Simple Multi-dataset Detection

Xingyi Zhou[1] Vladlen Koltun[2] Phillip Krähenbühl[1]
[1]The University of Texas at Austin [2]Apple

Abstract

How do we build a general and broad object detection system? We use all labels of all concepts ever annotated. These labels span diverse datasets with potentially inconsistent taxonomies. In this paper, we present a simple method for training a unified detector on multiple large-scale datasets. We use dataset-specific training protocols and losses, but share a common detection architecture with dataset-specific outputs. We show how to automatically integrate these dataset-specific outputs into a common semantic taxonomy. In contrast to prior work, our approach does not require manual taxonomy reconciliation. Experiments show our learned taxonomy outperforms a expert-designed taxonomy in all datasets. Our multi-dataset detector performs as well as dataset-specific models on each training domain, and can generalize to new unseen dataset without fine-tuning on them. Code is available at https://github.com/xingyizhou/UniDet.

1. Introduction

Computer vision aims to produce broad, general-purpose perception systems that work in the wild. Yet object detection is fragmented into datasets [18, 22, 24, 33] and our models are locked into the corresponding domains. This fragmentation brought rapid progress in object detection [5, 10, 20, 31, 39, 45] and instance segmentation [14], 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 paper, 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 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 advantages of all training data, performs well on training domains, and generalizes

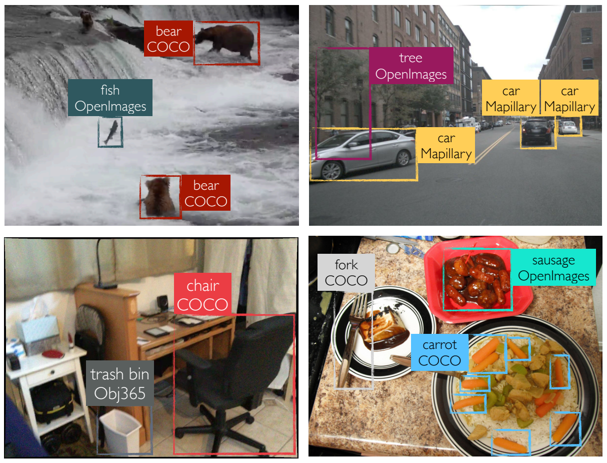

Figure 1. Different datasets span diverse semantic and visual domains. We learn to unify the label spaces of multiple datasets and train a single object detector that generalizes across datasets.

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 [19, 47], 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 [22], Objects365 [33], and OpenImages [18]. 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 to new domains *without retraining*, and outperform single-dataset models.

2. Related Work

Training on multiple datasets. In recent years, training on multiple diverse datasets has emerged as an effective tool to improve model robustness for depth estimation [29], stereo matching [43], and person detection [13]. In these domains, unifying the output space involves modeling different camera transformations or depth ambiguities. In contrast, for recognition, dataset unification involves merging different *semantic* concepts. MSeg [19] manually unified the taxonomies of 7 semantic segmentation datasets and used Amazon Mechanical Turk to resolve inconsistent annotations between datasets. In contrast, we propose to learn a label space from visual data automatically, without requiring any manual effort.

Wang et al. [40] train a universal object detector on multiple datasets, and gain robustness by joining diverse sources of supervision. This is similar to our partitioned detector, while they work on small datasets and didn't model the training differences between different datasets. Universal-RCNN [42] trains an partitioned detector on three large datasets [17,22,48] and models the class relations with a inter-dataset attention module. However again they use the same training recipe for all datasets, and produce duplicated outputs for the same object if it occurs in more one dataset. Both Wang et al. [40] and MSeg [19] 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, 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. [47] trains a universal detector on multiple datasets: COCO [22], Pascal VOC [6], and SUN-RGBD [37], 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 [30] combines detection and classification datasets to expand the detection vocabulary. LVIS [12] 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.

Zero-shot classification and detection reasons about novel object categories outside the training set [1,8]. This is often realized by representing a novel class by a semantic embedding [25] or auxiliary attribute annotations [7]. In zero-shot detection, Bansal et al. [1] proposed a statically assigned background model to avoid novel classes being detected as background. Rahman et al. [28] used test-time training to progressively generate new class labels based on word embeddings. Li et al. [21] leveraged external text descriptions for novel objects. Our program is complementary: we aim to build a sufficiently large label space by merging diverse datasets during training, such that the trained detector transfers well across domains even without machinery such as word embeddings or attributes. Such machinery can be added, if desired, to further expand our model's vocabulary.

3. Preliminaries

Object detection aims to predict a location $b_i \in \mathbb{R}^4$ and a class-wise detection score $d_i \in \mathbb{R}^{|L|}$ for each object i in image I. The detection score describes the confidence that a bounding box belongs to an object with label $c \in L$, where L is the set of all classes (label space) of the dataset $\mathcal{D}$.

Many existing works on object detection focus on the COCO dataset [22], which contains balanced annotations for 80 common object classes. This class balance simplifies training and yields good generalization. Training an object detector on COCO follows a simple recipe: Minimize a loss ℓ, usually box-level log-likelihood, over an sampled image $\hat{I}$ and its corresponding annotated bounding boxes annotations $\hat{B}$ from the dataset $\mathcal{D}$:

$$\min_{\Theta} \mathbb{E}_{(\hat{I},\hat{B})\sim\mathcal{D}} \left[\ell(\mathcal{M}(\hat{I};\Theta),\hat{B})\right]. \tag{1}$$

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 $L_1, L_2, \ldots$. 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 $L = L_1 \cup L_2 \cup \ldots$. Labels that repeat across datasets are merged. We then optimize the same loss with more data:

$$\min_{\Theta} \mathbb{E}_{(\hat{I},\hat{B})\sim\mathcal{D}_1\cup\mathcal{D}_2\cup\ldots} \left[\ell(\mathcal{M}(\hat{I};\Theta),\hat{B})\right]. \tag{2}$$

This has shown promise on smaller, evenly distributed datasets [6, 40, 41]. 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 [33] contains $5\times$ more images than COCO and OpenImages [18] is $18\times$ larger than COCO. While the top 20% of classes in Objects365 and OpenImages contain $19\times$ and $20\times$ more images than COCO, respectively, the bottom 20% classes actually have fewer images than COCO. 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 (1) works for all datasets. Most successful Objects365 models [9] employ class-aware sampling [35]. OpenImages models treat rare classes differently [38] and model the hierarchy of classes in the loss [26].

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

$$\min_{\Theta} \mathbb{E}_{(\hat{I},\hat{B})\sim\mathcal{D}_k}\left[\ell_k(\mathcal{M}_k(\hat{I};\Theta),\hat{B})\right]. \tag{3}$$

No single loss generalizes to all datasets. In the next section, we present a different view of multi-dataset training and show how to train a model that performs well on all datasets.

4. Training a multi-dataset detector

Our goal is to train a single detector $\mathcal{M}$ on K datasets $\mathcal{D}_1,\ldots,\mathcal{D}_K$ with label spaces $L_1,\ldots,L_K$, 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 2b). We train a partitioned detector over all datasets by minimizing the K dataset-specific losses:

$$\min_{\Theta} \mathbb{E}_{\mathcal{D}_k}\left[\mathbb{E}_{(\hat{I},\hat{B})\sim\mathcal{D}_k}\left[\ell_k(\mathcal{M}_k(\hat{I};\Theta),\hat{B})\right]\right]. \tag{4}$$

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 5.

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 $L = L_1 \cup L_2 \cup \ldots$.

4.1. Learning a unified label space

Consider multiple datasets, each with its own label space $L_1, L_2, \ldots$. Our goal is to jointly learn a common label space L for all datasets, and define a mapping between this common label space and dataset-specific labels $\mathcal{T}_k : L \to L_k$. Mathematically, $\mathcal{T}_k \in \{0,1\}^{|L_k|\times|L|}$ is a Boolean linear transformation. In this work, we only consider direct mappings. Each joint label $c \in L$ maps to at most one dataset-specific label $\hat{c} \in L_k$: $\mathcal{T}_k^\top \mathbf{1} \le \mathbf{1}$. I.e., no dataset contains duplicated classes itself. Also, each dataset-specific label matches to exactly one joint label: $\mathcal{T}_k\mathbf{1} = \mathbf{1}$. In particular, we do not hierarchically relate concepts across datasets. When there are different label granularities, we keep them all in our label-space, and expect to predict all of them[1].

Given a set of partitioned detector outputs $d_i^1 \in \mathbb{R}^{|L_1|}, d_i^2 \in \mathbb{R}^{|L_2|}, \ldots$ for a bounding box b_i, we build a joint detection score d_i by simply averaging the outputs of common classes:

$$d_i = \frac{\sum_k \mathcal{T}_k^\top \boldsymbol{d}^k}{\sum_k \mathcal{T}_k^\top \mathbf{1}}, \tag{5}$$

where the division is elementwise. Figure 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 L such that the joint classifier does not degrade in performance.

Simple baselines include hand-designed mappings $\mathcal{T}$ and label spaces L [19, 47], or language-based merging. One issue with these techniques is 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 $\mathcal{L}_c$ be a loss function that measures the quality of the merged label space d_i and its re-projections $\tilde{d}_i^k$ 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 $D = \frac{\sum_k \mathcal{T}_k^\top D^k}{\sum_k \mathcal{T}_k^\top \mathbf{1}}$ 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

$$\begin{aligned} &\text{minimize}_{L,\mathcal{T}} \quad E_{\mathcal{D}_k}\left[\sum_{c\in L_k} \mathcal{L}_c(D_c^k, \tilde{D}_c^k)\right] + \lambda|L| \qquad (6)\\ &\text{subject to} \quad \mathcal{T}_k\mathbf{1} = \mathbf{1} \quad \text{and} \quad \mathcal{T}_k^\top \mathbf{1} \le \mathbf{1} \quad \forall_k. \end{aligned}$$

The cardinality penalty $\lambda|L|$ encourages a small and compact label space. A factorization of the loss $\mathcal{L}_c$ over the

[1]This follows the official evaluation protocol of OpenImages [18].

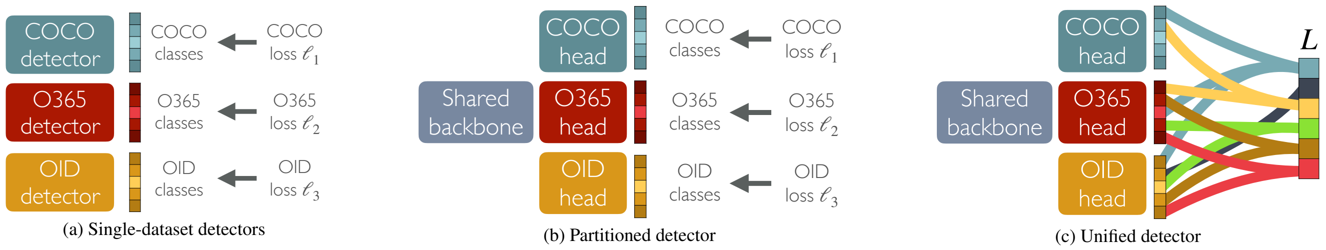

Figure 2. Standard detectors (a) are trained on one dataset with a dataset-specific loss. We train a single partitioned detector (b) on multiple datasets with shared backbone, dataset-specific outputs and loss. Finally, we unify the outputs of the partitioned detector in a common taxonomy completely automatically (c).

output space $c \in L_k$ may seem restrictive. However, it does include the most common loss functions in detection: score distortion and Average Precision (AP). Section 4.2 discusses the exact loss functions used in our optimization.

Objective 6 mixes combinatorial optimization over L 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 L simply corresponds to the number of columns in $\mathcal{T}$. Furthermore, we merge at most one label per dataset $\mathcal{T}_k^\top \mathbf{1} \leq \mathbf{1}$. Hence, for each dataset $\mathcal{D}_k$ a column $\mathcal{T}_k(c) \in \mathbb{T}_k$ takes one of $|\hat{L}_k| + 1$ values: $\mathbb{T}_k = \{\mathbf{0}, \mathbf{1}_1, \mathbf{1}_2, \ldots\}$, where $\mathbf{1}_i \in \{0,1\}^{|L_k|}$ 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 L and transformation $\mathcal{T}$ directly, we instead use combinatorial optimization over the potential column values of $\boldsymbol{t} \in \mathbb{T}$. Let $x_{\boldsymbol{t}} \in \{0,1\}$ be the indicator of combination $\boldsymbol{t} \in \mathbb{T}$. $x_{\boldsymbol{t}} = 1$ means we apply the class combination specified by $\boldsymbol{t}$, and otherwise not. In this formulation, the constraint $\mathcal{T}_k\mathbf{1} = \mathbf{1}\forall_k$ translates to $\sum_{\boldsymbol{t}\in\mathbb{T}|\boldsymbol{t}(c)=1} x_{\boldsymbol{t}} = 1$ for all dataset-specific labels c. Furthermore, the objective of the optimization simplifies to

$$\sum_{\boldsymbol{t}\in\mathbb{T}} \underbrace{x_{\boldsymbol{t}}\, E_{\mathcal{D}_k}\left[\sum_{c\in L_k|\boldsymbol{t}(c)=1} \mathcal{L}_c(D_c^k, \tilde{D}_c^k)\right]}_{c_{\boldsymbol{t}}} + \lambda \sum_{\boldsymbol{t}\in\mathbb{T}} x_{\boldsymbol{t}}. \tag{7}$$

Crucially, the merge cost $c_{\boldsymbol{t}}$ can be precomputed for any subset of labels $\boldsymbol{t}$. This leads to a compact integer linear programming formulation of objective 6:

$$\begin{aligned} \text{minimize}_x \quad & \sum_{\boldsymbol{t}\in\mathbb{T}} x_{\boldsymbol{t}}\,(c_{\boldsymbol{t}} + \lambda) \\ \text{subject to} \quad & \sum_{\boldsymbol{t}\in\mathbb{T}|\boldsymbol{t}_c=1} x_{\boldsymbol{t}} = 1 \qquad \forall_c \end{aligned} \tag{8}$$

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 [23].

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

4.2. Loss functions

The loss function in our constrained objective 6 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:

$$\mathcal{L}_c^{\text{dist}}(D_c^k, \tilde{D}_c^k) = \left(D_c^k - \tilde{D}_c^k\right)^2. \tag{9}$$

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 $\text{AP}_c(\tilde{D}_c^k)$ of each output class c on the validation set of $\mathcal{D}_k$. Our loss measures the improvement in AP:

$$\mathcal{L}_c^{\text{AP}}(D_c^k, \tilde{D}_c^k) = \frac{1}{|L_k|}\left(\text{AP}_c(D_c^k) - \text{AP}_c(\tilde{D}_c^k)\right). \tag{10}$$

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.

5. 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 5.1, we analyze our key design choices for a partitioned detector baseline. In Section 5.2, we evaluate our unified detector and our unified label space learning algorithm. We further evaluate the unified detector in new test datasets in a cross-dataset evaluation (Section 5.3) without any training on the test domain.

Datasets. Our main training datasets are adopted from the Robust Vision Challenge (RVC)[2]. These are four large datasets for object detection: COCO [22], OpenImages [18], Objects365 [33], and optionally Mapillary [24]. To evaluate the generalization ability of the models, we follow MSeg [19] to set up a ross-dataset evaluation protocol: we evaluate models on new test dataset *without training on them*. Specifically, we test on VIPER [32], Cityscapes [3], ScanNet [4], WildDash [44], KITTI [11], Pascal VOC [6], and CrowdHuman [34]. A detailed description of all datasets is contained in the supplement. 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 [18]. For the small datasets in cross-dataset evaluation, we use mAP at IoU threshold 0.5 for consistency with PascalVOC [6].

Implementation details. We use the CascadeRCNN detector [2] 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 detector. For the partitioned detector, the last classification layers of all cascade stages are split between datasets. The unified detector uses CascadeRCNN [2] as is.

Our implementation is based on Detectron2 [41]. We adopt most of the default hyper-parameters 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 [15] 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) [41] in most experiments unless specified otherwise, regardless of the training data size.

[2]http://www.robustvision.net

	COCO	O365	OImg	*mean*
Simple merge [40]	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 1. **Effectiveness of our multi-dataset training strategies**. We start with a simple merging of datsets [40], 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 (fourth row). Our partitioned detector combines these sampling strategies with a dataset-specific loss (last row).

5.1. 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 4, our baseline highlights two basic components: 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 [40, 41]. They simply collect all data from all datasets and train with a common loss. As is shown in Table 1, 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 [26, 35], 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 [18] 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 [18]. 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 $+2.7$mAP improvement on OpenImages alone, and does not degrades other datasets.

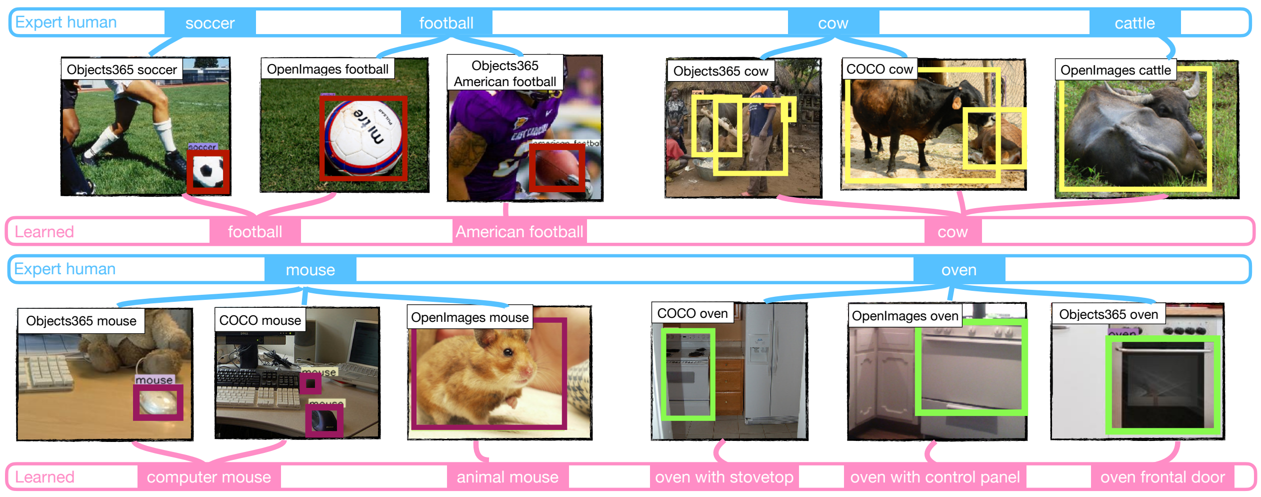

Figure 3. **Sampled results of the learned unified label space**. We show example differences between an expert-designed label space provided as part of the Robust Vision Challenge (top of each row, blue) and our learned label space (bottom of each row, pink). Our learned label space captures detailed visual differences. Zoom in for details.

Dateset-specific vs. partitioned detectors. In our partitioned detector, training on multiple datasets resembles training separate individual models but with a shared detector. Table 2 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 our partitioned detector. In a 2$\times$ 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$\times$ schedule, the partitioned detector starts to match dataset-specific models, and outperforms 2$\times$ dataset-specific models under the same total iterations. In a 8$\times$ schedule, all models converge. The partitioned detector surpasses the single-dataset model on COCO, and matches OpenImages and Objects365 models.

5.2. Unified multi-dataset detection

Next, we evaluate different ways to unify the label space. **Unified label space** We run our label space learning algorithm from Section 4.1 based on the output of a partitioned detector with a ResNeSt backbone [46] 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 $|L| = 701$. we compare our automated data-driven unification to human and language-based baselines. We use the official manually-crafted RVC taxonomy as the human expert baseline[3].

Over two-thirds of our learned label space agrees with the human expert. Figure 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.

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

	2×			6×			8×		
	COCO	Objects365	OImg.	COCO	Objects365	OImg.	COCO	Objects365	OImg.
Partitioned detector	**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	-	-
Objects365	-	**23.8**	-	-	**25.0**	-	-	**24.9**	-
OpenImages	-	-	**64.6**	-	-	**65.4**	-	-	65.7

Table 2. **Dateset-specific vs partitioned detectors.** We show validation mAP of our partitioned model and the three dataset-specific models under different training schedules. The performance of a partitioned model matches dataset-specific models on long schedules.

	$\vert L\vert$	COCO	O365	OImg.	*mean*
GloVe embedding	696	$41.6_{\pm 0.00}$	$20.3_{\pm 0.12}$	$62.4_{\pm 0.06}$	$41.4_{\pm 0.05}$
Learned, distortion	682	$41.6_{\pm 0.15}$	$20.7_{\pm 0.06}$	$62.6_{\pm 0.06}$	$41.7_{\pm 0.09}$
Learned, AP (ours)	701	$\mathbf{41.9}_{\pm \mathbf{0.10}}$	$\mathbf{20.8}_{\pm \mathbf{0.10}}$	$\mathbf{63.0}_{\pm \mathbf{0.21}}$	$\mathbf{41.9}_{\pm \mathbf{0.02}}$
Expert human	659	$41.5_{\pm 0.06}$	$20.7_{\pm 0.06}$	$62.6_{\pm 0.06}$	$41.6_{\pm 0.04}$

Table 3. **Evaluation of unified label spaces.** We show label space size ($|L|$) and 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 ResNet50 CascadeRCNN trained in a 2× schedule. We show the mean and standard deviation based on 3 repeated runs. Our learned label space works better than the language and the human counterparts.

λ	τ	$\vert L\vert$	COCO	O365	Oimg.	mean
0.1	0.25	700	**41.9**	20.6	62.9	41.8
0.5*	0.25*	701	**41.9**	20.8	**63.0**	**41.9**
1.0	0.25	703	**41.9**	**20.9**	**63.0**	**41.9**
0.5	0.2	668	41.6	20.7	62.9	41.7
0.5	0.3	728	41.8	**20.9**	62.9	**41.9**

Table 4. **Hyper-parameter choices**. We change λ and τ of the label space learning algorithm. We show the size of the resulting label space and the mAP on 3 datasets. *: the default option. The pruning threshold τ impacts the label space size, but not mAP.

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 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 4.2 with the cosine distance between the GloVe word embeddings [27], and run the same integer linear program. Table 3 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.

Hyper-parameter choices. Table 4 ablates the hyper-parameters λ and τ of the label space learning algorithm (Section 4.1). Our algorithm is robust to the cardinality penalty factor λ. Varying the cardinality penalty λ from 0.1 to 1.0 only affects the size of the label space by 3. The pruning threshold τ has a larger impact on the label space size, but not the final performance. We use $\lambda = 0.5$ and $\tau = 0.25$ for a good balance between the label space size and overage performance.

	COCO	O365	OImg.	*mean*
Unified (naive merge)	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 5. **Unified vs. partitioned detectors**. We show validation mAP on training domains for a unified detector directly from merging partitioned detector weights (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.

Unified vs. partitioned detectors. We next compare unified detectors with and without retraining using the joint taxonomy, a partitioned detector, and an ensemble of dataset-specific detectors. The partitioned 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 without any modification in new domains, while the alternatives must know which domain they are in. Table 5 shows the results. A partitioned detector outperforms a dataset-specific ensemble under the same conditions (Table 5 bottom), especially on the "small" COCO dataset. An offline unification loses some accuracy, but this is regained when retraining the model under the unified taxonomy (Table 5 top). Crucially, the unified models do not need to know what domain they are in at test time.

5.3. 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-to-train label correspondence. Specifically, we calculate the GloVe [27] 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 [24] in a 2× fine-tuning schedule with 10× smaller learning rate. We compare all models under the same schedule [5], hyperparameters, and detection

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

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

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

Table 6. **Cross-dataset evaluation**. We show mAP50 on the validation sets of datasets that were not seen during training. We compare models trained on each single training dataset (Rows 1-4), the ensemble of the 4 single dataset models (row 5), a partitioned detector (row 6), and the unified detector with our learned unified label space (row 7). For reference, we show the "oracle" models that are trained on the training set of each test dataset on row 8. The columns refer to test datasets. Each model is a ResNet-50 CascadeRCNN trained until converge or at most an 8× schedule.

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 evaluate on the training set and do not provide the oracle model.

Table 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 (row 6) or a unified one (row 7) yields generally good performance on all test datasets. Notably, both our detectors perform better than the ensemble of the 4 single dataset models (row 5), showing that the multi-dataset models learned more general features. On Pascal VOC, both multi-dataset models outperform the VOC-trained upper-bound without seeing VOC training images. Our unified model outperforms the partitioned detector overall and operates on a unified taxonomy.

5.4. Scale up to large models

Next, we scale up our unified detector with a large backbone to develop a ready-to-deploy object detector. We used a ResNeSt200 backbone [46] and followed the same training procedure as in Section 5.2 with an 8× schedule. The training took ∼16 days on a server with 8 Quadro RTX 6000 GPUs. Table 7 shows our *single* model achives 52.9 mAP on COCO, 60.6 mAP on OpenImages, and 33.7 mAP on Objects 365. We compare to state-of-the-art results with comparable baselines on each individual dataset. On COCO, our result improves the COCO-only ResNeSt200 [46] model, by 2 mAP with the same detector, thanks to our ability to train with more data. On OpenImages, our result matches the best single model in the OpenImages 2019 Challenge, TSD [36], with a comparable backbone (SENet154-DCN [16] of TSD). On Objects365, we outperform the 2019 Object365 detection challenge winner [9] by 2 mAP points.

	COCO	OImg.	Mapillary	O365
Ours	**52.9**	**60.6/56.8**	**25.3**	**33.7**
ResNeSt200 [46]	50.9	-	-	-
TSD [36]	-	60.5/-	-	-
CACascade RCNN [9]	-	-	-	31.6

Table 7. **Scale up to large models**. We show results on COCO test-challenge set, OpenImages challenge 2019 test sets (public test set/ private test set), Mapillary test set, and Objects365 validation set. Top row: our detector with a ResNeSt200 backbone. 2-4 rows: state-of-the-art single-dataset models with comparable backbones (without model ensembles or test-time augmentation).

6. Conclusion

We presented a simple recipe for training a single object detector across multiple datasets and a formulation to automatically construct a unified taxonomy. Our resulting detector can be deployed in new domains without additional knowledge. We hope our model makes object detection more accessible to general users.

Limitations. Our label space learning algorithm currently uses only visual cues, integrating language cues as auxiliary information may further improve the performance. Our formulation currently does not consider label hierarchies, and the resulting label space treats COCO person and OpenImages boy as two independent classes. We leave incorporating label hierarchies as exciting future work.

Acknowledgments. This material is based upon work supported by the National Science Foundation under Grant No. IIS-1845485 and IIS-2006820. Xingyi is supported by a Facebook Fellowship.

References

[1] Ankan Bansal, Karan Sikka, Gaurav Sharma, Rama Chellappa, and Ajay Divakaran. Zero-shot object detection. In *ECCV*, 2018. 2

[2] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: High quality object detection and instance segmentation. *TPAMI*, 2019. 5

[3] 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. 5

[4] 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. 5

[5] Xiyang Dai, Yinpeng Chen, Bin Xiao, Dongdong Chen, Mengchen Liu, Lu Yuan, and Lei Zhang. Dynamic head: Unifying object detection heads with attentions. In *CVPR*, 2021. 1

[6] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The PASCAL visual object classes (VOC) challenge. *IJCV*, 2010. 2, 5

[7] Ali Farhadi, Ian Endres, Derek Hoiem, and David Forsyth. Describing objects by their attributes. In *CVPR*, 2009. 2

[8] Yanwei Fu, Tao Xiang, Yu-Gang Jiang, Xiangyang Xue, Leonid Sigal, and Shaogang Gong. Recent advances in zero-shot recognition: Toward data-efficient understanding of visual content. *IEEE Signal Processing Magazine*, 2018. 2

[9] 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. 3, 8

[10] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. Yolox: Exceeding yolo series in 2021. *arXiv:2107.08430*, 2021. 1

[11] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012. 5

[12] Agrim Gupta, Piotr Dollar, and Ross Girshick. Lvis: A dataset for large vocabulary instance segmentation. In *CVPR*, 2019. 2

[13] Irtiza Hasan, Shengcai Liao, Jinpeng Li, Saad Ullah Akram, and Ling Shao. Generalizable pedestrian detection: The elephant in the room. In *CVPR*, 2021. 2

[14] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *CVPR*, 2017. 1

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 5

[16] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *CVPR*, 2018. 8

[17] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *IJCV*, 2017. 2

[18] 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. 1, 3, 5

[19] John Lambert, Zhuang Liu, Ozan Sener, James Hays, and Vladlen Koltun. MSeg: A composite dataset for multi-domain semantic segmentation. In *CVPR*, 2020. 1, 2, 3, 5

[20] 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. *CVPR*, 2021. 1

[21] Zhihui Li, Lina Yao, Xiaoqin Zhang, Xianzhi Wang, Salil Kanhere, and Huaxiang Zhang. Zero-shot object detection with textual descriptions. In *AAAI*, 2019. 2

[22] 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, 5

[23] 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. 4

[24] Gerhard Neuhold, Tobias Ollmann, Samuel Rota Bulò, and Peter Kontschieder. The mapillary vistas dataset for semantic understanding of street scenes. In *ICCV*, 2017. 1, 5, 7

[25] Mohammad Norouzi, Tomas Mikolov, Samy Bengio, Yoram Singer, Jonathon Shlens, Andrea Frome, Greg S Corrado, and Jeffrey Dean. Zero-shot learning by convex combination of semantic embeddings. In *ICLR*, 2014. 2

[26] 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. 3, 5

[27] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, 2014. 7

[28] Shafin Rahman, Salman Khan, and Nick Barnes. Transductive learning for zero-shot object detection. In *ICCV*, 2019. 2

[29] 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. 2

[30] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *CVPR*, 2017. 2

[31] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 1

[32] Stephan R Richter, Zeeshan Hayder, and Vladlen Koltun. Playing for benchmarks. In *ICCV*, 2017. 5

[33] 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. 1, 3, 5

[34] 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. 5

[35] Li Shen, Zhouchen Lin, and Qingming Huang. Relay back-propagation for effective learning of deep convolutional neural networks. In *ECCV*, 2016. 3, 5

[36] Guanglu Song, Yu Liu, and Xiaogang Wang. Revisiting the sibling head in object detector. In *CVPR*, 2020. 8

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

[38] 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. 3

[39] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Scaled-yolov4: Scaling cross stage partial network. *CVPR*, 2021. 1

[40] Xudong Wang, Zhaowei Cai, Dashan Gao, and Nuno Vasconcelos. Towards universal object detection by domain attention. In *CVPR*, 2019. 2, 5

[41] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019. 2, 5

[42] Hang Xu, Linpu Fang, Xiaodan Liang, Wenxiong Kang, and Zhenguo Li. Universal-rcnn: Universal object detector via transferable graph r-cnn. In *AAAI*, 2020. 2

[43] Gengshan Yang, Joshua Manela, Michael Happold, and Deva Ramanan. Hierarchical deep stereo matching on high-resolution images. In *CVPR*, 2019. 2

[44] Oliver Zendel, Katrin Honauer, Markus Murschitz, Daniel Steininger, and Gustavo Fernandez Dominguez. Wilddash-creating hazard-aware benchmarks. In *ECCV*, 2018. 5

[45] Haoyang Zhang, Ying Wang, Feras Dayoub, and Niko Sünderhauf. Varifocalnet: An iou-aware dense object detector. *CVPR*, 2021. 1

[46] 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. 6, 8

[47] 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. 1, 2, 3

[48] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *CVPR*, 2017. 2