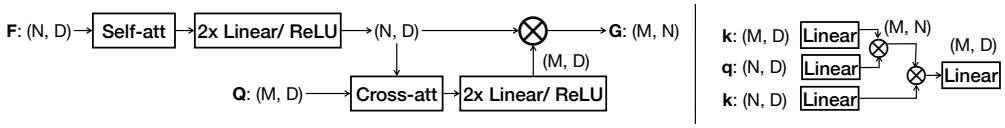


Figure 6.2: Left: detailed network architecture of GTR. Right: detailed structure of both self-att and cross-att blocks. We omit multi-head [200] in the figure for simplicity. For self-attention, $q = k = F$. For cross attention, $q = Q$, $k = F$. We list data dimensionalities in parentheses. $\bigotimes$ indicates matrix multiplication (transpose when needed).

6.1.5 Connection to embedding learning and ReID

Consider a variation of GTR with just a dot-product association score $g_i^t(q_k, F) = q_k \cdot F_i^t$. Further consider learning all trajectory queries $Q = \{q_1, \ldots, q_k\}$ as free parameters, one per training trajectory τ_k. In this variation, the softmax assignment in Equation (6.1) reduces to a classification problem. For each proposal, we classify it as a specific training instance or as background. This is exactly the objective of classification-based embedding learning in person-ReID [123], as used in ReID-based trackers [207, 247].

The two key differences between embedding learning and GTR are: first, our transformer does not assume any factorization of g_i^t and allows the model to reason about all boxes at once when computing associations. A dot-product-based ReID network on the other hand assumes that all boxes independently produce a compatible embedding. See Section 6.2.5 for an ablation of this transformer structure. Second, our trajectory queries are not learned. This allows our transformer to produce long-term associations in a single forward pass, while ReID-based trackers rely on a separate cosine-distance-based grouping step [207, 247].

6.2 Experiments

We evaluate our method on two tracking benchmarks: TAO [32] and MOT17 [126].

TAO [32] tracks a wide variety of objects. The images are adopted from 6 existing video datasets, including indoor, outdoor, and driving scenes. The dataset requires tracking objects with a large vocabulary of 488 classes in a long-tail setting. It contains 0.5k, 1k, and 1.5k videos for training, validation, and testing, respectively. Each video contains around 40 annotated frames at 1 annotated frame per second. There is significant motion between adjacent annotated frames. The training annotations are incomplete. We thus do not use the training set and solely train on LVIS [61] and use the TAO validation and test set for evaluation.

MOT [126] tracks pedestrians in crowd scenes. It contains 7 training sequences and 7 test sequences. The sequences contain 500 to 1500 frames, recorded and annotated at 25-30 FPS. We follow CenterTrack [255] and split each training sequence in half. We use the first half for training and the second half for validation. We perform ablation studies mainly on this validation set, and compare to other approaches on the official hidden test set. We evaluate under the private detection protocol.

6.2.1 Evaluation metrics

We first evaluate under the official metrics for each dataset. TAO [32] uses tracking mAP@0.5 as the official metric, which is based on standard object

detection mAP [109] but changes the 2D bounding box IoU to 3D temporal-spatial IoU between the predicted trajectory and the ground-truth trajectory. The overall tracking mAP is averaged across all classes. MOT [126] uses Multi-Object Tracking Accuracy (MOTA) as the official metric. MOTA $= 1 - \frac{\sum_t (FP_t + FN_t + IDSW_t)}{\sum_t GT_t}$, where GT_t is the number of ground truth objects in frame t, and FP_t, FN_t, and $IDSW_t$ measure the errors of false positives, false negatives, and ID switches, respectively.

As suggested by the MOT benchmark, we additionally report HOTA, a new tracking metric [122]. HOTA is defined as the geometric mean of detection accuracy (DetA) and association accuracy (AssA). Both DetA and AssA have the form $\frac{|TP|}{|TP|+|FN|+|FP|}$, with their respective true/false criteria. In our experiments, we mainly use AssA to access tracking performance.

6.2.2 Training and inference details

TAO training. Our implementation is based on detectron2 [216] (Apache License). For TAO [32] experiments, we use Res2Net [53] with deformable convolution [29] as the backbone. We adopt CenterNet2 [256] as the detector, which equips a CenterNet [258] proposal network with cascaded RoI heads [14]. Following the guideline of TAO dataset [32], we train the object detector on the combination of LVISv1 [61] and COCO [109]. We additionally incorporate a federated loss [256] to improve long-tail detection. We first train a single-frame detector. The training uses SGD with learning rate 0.04 and batch size 32 for 180K iterations (the 4× schedule [216]). We use training

resolution 896 × 896 following the scale-and-crop augmentation of EfficientDet [186]. The detector yields 37.1 mAP on the LVISv1 validation set and 27.3 mAP on the TAO validation set.

TAO only provides a small training set for tuning tracking hyperparameters, but not for training the tracker. We empirically observed that training on the TAO training set hurts detection performance, and overall does not yield good tracking accuracy. We find that training only on static image datasets [61] with data augmentation is sufficient for tracking. Our training strategy follows CenterTrack [255]. Specifically, we apply two different data augmentations to an image, and use them as the starting and ending frame of a video. We then interpolate the images and annotations linearly to generate a smooth video for training.

With the synthetic video, we fine-tune the network with the tracking transformer head end-to-end from the single-frame detector. Our fine-tuning protocol follows DETR [17] and uses the AdamW optimizer [120], multiplies the backbone learning rate by a factor of 0.1, and clamps the gradient norm at 0.1. We use a base learning rate of 0.0001. We generate video clips of length $T = 8$ and train with a batch size of 8 videos on 8 GPUs, resulting in an effective batch size of 64. We fine-tune the network for 22,500 iterations (a 2× schedule). The fine-tuning takes around 8 hours on 8 Quadro RTX 6000 GPUs.

MOT training. For our MOT model, we follow past works [247, 255] and use CenterNet [258] with a DLA-34 backbone [233] as the object

detector. We adopt the improvements in CenterNet2 [256] to use BiFPN [186] for upsampling layers instead of the original deformable-convolution-based [29] hierarchical upsampling [233]. We use 3 BiFPN layers with 160 channels and 5 FPN levels (stride 8–128) in our experiments. We use RoIAlign [63] to extract features for our tracking transformer, but still use the boxes from CenterNet proposals. We use a training size of 1280×1280 and a test size of 1560 (longer edge). Following CenterTrack [255], we pretrain the detector on Crowdhuman [161] for 96 epochs. We then fine-tune with the GTR head on Crowdhuman (with augmentation) and the MOT training set in a $1:1$ ratio [179]. We again use $T = 8$ frames for a video clip with a batch size of 8 clips. We fine-tune the network for 8K iterations, which corresponds to $\sim$9 epochs of Crowdhuman and 16 epochs of MOT. This takes $\sim$1.5 hours on 8 Quadro RTX 6000 GPUs.

Inference. During testing, we set the output score threshold to 0.55 for MOT and the proposal score threshold to 0.4 for TAO, based on a sweep on the validation set. We do not set an output threshold for TAO. For both datasets, we set the new-track association threshold to $\theta = 0.2$. Since the MOT dataset has high frame rate, we find it beneficial to use location information during association. We associate tracks based on the maximum of the trajectory association score and the box-trajectory IoU. This is examined in a controlled experiment in Section 6.2.5. We further remove trajectories of length < 5 [12].

Runtime. On the MOT17 dataset our backbone detector runs in 47ms and the global tracking transformer in 4ms per frame. On TAO the backbone

	TAO				MOT17				
	Track mAP	HOTA	DetA	AssA	MOTA	IDF1	HOTA	DetA	AssA
IoU [8]	8.8	32.7	30.5	35.4	68.9	65.0	57.4	59.2	56.1
IoU+ReID [247]	11.0	34.9	31.2	39.5	70.9	74.0	61.7	60.0	63.9
Ours (T=2)	13.6	42.0	35.8	49.8	71.3	65.1	57.8	60.6	55.8
Ours (T=4)	17.7	44.0	36.4	53.6	**71.6**	69.6	59.9	**60.8**	59.6
Ours (T=8)	19.5	45.6	36.8	56.8	71.3	72.2	61.1	60.7	62.0
Ours (T=16)	**22.5**	**45.8**	**36.8**	**57.4**	71.4	75.1	62.5	60.6	65.0
Ours (T=32)	22.1	44.9	35.9	56.7	71.3	**75.9**	**63.0**	60.4	**66.2**

Table 6.1: Comparing greedy trackers [8, 247] (top) with our global tracker under different temporal windows on the TAO and MOT17 validation sets. We show the official metrics (Track mAP for TAO and MOTA/IDF1 for MOT17) as well as HOTA metrics. Higher is better for all metrics.

runs in 86ms, the transformer in 3ms.

6.2.3 Global versus local association

We first validate our main contribution: global association. We compare to two popular local trackers, one based on location (SORT [8]) and and one based on joint location and identity (FairMOT [247]). We use the detection output of our detector with the same output threshold so that the detections are the same. We adopt the implementation from FairMOT [247] (MIT License) for both baselines, with the default hyperparameters[2] and tricks, including a track-rebirth mechanism for up to 30 frames.

Table 6.1 shows the results on the TAO [32] and MOT17 [126] validation sets. First, despite close MOTA and DetA, ReID-based methods (Fair-

[2]We tuned the hyperparameters for both baselines, but observed that the default settings performed best.

MOT [247] and ours) generally achieve higher tracking accuracy than the location-only baseline [8]. For our method, when $T = 2$ it reduces to a local tracker that only associates across consecutive pairs of frames. This tracker cannot recover from any occlusion or missing detection, yielding a relatively low AssA. However, when we gradually increase the temporal window T, we observe a consistent increase in association accuracy. On MOT17 with $T = 32$, our method outperforms FairMOT [247] by a healthy 2.3 AssA and 1.9 IDF1, showing the advantage of our global tracking formulation. On TAO the performance saturates at $T = 16$. This may be due to the lack of training on video data and the much lower frame-rate in the TAO dataset.

6.2.4 Comparison to the state of the art

Next we compare to other trackers with different detections on the corresponding test sets. Table 6.2 shows the results on TAO validation and test sets. TAO [32] is a relatively new benchmark with few public entries [32, 134]. Our method substantially outperforms the official SORT baseline [32] and the prior best result (QDTrack [134]), yielding a relative improvement of 62% in mAP on the test set. While part of the gain is from our stronger detector, this highlights one of the advantages of our model: it is end-to-end jointly trainable with state-of-the-art detection systems.

Table 6.3 compares our tracker with other entries on the MOT17 leaderboard. Our entry achieves a decent 74.1 MOTA, 71.1 IDF1, and 59.0 HOTA. This is better than concurrent transformer-based trackers, including Track-

	Validation				Test			
	mAP50	mAP_s	mAP_m	mAP_l	mAP50	mAP_s	mAP_m	mAP_l
SORT_TAO [32]	13.2	-	-	-	10.2	7.7	8.2	15.2
QDTrack [134]	16.1	2.4	4.6	19.6	12.4	3.7	8.3	18.8
Ours	**22.5**	**7.8**	**16.9**	**33.7**	**20.1**	**10.0**	**16.6**	**25.5**

Table 6.2: Results on TAO validation and test sets. We show the official metrics from the leaderboard, including tracking mAP50 and its breakdown to small, medium, and large objects.

former [125], MOTR [238], and TransCenter [222]. Our model currently underperforms CorrTracker [204]. One possible reason is that CorrTracker [204] has been trained on additional person detection and ReID datasets (following the setting of JDE [205]), which may lead to better generalization in both detection and association.

6.2.5 Design choice experiments

Here we ablate our key design choices. All experiments are conducted under the best setting of Table 6.1, with $T = 32$. The random noise across different runs is within 0.2 MOTA and 0.5 AssA.

Attention structure. We first verify the necessity of using a transformer structure for the association head. As the counterpart, we remove both the self-attention layer and the cross-attention layer in Figure 6.2, and directly dot-product the association features after the linear layers. Table 6.4a shows that this decreases AssA considerably. Further adding self-attention layers between queries in the decoder as in DETR [17] does not improve performance, thus we just use encoder attention.

	MOTA↑	HOTA↑	DetA↑	AssA↑	FP↓	FN↓	IDS↓	IDF1↑
Trackformer [125]	65.0	-	-	-	70443	123552	3528	63.9
MOTR [238]	65.1	-	-	-	45486	149307	2049	66.4
ChainedTracker [139]	66.6	49.0	53.6	45.2	22284	160491	5529	57.4
CenterTrack [255]	67.8	52.2	53.8	51.0	18498	160332	3039	64.7
QDTrack [134]	68.7	53.9	55.6	52.7	26589	146643	3378	66.3
TraDeS [214]	69.1	52.7	55.2	50.8	20892	150060	3555	63.9
TransCenter [222]	73.2	54.5	60.1	49.7	23112	123738	4614	62.2
GSDT [205]	73.2	55.2	60.0	51.0	26397	120666	3891	66.5
FairMOT [247]	73.7	59.3	60.9	58.0	27507	117477	3303	72.3
TransTrack [179]	74.5	53.9	60.5	48.3	28323	112137	3663	63.9
CSTrack [106]	74.9	59.3	61.1	57.9	23847	114303	3567	72.6
FUFET [159]	76.2	57.9	62.9	53.6	32796	98475	3237	68.0
CorrTracker [204]	76.5	60.7	62.8	58.9	29808	99510	3369	73.6
Ours	74.1	59.0	61.7	56.7	36663	104772	3369	71.1

Table 6.3: Comparison to the state of the art on the MOT17 test set (private detection). We show the official metrics from the leaderboard. ↑ means that higher is better and ↓ means that lower is better.

	HOTA	DetA	AssA	MOTA
Dot product	61.3	59.5	63.6	70.5
*Enc. att.	63.0	60.4	66.2	71.3
Enc.+Dec att.	62.3	60.5	64.5	71.2

(a) With/without attention layers.

	HOTA	DetA	AssA	MOTA
*no emb.	63.0	60.4	66.2	71.3
pos. emb.	62.5	60.7	65.0	71.7
pos.+ temp. emb.	62.4	60.7	64.6	71.7

(b) Different positional embeddings.

Enc.	Dec.	HOTA	DetA	AssA	MOTA
*1	1	63.0	60.4	66.2	71.3
1	2	62.7	60.4	65.0	71.2
2	1	63.0	60.9	66.0	71.7

(c) Number of transformer layers.

	MOT17				TAO
	HOTA	DetA	AssA	MOTA	mAP50
w/o loc.	61.7	60.6	63.3	71.3	22.5
*w/ loc.	63.0	60.4	66.2	71.3	22.5

(d) With/without using location in testing.

Table 6.4: Design choice experiments on the MOT17 validation set. * means our default setting.

Positional embedding. Positional embedding is a common component in transformers. We have implemented a learned positional embedding as well as a learned temporal embedding. However, we didn't observe an improvement in association accuracy from these, as shown in Table 6.4b. We thus don't use any positional embedding in our final model.

Transformer layers. Table 6.4c shows the results of using different numbers of attention layers in the transformer encoder and decoder. While most other transformer-based trackers [125,179] require 6 encoder and decoder layers to work well, we observe that 1 layer in each is sufficient in our model. One possible reason for this is that other trackers take pixel features as the encoder input, while we use object features from the proposal network, which makes the task easier.

Using location in testing. As described in Section 6.1.3, we combine the trajectory probability and location-based IoU during inference. Table 6.4d examines this choice. On MOT17, using location information improves AssA by 3, likely due to the high frame-rate on the MOT dataset. On TAO, where frame-rate is low, using our predicted trajectory probability alone works fine.

Summary In this chapter, we presented a unified framework for joint object detection and tracking. The key component is a global tracking transformer that takes object proposals from all frames and groups objects into trajectories. Our framework produces trajectories in a global manner. It can be jointly trained with the detector and runs online during testing with very low

overhead. It performs competitively on the MOT17 benchmark and sets a new state-of-the-art on the challenging TAO dataset.

Chapter 7

UniDet: Unified detection on multiple datasets

[1] The detectors and trackers described in the previous chapters have shown promising results on their corresponding trained benchmarks: COCO [109] or MOT [126]. However, computer vision aims to produce broad, general-purpose perception systems that work in the wild, and is likely not covered by a single dataset. Yet object detection is fragmented into datasets [89, 109, 129, 160] and our models are locked into the corresponding domains. This fragmentation brought rapid progress in object detection [153] and instance segmentation [63], but comes with a drawback. Single datasets are limited in both image domains and label vocabularies and do not yield general-purpose recognition systems. Can we alleviate these limitations by unifying diverse detection datasets?

In this chapter, we first make training an object detector on a collection of disparate datasets as straightforward as training on a single one. Different datasets are usually trained under different training losses, data sampling

[1]The work in this chapter was originally published as: Simple multi-dataset detection, Xingyi Zhou, Vladlen Koltun, Philipp Krähenbühl, in CVPR 2022. Code and models are released at https://github.com/xingyizhou/UniDet.

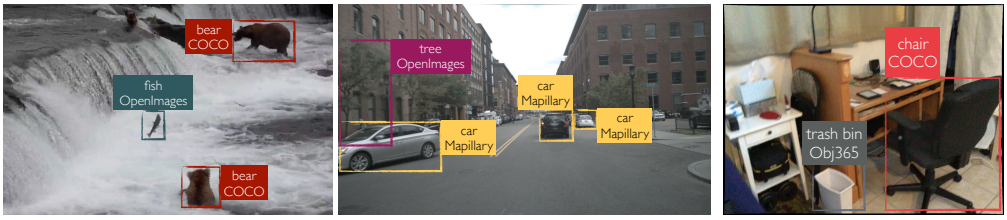


Figure 7.1: Different datasets span diverse semantic and visual domains. We learn to unify the label spaces of multiple datasets and train a single object detector that generalizes across datasets.

strategies, and schedules. We show that we can train a single detector with separate outputs for each dataset, and apply dataset-specific supervision to each. Our training mimics training parallel dataset-specific models with a common network. As a result, our single detector takes full advantage of all training data, performs well on training domains, and generalizes better to new unseen domains. However, this detector produces duplicate outputs for classes that occur in multiple datasets.

A core challenge is integrating different datasets into a common taxonomy, and training a detector that reasons about general objects instead of dataset-specific classes. Traditional approaches create this taxonomy by hand [91, 250], which is both time-consuming and error-prone. We present a fully automatic way to unify the output space of a multi-dataset detection system using visual data only. We use the fact that object detectors for similar concepts from different datasets fire on similar novel objects. This allows us to define the cost of merging concepts across datasets, and optimize for a common taxonomy fully automatically. Our optimization jointly finds a unified

taxonomy, a mapping from this taxonomy to each dataset, and a detector over the unified taxonomy using a novel 0-1 integer programming formulation. An object detector trained on this unified taxonomy has a large, automatically constructed vocabulary of concepts from all training datasets.

We evaluate our unified object detector at an unprecedented scale. We train a unified detector on 3 large and diverse datasets: COCO [109], Objects365 [160], and OpenImages [89]. For the first time, we show that a single detector performs as well as dataset-specific models on each individual dataset. A unified taxonomy further improves this detector. Crucially, we show that models trained on diverse training sets generalize zero-shot to new domains, and outperform single-dataset models. Our models ranked first in the object detection and instance segmentation tracks of the ECCV 2020 Robust Vision Challenge across all evaluation datasets.

7.1 Multi-dataset training

Our goal is to train a single detector $\mathcal{M}$ on K datasets $\mathcal{D}_1, \ldots, \mathcal{D}_K$ with label spaces $C_1, \ldots, C_K$, and dataset-specific training objectives $\ell_1, \ldots, \ell_K$. Our core insight is that we can train a unified detector in the same way as we train multiple dataset-specific detectors separately, as long as we do not attempt to merge label spaces between different datasets. This can be considered training K dataset-specific detectors $\mathcal{M}_1, \ldots, \mathcal{M}_K$ in parallel, while *sharing their backbone architecture* $\mathcal{M}$. Each dataset-specific architecture shares all but the last layer with the common backbone. Each dataset uses its own

classification layer at the end. We call this a **partitioned detector** (Figure 7.2b). We train a partitioned detector over all datasets by minimizing the K dataset-specific losses:

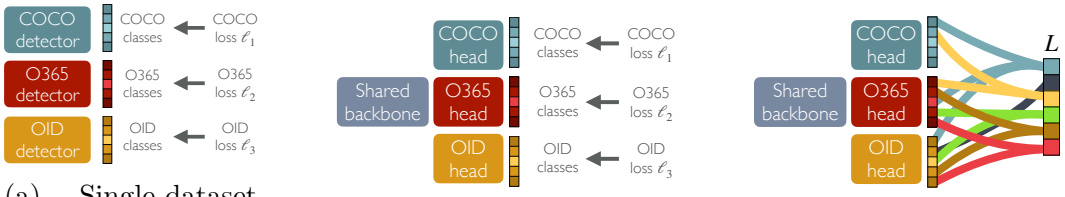
$$\min_{\Theta} \mathbb{E}_{\mathcal{D}_k} \left[\mathbb{E}_{(\hat{I},\hat{B})\sim\mathcal{D}_k} \left[\ell_k(\mathcal{M}_k(\hat{I};\Theta), \hat{B}) \right] \right]. \tag{7.1}$$

Here, evenly sampling datasets, i.e. showing the partitioned detector the same number of images from each dataset, works best empirically, as we will show in Section 7.3.1.

While the partitioned detector learns to detect all classes, it still produces different dataset-specific outputs. For example, it predicts a COCO-person separately from an Objects365-Person, etc. Next we show how to convert this partitioned model into a joint detector that reasons about a *unified* set of output labels $C = C_1 \cup C_2 \cup \ldots$.

7.2 Learning a unified label space

Consider multiple datasets, each with its own label space $C_1, C_2, \ldots$. Our goal is to jointly learn a common label space C for all datasets, and define a mapping between this common label space and dataset-specific labels $\mathcal{T}_k : C \to C_k$. Mathematically, $\mathcal{T}_k \in \{0,1\}^{|C_k|\times|C|}$ is a Boolean linear transformation. In this work, we only consider direct mappings. Each joint label $c \in C$ maps to at most one dataset-specific label $\hat{c} \in C_k$: $\mathcal{T}_k^\top \vec{1} \leq \vec{1}$. I.e., no dataset contains duplicated classes itself. Also, each dataset-specific label matches to exactly one joint label: $\mathcal{T}_k \vec{1} = \vec{1}$. In particular, we do not hierarchically relate



(a) Single-dataset detectors

(b) Partitioned detector

(c) Unified detector

Figure 7.2: Standard detectors (a) are trained on one dataset with a dataset-specific loss. We train a single partitioned detector (b) on multiple datasets with shared backbone, dataset-specific outputs and loss. Finally, we unify the outputs of the partitioned detector in a common taxonomy completely automatically (c).

concepts across datasets. When there are different label granularities, we keep them all in our label-space, and expect to predict all of them[2].

Given a set of partitioned detector outputs $d_i^1 \in \mathbb{R}^{|C_1|}, d_i^2 \in \mathbb{R}^{|C_2|}, \ldots$ for a bounding box b_i, we build a joint detection score d_i by simply averaging the outputs of common classes:

$$d_i = \frac{\sum_k \mathcal{T}_k^\top \vec{d}^k}{\sum_k \mathcal{T}_k^\top \vec{1}}, \tag{7.2}$$

where the division is elementwise. Figure 7.2c provides an overview. From this joint detector, we recover dataset-specific outputs $\tilde{d}_i^k = \mathcal{T}_k d_i$. Our goal is to find a set of mappings $\mathcal{T}^\top = \left[\mathcal{T}_1^\top \ldots, \mathcal{T}_N^\top\right]$ and implicitly define a joint label-space C such that the joint classifier does not degrade in performance.

Simple baselines include hand-designed mappings $\mathcal{T}$ and label spaces C [91, 250], or language-based merging. One issue with these techniques is

[2]This follows the official evaluation protocol of OpenImages [89].

that word labels are ambiguous. Instead, we let the data speak and optimize a label space automatically based on correlations in the firings of a pre-trained partitioned detector on different images, which is a proxy for perceptual similarity.

For a specific output class c, let $\mathcal{C}_c$ be a loss function that measures the quality of the merged label space d_i and its re-projections $\hat{d}_i^k$ compared to the original disjoint label-space d_i^k on a single box i. Let $D^k = [d_1^k, d_2^k, \ldots]$ be the outputs of the partitioned detection head for dataset $\mathcal{D}_k$. Let $D = \frac{\sum_k \mathcal{T}_k^\top D^k}{\sum_k \mathcal{T}_k^\top \vec{1}}$ be the merged detection scores, and $\tilde{D}^k = \mathcal{T}_k D$ be the reprojection. Our goal is to optimize this loss over all detector outputs given the Boolean constraints on our mapping

$$\begin{aligned} &\text{minimize}_{C,\mathcal{T}} && E_{\mathcal{D}_k}\left[\sum_{c \in C_k} \mathcal{L}_c(D_c^k, \tilde{D}_c^k)\right] + \lambda|C| \\ &\text{subject to} && \mathcal{T}_k\vec{1} = \vec{1} \quad \text{and} \quad \mathcal{T}_k^\top \vec{1} \leq \vec{1} \qquad \forall_k. \end{aligned} \tag{7.3}$$

The cardinality penalty $\lambda|C|$ encourages a small and compact label space. A factorization of the loss $\mathcal{L}_c$ over the output space $c \in C_k$ may seem restrictive. However, it does include the most common loss functions in detection: score distortion and Average Precision (AP). Section 7.2.1 discusses the exact loss functions used in our optimization.

Objective 7.3 mixes combinatorial optimization over C with a 0-1 integer program over $\mathcal{T}$. However, there is a simple reparametrization that lends itself to efficient optimization.

First, observe that the label set C simply corresponds to the number of columns in $\mathcal{T}$. Furthermore, we merge at most one label per dataset $\mathcal{T}_k^\top \vec{1} \leq \vec{1}$. Hence, for each dataset $\mathcal{D}_k$ a column $\mathcal{T}_k(c) \in \mathbb{T}_k$ takes one of $|\hat{C}_k| + 1$ values: $\mathbb{T}_k = \{\vec{0}, \vec{1}_1, \vec{1}_2, \ldots\}$, where $\vec{1}_i \in \{0,1\}^{|C_k|}$ is an indicator vector of the i-th element. Each column $\mathcal{T}(c) \in \mathbb{T}$ then only chooses from a small set of potential values $\mathbb{T} = \mathbb{T}_1 \times \mathbb{T}_2 \times \ldots$, where $\times$ represents the Cartesian product. Instead of optimizing over the label set C and transformation $\mathcal{T}$ directly, we instead use combinatorial optimization over the potential column values of $\vec{t} \in \mathbb{T}$. Let $x_{\vec{t}} \in \{0,1\}$ be the indicator of combination $\vec{t} \in \mathbb{T}$. $x_{\vec{t}} = 1$ means we apply the class combination specified by $\vec{t}$, and otherwise not. In this formulation, the constraint $\mathcal{T}_k \vec{1} = \vec{1} \forall_k$ translates to $\sum_{\vec{t} \in \mathbb{T} | \vec{t}(c)=1} x_{\vec{t}} = 1$ for all dataset-specific labels c. Furthermore, the objective of the optimization simplifies to

$$\sum_{\vec{t} \in \mathbb{T}} x_{\vec{t}} \underbrace{E_{\mathcal{D}_k}\left[\sum_{c \in C_k | \vec{t}(c)=1} \mathcal{L}_c(D_c^k, \tilde{D}_c^k)\right]}_{c_{\vec{t}}} + \lambda \sum_{\vec{t} \in \mathbb{T}} x_{\vec{t}}. \tag{7.4}$$

Crucially, the merge cost $c_{\vec{t}}$ can be precomputed for any subset of labels $\vec{t}$. This leads to a compact integer linear programming formulation of objective 7.3:

$$\begin{aligned} &\text{minimize}_x && \sum_{\vec{t} \in \mathbb{T}} x_{\vec{t}} (c_{\vec{t}} + \lambda) \\ &\text{subject to} && \sum_{\vec{t} \in \mathbb{T} | \vec{t}_c = 1} x_{\vec{t}} = 1 \qquad \forall_c \end{aligned} \tag{7.5}$$

For two datasets, the above objective is equivalent to a weighted bipartite matching. For a higher number of datasets, it reduces to weighted graph

matching and is NP-hard, but is practically solvable with integer linear programming [110].

One drawback of the combinatorial reformulation is that the set of potential combinations $\mathbb{T}$ grows exponentially in the datasets used: $|\mathbb{T}| = O(|\hat{C}_1||\hat{C}_2||\hat{C}_3|\ldots)$. However, most merges $\vec{t} \in \mathbb{T}$ are bad and incur a large merge cost $c_{\vec{t}}$. In practice, we use a linear-time greedy enumeration algorithm for low-cost merges. Considering only low-cost matches, standard integer linear programming solvers find an optimal solution within seconds for all label spaces we tried, even for $|C| > 600$ and up to 6 datasets.

7.2.1 Loss functions

The loss function in our constrained objective 7.3 is quite general and captures a wide range of commonly used losses. We highlight two: an unsupervised objective based on the distortion between partitioned and unified outputs, and Average Precision (AP) on a validation set.

Distortion measures the difference in detection scores between partitioned and unified detectors:

$$\mathcal{L}_c^{\text{dist}}(D_c^k, \tilde{D}_c^k) = \left(D_c^k - \tilde{D}_c^k\right)^2. \tag{7.6}$$

A drawback of this distortion measure is that it does not take task performance into consideration when optimizing the joint label space.

Average Precision. Given a reprojected dataset-specific output $\tilde{D}_c^k$, we can measure the average precision $\mathrm{AP}_c(\tilde{D}_c^k)$ of each output class c on the validation set of $\mathcal{D}_k$. Our loss measures the improvement in AP:

$$\mathcal{L}_c^{\mathrm{AP}}(D_c^k, \tilde{D}_c^k) = \frac{1}{|C_k|}\left(\mathrm{AP}_c(D_c^k) - \mathrm{AP}_c(\tilde{D}_c^k)\right). \tag{7.7}$$

The AP computation is computationally quite expensive. We will provide an optimized joint evaluation in our code.

These two loss functions allow us to train a partitioned detector and merge its output space after training, either maximizing the original evaluation metric (AP) or minimizing the change incurred by the unification.

7.3 Experiments

Our goal is to facilitate the training of a single model that performs well across datasets. In this section, we first introduce our dataset setup and implementation details. In Section 7.3.1, we analyze our key design choices for a partitioned detector baseline. In Section 7.2, we evaluate our unified detector and our unified label space learning algorithm. We further evaluate the unified detector in new test datasets in a zero-shot setting (Section 7.3.3) and report its performance in the ECCV 2020 Robust Vision Challenge (Section 7.3.4).

Datasets. Our main training datasets are adopted from the ECCV 2020 Robust Vision Challenge (RVC). These are four large object detection datasets: COCO [109], OpenImages [89], Objects365 [160], and optionally Mapillary [129]. To evaluate the generalization ability of the models, we follow

Dataset name	Domain	# Cat.	# Img.
Train & Validation			
COCO	Internet images	80	118k
Objects365	Internet images	365	600k
OpenImages	Internet images	500	1.8M
Mapillary	Traffic	38	18k
Test			
ScanNet	Indoor	20	25k
VIPER	Virtual	10	13k
Cityscapes	Traffic	8	12k
WildDash	Traffic	13	4k
KITTI	Traffic	8	200
Pascal VOC	Internet images	20	16k
CrowdHuman	Internet images	1	15k

Table 7.1: Datasets we used. Top: datasets we used in training and validation, which are from the ECCV 2020 Robust Vision Challenge. Bottom: datasets we used for zero-shot cross-dataset testing.

MSeg [91] to set up a zero-shot cross-dataset evaluation protocol: we evaluate models on new test dataset *without training on them*. Specifically, we test on VIPER [155], Cityscapes [27], ScanNet [28], WildDash [237], KITTI [55], Pascal VOC [42], and CrowdHuman [161]. A detailed description of all datasets is provided in Table 7.1. In our main evaluation, we use large and general datasets: COCO, Objects365, and OpenImages. Mapillary is relatively small and is specific to traffic scenes; we only add it for the RVC and cross-dataset experiments.

For each dataset, we use its official evaluation metric: for COCO, Objects365, and Mapillary, we use mAP at IoU thresholds 0.5 to 0.95. For OpenImages, we use the official modified mAP@0.5 that excludes unlabeled

classes and enforces hierarchical labels [89]. For the small datasets in cross-dataset evaluation, we use mAP at IoU threshold 0.5 for consistency with PascalVOC [42].

Implementation details. We use the CascadeRCNN detector [14] with a shared region proposal network (RPN) across datasets. We evaluate two models in our experiments: a partitioned detector (i.e., detector with dataset-specific output heads) and a unified model). For the partitioned detector, the last classification layer of each cascade stage is split between datasets. The unified model uses the CascadeRCNN as is.

Our implementation is based on Detectron2 [216]. We adopt most of the default hyperparameters for training. We use the standard data augmentation, including random flip and scaling of the short edge in the range $[640, 800]$. We use SGD with base learning rate 0.01 and batch size 16 over 8 GPUs. We use ResNet50 [65] as the backbone in our controlled experiments unless specified otherwise. We use a $2\times$ training schedule (180k iterations with learning rate dropped at the 120k and 160k iterations) [216] in most experiments unless specified otherwise, regardless of the training data size.

7.3.1 Multi-dataset detection

We first evaluate the partitioned detector. We use dataset-specific outputs and do not merge classes between different datasets. During evaluation, we assume the target dataset is known and only look at the corresponding output head. As discussed in Section 7.1, our baseline highlights two basic com-

	COCO	O365	OImg	*mean*
Simple merge [207]	34.2	14.6	50.8	33.2
w/ uniform dataset sampling	41.1	16.5	46.0	34.5
w/ class-aware sampling	35.3	18.5	61.8	38.5
w/ dataset+class-aware sampling	**41.8**	20.3	60.0	40.6
Partitioned detector (ours)	**41.8**	**20.6**	**62.7**	**41.7**

Table 7.2: Ablation of our multi-dataset training strategies. We start with simple merging of datsets [207], then add a uniform sampling of images between different training datasets (second row), class-aware sampling within Objects365 and OpenImages (third row), and both sampling strategies (forth row). Our partitioned detector combines these sampling strategies with a dataset-specific loss (last row).

ponents: uniform sampling of images between datasets and dataset-specific training objective. For these experiments we distinguish between modifications of the objective that merely sample data differently within each dataset (e.g. class-aware sampling), and changes to the loss functions (e.g. hierarchical losses).

We start from the baseline of [207, 216]. They simply collect all data from all datasets and train with a common loss. As is shown in Table 7.2,

	2×			6×			8×		
	COCO	O365	OImg.	COCO	O365	OImg.	COCO	O365	OImg.
Partitioned	**41.8**	20.6	62.7	**44.6**	23.6	64.8	**45.5**	24.6	**66.0**
COCO	41.5	-	-	42.5	-	-	42.5	-	-
O365	-	**23.8**	-	-	**25.0**	-	-	**24.9**	-
OpenImages	-	-	**64.6**	-	-	**65.4**	-	-	65.7

Table 7.3: Comparison of a partitioned detector to single-dataset models under different training schedules in terms of validation mAP.

this biases the model to large datasets (OpenImages) and yields low performance for relatively small datasets (COCO). Sampling datasets uniformly (second row) trades the performance on smaller datasets with large datasets, and overall improves performance. On the other hand, both OpenImages and Objects365 are long-tailed and best train with advanced inter-dataset sampling strategy [140, 164], namely class-aware sampling. Class-aware sampling significantly improves accuracy on OpenImages and Objects365. Combining the uniform dataset sampling and the intra-dataset class-aware sampling gives a further boost. Finally, OpenImages [89] requires predicting a label hierarchy. For example, it requires predicting "vehicle" *and* "car" for all cars. This breaks the default cross-entropy loss that assumes exclusive class labels per object. We instead use a dedicated hierarchy-aware sigmoid cross-entropy loss for OpenImages [89]. Specifically, for an annotated class label in OpenImages, we set all its parent classes as positives and ignore the losses over its descendant classes. Our partitioned detector combines both sampling strategies and the dataset-specific loss. The hierarchy-aware loss yields a significant +2.7mAP improvement on OpenImages alone, and does not degrades other datasets.

Dateset-specific vs. partitioned detection. In our partitioned detector, training on multiple datasets resembles training separate individual models but with a shared detector. Table 7.3 compares training a partitioned detector on all datasets with dataset-specific models. We compare detectors under different training schedules ($n\times$ the COCO default schedule). Each of the three dataset-specific models sees the same number of gradient updates as

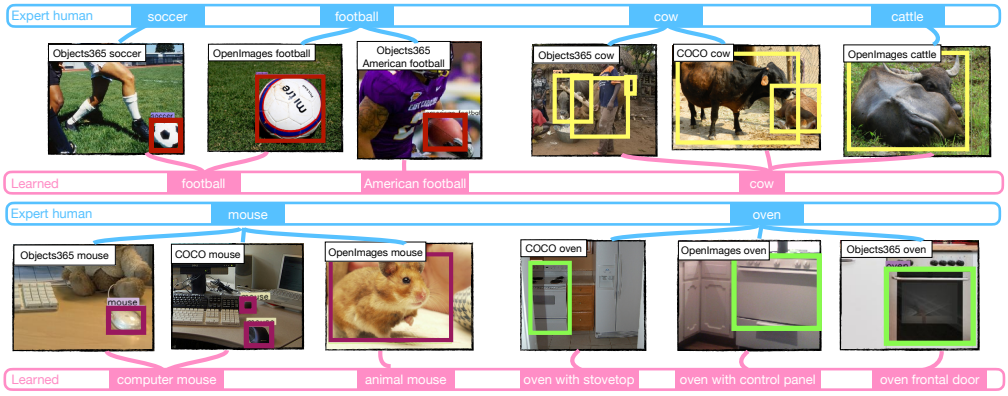


Figure 7.3: Some differences between an expert-designed label space provided as part of the ECCV 2020 Robust Vision Challenge (top of each row, blue) and our learned label space (bottom of each row, pink). Zoom in for details.

our partitioned detector. In a 2× training schedule (180k iterations), single-dataset models generally perform better than a partitioned model, as each dataset is only trained for a $\frac{1}{3}\times$ schedule in the partitioned model. At a 6× schedule, the partitioned detector starts to match dataset-specific models, and outperforms 2× dataset-specific models under the same total iterations. In a 8× schedule, all models converge. Here the partitioned detector surpasses the single-dataset model on COCO and OpenImages, and closely matches the Objects365 model.

7.3.2 Unified multi-dataset detection

Next, we evaluate different unification strategies for the label space.

Unification strategies. We run our label space learning algorithm from Section 7.2 based on the output of a partitioned detector with a ResNeSt backbone [240] trained on COCO, Objects365, and OpenImages, with a total

of 945 disjoint classes. The hyperparameters are $\lambda = 0.5$ and $\tau = 0.25$. The optimization ends up with a unified label space with cardinality $|C| = 701$. we compare our automated data-driven unification to human and language-based baselines. We use the official manually-crafted ECCV RVC taxonomy as the human expert baseline[3].

Over two-thirds of our learned label space agrees with the human expert. Figure 7.3 highlights some of the differences. Our unification successfully groups similar concepts with different descriptions ("Cow" and "Cattle"), and is not distracted by spurious linguistic matches ("American football" and "football"). Interestingly, the learned label space splits the "oven" classes from COCO, Objects365, and OpenImages, even though they share the same word. A visual examination reveals that they are visually dissimilar due to different underlying definitions of the "oven" concept in the different datasets: COCO ovens include the cooktop, OpenImages ovens include the control panel, and Objects365 ovens are just the front door. Our data-driven taxonomy reconciliation is able to detect such distinctions, which are missed by word-level approaches.

We next quantitatively compare our learned label space with alternatives. For each label space, we retrain a multi-dataset detector with that label space. During training, as with our partitioned model, we only apply training losses to the classes that are annotated in the source dataset. We compare our

[3] https://github.com/ozendelait/rvc_devkit/blob/master/objdet/obj_det_mapping.csv

	COCO	O365	OImg.	*mean*
GloVe embedding	41.6±0.00	20.3±0.12	62.4±0.06	41.4±0.05
Learned, distortion	41.6±0.15	20.7±0.06	62.6±0.06	41.7±0.09
Learned, AP (ours)	**41.9**±0.10	**20.8**±0.10	**63.0**±0.21	**41.9**±0.02
Expert human	41.5±0.06	20.7±0.06	62.6±0.06	41.6±0.04

Table 7.4: Evaluation of unified label spaces. We measure mAP on the validation sets of the training domains. We compare to a language-based baseline (GloVe) and a manual unification by a human expert. Each model is a ResNet-50 CascadeRCNN trained in a 2× schedule. We show the mean and standard deviation based on 3 repeated runs under different random seeds.

learned label space to a "best effort" human baseline and a language-based baseline. For the language-based baseline, we replace the cost measurement defined in Section 7.2.1 with the cosine distance between the GloVe word embeddings [141], and run the same integer linear program. Table 7.4 shows the results. We repeat the training for three runs with different random seeds and report the mean and standard deviation. The four label spaces agree on most classes and the overall mAP is thus close. Our automatically constructed label space consistently outperforms the human expert baseline, with a healthy 0.3 mAP margin on average. The improvement appears statistically stable under multiple training runs. Notably, the relative improvement of our model over the expert is larger than the expert's improvement over the language-based baseline.

Unified vs. partitioned detectors. We next compare a partitioned detector, a unified detector with and without retraining using the joint taxonomy, and an ensemble of dataset-specific detectors. Note that the par-

	COCO	O365	OImg.	*mean*
Unified (partitioned)	44.4	23.6	65.3	44.4
Unified (retrained)	45.4	24.4	66.0	45.3
Partitioned (oracle)	45.5	24.6	66.0	45.4
Ensemble (oracle)	42.5	24.9	65.7	44.4

Table 7.5: Validation mAP on training domains for our unified detector (top), the same detector retrained on the joint taxonomy (second), a partitioned detector knowing the target domain (thrid), and an ensemble of three dataset-specific detectors (bottom). The bottom two rows require a known test dataset source and the top two rows do not. All models use a ResNet-50 CascadeRCNN trained in an 8× schedule.

titioned detector and the ensemble need to know the target domain at test time, while the unified models do not. This means that the unified models can be deployed zero-shot in new domains, while the alternatives must know which domain they are in. Table 7.5 shows the results. A partitioned detector outperforms a dataset-specific ensemble under the same experimental conditions (known test domain), especially on the "small" COCO dataset. An offline unification loses some accuracy, but this is regained when retraining the model under the unified taxonomy. Crucially, the unified models do not need to know what domain they are in at test time.

7.3.3 Zero-shot cross-dataset evaluation

We evaluate the generalization ability of object detectors by evaluating them in new test domains not seen during training. In this setting, we do not assume to know the test classes ahead of time. To allow for a fair and unbiased evaluation, we use a simple language-based matching to find the test-

	VOC	VIPER	CityS.	ScanNet	WildDash	CrowdH.	KITTI	*mean*
COCO	80.0	13.9	39.6	17.4	25.9	**73.9**	30.5	40.2
Objects365	71.9	20.7	43.4	24.9	27.6	71.8	32.2	41.8
OpenImages	64.4	10.4	29.8	24.2	20.3	66.7	21.8	33.9
Mapillary	11.4	15.2	44.7	0.0	23.4	49.3	37.8	26.0
Ensemble	79.7	16.8	46.0	30.1	32.1	**73.9**	34.3	44.7
Partitioned	**83.1**	20.9	48.4	**32.2**	34.4	70.0	38.9	46.8
Unified (retrained)	**82.9**	**21.3**	**52.6**	29.8	**34.7**	70.7	**39.9**	**47.3**
Dataset-specific	80.3	31.8	54.6	44.7	-	80.0	-	-

Table 7.6: Zero-shot cross-dataset object detection performance on the validation sets of datasets that were not seen during training. We compare models trained on each single training dataset (top 4 rows), the ensemble of the 4 single dataset models (5th row), a partitioned detector (6th row), and the unified detector with our learned unified label space (7th row). For reference, we show the "oracle" models that are trained on the training set of each test dataset on the bottom row. The columns refer to test datasets. Each model is a ResNet-50 CascadeRCNN trained until converge or at most an 8× schedule.

to-train label correspondence. Specifically, we calculate the GloVe [141] word embedding distances between each test label and the training label, and match the test label to its closest training label. If multiple training labels match, we break ties in a fixed order: COCO, Objects365, OpenImages, and Mapillary[4].

We compare both our multi-dataset models (partitioned or unified) to single-dataset models. We use all four RVC training sets to train the multi-dataset models. Specifically, we start from a 6× schedule model trained on the three large datasets, and add Mapillary [129] in a 2× fine-tuning schedule with 10× smaller learning rate. We compare all models under the same sched-

[4]We also tried evaluating under different orders, and find the listed order to perform best for all methods.

ule [5], hyperparameters, and detection models. In addition, we also compare to the ensemble of the four single-dataset models trained analogously to the partitioned model. For reference, we also show the performance of detectors trained on the training set of each test dataset. This serves as an oracle "upper bound" that has seen the test domain and label space. Note that KITTI and WildDash are small and do not have a validation set. We thus directly evaluate on the training set and do not provide the oracle model.

Table 7.6 shows the results. The COCO model exhibits reasonable performances of some test datasets, such as Pascal VOC and CrowdHuman. However, its performance is less than satisfactory on datasets such as ScanNet, whose label space differs significantly from COCO. Training on the more diverse Objects365 dataset yields higher accuracy in the indoor domain, but loses ground on VOC and CrowdHuman, which are more similar to COCO. Training on all datasets, either with a partitioned detector or a unified one yields generally good performance on all test datasets. Notably, both our detectors perform better than the ensemble of the 4 single dataset models, showing that the multi-dataset models learned more general features. On Pascal VOC, both multi-dataset models outperform the VOC-trained upper-bound without ever seeing VOC training images. Our unified model outperforms the partitioned detector overall and operates on a single unified taxonomy.

	COCO	OImg.	Mapillary	O365
Ours	**52.9**	**60.6/56.8**	**25.3**	**33.7**
WiseDet_RVC	40.0	56.1/53.3	22.5	-
FRCNN_R50_GN_RVC	34.0	21.4/19.9	8.1	-
DetectoRS	**53.3**	-	-	-
TSD	-	60.5/-	-	-
CACascade RCNN	-	-	-	31.6

Table 7.7: RVC results: COCO test-challenge set, OpenImages challenge 2019 test sets (public test set/ private test set), Mapillary test set, and Objects365 validation set. Top: results of RVC challenge participants. Bottom: published state-of-the-art models on each specific dataset (without model ensembles or test-time augmentation). Objects365 was initially part of the challenge but was removed in the final evaluation.

7.3.4 ECCV Robust Vision Challenge

We submitted a model trained with the presented approach to the ECCV 2020 Robust Vision Challenge (RVC). We used a heavy ResNeSt200 backbone [240] and followed the same training procedure as in Section 7.3.2 with an 8× schedule. We used a unified label space of 682 classes learned with the distortion loss. The training took ∼16 days on a server with 8 Quadro RTX 6000 GPUs. Table 7.7 summarizes the results of the challenge. Our model won the challenge, outperforming all other RVC entries on all datasets by a large margin. Notably, WiseDet_RVC used a stronger detector [144], but is trained under a uniform sampling strategy and uses the same training loss for all datasets. We also compare to state-of-the-art results on each individual

[5]except for the Mapillary model, for which a 2× schedule performs better than longer schedules.

dataset. On COCO, our result compares well to DetectoRS [144], which is by default 2.4 mAP higher than our ResNeSt200 backbone (50.9 mAP) [240]. On OpenImages, our result matches the best single model in the OpenImages 2019 Challenge, TSD [174], with a comparable backbone (SENet154-DCN [68]). On Objects365, we outperform the 2019 challenge winner [54] by 2 mAP points.

Summary In this chapter, we presented a simple approach for training an object detector across multiple datasets and automatically constructing a unified taxonomy for deploying the detector without knowledge of the test-time domain. Our models match dataset-specific performance in training domains, and outperform all baselines when evaluated on new test domains.

Chapter 8

Detic: Unified detector with image classes

[1]In the previous chapter, we expand the object detector's vocabulary by combining detection datasets. This expands the vocabulary from a few hundred to more than thousands. However, detection datasets are inherently smaller than classification datasets in both the overall size and vocabulary, due to the difficulties in annotations. Image classification, on the other hand, reasons about much larger vocabularies in wider domains. In this chapter, we seek ways to expand the detection vocabulary to tens of thousands, or even to infinite. The key ingredient is to use weakly-supervised learning on image-classification datasets.

Object detection consists of two sub-problems - finding the object (localization) and naming it (classification). Traditional methods tightly couple these two sub-problems and thus rely on box labels for all classes. Despite many data collection efforts, detection datasets [61, 90, 109, 160] are much smaller in overall size and vocabularies than classification datasets [34]. For

[1]The work in this chapter is based on: Detecting Twenty-thousand Classes using Image-level Supervision, Xingyi Zhou, Rohit Girdhar, Armand Joulin, Philipp Krähenbühl, Ishan Misra, in arXiv 2201.02605. Code is available at https://github.com/facebookresearch/Detic.

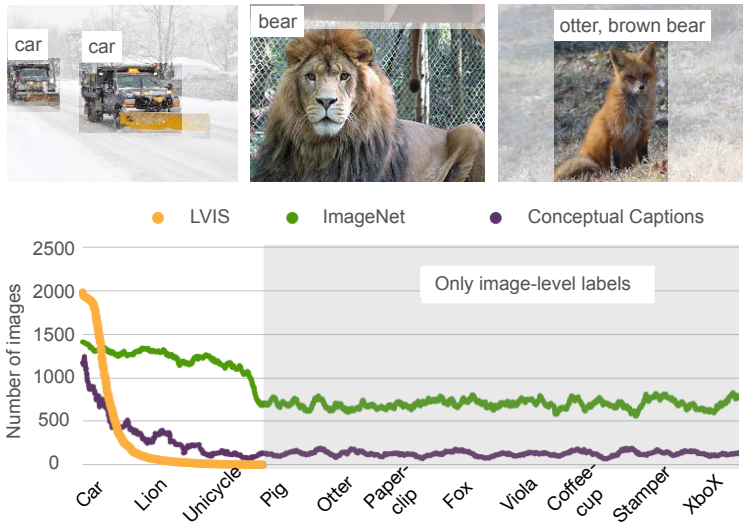


Figure 8.1: **Top:** Typical detection results from a strong open-vocabulary LVIS detector. The detector misses objects of "common" classes. **Bottom:** Number of images in LVIS, ImageNet, and Conceptual Captions per class (smoothed by averaging 100 neighboring classes). Classification datasets have a much larger vocabulary than detection datasets.

example, the recent LVIS detection dataset [61] has 1000+ classes with 120K images; OpenImages [90] has 500 classes in 1.8M images. Moreover, not all classes contain sufficient annotations to train a robust detector (see Figure 8.1 Top). In classification, even the ten-year-old ImageNet [34] has 21K classes and 14M images (Figure 8.1 Bottom).

In this chapter, we propose **Det**ector with **i**mage **c**lasses (Detic) that uses image-level supervision in addition to detection supervision. We observe that the localization and classification sub-problems can be decoupled. Modern region proposal networks already localize many 'new' objects using existing detection supervision. Thus, we focus on the classification sub-problem and

use image-level labels to train the classifier and broaden the vocabulary of the detector. We propose a simple classification loss that applies the image-level supervision to the proposal with the largest spatial size, and do not supervise other outputs for image-labeled data. This is extremely easy to implement and massively expands the detector's vocabulary.

Most existing weakly-supervised detection techniques [45, 70, 114, 187, 221] use the weakly labeled data to supervise *both* the localization and classification sub-problems of detection. Since image-classification data has no box labels, these methods develop various label-to-box assignment techniques *based on model predictions* to obtain supervision. For example, YOLO9000 [150] and DLWL [147] assign the image label to proposals that have high prediction scores on the labeled class. Unfortunately, this prediction-based assignment requires good initial detections which leads to a chicken-and-egg problem—we need a good detector for good label assignment, but we need many boxes to train a good detector. Our method completely side-steps the prediction-based label assignment process by supervising the classification sub-problem alone when using classification data. This also enables our method to learn detectors for new classes which would have been impossible to predict and assign.

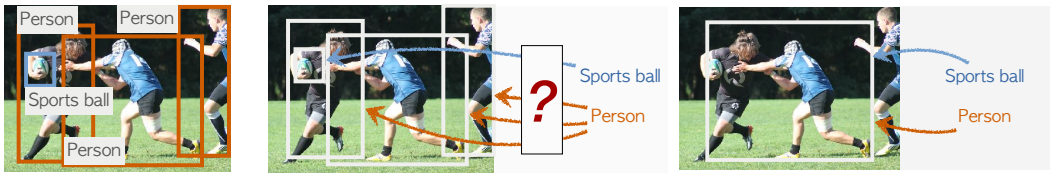
Experiments on the open-vocabulary LVIS [60,61] and the open-vocabulary COCO [5] benchmarks show that our method can significantly improve over a strong box-supervised baseline, on both novel and base classes. With image-level supervision from ImageNet-21K [34], our model trained without novel class detection annotations improves the baseline by 8.3 point and matches

the performance of using full class annotations in training. With the standard LVIS annotations, our model reaches 41.7 mAP and 41.7 mAP_{rare}, closing the gap between rare classes and all classes. On open-vocabulary COCO, our method outperforms the previous state-of-the-art OVR-CNN [236] by 5 point with the same detector and data. Finally, we train a detector using the full ImageNet-21K with more than twenty-thousand classes. Our detector generalizes much better to new datasets [90, 160] with disjoint label spaces, reaching 21.5 mAP on Objects365 and 55.2 mAP50 on OpenImages, without seeing any images from the corresponding training sets. Our contributions are summarized below:

- We identify issues and propose a simpler alternative to existing weakly-supervised detection techniques in the open-vocabulary setting.
- Our proposed family of losses significantly improves detection performance on novel classes, closely matching the supervised upper bound.
- Our detector transfers to new datasets without finetuning.
- We release our code (in supplement). It is ready-to-use for open-vocabulary detection in the real world. See examples in supplement.

8.1 Approach

As shown in Figure 8.2, our method leverages the box labels from detection datasets $\mathcal{D}^{\text{det}}$ and image-level labels from classification datasets $\mathcal{D}^{\text{cls}}$.



(a) Standard detection (b) Prediction-based label assignment (c) Our non-prediction-based loss

Figure 8.2: **Left:** Standard detection requires ground-truth labeled boxes and cannot leverage image-level labels. **Center:** Existing prediction-based weakly supervised detection methods [9, 147, 150] use image-level labels by assigning them to the detector's predicted boxes (proposals). Unfortunately, this assignment is error-prone, especially for large vocabulary detection. **Right:** Detic simply assigns the image-labels to the *max-size* proposal. We show that this loss is both simpler and performs better than prior work.

During training, we compose a mini-batch using images from both types of datasets. For images with box labels, we follow the standard two-stage detector training [153]. For image-level labeled images, we only train the features from a fixed region proposal for classification. Thus, we only compute the localization losses (RPN loss and bounding box regression loss) on images with ground truth box labels. Below we describe our modified classification loss for image-level labels.

A sample from the weakly labeled dataset $\mathcal{D}^{\text{cls}}$ contains an image $\mathbf{I}$ and a set of K labels $\{c_k\}_{k=1}^{K}$. We use the region proposal network to extract N object features $\{(\mathbf{b}, \mathbf{f}, o)_j\}_{j=1}^{N}$. Prediction-based methods try to assign image labels to regions, and aim to train both localization and classification abilities. Instead, we propose simple ways to use the image labels $\{c_k\}_{k=1}^{K}$ and only improve classification. Our key idea is to use a fixed way to assign image labels to regions, and side-step a complex prediction-based assignment. We

allow the fixed assignment schemes miss certain objects, as long as they miss fewer objects than the prediction-based counterparts, thus leading to better performance.

Non-prediction-based losses. We now describe a variety of simple ways to use image labels and evaluate them empirically in Table 8.1. Our first idea is to use the whole image as a new "proposal" box. We call this loss **image-box**. We ignore all proposals from the RPN, and instead use an injected box of the whole image $\mathbf{b}' = (0, 0, w, h)$. We then apply the classification loss to its RoI features $\mathbf{f}'$ for all classes $c \in \{c_k\}_{k=1}^{K}$:

$$L_{\text{image-box}} = BCE(\mathbf{W}\mathbf{f}', c)$$

where $BCE(s, c) = -log\sigma(s_c) - \sum_{k \neq c} log(1 - \sigma(s_k))$ is the binary cross-entropy loss, and σ is the sigmoid activation. Thus, our loss uses the features from the same 'proposal' for solving the classification problem for all the classes $\{c_k\}$.

In practice, the image-box can be replaced by smaller boxes. We introduce two alternatives: the proposal with the **max object score** or the proposal with the **max size**:

$$L_{\text{max-object-score}} = BCE(\mathbf{W}\mathbf{f}_j, c), j = \text{argmax}_j o_j$$

$$L_{\text{max-size}} = BCE(\mathbf{W}\mathbf{f}_j, c), j = \text{argmax}_j(\text{size}(\mathbf{b}_j))$$

We show that all these three losses can effectively leverage the image-level supervision, while the max-size loss performs the best. We thus use the max-size loss by default for image-supervised data. We also note that the classification

parameters $\mathbf{W}$ are shared across both detection and classification data, which greatly improves detection performance. The overall training objective is

$$L(\mathbf{I}) = \begin{cases} L_{\text{rpn}} + L_{\text{reg}} + L_{\text{cls}}, & \text{if } \mathbf{I} \in \mathcal{D}^{\text{det}} \\ \lambda L_{\text{max-size}}, & \text{if } \mathbf{I} \in \mathcal{D}^{\text{cls}} \end{cases}$$

where L_{rpn}, L_{reg}, L_{cls} are standard losses in a two-stage detector, and $\lambda = 0.1$ is the weight of our loss.

Relation to prediction-based assignments. In traditional weakly-supervised detection [9, 147, 150], a popular idea is to assign the image to the proposals based on model prediction. Let $\mathbf{F} = (\mathbf{f}_1, \ldots, \mathbf{f}_N)$ be the stacked feature of all object proposals and $\mathbf{S} = \mathbf{WF}$ be their classification scores. For each $c \in \{c_k\}_{k=1}^K$, $L = BCE(\mathbf{S}_j, c), j = \mathcal{F}(\mathbf{S}, c)$, where $\mathcal{F}$ is the label-to-box assignment process. In most methods, $\mathcal{F}$ is a function of the prediction $\mathbf{S}$. For example, $\mathcal{F}$ selects the proposal with max score on c. Our key insight is that $\mathcal{F}$ should *not* depend on the prediction $\mathbf{S}$. In large-vocabulary detection, the initial recognition ability of rare or novel classes is low, making the label assignment process inaccurate. Our method side-steps this prediction-and-assignment process entirely and relies on a fixed supervision criteria.

8.2 Experiments

We evaluate Detic on the large-vocabulary object detection dataset LVIS [61]. We mainly use the open-vocabulary setting proposed by Gu *et al.* [60], and also report results on the standard LVIS setting. We describe our experiment setup below.

LVIS. The LVIS [61] dataset has object detection and instance segmentation labels for 1203 classes with 100K images. The classes are divided into three groups - frequent, common, rare based on the number of training images. We refer to this standard LVIS training set as *LVIS-all*. Following ViLD [60], we remove the labels of 337 rare-class from training and consider them as novel classes in testing. We refer to this partial training set with only frequent and common classes as *LVIS-base*. We report mask mAP which is the official metric for LVIS. While our model is developed for box detection, we use a standard class-agnostic mask head [64] to produce segmentation masks for boxes. We train the mask head only on detection data.

Image-supervised data. We use two sources of image-supervised data: ImageNet-21K [34] and Conceptual Captions [162]. ImageNet-21K (IN-21K) contains 14M images for 21K classes. For ease of training and evaluation, most of our experiments use the 997 classes that overlap with the LVIS vocabulary and denote this subset as IN-L. Conceptual Captions [162] (CC) is an image captioning dataset containing 3M images. We extract image labels from the captions using exact text-matching and keep images whose captions mention at least one LVIS class. See § 8.3.2 for results of directly using captions. The resulting dataset contains 1.5M images with 992 LVIS classes. We summarize the datasets used below.

Notation	Definition	#Images	#Classes
LVIS-all	The original LVIS dataset [61]	100K	1203
LVIS-base	LVIS without rare-class annotations	100K	866
IN-21K	The original ImageNet-21K dataset [34]	14M	21k
IN-L	997 overlapping IN-21K classes with LVIS	1.2M	997
CC	Conceptual Captions [162] with LVIS classes	1.5M	992

8.2.1 Implementation details

Box-Supervised: a strong LVIS baseline. We first establish a strong baseline on LVIS to demonstrate that our improvements are orthogonal to recent advances in object detection. The baseline only uses the supervised bounding box labels. We use the CenterNet2 [256] detector with ResNet50 [65] backbone. We use Federated Loss [256] and repeat factor sampling [61]. We use large scale jittering [56] with input resolution 640×640 and train for a $4\times$ ($\sim$48 LVIS epochs) schedule. To show our method is compatible with better pretraining, we use ImageNet-21k pretrained backbone weights [156]. As described in **??**, we use the CLIP [145] embedding as the classifier. Our baseline is 9.1 mAP higher than the detectron2 baseline [216] (31.5 vs. 22.4 mAP^{mask}) and trains in a similar time (17 vs. 12 hours on 8 V100 GPUs). See § 8.3.3 for more details.

Resolution change for image-labeled images. ImageNet images are inherently smaller and more object-focused than LVIS images [239]. In practice, we observe it is important to use smaller image resolution for ImageNet images. Using smaller resolution in addition allows us to increase the batch-size with

the same computation. In our implementation, we use 320×320 for ImageNet and CC and ablate this in § 8.3.4.

Multi-dataset training. We sample detection and classification mini-batches in a 1 : 1 ratio, regardless of the original dataset size. We group images from the same dataset on the same GPU to improve training efficiency [257].

Training schedules. To shorten the experimental cycle and have a good initialization for prediction-based WSOD losses [147, 150], we always first train a converged base-class-only model (4× schedule) and finetune on it with additional image-labeled data for another 4× schedule. We confirm finetuning the model using only box supervision does not improve the performance. The 4× schedule for our joint training consists of ∼24 LVIS epochs plus ∼4.8 ImageNet epochs or ∼3.8 CC epochs. Training our ResNet50 model takes ∼22 hours on 8 V100 GPUs. The large 21K Swin-B model trains in ∼24 hours on 32 GPUs.

8.2.2 Prediction-based *vs* non-prediction-based methods

Table 8.1 shows the results of the box-supervised baseline, existing prediction-based methods, and our proposed non-prediction-based methods. The baseline (Box-Supervised) is trained without access to novel class bounding box labels. It uses the CLIP classifier [60] and has open-vocabulary capabilities with 15.4 $\text{mAP}_{\text{novel}}$. In order to leverage additional image-labeled data like ImageNet or CC, we use prior prediction-based methods or our non-prediction-based method.

We compare a few prediction-based methods that assign image labels

	IN-L (object-centric)		CC (non object-centric)	
	mAP^{mask}	$\text{mAP}^{\text{mask}}_{\text{novel}}$	mAP^{mask}	$\text{mAP}^{\text{mask}}_{\text{novel}}$
Box-Supervised (baseline)	$30.0_{\pm 0.4}$	$16.3_{\pm 0.7}$	$30.0_{\pm 0.4}$	$16.3_{\pm 0.7}$
Prediction-based methods				
Self-training [172]	$30.3_{\pm 0.0}$	$15.6_{\pm 0.1}$	$30.1_{\pm 0.2}$	$15.9_{\pm 0.8}$
WSDDN [9]	$29.8_{\pm 0.2}$	$15.6_{\pm 0.3}$	$30.0_{\pm 0.1}$	$16.5_{\pm 0.8}$
DLWL* [147]	$30.6_{\pm 0.1}$	$18.2_{\pm 0.2}$	$29.7_{\pm 0.3}$	$16.9_{\pm 0.6}$
YOLO9000 [150]	$31.2_{\pm 0.3}$	$20.4_{\pm 0.9}$	$29.4_{\pm 0.1}$	$15.9_{\pm 0.6}$
Non-prediction-based methods				
Detic (Max-object-score)	$32.2_{\pm 0.1}$	$24.4_{\pm 0.3}$	$29.8_{\pm 0.1}$	$18.2_{\pm 0.6}$
Detic (Image-box)	$\mathbf{32.4}_{\pm 0.1}$	$23.8_{\pm 0.5}$	$\mathbf{30.9}_{\pm 0.1}$	$\mathbf{19.5}_{\pm 0.5}$
Detic (Max-size)	$\mathbf{32.4}_{\pm 0.1}$	$\mathbf{24.6}_{\pm 0.3}$	$\mathbf{30.9}_{\pm 0.2}$	$\mathbf{19.5}_{\pm 0.3}$
Fully-supervised (all classes)	$31.1_{\pm 0.4}$	$25.5_{\pm 0.7}$	$31.1_{\pm 0.4}$	$25.5_{\pm 0.7}$

Table 8.1: **Prediction-based *vs* non-prediction-based methods.** We show overall and novel-class mAP on open-vocabulary LVIS [60] (with 866 base classes and 337 novel classes) with different image-labeled datasets (IN-L or CC). The models are trained using our strong baseline § 8.2.1 (top row). This baseline is trained on boxes from the base classes and has non-zero novel-class mAP as it uses the CLIP classifier. All models in the following rows are finetuned from the baseline model and leverage image-labeled data. We repeat experiments for 3 runs and report mean/ std. All variants of our proposed non-prediction-based losses outperform existing prediction-based counterparts.

to proposals based on predictions. Self-training assigns predictions of Box-Supervised as pseudo-labels *offline* with a fixed score threshold (0.5). The other prediction-based methods use different losses to assign predictions to image labels online. See § 8.3.5 for implementation details. For DLWL [147], we implement a simplified version that does not include bootstrapping and refer to it as DLWL*.

Table 8.1 (third block) shows the results of our non-prediction-based methods in § 8.1. All variants of our proposed simpler method outperform the complex prediction-based counterparts, with both image-supervised datasets. On the novel classes, Detic provides a significant gain of $\sim$ 4.2 points with ImageNet over the best prediction-based methods.

Using non-object centric images from Conceptual Captions. ImageNet images typically have a single large object [61]. Thus, our non-prediction-based methods, for example image-box which considers the entire image as a bounding box, are well suited for ImageNet. To test whether our losses work with different image distributions with multiple objects, we test it with the Conceptual Captions (CC) dataset. Even on this challenging dataset with multiple objects/labels per image, Detic provides a gain of $\sim$ 2.6 points on novel class detection over the best prediction-based methods. This suggests that our simpler Detic method can generalize to different types of image-labeled data. Overall, the results from Table 8.1 suggest that complex prediction-based methods that overly rely on model prediction scores do not perform well for open-vocabulary detection. Amongst our non-prediction-based variants, the max-size loss consistently performs the best, and is the default for Detic in our following experiments.

Why does max-size work? Intuitively, our simpler non-prediction methods outperform the complex prediction-based method by side-stepping a hard assignment problem. Prediction-based methods rely on strong initial detections to assign image-level labels to predicted boxes. When the initial predictions

are reliable, prediction-based methods are ideal. However, in open-vocabulary scenarios, such strong initial predictions are absent, which explains the limited performance of prediction-based methods. Detic's simpler assignment does not rely on strong predictions and is more robust under the challenges of open-vocabulary setting.

We now study two additional advantages of the Detic max-size variant over prediction-based methods that may contribute to improved performance: 1) the selected max-size proposal can safely *cover* the target object; 2) the selected max-size proposal is consistent during different training iterations.

Figure 8.3 provides typical qualitative examples of the assigned region for the prediction-based method and our max-size variant. On an annotated subset of IN-L, Detic max-size covers 92.8% target objects, vs. 69.0% for the prediction-based method. Overall, unlike prediction-based methods, Detic's simpler assignment yields boxes that are more likely to contain the object. Indeed, Detic may miss certain objects (especially small objects) or supervise to a loose region. However, in order for Detic to yield a good detector, the selected box need not be perfect, it just needs to 1) provide meaningful training signal (cover the objects and be consistent during training); 2) be 'more correct' than the box selected by the prediction-based method. We provide details about our metrics, more quantitative evaluation, and more discussions in § 8.3.6.

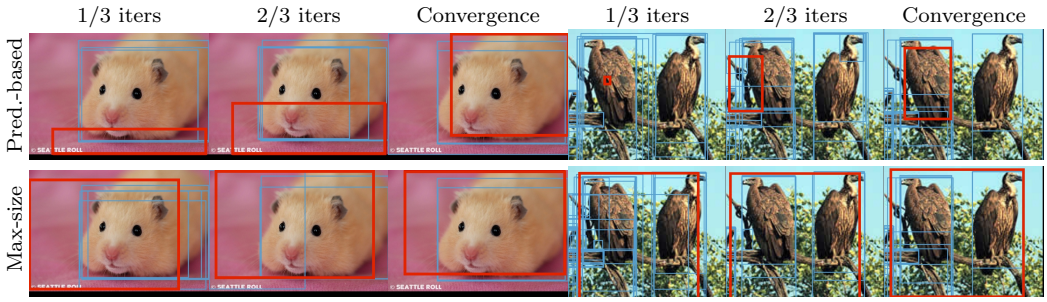


Figure 8.3: **Visualization of the assigned boxes during training.** We show all boxes with score > 0.5 in blue and the assigned (selected) box in red. **Top**: The prediction-based method selects different boxes across training, and the selected box may not cover the objects in the image. **Bottom**: Our simpler max-size variant selects a box that covers the objects and is more consistent across training.

8.2.3 Comparison with a fully-supervised detector

In Table 8.1, compared with the strong baseline Box-Supervised, Detic improves the detection performance by 2.4 mAP and 8.3 $\text{mAP}_{\text{novel}}$. Thus, Detic with image-level labels leads to strong open-vocabulary detection performance

	mAP^{mask}	$\text{mAP}^{\text{mask}}_{\text{novel}}$	$\text{mAP}^{\text{mask}}_{\text{c}}$	$\text{mAP}^{\text{mask}}_{\text{f}}$
ViLD-text [60]	24.9	10.1	23.9	**32.5**
ViLD [60]	22.5	16.1	20.0	28.3
ViLD-ensemble [60]	25.5	16.6	24.6	30.3
Detic	**26.8**	**17.8**	**26.3**	31.6

Table 8.2: **Open-vocabulary LVIS compared to ViLD [60].** We train our model *using their training settings and architecture* (MaskRCNN-ResNet50, training from scratch). We report mask mAP and its breakdown to novel (rare), common, and frequent classes. Variants of ViLD use distillation (ViLD) or ensembling (ViLD-ensemble.). Detic (with IN-L) uses a single model and improves both mAP and $\text{mAP}_{\text{novel}}$.

and can provide orthogonal gains to existing open-vocabulary detectors [5]. To further understand the open-vocabulary capabilities of Detic, we also report the *top-line* results trained with box labels for all classes (Table 8.1 last row). Despite not using box labels for the novel classes, Detic with ImageNet performs favorably compared to the fully-supervised detector. This result also suggests that bounding box annotations may not be required for new classes. Detic combined with large image classification datasets is a simple and effective alternative for increasing detector vocabulary.

8.2.4 Comparison with the state-of-the-art

We compare Detic's open-vocabulary object detectors with state-of-the-art methods on the open-vocabulary LVIS and the open-vocabulary COCO benchmarks. In each case, we strictly follow the architecture and setup from prior work to ensure fair comparisons.

Open-vocabulary LVIS. We compare to ViLD [60], which first uses CLIP embeddings [145] for open-vocabulary detection. We strictly follow their training setup and model architecture (§ 8.3.7) and report results in Table 8.2. Here ViLD-text is exactly our Box-Supervised baseline. Detic provides a gain of 7.7 points on $\text{mAP}_{\text{novel}}$. Compared to ViLD-text, ViLD, which uses knowledge distillation from the CLIP visual backbone, improves $\text{mAP}_{\text{novel}}$ at the cost of hurting overall mAP. Ensembling the two models, ViLD-ens provides improvements for both metrics. On the other hand, Detic uses a single model which improves both novel and overall mAP, and outperforms the ViLD ensemble.

	$\text{mAP50}^{\text{box}}_{\text{all}}$	$\text{mAP50}^{\text{box}}_{\text{novel}}$	$\text{mAP50}^{\text{box}}_{\text{base}}$
Base-only†	39.9	0	**49.9**
Base-only (CLIP)	39.3	1.3	48.7
WSDDN [9]†	24.6	20.5	23.4
Cap2Det [230]†	20.1	20.3	20.1
SB [5]‡	24.9	0.31	29.2
DELO [266]‡	13.0	3.41	13.8
PL [146]‡	27.9	4.12	35.9
OVR-CNN [236]†	39.9	22.8	46.0
Detic	**45.0**	**27.8**	47.1

Table 8.3: **Open-vocabulary COCO [5].** We compare Detic using the same training data and architecture from OVR-CNN [236]. We report box mAP at IoU threshold 0.5 using Faster R-CNN with ResNet50-C4 backbone. Detic builds upon the CLIP baseline (second row) and shows significant improvements over prior work. †: results quoted from OVR-CNN [236] paper or code. ‡: results quoted from the original publications.

Open-vocabulary COCO. Next, we compare with prior works on the popular open-vocabulary COCO benchmark [5] (see benchmark and implementation details in § 8.3.8). We strictly follow OVR-CNN [236] to use Faster R-CNN with ResNet50-C4 backbone and do not use any improvements from § 8.2.1. Following [236], we use COCO captions as the image-supervised data. We extract nouns from the captions and use both the image labels and captions as supervision.

Table 8.3 summarizes our results. As the training set contains only 48 base classes, the base-class only model (second row) yields low mAP on novel classes. Detic improves the baseline and outperforms OVR-CNN [236] by a large margin, using exactly the same model, training recipe, and data.

	Objects365 [160]		OpenImages [90]	
	mAP^{box}	$\text{mAP}^{\text{box}}_{\text{rare}}$	$\text{mAP50}^{\text{box}}$	$\text{mAP50}^{\text{box}}_{\text{rare}}$
Box-Supervised	19.1	14.0	46.2	61.7
Detic w. IN-L	21.2	17.8	53.0	67.1
Detic w. IN-21k	**21.5**	**20.0**	**55.2**	**68.8**
Dataset-specific oracles	31.2	22.5	69.9	81.8

Table 8.4: **Detecting 21K classes across datasets.** We use Detic to train a detector and evaluate it on multiple datasets *without retraining*. We report the bounding box mAP on Objects365 and OpenImages. Compared to the Box-Supervised baseline (trained on LVIS-all), Detic leverages image-level supervision to train robust detectors. The performance of Detic is 70%-80% of dataset-specific models (bottom row) that use dataset specific box labels.

Additionally, similar to Table 8.1, we compare to prior prediction-based methods on the open-vocabulary COCO benchmark in § 8.3.8. In this setting too, Detic improves over prior work providing significant gains on novel class detection and overall detection performance.

8.2.5 Detecting 21K classes across datasets without finetuning

Next, we train a detector with the full 21K classes of ImageNet. We use our strong recipe with Swin-B [115] backbone. In practice, training a classification layer of 21K classes is computationally involved.[2] We adopt a modified Federated Loss [256] that uniformly samples 50 classes from the vocabulary at every iteration. We only compute classification scores and back-propagate on the sampled classes.

[2]This is more pronounced in detection than classification, as the "batch-size" for the classification layer is 512× image-batch-size, where 512 is #RoIs per image.

As there are no direct benchmark to evaluate detectors with such large vocabulary, we evaluate our detectors on new datasets *without finetuning*. We evaluate on two large-scale object detection datasets: Objects365v2 [160] and OpenImages [90], both with around 1.8M training images. We follow LVIS to split $\frac{1}{3}$ of classes with the fewest training images as rare classes. Table 8.4 shows the results. On both datasets, Detic improves the Box-Supervised baseline by a large margin, especially on classes with fewer annotations. Using all the 21k classes further improves performance owing to the large vocabulary. Our single model significantly reduces the gap towards the dataset-specific oracles and reaches 70%-80% of their performance without using the corresponding 1.8M detection annotations. See Figure 8.4 for qualitative results.

Figure 8.4: **Qualitative results of our 21k-class detector.** We show random samples from images containing novel classes in OpenImages (top) and Objects365 (bottom) validation sets. We use the CLIP embedding of the corresponding vocabularies. We show LVIS classes in purple and novel classes in green. We use a score threshold of 0.5 and show the most confident class for each box. Best viewed on screen.

Classifier	Box-Supervised		Detic	
	mAP^{mask}	$\text{mAP}^{\text{mask}}_{\text{novel}}$	mAP^{mask}	$\text{mAP}^{\text{mask}}_{\text{novel}}$
*CLIP [145]	30.2	16.4	32.4	24.9
Trained	27.4	0	31.7	17.4
FastText [75]	27.5	9.0	30.9	19.2
OpenCLIP [72]	27.1	8.9	30.7	19.4

Table 8.5: **Detic with different classifiers.** We vary the classifier used with Detic and observe that it works well with different choices. While CLIP embeddings give the best performance (* indicates our default), all classifiers benefit from our Detic.

8.2.6 Ablation studies

We now ablate our key components under the open-vocabulary LVIS setting with IN-L as the image-classification data. We use our strong training recipe as described in § 8.2.1 for all these experiments.

Classifier weights. We study the effect of different classifier weights $\mathbf{W}$. While our main open-vocabulary experiments use CLIP [145], we show the gain of Detic is independent of CLIP. We train Box-Supervised and Detic with different classifiers, including a standard random initialized and trained classifier, and other *fixed* language models [72,75] The results are shown in Table 8.5. By default, a trained classifier cannot recognize novel classes. However, Detic enables novel class recognition ability even in this setting (17.4 $\text{mAP}_{\text{novel}}$ for classes without detection labels). Using language models such as FastText [75] or an open-source version of CLIP [72] leads to better novel class performance. CLIP [145] performs the best among them.

Effect of Pretraining. Many existing methods use additional data only for

pretraining [35, 236, 239], while we use image-labeled data for co-training. We present results of Detic with different types of pretraining in Table 8.6. Detic provides similar gains across different types of pretraining, suggesting that our gains are orthogonal to advances in pretraining. We believe that this is because pretraining improves the overall features, while Detic uses co-training which improves both the features and the classifier.

8.2.7 The standard LVIS benchmark

Finally, we evaluate Detic on the standard LVIS benchmark [61]. In this setting, the baseline (Box-Supervised) is trained with box and mask labels for all classes while Detic uses additional image-level labels from IN-L. We train Detic with the same recipe in § 8.2.1 and use a strong Swin-B [115] backbone and 896×896 input size. We report the mask mAP across all classes and also split into rare, common, and frequent classes. Notably, Detic achieves 41.7 mAP and 41.7 mAP_r, closing the gap between the overall mAP and the rare mAP. This suggests Detic effectively uses image-level labels to improve

	Pretrain data	mAP^mask	$\text{mAP}^\text{mask}_\text{novel}$
Box-Supervised	IN-1K	26.1	13.6
Detic	IN-1K	28.8 (+2.7)	21.7 (+8.1)
Box-Supervised	IN-21K	30.2	16.4
Detic	IN-21K	32.4 (+2.2)	24.9 (+8.5)

Table 8.6: **Detic with different pretraining data.** Top: our method using ImageNet-1K as pretraining and ImageNet-21K as co-training; Bottom: using ImageNet-21K for both pretraining and co-training. Co-training helps pretraining in both cases.

	Backbone	mAP^{mask}	mAP^{mask}_r	mAP^{mask}_c	mAP^{mask}_f
MosaicOS† [239]	ResNeXt-101	28.3	21.7	27.3	32.4
CenterNet2 [256]	ResNeXt-101	34.9	24.6	34.7	42.5
AsyncSLL† [62]	ResNeSt-269	36.0	27.8	36.7	39.6
SeesawLoss [203]	ResNeSt-200	37.3	26.4	36.3	**43.1**
Copy-paste [56]	EfficientNet-B7	38.1	32.1	37.1	41.9
Tan et al. [185]	ResNeSt-269	38.8	28.5	39.5	42.7
Baseline	Swin-B	40.7	35.9	40.5	**43.1**
Detic†	Swin-B	**41.7**	**41.7**	**40.8**	42.6

Table 8.7: **Standard LVIS.** We evaluate our baseline (Box-Supervised) and Detic using different backbones on the LVIS dataset. We report the mask mAP. We also report prior work on LVIS using large backbone networks (single-scale testing) for references (not for apple-to-apple comparison). †: detectors using additional data. Detic improves over the baseline with increased gains for the rare classes.

the performance of classes with very few boxes labels. § 8.3.9 provides more comparisons to prior work [239] on LVIS. § 8.3.10 shows Detic generalizes to DETR-based [269] detectors.

8.3 Additional experiments

8.3.1 Region proposal quality

In this section, we show the region proposal network trained on LVIS [61] is satisfactory and generalizes well to new classes by default. We experiment under our strong baseline in § 8.2.1. Table 8.8a shows the proposal recalls with or without rare classes in training. First, we observe the recall gaps between the two models on rare classes are small (79.7 vs. 78.5); second, the

	AR_r50@100	AR_r50@300	AR_r50@1k	AR50@1k
LVIS-all	63.3	76.3	79.7	80.9
LVIS-base	62.2	76.2	78.5	81.0

(a) **Proposal networks trained with (top) and without (bottom) rare classes.** We report recalls on rare classes and all classes at IoU threshold 0.5 with different number of proposals. Proposal networks trained *without* rare classes can generalize to rare classes in testing.

	$AR_{\text{half-1st}}$50@1k	$AR_{\text{half-2nd}}$50@1k
LVIS-half-1st	80.8	69.6
LVIS-half-2nd	62.9	82.2

(b) **Proposal networks trained on half of the LVIS classes.** We report recalls at IoU threshold 0.5 on the other half classes. Proposal networks produce non-trivial recalls on novel classes.

Table 8.8: **Proposal network generalization ability evaluation. (a)**: Generalize from 866 LVIS base classes to the 337 rare classes; **(b)**: Generalize from uniformly sampled half LVIS classes (601/ 602 classes) to the other half.

gaps between rare classes and all classes are small (79.7 vs. 80.9); third, the absolute recall is relatively high ($\sim$80%, note recall at IoU threshold 0.5 can be translated into oracle mAP-pool [31] given perfect classifier and regressor). All observations indicate the proposals have good generalization abilities to new classes even though they are supervised to background during training. We consider the proposal generalization is currently not the performance bottleneck in open-vocabulary detection. This especially the case as modern detectors use an over-sufficient number of proposals in testing (1K proposals for $<$ 20 objects per image). Our observations are consistent with ViLD [60].

We in addition evaluate a more strict setting, where we uniformly split

	Supervision	mAP^{mask}	$\text{mAP}^{\text{mask}}_{\text{novel}}$
Box-Supervised	-	30.2	16.4
Detic w. CC	Image label	**31.0**	19.8
Detic w. CC	Caption	30.4	17.4
Detic w. CC	Both	**31.0**	**21.3**
		$\text{mAP50}^{\text{box}}_{\text{all}}$	$\text{mAP50}^{\text{box}}_{\text{novel}}$
Box-Supervised	-	39.3	1.3
Detic w. COCO-cap.	Image label	44.7	24.1
Detic w. COCO-cap.	Caption	43.8	21.0
Detic w. COCO-cap.	Both	**45.0**	**27.8**

Table 8.9: **Direct caption supervision.** Top: Open-vocabulary LVIS with Conceptual Caption as weakly-labeled data; Bottom block: Open-vocabulary COCO with COCO-caption as weakly-labeled data. Directly using caption embeddings as a classifier is helpful on both benchmarks; the improvements are complementary to Detic.

LVIS classes into two halves. I.e., we use classes ID $1, 3, 5, \cdots$ as the first half, and the rest as the second half. These two subsets have completely different definitions of "objects". We then train a proposal network on each of them, and evaluate on both subsets. As shown in Table 8.8b, the proposal networks give non-trivial recalls at the complementary other half (69.6% over 82.2% percent of the full generalizability). This again supports proposal networks trained on a diverse vocabulary learned a general concept of objects.

8.3.2 Direct captions supervision

As we are using a language model CLIP [145] as the classifier, our framework can seamlessly incorporate the free-form caption text as image-

supervision. Using the notations in § 8.1, here $\mathcal{D}^{\text{cls}} = \{(\mathbf{I}, t)_i\}$ where t is a free-form text. In our open-vocabulary detection formulation, text t can natrually be converted to an embedding by the CLIP [145] language encoder $\mathcal{L}$: $w = \mathcal{L}(t)$. Given a minibatch of B samples $\{(\mathbf{I}, t)_i\}_{i=1}^{B}$, we compose a dynamic classification layer by stacking all caption features within the batch $\widetilde{\mathbf{W}} = \mathcal{L}(\{t_i\}_{i=1}^{B})$. For the i-th image in the minibatch, its "classification" label is the i-th text, and other texts are negative samples. We use the injected whole image box to extract RoI feature $\mathbf{f}'_i$ for image i. We use the same binary cross entropy loss as classifying image labels:

$$L_{cap} = \sum_{i=1}^{B} BCE(\widetilde{\mathbf{W}}\mathbf{f}'_i, i)$$

We do not back-propagate into the language encoder.

We evaluate the effectiveness of the caption loss in Table 8.9 on both open-vocabulary LVIS and COCO (see dataset details in § 8.3.8). We compare individually applying the max-size loss for image labels and the caption loss, and applying both of them. Both image labels and captions can improve both overall mAP and novel class mAP. Combining both losses gives a more significant improvement. Our open-vocabulary COCO results in Table 8.3 uses both the max-size loss and the caption loss.

	mAP^{box}	mAP^{box}_r	mAP^{mask}	mAP^{mask}_r	T
D2 baseline [216]	22.9	11.3	22.4	11.6	12h
+Class-agnostic box&mask	22.3	10.1	21.2	10.1	12h
+Federated loss [256]	27.0	20.2	24.6	18.2	12h
+CenterNet2 [256]	30.7	22.9	26.8	19.4	13h
+LSJ 640×640, 4× sched. [56]	31.0	21.6	27.2	20.1	17h
+CLIP classifier [145]	31.5	24.2	28	22.5	17h
+Adam optimizer, lr2e-4 [84]	30.4	23.6	26.9	21.4	17h
+IN-21k pretrain [156]*	35.3	28.2	31.5	25.6	17h
+Input size 896×896	37.1	29.5	33.2	26.9	25h
+Swin-B backbone [115]	45.4	39.9	40.7	35.9	43h
*Remove rare class ann. [60]	33.8	17.6	30.2	16.4	17h

Table 8.10: **LVIS baseline evolution.** First row: the configuration from the detectron2 model zoo. The following rows change components one by one. Last row: removing rare classes from the "+IN-21k pretrain*" row. The two gray-filled rows are the baselines in our main paper, for full LVIS and open-vocabulary LVIS, respectively. We show rough wall-clock training times (T) on our machine with 8 V100 GPUs in the last column.

8.3.3 LVIS baseline details

We first describe the standard LVIS baseline from the detectron2 model zoo[3]. This baseline uses ResNet-50 FPN backbone and a 2× training schedule ($180k$ iterations with batch-size 16)[4]. Data augmentation includes horizontal flip and random resize short side [640, 800], long side < 1333. The baseline uses SGD optimizer with a learning rate 0.02 (dropped by 10× at $120k$ and $160k$ iteration). The bounding box regression head and the mask head are

[3] https://github.com/facebookresearch/detectron2/blob/main/configs/LVISv1-InstanceSegmentation/mask_rcnn_R_50_FPN_1x.yaml

[4] We are aware different projects use different notations of a 1× schedule. In this paper we always refer 1× schedule to $16 \times 90k$ images

class-specific.

Table 8.10 shows the roadmap from the detectron2 baseline to our baseline (§ 8.2.1). First, we prepare the model for new classes by making the box and mask heads class-agnostic. This slightly hurts performance. We then use Federated loss [256] and upgrade the detector to CenterNet2 [256] (i.e., replacing RPN with CenterNet and multiplying proposal score to classification score). Both modifications improve mAP and mAP_rsignificantly, and CenterNet2 slightly increases the training time.

Next, we use the EfficientDet [56, 186] style large-scale jittering and train a longer schedule (4×). To balance the training time, we also reduce the training image size to 640×640 (the testing size is unchanged at 800×1333) and increase batch-size to 64 (with the learning rate scaled up to 0.08). The resulting augmentation and schedule is slightly better than the default multi-scale training, with 30% more training time. A longer schedule is beneficial when using more data, and can be improved by larger resolution.

Next, we switch in the CLIP classifier [145]. We follow ViLD [60] to L2 normalize the embedding and RoI feature before dot-product. Note CenterNet2 uses a cascade classifier [14]. We use CLIP for all of them. Using CLIP classifier improves rare class mAP.

Finally, we use an ImageNet-21k pretrained ResNet-50 model from Ridnik *et al.* [156]. We remark the ImageNet-21k pretrained model requires using Adam optimizer (with learning rate $2e$-4). Combing all the improvements re-

	Ratio	Size	mAP^{mask}	$\text{mAP}^{\text{mask}}_{\text{novel}}$
Bos-Supervised	1: 0	-	30.2	16.4
Detic w. IN-L	1: 1	640	30.9	23.3
Detic w. IN-L	1: 1	320	32.0	24.0
Detic w. IN-L	1: 4	640	31.1	23.5
Detic w. IN-L	1: 4	320	**32.4**	**24.9**
Detic w. CC	1: 1	640	30.8	21.6
Detic w. CC	1: 1	320	30.8	21.5
Detic w. CC	1: 4	640	30.7	21.0
Detic w. CC	1: 4	320	**31.1**	**21.8**

Table 8.11: **Ablations of the resolution change.** We report mask mAP on the open-vocabulary LVIS following the setting of Table 8.1. Top: ImageNet as the image-labeled data. Bottom: CC as the image-labeled data.

sults in 35.3 mAP^{box} and 31.5 mAP^{mask}, and trains in a favorable time (17h on 8 V100 GPUs). We use this model as our baseline in the main paper.

Increasing the training resolution or using a larger backbone [115] can further increase performance significantly, at a cost of longer training time. We use the large models only when compared to the state-of-the-art models.

8.3.4 Resolution change for classification data

Table 8.11 ablates the resolution change in § 8.2.1. Using a smaller input resolution improves ~ 1 point for both mAP and $\text{mAP}_{\text{novel}}$ with ImageNet, but does not impact much with CC. Using more batches for the weak datasets is slightly better than a $1:1$ ratio.

8.3.5 WSOD losses implementation details

Following the notations in § 8.1, we implement the prediction-based WSOD losses as below:

WSDDN [9] learns a soft weight on the proposals to weight-sum the proposal classification scores into a single image classification score:

$$L_{\text{WSDDN}} = BCE(\sum_j (\text{softmax}(\mathbf{W}'\mathbf{F})_j * \mathbf{S}_j), c)$$

where $\mathbf{W}'$ is a learnable network parameter.

Predicted [150] selects the proposal with the max predicted score on class c:

$$L_{\text{Predicted}} = BCE(\mathbf{S}_j, c), j = \text{argmax}_j \mathbf{S}_{jc}$$

DLWL* [147] first runs a clustering algorithm with IoU threshold 0.5. Let $\mathcal{J}$ be the set of peaks of each cluster (i.e., the proposal within the cluster and has the max predicted score on class c), We then select the top $N_c = 3$ peaks with the highest prediction scores on class c.

$$L_{\text{DLWL}^*} = \frac{1}{N_c} \sum_{t=1}^{N_c} BCE(\mathbf{S}_{j_t}, c),$$
$$j_t = \text{argmax}_{j \in \mathcal{J}, j \neq \{j_1, \ldots, j_{t-1}\}} \mathbf{S}_{jc}$$

The original DLWL [147] in addition upgrades $\mathbf{S}$ using an IoU-based assignment matrix from self-training and bootstrapping (See their Section 3.2). In our implementation, we did not include this part, as our goal is to only compare the training losses.

	Dataset	Backbone	mAP^{mask}	$\text{mAP}^{\text{mask}}_{\text{novel}}$
Box-Supervised			30.2	16.4
Predicted	LVIS-base	Res50	31.2	20.4
Max-size			32.4 (+1.2)	24.6 (+4.2)
Box-Supervised			38.4	21.9
Predicted	LVIS-base	SwinB	40.0	31.7
Max-size			40.7 (+0.7)	33.8 (+2.1)
Box-Supervised			31.5	25.6
Predicted	LVIS-all	Res50	32.5	28.4
Max-size			33.2 (+0.7)	29.7 (+1.3)
Box-Supervised			40.7	35.9
Predicted	LVIS-all	SwinB	40.6	39.8
Max-size			41.3 (+0.7)	40.9 (+1.1)

(a) **Predicted loss and max-size loss with different prediction qualities.** We show the mask mAP of the box-supervised baseline, Predicted loss [150], and our max-size loss. We show the delta between max-size loss and predicted loss in green. Improving the backbone and including rare classes in training can both narrow the gap. Max-size consistently performs better.

	Cover rate		Consistency		
	IN-L	COCO	IN-L	CC	COCO
Predicted	69.0	73.8	71.5	30.0	57.7
Max-size	92.8	80.0	87.9	73.0	62.8

(b) **Assigned proposal cover rate and consistency.** Left: ratio of assigned proposal covering the ground truth both. We evaluate on an ImageNet subset that has box ground truth and the annotated COCO training set; Right: average assigned bounding box IoU of between the final model and the half-schedule model.

Table 8.12: **Comparison between predicted loss and and max-size loss. (a)**: comparison under different baselines. **(b)**: comparison in customized metrics.

8.3.6 More comparison between prediction-based and non-prediction-based methods

Our non-prediction-based losses perform significantly better than prediction-based losses as is shown in Table 8.1. In this section, we take the max-size loss and the predicted-loss as the representitives and conduct more detailed comparisons between them. A straightforward reason is that the predicted loss requires a good initial prediction to guide the pseudo-label-based training. However in the open-vocabulary detection setting the initial predictions are inherently flawed. To verify this, in Table 8.12a, we show both improving the backbone and including rare classes in training can narrow the gap. However in the current performance regime, our max-size loss performs better.

We highlight two additional advantages of the max-size loss that may contribute to the good performance: (1) the max-size loss is a safe approximation of object regions; (2) the max-size loss is consistent during training. Figure 8.3 provides qualitative examples of the assigned region for the predicted loss and the max-size loss. First, we observe that while being coarse at the boundary, the max-size loss can *cover* the target object in most cases. Second, the assigned regions of the predicted loss are usually different across training iterations, especially in the early phase where the model predictions are unstable. On the contrary, max-size loss supervises consistent regions across training iterations.

Table 8.12b quantitatively evaluates these two properties. We use the ground truth box annotation in the full COCO detection dataset and a subset

of ImageNet with bounding box annotation [5] to evaluate the cover rate. We define cover rate as the ratio of image labels whose ground-truth box has > 0.5 intersection-over-area with the assigned region. We define the consistency metric as the average assigned-region IoU of the same image between the 1/2 schedule and the final schedule. Table 8.12b shows max-size loss is more favorable than predicted loss on these two metrics. However we highlight that these two metrics alone do not always correlate to the final performance, as the **image-box** loss is perfect on both metrics but underperforms max-size loss.

8.3.7 ViLD baseline details

The baseline in ViLD [60] is very different from detectron2. They use MaskRCNN detector [64] with Res50-FPN backbone, but trains the network from scratch without ImageNet pretraining. They use large-scale jittering [56] with input resolution 1024×1024 and train a $32\times$ schedule. The optimizer is SGD with batch size 256 and learning rate 0.32. We first reproduce their baselines (both the oracle detector and ViLD-text) under the same setting. We observe half of their schedule ($16\times$) is sufficient to closely match their numbers. The half training schedule takes 4 days on 4 nodes (each with 8 V100 GPUs). We then finetune another $16\times$ schedule using ImageNet data with our max-size loss.

[5] https://image-net.org/download-bboxes.php. 213K of the 1.2M IN-L images have bounding box annotations.

8.3.8 Open-vocabulary COCO benchmark details

Open-vocabulary COCO is proposed by Bansal et al. [5]. They manually select 48 classes from the 80 COCO classes as base classes, and 17 classes as novel classes. The training set is the same as the full COCO, but only images containing at least one base class are used. During testing, we report results under the "generalized zero-shot detection" setting [5], where all COCO validation images are used.

We strictly follow the literatures [5, 146, 236] to use FasterRCNN [153] with ResNet50-C4 backbone and the $1\times$ training schedule ($90k$ iterations). We use horizontal flip as the only data augmentation in training and keep the input resolution fixed to 800×1333 in both training and testing. We use SGD optimizer with a learning rate 0.02 (dropped by $10\times$ at $60k$ and $80k$ iteration) and batch size 16. The evaluation metric on open-vocabulary COCO is box mAP at IoU threshold 0.5. Our reproduced baseline matches OVR-CNN [236]. Our model is finetuned on the baseline model with another $1\times$ schedule. We sample detection data and image-supervised data in a $1:1$ ratio.

Table 8.13 repeats the experiments in Table 8.1 on open-vocabulary COCO. The observations are consistent: our proposed non-prediction-based methods outperform existing prediction-based counterparts, and the max-size loss performs the best among our variants.

	$\text{mAP50}^{\text{box}}_{\text{all}}$	$\text{mAP50}^{\text{box}}_{\text{novel}}$
Box-Supervised (base cls)	39.3	1.3
Self-training [172]	39.5	1.8
WSDDN [9]	39.9	5.9
DLWL* [147]	42.9	19.6
Predicted [150]	41.9	18.7
Detic (Max-object-score)	43.3	20.4
Detic (Image-box)	43.4	21.0
Detic (Max-size)	**44.7**	**24.1**
Box-Supervised (all cls)	54.9	60.0

Table 8.13: **Different ways to use image supervision on open-vocabulary COCO.** The models are trained using ResNet50-C4 [5]. We follow setups in Table 8.1. The observations are consistent with LVIS.

	mAP^{mask}	$\text{mAP}^{\text{mask}}_{\text{r}}$
Box-Supervised [239]	22.6	12.3
MosaicOS [239]	24.5 (+1.9)	18.3 (+6.0)
Box-Supervised (Reproduced)	22.6	12.3
Detic (default classifier)	25.1 (+2.5)	18.6 (+6.3)
Box-Supervised (CLIP classifier)	22.3	14.1
Detic (CLIP classifier)	**24.9 (+2.6)**	**20.7 (+6.5)**

Table 8.14: **Standard LVIS compared to MosiacOS [239].** Top block: results quoted from MosiacOS paper; Middle block: Detic with the default random intialized and trained classifier; Bottom block: Detic with CLIP classifier.

8.3.9 Compare to MosaicOS [239]

MosaicOS [239] first uses image-level annotations to improve LVIS detectors. We compare to MosaicOS [239] by strictly following their baseline setup (without any improvements in § 8.2.1). The detailed hyper-parameters

	mAP^{box}	mAP^{box}_r	mAP^{box}_c	mAP^{box}_f
Box-Supervised	31.7	21.4	30.7	**37.5**
Detic	**32.5**	**26.2**	**31.3**	36.6

Table 8.15: **Detic applied to Deformable-DETR [269].** We report Box mAP on full LVIS. Our method improves Deformable-DETR.

follow the detectron2 baseline as described in § 8.3.3. We finetune on the Box-supervised model with an additional 2× schedule with Adam optimizer. Table 8.14 shows our re-trained baseline exactly matches their reported results from the paper. Our method is developed based on the CLIP classifier, and we also report our baseline with CLIP. The baseline has slightly lower mAP and higher mAP_r. MosaicOS uses IN-L and additional web-search images as image-supervised data. Detic outperforms MosaicOS [239] in mAP and mAP_r, without using their multi-stage training and mosaic augmentation. Our relative improvements over the baseline are slightly higher than MosiacOS [239]. We highlight our training framework is simpler and we use less additional training data (Google-searched images).

8.3.10 Generalization to Deformable-DETR.

We apply Detic to the recent Transformer based Deformable-DETR [269] to study its generalization. We use their default training recipe, Federated Loss [256] and train for a 4× schedule (∼ 48 LVIS epochs). We apply the image supervision to the query from the encoder with the max predicted size. Table 8.15 shows that Detic improves over the baseline (+0.8 mAP and +4.8 mAP_r) and generalizes to Transformer based detectors.

8.3.11 Improvements breakdown to classes

Table 8.16 shows mAP breakdown into classes with and without image labels for both the Box-Supervised baseline and Detic. As expected, most of the improvements are from classes with image-level labels. On ImageNet, Detic also improves classes without image labels thanks to the CLIP classifier which leverages inter-class relations.

8.3.12 $\text{mAP}^{\text{Fixed}}$ evaluation

Table 8.17 compares our improvements under the new mAP^{fix} proposed in Dave *et al.* [31]. Our improvements are consistent under the new metric.

Limitations and Conclusions We present Detic which is a simple way to use image supervision in large-vocabulary object detection. While Detic is simpler than prior assignment-based weakly-supervised detection methods, it supervises all image labels to the same region and does not consider

	mAP^{mask}	$\text{mAP}^{\text{mask}}_{\text{IN-L}}$	$\text{mAP}^{\text{mask}}_{\text{non-IN-L}}$
Box-Supervised	30.2	30.6	27.6
Max-size	32.4	33.5	28.1
	mAP^{mask}	$\text{mAP}^{\text{mask}}_{\text{CC}}$	$\text{mAP}^{\text{mask}}_{\text{non-CC}}$
Box-Supervised	30.2	30.1	29.5
Max-size	30.9	31.7	28.6

Table 8.16: **mAP breakdown into classes with and without image labels.** Top: Detic trained on ImageNet. Bottom: Detic trained on CC. Most of the improvements are from classes with image-level labels. On ImageNet Detic also improves classes without image labels thanks to the CLIP classifier.

Datasets	mAP^{box}	$\text{mAP}^{\text{box}}_{\text{novel}}$	$\text{mAP}^{\text{Fixed}}$	$\text{mAP}^{\text{Fixed}}_{\text{novel}}$
Box-Supervised	30.2	16.4	31.2	18.2
Detic	32.4 (+2.2)	24.9 (+8.5)	33.4 (+2.3)	26.7 (+8.5)

Table 8.17: **$\text{mAP}^{\text{Fixed}}$ evaluation**. Middle: the original box mAP metric used in the main paper. Right: the new box mAP^{Fix} metric. Our improvements are consistent under the new metric.

overall dataset statistics. We leave incorporating such information for future work. The generalization capabilities of Detic also benefit from the large-scale pretraining of CLIP, and it is an open question whether there are other ways to train the classifier. Moreover, open vocabulary generalization has no guarantees in extreme domains. Our experiments show Detic improves large-vocabulary detection with various weak data sources, classifiers, detector architectures, and training recipes. We hope Detic makes object detection easier to deploy in the real world.

Chapter 9

Future Works and broader impacts

In the preceding chapters, I presented my completed works on the goal of a unified computer vision model, including a unified object representation (CenterNet in Chapter 4), a unified framework for detection and tracking (CenterTrack in Chapter 5 and GTR in Chapter 6), and a unified vocabulary for detection and classification data (UniDet inin Chapter 7 and Detic in Chapter 8). However, there are still missing components and unaddressed problems towards the goal in Figure 8.1. In this chapter, I outline a few potential future research directions that I will pursue beyond my thesis.

9.1 Mixing annotations from different datasets

Our point-based detector serves as a unified framework for many object-related tasks. In our practice, we train different network weights for different tasks on their specific datasets. While it simplified the training procedure for each task, it misses to the opportunity to leverage the interplay of different tasks. An exciting future direction is leveraging techniques in multi-task learning to train a unified computer vision model for all object-based tasks. For example, a system can learn about object categorization from one dataset

(e.g. MSCOCO), 3D layout from another (e.g. nuScenes), and human pose from a third (e.g. CMU Panoptic), and so on. This inspires techniques in semi-supervised training framework or progressively label propagation.

9.2 Object-based video representation

The current dominant video models are all based on stack of pixels in all frames [3, 43, 46, 116]. This is not only expensive in computation but also wastes the temporal cues between the frames. Objects and their trajectories are rich and compact abstractions in a scene. Given the progress in object detection and tracking in recent years (outlined in this thesis), it is possible to use class-agnostic object features as an intermediate representation of a video. A strong off-the-shelf class-agnostic object tracker can hopefully produce trajectory proposals as metadata for all videos on the internet. This metadata can inspire a video recognition framework that highlights the object regions or tubes in the video, instead of processing all pixels. This can hopefully significantly speed up video recognition systems.

I hope my research will make object recognition easier to use for general users: our generalized detection framework can be used for detecting anomalies in medical images, for analyzing traffic to make driving safer, and for helping teach children to recognize novel objects. I hope all users can use computer vision models without any customized retraining.

Bibliography

[1] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. 2d human pose estimation: New benchmark and state of the art analysis. In *CVPR*, 2014. 1

[2] Pablo Arbeláez, Jordi Pont-Tuset, Jonathan T Barron, Ferran Marques, and Jitendra Malik. Multiscale combinatorial grouping. In *CVPR*, 2014. 18

[3] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. Vivit: A video vision transformer. In *ICCV*, 2021. 154

[4] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv:1607.06450*, 2016. 84

[5] Ankan Bansal, Karan Sikka, Gaurav Sharma, Rama Chellappa, and Ajay Divakaran. Zero-shot object detection. In *ECCV*, 2018. xvi, xvii, 18, 28, 119, 131, 132, 148, 149

[6] Jerome Berclaz, Francois Fleuret, Engin Turetken, and Pascal Fua. Multiple object tracking using k-shortest paths optimization. *TPAMI*, 2011. 77

[7] Philipp Bergmann, Tim Meinhardt, and Laura Leal-Taixe. Tracking without bells and whistles. In *ICCV*, 2019. 14, 15, 25, 54, 66, 67, 68, 76

[8] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *ICIP*, 2016. xiv, 13, 25, 53, 68, 73, 76, 90, 91

[9] Hakan Bilen and Andrea Vedaldi. Weakly supervised deep detection networks. In *CVPR*, 2016. xx, 18, 20, 121, 123, 127, 132, 144, 149

[10] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv:2004.10934*, 2020. 19, 50

[11] Navaneeth Bodla, Bharat Singh, Rama Chellappa, and Larry S Davis. Soft-nms—improving object detection with one line of code. In *ICCV*, 2017. 9

[12] Guillem Brasó and Laura Leal-Taixé. Learning a neural solver for multiple object tracking. In *CVPR*, 2020. 15, 25, 77, 89

[13] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A multimodal dataset for autonomous driving. In *CVPR*, 2020. 1, 2, 30, 32, 35, 42, 46, 56, 61, 62, 63, 65, 67, 69

[14] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *CVPR*, 2018. 22, 46, 47, 51, 87, 106, 142

[15] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields. In *arXiv:1812.08008*, 2018. 1, 30

[16] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *CVPR*, 2017. 9, 11, 23, 30, 35, 44, 45

[17] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020. 5, 15, 21, 78, 82, 84, 88, 92

[18] Joao Carreira, Pulkit Agrawal, Katerina Fragkiadaki, and Jitendra Malik. Human pose estimation with iterative error feedback. In *CVPR*, 2016. 12, 55

[19] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 2017. 2

[20] Florian Chabot, Mohamed Chaouch, Jaonary Rabarisoa, Celine Teuliere, and Thierry Chateau. Deep manta: A coarse-to-fine many-task network for joint 2d and 3d vehicle analysis from monocular image. In *CVPR*, 2017. 10

[21] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 1

[22] Nadine Chang, Zhiding Yu, Yu-Xiong Wang, Anima Anandkumar, Sanja Fidler, and Jose M Alvarez. Image-level or object-level? a tale of two resampling strategies for long-tailed detection. *ICML*, 2021. 19

[23] Liangyu Chen, Tong Yang, Xiangyu Zhang, Wei Zhang, and Jian Sun. Points as queries: Weakly semi-supervised object detection by points. In *CVPR*, 2021. 19

[24] Yilun Chen, Zhicheng Wang, Yuxiang Peng, Zhiqiang Zhang, Gang Yu, and Jian Sun. Cascaded pyramid network for multi-person pose estimation. In *CVPR*, 2018. 23

[25] Yuntao Chen, Chenxia Han, Naiyan Wang, and Zhaoxiang Zhang. Revisiting feature alignment for one-stage object detection. *arXiv:1908.01570*, 2019. 40

[26] Wongun Choi and Silvio Savarese. Multiple target tracking in world coordinate with single, minimally calibrated camera. In *ECCV*, 2010. 14

[27] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016. 105

[28] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, 2017. 105

[29] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *ICCV*, 2017. 29, 87, 89

[30] Peng Dai, Renliang Weng, Wongun Choi, Changshui Zhang, Zhangping He, and Wei Ding. Learning a proposal classifier for multiple object tracking.

CVPR, 2021. 15

[31] Achal Dave, Piotr Dollár, Deva Ramanan, Alexander Kirillov, and Ross Girshick. Evaluating large-vocabulary object detectors: The devil is in the details. *arXiv:2102.01066*, 2021. 138, 151

[32] Achal Dave, Tarasha Khurana, Pavel Tokmakov, Cordelia Schmid, and Deva Ramanan. Tao: A large-scale benchmark for tracking any object. In *ECCV*, 2020. 2, 5, 25, 79, 86, 87, 90, 91, 92

[33] Achal Dave, Pavel Tokmakov, and Deva Ramanan. Towards segmenting anything that moves. In *ICCVW*, 2019. 20

[34] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 8, 117, 118, 119, 124, 125

[35] Karan Desai and Justin Johnson. VirTex: Learning Visual Representations from Textual Annotations. In *CVPR*, 2021. 20, 136

[36] Bowen Dong, Zitong Huang, Yuelin Guo, Qilong Wang, Zhenxing Niu, and Wangmeng Zuo. Boosting weakly supervised object detection via learning bounding box adjusters. In *ICCV*, 2021. 19

[37] Zhiwei Dong, Guoxuan Li, Yue Liao, Fei Wang, Pengju Ren, and Chen Qian. Centripetalnet: Pursuing high-quality keypoint pairs for object detection. In *CVPR*, 2020. 51

[38] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. 5, 78

[39] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. Centernet: Object detection with keypoint triplets. *ICCV*, 2019. 10, 39, 40, 51

[40] Kaiwen Duan, Lingxi Xie, Honggang Qi, Song Bai, Qingming Huang, and Qi Tian. Corner proposal network for anchor-free, two-stage object detection. *ECCV*, 2020. 51

[41] Georgios D Evangelidis and Emmanouil Z Psarakis. Parametric image alignment using enhanced correlation coefficient maximization. *TPAMI*, 2008. 14

[42] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *ICCV*, 2010. 17, 26, 105, 106

[43] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. In *ICCV*, 2021. 154

[44] Kuan Fang, Yu Xiang, Xiaocheng Li, and Silvio Savarese. Recurrent autoregressive networks for online multi-object tracking. In *WACV*, 2018. 13

[45] Shijie Fang, Yuhang Cao, Xinjiang Wang, Kai Chen, Dahua Lin, and Wayne Zhang. Wssod: A new pipeline for weakly-and semi-supervised object detection. *arXiv:2105.11293*, 2021. 19, 119

[46] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *ICCV*, 2019. 154

[47] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Detect to track and track to detect. In *ICCV*, 2017. 14, 54

[48] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. In *TPAMI*, 2009. 13, 53

[49] Chengjian Feng, Yujie Zhong, and Weilin Huang. Exploring classification equilibrium in long-tailed object detection. In *ICCV*, 2021. 19

[50] Weitao Feng, Zhihao Hu, Wei Wu, Junjie Yan, and Wanli Ouyang. Multi-object tracking with multiple cues and switcher-aware classification. *arXiv:1901.06129*, 2019. 13, 67, 68

[51] Mihai Fieraru, Anna Khoreva, Leonid Pishchulin, and Bernt Schiele. Learning to refine human pose estimation. In *CVPR Workshops*, 2018. 12

[52] Davi Frossard and Raquel Urtasun. End-to-end learning of multi-sensor 3D tracking by detection. In *ICRA*, 2018. 25

[53] Shanghua Gao, Ming-Ming Cheng, Kai Zhao, Xin-Yu Zhang, Ming-Hsuan Yang, and Philip HS Torr. Res2net: A new multi-scale backbone architecture. *TPAMI*, 2019. 47, 50, 87

[54] Yuan Gao, Hui Shen, Donghong Zhong, Jian Wang, Zeyu Liu, Ti Bai, Xiang Long, and Shilei Wen. A solution for densely annotated large scale object detection task. 2019. 27, 116

[55] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012. 1, 10, 30, 56, 61, 62, 63, 65, 67, 68, 105

[56] Golnaz Ghiasi, Yin Cui, Aravind Srinivas, Rui Qian, Tsung-Yi Lin, Ekin D Cubuk, Quoc V Le, and Barret Zoph. Simple copy-paste is a strong data augmentation method for instance segmentation. In *CVPR*, 2021. 125, 137, 141, 142, 147

[57] Golnaz Ghiasi, Xiuye Gu, Yin Cui, and Tsung-Yi Lin. Open-vocabulary image segmentation. *arXiv:2112.12143*, 2021. 18

[58] Ross Girshick. Fast r-cnn. In *ICCV*, 2015. 9, 24, 29

[59] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 8, 10, 29

[60] Xiuye Gu, Tsung-Yi Lin, Weicheng Kuo, and Yin Cui. Open-vocabulary object detection via vision and language knowledge distillation. *ICLR*, 2022.

xv, 18, 20, 28, 119, 123, 124, 126, 127, 130, 131, 138, 141, 142, 147

[61] Agrim Gupta, Piotr Dollar, and Ross Girshick. Lvis: A dataset for large vocabulary instance segmentation. In *CVPR*, 2019. 17, 19, 86, 87, 88, 117, 118, 119, 123, 124, 125, 128, 136, 137

[62] Jianhua Han, Minzhe Niu, Zewei Du, Longhui Wei, Lingxi Xie, Xiaopeng Zhang, and Qi Tian. Joint coco and lvis workshop at eccv 2020: Lvis challenge track technical report: Asynchronous semi-supervised learning for large vocabulary instance segmentation. 2020. 137

[63] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *CVPR*, 2017. xii, 10, 23, 29, 40, 44, 45, 77, 79, 89, 96

[64] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017. 124, 147

[65] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 8, 10, 36, 106, 125

[66] Jan Hendrik Hosang, Rodrigo Benenson, and Bernt Schiele. Learning non-maximum suppression. In *CVPR*, 2017. 29

[67] Hou-Ning Hu, Qi-Zhi Cai, Dequan Wang, Ji Lin, Min Sun, Philipp Krähenbühl, Trevor Darrell, and Fisher Yu. Joint monocular 3d vehicle detection and tracking. *ICCV*, 2019. 34, 54, 61, 68

[68] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *CVPR*, 2018. 116

[69] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, et al. Speed/accuracy trade-offs for modern convolutional object detectors. In *CVPR*, 2017. 29

[70] Zeyi Huang, Yang Zou, Vijayakumar Bhagavatula, and Dong Huang. Comprehensive attention self-distillation for weakly-supervised object detection. *NeurIPS*, 2020. 18, 119

[71] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *CVPR*, 2017. 74

[72] Gabriel Ilharco, Mitchell Wortsman, Nicholas Carlini, Rohan Taori, Achal Dave, Vaishaal Shankar, Hongseok Namkoong, John Miller, Hannaneh Hajishirzi, Ali Farhadi, and Ludwig Schmidt. Openclip, July 2021. 135

[73] Borui Jiang, Ruixuan Luo, Jiayuan Mao, Tete Xiao, and Yuning Jiang. Acquisition of localization confidence for accurate object detection. In *ECCV*, 2018. 29

[74] Hanbyul Joo, Tomas Simon, Xulong Li, Hao Liu, Lei Tan, Lin Gui, Sean Banerjee, Timothy Scott Godisart, Bart Nabbe, Iain Matthews, Takeo Kanade, Shohei Nobuhara, and Yaser Sheikh. Panoptic studio: A massively multiview system for social interaction capture. *TPAMI*, 2017. 2

[75] Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve

Jégou, and Tomas Mikolov. Fasttext. zip: Compressing text classification models. *arXiv:1612.03651*, 2016. 135

[76] Aishwarya Kamath, Mannat Singh, Yann LeCun, Ishan Misra, Gabriel Synnaeve, and Nicolas Carion. Mdetr–modulated detection for end-to-end multimodal understanding. *ICCV*, 2021. 20

[77] Kai Kang, Hongsheng Li, Tong Xiao, Wanli Ouyang, Junjie Yan, Xihui Liu, and Xiaogang Wang. Object detection in videos with tubelet proposal networks. In *CVPR*, 2017. 14

[78] Kai Kang, Hongsheng Li, Junjie Yan, Xingyu Zeng, Bin Yang, Tong Xiao, Cong Zhang, Zhe Wang, Ruohui Wang, Xiaogang Wang, et al. T-cnn: Tubelets with convolutional neural networks for object detection from videos. *Circuits and Systems for Video Technology*, 2017. 14

[79] Hasith Karunasekera, Han Wang, and Handuo Zhang. Multiple object tracking with attention to appearance, structure, motion and size. *IEEE Access*, 2019. 68

[80] Wei Ke, Tianliang Zhang, Zeyi Huang, Qixiang Ye, Jianzhuang Liu, and Dong Huang. Multiple anchor learning for visual object detection. In *CVPR*, 2020. 51

[81] Margret Keuper, Siyu Tang, Bjorn Andres, Thomas Brox, and Bernt Schiele. Motion segmentation & multiple object tracking by correlation co-clustering. *TPAMI*, 2018. 66

[82] Dahun Kim, Tsung-Yi Lin, Anelia Angelova, In So Kweon, and Weicheng Kuo. Learning open-world object proposals without learning to classify. *arXiv:2108.06753*, 2021. 20

[83] Kang Kim and Hee Seok Lee. Probabilistic anchor assignment with iou prediction for object detection. In *ECCV*, 2020. 22, 51

[84] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 37, 64, 141

[85] Sachin Konan, Kevin J Liang, and Li Yin. Extending one-stage detection with open-world proposals. *arXiv:2201.02302*, 2022. 20

[86] Tao Kong, Fuchun Sun, Huaping Liu, Yuning Jiang, and Jianbo Shi. Foveabox: Beyond anchor-based object detector. *arXiv:1904.03797*, 2019. 41

[87] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 9

[88] Abhijit Kundu, Yin Li, and James M. Rehg. 3d-rcnn: Instance-level 3d object reconstruction via render-and-compare. In *CVPR*, 2018. 10

[89] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Alexander Kolesnikov, Tom Duerig, and Vittorio Ferrari. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *IJCV*, 2020. 26, 96, 98, 100, 104, 106, 108

[90] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Alexander Kolesnikov, et al. The open images dataset v4. *IJCV*, 2020. 117, 118, 120, 133, 134

[91] John Lambert, Zhuang Liu, Ozan Sener, James Hays, and Vladlen Koltun. MSeg: A composite dataset for multi-domain semantic segmentation. In *CVPR*, 2020. 5, 16, 97, 100, 105

[92] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *CVPR*, 2019. 32, 43

[93] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In *ECCV*, 2018. 10, 23, 24, 31, 33, 36, 39, 40, 41, 51

[94] Hei Law, Yun Teng, Olga Russakovsky, and Jia Deng. Cornernet-lite: Efficient keypoint based object detection. *arXiv:1904.08900*, 2019. 38, 40, 41

[95] Laura Leal-Taixé, Cristian Canton-Ferrer, and Konrad Schindler. Learning by tracking: Siamese cnn for robust target association. In *CVPR Workshops*, 2016. 13

[96] Laura Leal-Taixé, Anton Milan, Konrad Schindler, Daniel Cremers, Ian Reid, and Stefan Roth. Tracking the trackers: an analysis of the state of the art in multiple object tracking. *arXiv:1704.02781*, 2017. 63

[97] Boyi Li, Kilian Q Weinberger, Serge Belongie, Vladlen Koltun, and René Ranftl. Language-driven semantic segmentation. *ICLR*, 2022. 18

[98] Sijin Li and Antoni B Chan. 3d human pose estimation from monocular images with deep convolutional neural network. In *ACCV*, 2014. 11

[99] Xiang Li, Wenhai Wang, Xiaolin Hu, Jun Li, Jinhui Tang, and Jian Yang. Generalized focal loss v2: Learning reliable localization quality estimation for dense object detection. *arXiv preprint*, 2020. 51

[100] Xiang Li, Wenhai Wang, Lijun Wu, Shuo Chen, Xiaolin Hu, Jun Li, Jinhui Tang, and Jian Yang. Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection. *NeurIPS*, 2020. 48, 51

[101] Xiaoyan Li, Meina Kan, Shiguang Shan, and Xilin Chen. Weakly supervised object detection with segmentation collaboration. In *ICCV*, 2019. 18

[102] Yan Li, Junge Zhang, Kaiqi Huang, and Jianguo Zhang. Mixed supervised object detection with robust objectness transfer. *TPAMI*, 2018. 19

[103] Yanghao Li, Yuntao Chen, Naiyan Wang, and Zhaoxiang Zhang. Scale-aware trident networks for object detection. *ICCV*, 2019. 29, 40, 41, 51

[104] Yu Li, Tao Wang, Bingyi Kang, Sheng Tang, Chunfeng Wang, Jintao Li, and Jiashi Feng. Overcoming classifier imbalance for long-tail object detection with balanced group softmax. In *CVPR*, 2020. 19

[105] Zhihui Li, Lina Yao, Xiaoqin Zhang, Xianzhi Wang, Salil Kanhere, and Huaxiang Zhang. Zero-shot object detection with textual descriptions. In *AAAI*, 2019. 18

[106] Chao Liang, Zhipeng Zhang, Yi Lu, Xue Zhou, Bing Li, Xiyong Ye, and Jianxiao Zou. Rethinking the competition between detection and reid in multi-object tracking. *arXiv:2010.12138*, 2020. 93

[107] Tsung-Yi Lin, Piotr Dollár, Ross B Girshick, Kaiming He, Bharath Hariharan, and Serge J Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 41, 47

[108] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. *ICCV*, 2017. xii, 9, 10, 21, 22, 24, 29, 40, 46, 48, 82

[109] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014. 1, 2, 8, 17, 25, 38, 44, 87, 96, 98, 104, 117

[110] Jeffrey T Linderoth and Ted K Ralphs. Noncommercial software for mixed-integer linear programming. *Integer programming: theory and practice*, 3(253-303):144–189, 2005. 103

[111] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *CVPR*, 2018. 29, 40, 41

[112] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *ECCV*, 2016. 9

[113] Yan Liu, Zhijie Zhang, Li Niu, Junjie Chen, and Liqing Zhang. Mixed supervised object detection by transferringmask prior and semantic similarity. In *NeurIPS*, 2021. 19

[114] Yen-Cheng Liu, Chih-Yao Ma, Zijian He, Chia-Wen Kuo, Kan Chen, Peizhao Zhang, Bichen Wu, Zsolt Kira, and Peter Vajda. Unbiased teacher for semi-supervised object detection. *ICLR*, 2021. 119

[115] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *arXiv:2103.14030*, 2021. 78, 133, 136, 141, 143

[116] Ze Liu, Jia Ning, Yue Cao, Yixuan Wei, Zheng Zhang, Stephen Lin, and Han Hu. Video swin transformer. *CVPR*, 2022. 154

[117] Ze Liu, Zheng Zhang, Yue Cao, Han Hu, and Xin Tong. Group-free 3D object detection via transformers. *arXiv:2104.00678*, 2021. 82

[118] Chen Long, Ai Haizhou, Zhuang Zijie, and Shang Chong. Real-time multiple people tracking with deeply learned candidate selection and person re-identification. In *ICME*, 2018. 66

[119] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 30

[120] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2017. 88

[121] Jonathon Luiten, Tobias Fischer, and Bastian Leibe. Track to reconstruct and reconstruct to track. *arXiv:1910.00130*, 2019. 68

[122] Jonathon Luiten, Aljosa Osep, Patrick Dendorfer, Philip Torr, Andreas Geiger, Laura Leal-Taixé, and Bastian Leibe. Hota: A higher order metric for evaluating multi-object tracking. *IJCV*, 2021. 87

[123] Hao Luo, Youzhi Gu, Xingyu Liao, Shenqi Lai, and Wei Jiang. Bag of tricks and a strong baseline for deep person re-identification. In *CVPR Workshops*, 2019. 85

[124] Muhammad Maaz, Hanoona Rasheed, Salman Khan, Fahad Shahbaz Khan, Rao Muhammad Anwer, and Ming-Hsuan Yang. Multi-modal transformers excel at class-agnostic object detection. *arXiv:2111.11430*, 2021. 20

[125] Tim Meinhardt, Alexander Kirillov, Laura Leal-Taixe, and Christoph Feichtenhofer. Trackformer: Multi-object tracking with transformers. *arXiv:2101.02702*. 15, 78, 79, 92, 93, 94

[126] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler. MOT16: A benchmark for multi-object tracking. *arXiv:1603.00831*, 2016. 55, 61, 62, 63, 64, 67, 79, 86, 87, 90, 96

[127] Gyeongsik Moon, Juyong Chang, and Kyoung Mu Lee. Posefix: Model-agnostic general human pose refinement network. In *CVPR*, 2019. 12

[128] Arsalan Mousavian, Dragomir Anguelov, John Flynn, and Jana Kosecka. 3d bounding box estimation using deep learning and geometry. In *CVPR*, 2017. 10, 31, 34

[129] Gerhard Neuhold, Tobias Ollmann, Samuel Rota Bulò, and Peter Kontschieder. The mapillary vistas dataset for semantic understanding of street scenes. In *ICCV*, 2017. 96, 104, 113

[130] Alejandro Newell, Zhiao Huang, and Jia Deng. Associative embedding: End-to-end learning for joint detection and grouping. In *NIPS*, 2017. 9, 11, 23, 30, 35, 44, 45

[131] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *ECCV*, 2016. 11, 23, 24, 30, 31, 36, 47

[132] Xuecheng Nie, Jiashi Feng, Jianfeng Zhang, and Shuicheng Yan. Single-stage multi-person pose machines. In *ICCV*, 2019. 44, 45

[133] Tai-Yu Pan, Cheng Zhang, Yandong Li, Hexiang Hu, Dong Xuan, Soravit Changpinyo, Boqing Gong, and Wei-Lun Chao. On model calibration for long-tailed object detection and instance segmentation. *NeurIPS*, 2021. 19

[134] Jiangmiao Pang, Linlu Qiu, Xia Li, Haofeng Chen, Qi Li, Trevor Darrell, and Fisher Yu. Quasi-dense similarity learning for multiple object tracking. In *CVPR*, 2021. 79, 91, 92, 93

[135] George Papandreou, Tyler Zhu, Liang-Chieh Chen, Spyros Gidaris, Jonathan Tompson, and Kevin Murphy. Personlab: Person pose estimation and in-

stance segmentation with a bottom-up, part-based, geometric embedding model. *ECCV*, 2018. 23, 35, 44, 45

[136] George Papandreou, Tyler Zhu, Nori Kanazawa, Alexander Toshev, Jonathan Tompson, Chris Bregler, and Kevin Murphy. Towards accurate multi-person pose estimation in the wild. In *CVPR*, 2017. 23

[137] Georgios Pavlakos, Xiaowei Zhou, Aaron Chan, Konstantinos G Derpanis, and Kostas Daniilidis. 6-dof object pose from semantic keypoints. In *ICRA*, 2017. 11, 23

[138] Georgios Pavlakos, Xiaowei Zhou, Konstantinos G Derpanis, and Kostas Daniilidis. Coarse-to-fine volumetric prediction for single-image 3d human pose. In *CVPR*, 2017. 11

[139] Jinlong Peng, Changan Wang, Fangbin Wan, Yang Wu, Yabiao Wang, Ying Tai, Chengjie Wang, Jilin Li, Feiyue Huang, and Yanwei Fu. Chained-tracker: Chaining paired attentive regression results for end-to-end joint multiple-object detection and tracking. In *ECCV*, 2020. 93

[140] Junran Peng, Xingyuan Bu, Ming Sun, Zhaoxiang Zhang, Tieniu Tan, and Junjie Yan. Large-scale object detection in the wild from imbalanced multi-labels. In *CVPR*, 2020. 27, 108

[141] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, 2014. 18, 111, 113

[142] Pedro O Pinheiro and Ronan Collobert. Weakly supervised semantic segmentation with convolutional networks. In *CVPR*, 2015. 19

[143] Charles R Qi, Yin Zhou, Mahyar Najibi, Pei Sun, Khoa Vo, Boyang Deng, and Dragomir Anguelov. Offboard 3d object detection from point cloud sequences. In *CVPR*, 2021. 1

[144] Siyuan Qiao, Liang-Chieh Chen, and Alan Yuille. Detectors: Detecting objects with recursive feature pyramid and switchable atrous convolution. *CVPR*, 2021. 51, 115, 116

[145] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. *arXiv:2103.00020*, 2021. 18, 27, 28, 125, 131, 135, 139, 140, 141, 142

[146] Shafin Rahman, Salman Khan, and Nick Barnes. Improved visual-semantic alignment for zero-shot object detection. In *AAAI*, 2020. 18, 132, 148

[147] Vignesh Ramanathan, Rui Wang, and Dhruv Mahajan. Dlwl: Improving detection for lowshot classes with weakly labelled data. In *CVPR*, 2020. xx, 19, 119, 121, 123, 126, 127, 144, 149

[148] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *TPAMI*, 2020. 16

[149] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016. 9, 29

[150] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *CVPR*, 2017. xx, 9, 17, 19, 119, 121, 123, 126, 127, 144, 145, 149

[151] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv:1804.02767*, 2018. 9, 21, 40, 41

[152] Jimmy Ren, Xiaohao Chen, Jianbo Liu, Wenxiu Sun, Jiahao Pang, Qiong Yan, Yu-Wing Tai, and Li Xu. Accurate single stage detector using recurrent rolling convolution. In *CVPR*, 2017. 13

[153] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. xii, xiii, 1, 9, 10, 13, 14, 21, 22, 29, 42, 46, 53, 67, 77, 81, 96, 121, 148

[154] Zhongzheng Ren, Zhiding Yu, Xiaodong Yang, Ming-Yu Liu, Alexander G Schwing, and Jan Kautz. Ufo^2: A unified framework towards omni-supervised object detection. In *ECCV*, 2020. 19

[155] Stephan R Richter, Zeeshan Hayder, and Vladlen Koltun. Playing for benchmarks. In *ICCV*, 2017. 105

[156] Tal Ridnik, Emanuel Ben-Baruch, Asaf Noy, and Lihi Zelnik-Manor. Imagenet-21k pretraining for the masses. In *NeurIPS*, 2021. 125, 141, 142

[157] Amir Sadeghian, Alexandre Alahi, and Silvio Savarese. Tracking the untrackable: Learning to track multiple cues with long-term dependencies. In *ICCV*, 2017. 65

[158] Samuel Schulter, Paul Vernaza, Wongun Choi, and Manmohan Chandraker. Deep network flow for multi-object tracking. In *CVPR*, 2017. 13

[159] Chaobing Shan, Chunbo Wei, Bing Deng, Jianqiang Huang, Xian-Sheng Hua, Xiaoliang Cheng, and Kewei Liang. Tracklets predicting based adaptive graph tracking. *arXiv:2010.09015*, 2020. 93

[160] Shuai Shao, Zeming Li, Tianyuan Zhang, Chao Peng, Gang Yu, Xiangyu Zhang, Jing Li, and Jian Sun. Objects365: A large-scale, high-quality dataset for object detection. In *ICCV*, 2019. 26, 96, 98, 104, 117, 120, 133, 134

[161] Shuai Shao, Zijian Zhao, Boxun Li, Tete Xiao, Gang Yu, Xiangyu Zhang, and Jian Sun. Crowdhuman: A benchmark for detecting human in a crowd. *arXiv:1805.00123*, 2018. 65, 67, 89, 105

[162] Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut. Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *ACL*, 2018. 20, 124, 125

[163] Sarthak Sharma, Junaid Ahmed Ansari, J Krishna Murthy, and K Madhava Krishna. Beyond pixels: Leveraging geometry and shape cues for online multi-object tracking. In *ICRA*, 2018. 13, 68

[164] Li Shen, Zhouchen Lin, and Qingming Huang. Relay backpropagation for effective learning of deep convolutional neural networks. In *ECCV*, 2016. 27, 108

[165] Yunhang Shen, Rongrong Ji, Yan Wang, Zhiwei Chen, Feng Zheng, Feiyue Huang, and Yunsheng Wu. Enabling deep residual networks for weakly supervised object detection. In *ECCV*, 2020. 18

[166] Yunhang Shen, Rongrong Ji, Yan Wang, Yongjian Wu, and Liujuan Cao. Cyclic guidance for weakly supervised joint detection and segmentation. In *CVPR*, 2019. 18

[167] Jianbo Shi and Carlo Tomasi. Good features to track. In *CVPR*, 1994. 53

[168] Andrea Simonelli, Samuel Rota Rota Bulò, Lorenzo Porzi, Manuel López-Antequera, and Peter Kontschieder. Disentangling monocular 3D object detection. In *ICCV*, 2019. 10, 32, 43, 65, 69

[169] Bharat Singh and Larry S Davis. An analysis of scale invariance in object detection–snip. In *CVPR*, 2018. 10, 29

[170] Bharat Singh, Hengduo Li, Abhishek Sharma, and Larry S Davis. R-fcn-3000 at 30fps: Decoupling detection and classification. In *CVPR*, 2018. 19

[171] Bharat Singh, Mahyar Najibi, and Larry S Davis. SNIPER: Efficient multi-scale training. *NIPS*, 2018. 29, 40, 41

[172] Kihyuk Sohn, Zizhao Zhang, Chun-Liang Li, Han Zhang, Chen-Yu Lee, and Tomas Pfister. A simple semi-supervised learning framework for object detection. *arXiv:2005.04757*, 2020. 127, 149

[173] Jeany Son, Mooyeol Baek, Minsu Cho, and Bohyung Han. Multi-object tracking with quadruplet convolutional neural networks. In *CVPR*, 2017. 65

[174] Guanglu Song, Yu Liu, and Xiaogang Wang. Revisiting the sibling head in object detector. In *CVPR*, 2020. 51, 116

[175] Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *CVPR*, 2015. 2, 17

[176] Rainer Stiefelhagen, Keni Bernardin, Rachel Bowers, John Garofolo, Djamel Mostefa, and Padmanabhan Soundararajan. The CLEAR 2006 evaluation. In *CLEAR*, 2006. 63

[177] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. In *CVPR*, 2019. 44, 46

[178] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *CVPR*, 2020. 1, 46

[179] Peize Sun, Jinkun Cao, Yi Jiang, Rufeng Zhang, Enze Xie, Zehuan Yuan, Changhu Wang, and Ping Luo. Transtrack: Multiple-object tracking with transformer. *arXiv:2012.15460*, 2020. 15, 78, 89, 93, 94

[180] Peize Sun, Rufeng Zhang, Yi Jiang, Tao Kong, Chenfeng Xu, Wei Zhan, Masayoshi Tomizuka, Lei Li, Zehuan Yuan, Changhu Wang, et al. Sparse r-cnn: End-to-end object detection with learnable proposals. *CVPR*, 2021. 82

[181] Xiao Sun, Jiaxiang Shang, Shuang Liang, and Yichen Wei. Compositional human pose regression. In *ICCV*, 2017. 35

[182] Zhiqing Sun, Shengcao Cao, Yiming Yang, and Kris Kitani. Rethinking transformer-based set prediction for object detection. *arXiv:2011.10881*, 2020. 82

[183] Jingru Tan, Xin Lu, Gang Zhang, Changqing Yin, and Quanquan Li. Equalization loss v2: A new gradient balance approach for long-tailed object detection. In *CVPR*, 2021. 19

[184] Jingru Tan, Changbao Wang, Buyu Li, Quanquan Li, Wanli Ouyang, Changqing Yin, and Junjie Yan. Equalization loss for long-tailed object recognition. In *CVPR*, 2020. 19, 27

[185] Jingru Tan, Gang Zhang, Hanming Deng, Changbao Wang, Lewei Lu, Quanquan Li, and Jifeng Dai. 1st place solution of lvis challenge 2020: A good box is not a guarantee of a good mask. *arXiv:2009.01559*, 2020. 137

[186] Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. In *CVPR*, 2020. xiii, 46, 47, 49, 50, 51, 88, 89, 142

[187] Peng Tang, Xinggang Wang, Song Bai, Wei Shen, Xiang Bai, Wenyu Liu, and Alan Yuille. Pcl: Proposal cluster learning for weakly supervised object detection. *TPAMI*, 2018. 18, 19, 119

[188] Siyu Tang, Mykhaylo Andriluka, Bjoern Andres, and Bernt Schiele. Multiple people tracking by lifted multicut and person re-identification. In *CVPR*, 2017. 13, 15, 18, 53, 54, 65, 68, 77

[189] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. FCOS: Fully convolutional one-stage object detection. In *ICCV*, 2019. xii, 21, 22, 40, 41, 46, 47, 51, 65

[190] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: A simple and strong anchor-free object detector. *TPAMI*, 2020. 49

[191] Carlo Tomasi and Takeo Kanade. Detection and tracking of point features. *Technical Report CMU-CS-91-132, Carnegie Mellon University*, 1991. 53

[192] Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, and Christoph Bregler. Efficient object localization using convolutional networks. In *CVPR*, 2015. 11

[193] Jonathan J Tompson, Arjun Jain, Yann LeCun, and Christoph Bregler. Joint training of a convolutional network and a graphical model for human pose estimation. In *NIPS*, 2014. 11

[194] Alexander Toshev and Christian Szegedy. Deeppose: Human pose estimation via deep neural networks. In *CVPR*, 2014. 11, 35

[195] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. *arXiv:2012.12877*, 2020. 78

[196] Zhuowen Tu. Auto-context and its application to high-level vision tasks. In *CVPR*, 2008. 12

[197] Shubham Tulsiani and Jitendra Malik. Viewpoints and keypoints. In *CVPR*, 2015. 11

[198] Jasper Uijlings, Stefan Popov, and Vittorio Ferrari. Revisiting knowledge transfer for training object class detectors. In *CVPR*, 2018. 19

[199] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *IJCV*, 2013. 8, 9, 18, 42

[200] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017. xix, 5, 78, 85

[201] Fang Wan, Chang Liu, Wei Ke, Xiangyang Ji, Jianbin Jiao, and Qixiang Ye. C-mil:continuation multiple instance learning for weakly supervised object detection. In *CVPR*, 2019. 18

[202] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Scaled-yolov4: Scaling cross stage partial network. *arXiv preprint arXiv:2011.08036*, 2020. 46, 47, 51

[203] Jiaqi Wang, Wenwei Zhang, Yuhang Zang, Yuhang Cao, Jiangmiao Pang, Tao Gong, Kai Chen, Ziwei Liu, Chen Change Loy, and Dahua Lin. Seesaw loss for long-tailed instance segmentation. In *CVPR*, 2021. 19, 137

[204] Qiang Wang, Yun Zheng, Pan Pan, and Yinghui Xu. Multiple object tracking with correlation learning. *CVPR*, 2021. 92, 93

[205] Yongxin Wang, Kris Kitani, and Xinshuo Weng. Joint object detection and multi-object tracking with graph neural networks. In *ICRA*, 2021. 92, 93

[206] Yuqing Wang, Zhaoliang Xu, Xinlong Wang, Chunhua Shen, Baoshan Cheng, Hao Shen, and Huaxia Xia. End-to-end video instance segmentation with transformers. In *CVPR*, 2021. 78

[207] Zhongdao Wang, Liang Zheng, Yixuan Liu, and Shengjin Wang. Towards real-time multi-object tracking. In *ECCV*, 2020. xiv, 5, 16, 26, 76, 85, 107

[208] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Convolutional pose machines. In *CVPR*, 2016. 11

[209] Xinshuo Weng and Kris Kitani. A baseline for 3D multi-object tracking. *arXiv:1907.03961*, 2019. xiii, 63, 68, 69, 73

[210] Nicolai Wojke and Alex Bewley. Deep cosine metric learning for person re-identification. In *WACV*, 2018. 76

[211] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *ICIP*, 2017. 13, 73

[212] Jiajun Wu, Yifan Wang, Tianfan Xue, Xingyuan Sun, William T Freeman, and Joshua B Tenenbaum. MarrNet: 3D Shape Reconstruction via 2.5D Sketches. In *NIPS*, 2017. 11

[213] Jiajun Wu, Tianfan Xue, Joseph J Lim, Yuandong Tian, Joshua B Tenenbaum, Antonio Torralba, and William T Freeman. Single image 3d interpreter network. In *ECCV*, 2016. 11

[214] Jialian Wu, Jiale Cao, Liangchen Song, Yu Wang, Ming Yang, and Junsong Yuan. Track to detect and segment: An online multi-object tracker. In *CVPR*, 2021. 93

[215] Jialian Wu, Liangchen Song, Tiancai Wang, Qian Zhang, and Junsong Yuan. Forest r-cnn: Large-vocabulary long-tailed object detection and instance segmentation. In *ACM Multimedia*, 2020. 19

[216] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019. xii, 26, 40, 87, 106, 107, 125, 141

[217] Yu Xiang, Alexandre Alahi, and Silvio Savarese. Learning to track: Online multi-object tracking by decision making. In *ICCV*, 2015. 54

[218] Bin Xiao, Haiping Wu, and Yichen Wei. Simple baselines for human pose estimation and tracking. In *ECCV*, 2018. 23, 31, 36, 37

[219] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017.

29

[220] Jiarui Xu, Yue Cao, Zheng Zhang, and Han Hu. Spatial-temporal relation networks for multi-object tracking. In *ICCV*, 2019. 13, 53, 54, 68, 76

[221] Mengde Xu, Zheng Zhang, Han Hu, Jianfeng Wang, Lijuan Wang, Fangyun Wei, Xiang Bai, and Zicheng Liu. End-to-end semi-supervised object detection with soft teacher. *ICCV*, 2021. 119

[222] Yihong Xu, Yutong Ban, Guillaume Delorme, Chuang Gan, Daniela Rus, and Xavier Alameda-Pineda. Transcenter: Transformers with dense queries for multiple-object tracking. *arXiv:2103.15145*, 2021. 79, 92, 93

[223] Ziang Yan, Jian Liang, Weishen Pan, Jin Li, and Changshui Zhang. Weakly- and semi-supervised object detection with expectation-maximization algorithm. *arXiv:1702.08740*, 2017. 19

[224] Fan Yang, Wongun Choi, and Yuanqing Lin. Exploit all the layers: Fast and accurate cnn object detector with scale dependent pooling and cascaded rejection classifiers. In *CVPR*, 2016. 13

[225] Gengshan Yang, Joshua Manela, Michael Happold, and Deva Ramanan. Hierarchical deep stereo matching on high-resolution images. In *CVPR*, 2019. 16

[226] Hao Yang, Hao Wu, and Hao Chen. Detecting 11k classes: Large scale object detection without fine-grained bounding boxes. In *ICCV*, 2019. 19

[227] Ke Yang, Dongsheng Li, and Yong Dou. Towards precise end-to-end weakly supervised object detection network. In *ICCV*, 2019. 18

[228] Wei Yang, Shuang Li, Wanli Ouyang, Hongsheng Li, and Xiaogang Wang. Learning feature pyramids for human pose estimation. In *ICCV*, 2017. 11

[229] Ze Yang, Shaohui Liu, Han Hu, Liwei Wang, and Stephen Lin. Reppoints: Point set representation for object detection. *ICCV*, 2019. 40, 41, 51

[230] Keren Ye, Mingda Zhang, Adriana Kovashka, Wei Li, Danfeng Qin, and Jesse Berent. Cap2det: Learning to amplify weak caption supervision for object detection. In *ICCV*, 2019. 20, 132

[231] Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl. Center-based 3d object detection and tracking. *CVPR*, 2021. 4, 46

[232] Fengwei Yu, Wenbo Li, Quanquan Li, Yu Liu, Xiaohua Shi, and Junjie Yan. Poi: Multiple object tracking with high performance detection and appearance feature. In *ECCV Workshops*, 2016. 65, 68

[233] Fisher Yu, Dequan Wang, Evan Shelhamer, and Trevor Darrell. Deep layer aggregation. In *CVPR*, 2018. 31, 36, 37, 64, 88, 89

[234] Qian Yu, Gérard Medioni, and Isaac Cohen. Multiple target tracking using spatio-temporal markov chain monte carlo data association. In *CVPR*, 2007. 77

[235] Lu Yuan, Dongdong Chen, Yi-Ling Chen, Noel Codella, Xiyang Dai, Jianfeng Gao, Houdong Hu, Xuedong Huang, Boxin Li, Chunyuan Li, et al. Florence: A new foundation model for computer vision. *arXiv:2111.11432*, 2021. 1

[236] Alireza Zareian, Kevin Dela Rosa, Derek Hao Hu, and Shih-Fu Chang. Open-vocabulary object detection using captions. In *CVPR*, 2021. xvi, 18, 20, 120, 132, 136, 148

[237] Oliver Zendel, Katrin Honauer, Markus Murschitz, Daniel Steininger, and Gustavo Fernandez Dominguez. Wilddash-creating hazard-aware benchmarks. In *ECCV*, 2018. 105

[238] Fangao Zeng, Bin Dong, Tiancai Wang, Cheng Chen, Xiangyu Zhang, and Yichen Wei. End-to-end multiple-object tracking with transformer. *arXiv:2105.03247*, 2021. 15, 79, 92, 93

[239] Cheng Zhang, Tai-Yu Pan, Yandong Li, Hexiang Hu, Dong Xuan, Soravit Changpinyo, Boqing Gong, and Wei-Lun Chao. Mosaicos: A simple and effective use of object-centric images for long-tailed object detection. *ICCV*, 2021. xvii, 19, 125, 136, 137, 149, 150

[240] Hang Zhang, Chongruo Wu, Zhongyue Zhang, Yi Zhu, Zhi Zhang, Haibin Lin, Yue Sun, Tong He, Jonas Mueller, R Manmatha, et al. Resnest: Split-attention networks. *arXiv: 2004.08955*, 2020. 47, 51, 109, 115, 116

[241] Li Zhang, Yuan Li, and Ramakant Nevatia. Global data association for multi-object tracking using network flows. In *CVPR*, 2008. 25, 77

[242] Shifeng Zhang, Cheng Chi, Yongqiang Yao, Zhen Lei, and Stan Z Li. Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection. In *CVPR*, 2020. 22, 46, 47, 48, 51

[243] Shifeng Zhang, Longyin Wen, Xiao Bian, Zhen Lei, and Stan Z. Li. Single-shot refinement neural network for object detection. In *CVPR*, 2018. 40

[244] Songyang Zhang, Zeming Li, Shipeng Yan, Xuming He, and Jian Sun. Distribution alignment: A unified framework for long-tail visual recognition. In *CVPR*, 2021. 19

[245] Wenwei Zhang, Hui Zhou, Shuyang Sun, Zhe Wang, Jianping Shi, and Chen Change Loy. Robust multi-modality multi-object tracking. In *ICCV*, 2019. 68

[246] Xiaosong Zhang, Fang Wan, Chang Liu, Rongrong Ji, and Qixiang Ye. Freeanchor: Learning to match anchors for visual object detection. In *Neural Information Processing Systems*, 2019. 51

[247] Yifu Zhang, Chunyu Wang, Xinggang Wang, Wenjun Zeng, and Wenyu Liu. Fairmot: On the fairness of detection and re-identification in multiple object tracking. *arXiv:2004.01888*, 2020. xiv, 25, 76, 85, 88, 90, 91, 93

[248] Zheng Zhang, Dazhi Cheng, Xizhou Zhu, Stephen Lin, and Jifeng Dai. Integrated object detection and tracking with tracklet-conditioned detection. *arXiv:1811.11167*, 2018. 14, 65, 66

[249] Hengshuang Zhao, Jiaya Jia, and Vladlen Koltun. Exploring self-attention for image recognition. In *CVPR*, 2020. 78

[250] Xiangyun Zhao, Samuel Schulter, Gaurav Sharma, Yi-Hsuan Tsai, Manmohan Chandraker, and Ying Wu. Object detection with a unified label space from multiple datasets. In *ECCV*, 2020. 17, 97, 100

[251] Yuanyi Zhong, Jianfeng Wang, Jian Peng, and Lei Zhang. Boosting weakly supervised object detection with progressive knowledge transfer. In *ECCV*. Springer, 2020. 19

[252] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsupervised learning of depth and ego-motion from video. In *CVPR*, 2017. 11

[253] Xingyi Zhou, Qixing Huang, Xiao Sun, Xiangyang Xue, and Yichen Wei. Towards 3d human pose estimation in the wild: A weakly-supervised approach. In *ICCV*, 2017. 11

[254] Xingyi Zhou, Arjun Karpur, Linjie Luo, and Qixing Huang. Starmap for category-agnostic keypoint and viewpoint estimation. In *ECCV*, 2018. 23, 30

[255] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. Tracking objects as points. *ECCV*, 2020. 4, 25, 86, 88, 89, 93

[256] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. Probabilistic two-stage detection. *arXiv preprint arXiv:2103.07461*, 2021. 4, 5, 19, 47, 79, 83, 87, 89, 125, 133, 137, 141, 142, 150

[257] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. Simple multi-dataset detection. In *CVPR*, 2022. 6, 126

[258] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arXiv:1904.07850*, 2019. 3, 50, 51, 54, 59, 61, 64, 65, 68, 69, 87, 88

[259] Xingyi Zhou, Tianwei Yin, Vladlen Koltun, and Philipp Krähenbühl. Global tracking transformers. In *In submission*, 2021. 5

[260] Xingyi Zhou, Tianwei Yin, Vladlen Koltun, and Philipp Krähenbühl. Global tracking transformers. In *CVPR*, 2022. 5

[261] Xingyi Zhou, Jiacheng Zhuo, and Philipp Krähenbühl. Bottom-up object detection by grouping extreme and center points. In *CVPR*, 2019. 10, 37, 38, 39, 40, 41

[262] Benjin Zhu, Jianfeng Wang, Zhengkai Jiang, Fuhang Zong, Songtao Liu, Zeming Li, and Jian Sun. Autoassign: Differentiable label assignment for dense object detection. *arXiv:2007.03496*, 2020. 51

[263] Chenchen Zhu, Fangyi Chen, Zhiqiang Shen, and Marios Savvides. Soft anchor-point object detection. *ECCV*, 2020. 51

[264] Chenchen Zhu, Yihui He, and Marios Savvides. Feature selective anchor-free module for single-shot object detection. *CVPR*, 2019. 29, 40, 41

[265] Ji Zhu, Hua Yang, Nian Liu, Minyoung Kim, Wenjun Zhang, and Ming-Hsuan Yang. Online multi-object tracking with dual matching attention networks. In *ECCV*, 2018. 13

[266] Pengkai Zhu, Hanxiao Wang, and Venkatesh Saligrama. Don't even look once: Synthesizing features for zero-shot detection. In *CVPR*, 2020. 132

[267] Tianyu Zhu, Markus Hiller, Mahsa Ehsanpour, Rongkai Ma, Tom Drummond, and Hamid Rezatofighi. Looking beyond two frames: End-to-end multi-object tracking using spatial and temporal transformers. *arXiv:2103.14829*, 2021. 15

[268] Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. Deformable convnets v2: More deformable, better results. *CVPR*, 2019. 29, 36, 37, 41

[269] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. In *ICLR*, 2021. xvii, 51, 82, 137, 150

[270] Xizhou Zhu, Yujie Wang, Jifeng Dai, Lu Yuan, and Yichen Wei. Flow-guided feature aggregation for video object detection. In *ICCV*, 2017. 14

[271] C Lawrence Zitnick and Piotr Dollár. Edge boxes: Locating object proposals from edges. In *ECCV*, 2014. 9

[272] Barret Zoph, Golnaz Ghiasi, Tsung-Yi Lin, Yin Cui, Hanxiao Liu, Ekin D Cubuk, and Quoc V Le. Rethinking pre-training and self-training. In *Neural Information Processing Systems*, 2020. 50

Vita

Xingyi Zhou was born in Zhejiang Province, China in November 1994. He attended high school at Wenzhou No.2 high school in 2013. He received his Bachelor of Science degree from the School of Computer Science of Technology at Fudan University in 2017. He started doing computer vision research at Institute for Media Computing at Fudan University in 2018, with guidance from Dr. Wei Zhang and Dr. Xiangyang Xue. In fall 2017, he started his graduate studies at the Department of Computer Science at The University of Texas at Austin, under the supervision of Dr. Philipp Krähenbühl. During his study, he has interned at Microsoft Research Asia, Google Research, Intel Labs, and Facebook AI Research. He received a Facebook Fellowship in computer vision in 2021.

Permanent email address: zhouxy2017@gmail.com

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.