USB: Universal-Scale Object Detection Benchmark *Supplementary Material*

Yosuke Shinya*
https://shinya7y.github.io/

independent researcher
Tokyo, Japan

A Discussions on Research Ethics

Limitations. In addition to the limitations described in the main text, this work has the following limitations. (1) USB depends on datasets with many instances. Reliable scale-wise metrics for small datasets should be considered. (2) USB does not cover the resolution of recent smartphone cameras (*e.g.*, 4000×3000). Such high-resolution images may encourage completely different methods. (3) USB, as well as UODB [61], has a large imbalance in the number of images. If participants train a unified detector [61, 67], they will need strategies for dataset sampling [67].

Potential negative societal impacts. Improving the accuracy and universality of object detectors could improve the performance of autonomous weapons. To mitigate the risk, we could develop more detectors for entertainment to increase people's happiness and decrease their hatred. Besides, detectors might be misused for surveillance systems (*e.g.*, as a part of person tracking methods). To mitigate the risk, the computer vision community will need to have discussions with national and international organizations to regulate them appropriately.

Existing assets. We used the assets listed in Table 10. See our codes for more details. Refer to the papers [2, 31, 48] and the URLs for how the datasets were collected.

Asset	Version	URL	License
COCO [31]	2017	https://cocodataset.org/	Annotations: CC-BY 4.0; images: various licenses
WOD [48]	1.2	https://waymo.com/open/	Custom license
Manga109-s [2, 33]	2020.12.18	http://www.manga109.org/	Custom license
COCO API [31]	2.0	https://github.com/cocodataset/cocoapi	2-Clause BSD License
WOD (code)	—	https://github.com/waymo-research/waymo-open-dataset	Apache License 2.0
Manga109 API	0.3.1	https://github.com/manga109/manga109api	MIT License
MMDetection [4]	2.25.0	https://github.com/open-mmlab/mmdetection	Apache License 2.0

Table 10: Existing assets we used.

Consent. See [2] for M109s. For the other datasets, we could not find whether and how consent was obtained. It will be impossible to obtain consent from people recorded in datasets for autonomous driving such as WOD [48].

Privacy. Faces and license plates in WOD [48] are blurred. COCO images may harm privacy because they probably contain personally identifiable information. However, COCO [31] is so popular that the computer vision community cannot stop using it suddenly. This paper will be a step toward reducing the dependence on COCO.

Offensive contents. M109s covers various contents [2]. This characteristic is useful to develop universal-scale object detectors. One of the authors checked many images of the three datasets with eyes and felt that some images in M109s may be considered offensive

*This work was done independently of the author's employer.

(*e.g.*, violence in battle manga and nudity in romantic comedy). Thus, researchers should be careful how they use it. It is also valuable to develop methods to detect such scenes using the dataset.

Compute. Considering the numbers of training images, training for USB takes about 1.7 ($\approx \frac{118287+79735+6467}{118287}$) times longer than that for COCO. This is reasonable as a next-generation benchmark after COCO. Furthermore, the proposed protocols provide incentives to avoid computationally intensive settings [55].

B Details of Related Work

B.1 Components for Multi-Scale Object Detection

Backbones and modules. Inception module [50] arranges 1×1, 3×3, and 5×5 convolutions to cover multi-scale regions. Residual block [22] adds multi-scale features from shortcut connections and 3×3 convolutions. ResNet-C and ResNet-D [23] replace the first layer of ResNet with the deep stem (three 3×3 convolutions) [51]. Res2Net module [18] stacks 3×3 convolutions hierarchically to represent multi-scale features. Res2Net-v1b [18] adopts deep stem with Res2Net module. Deformable convolution module in Deformable Convolutional Networks (DCN) [13] adjusts receptive field adaptively by deforming the sampling locations of standard convolutions. These modules are mainly used in backbones.

Necks. To combine and enhance backbones' representation, necks follow backbones. Feature Pyramid Networks (FPN) [32] adopt top-down path and lateral connections like architectures for semantic segmentation. Scale-Equalizing Pyramid Convolution (SEPC) [60] introduces pyramid convolution across feature maps with different resolutions and utilizes DCN to align the features. Dynamic Head (DyHead) [14] improves SEPC with two types of attention mechanisms.

Heads and training sample selection. Faster R-CNN [42] spreads multi-scale anchors over a feature map. SSD [34] spreads multi-scale anchors over multiple feature maps with different resolutions. Adaptive Training Sample Selection (ATSS) [66] eliminates the need for multi-scale anchors by dividing positive and negative samples according to object statistics across pyramid levels.

Multi-scale training and testing. Traditionally, the image pyramid is an essential technique to handle multi-scale objects [43]. Although recent detectors can output multi-scale objects from a single-scale input, many studies use multi-scale inputs to improve performance [33, 42, 60, 66]. In a popular implementation [9], multi-scale training randomly chooses a scale at each iteration for (training-time) data augmentation. Multi-scale testing infers multi-scale inputs and merges their outputs for Test-Time Augmentation (TTA). Scale Normalization for Image Pyramids (SNIP) [46] limits the range of object scales at each image scale during training and testing.

B.2 Scale-Wise Metrics

Many studies have introduced different scale-wise metrics [6, 15, 24, 31, 44, 59, 64]. Unlike these studies, we introduce two types of finer scale-wise metrics based on the absolute scale and relative scale [64]. More importantly, we evaluated them on the datasets that have extensive scale variations and many instances in multiple domains.

Volume	Genre
`15test` *set:*	
Aku-Ham	Four-frame cartoons
Bakuretsu! Kung Fu Girl	Romantic comedy
Doll Gun	Battle
Eva Lady	Science fiction
Hinagiku Kenzan!	Love romance
Kyokugen Cyclone	Sports
Love Hina vol. 1	Romantic comedy
Momoyama Haikagura	Historical drama
Tennen Senshi G	Humor
Uchi no Nyan's Diary	Animal
Unbalance Tokyo	Science fiction
Yamato no Hane	Sports
Youma Kourin	Fantasy
Yume no Kayoiji	Fantasy
Yumeiro Cooking	Love romance
`4val` *set:*	
Healing Planet	Science fiction
Love Hina vol. 14	Romantic comedy
Seijinki Vulnus	Battle
That's! Izumiko	Fantasy
`68train` *set: All the other volumes*	

Table 11: Manga109-s dataset splits (87 volumes in total).

C Details of Protocols

C.1 Dataset Splits of Manga109-s

The *Manga109-s* dataset (87 volumes) is a subset of the full *Manga109* dataset (109 volumes) [2]. Unlike the full Manga109 dataset, the Manga109-s dataset can be used by commercial organizations. The dataset splits for the full Manga109 dataset used in prior work [39] cannot be used for the Manga109-s dataset. We defined the Manga109-s dataset splits shown in Table 11. Unlike alphabetical order splits used in the prior work [39], we selected the volumes carefully. The `15test` set was selected to be well-balanced for reliable evaluation. Five volumes in the `15test` set were selected from the 10 test volumes used in [39] to enable partially direct comparison. All the authors of the `15test` and `4val` set are different from those of the `68train` set to evaluate generalizability.

C.2 Number of Images

There are 118,287 images in COCO `train2017`, 5,000 in COCO `val2017`, 79,735 in WOD `f0train`, 20,190 in WOD `f0val`, 6,467 in M109s `68train`, 399 in M109s `4val`, and 1,289 in M109s `15test`. Following prior work [39], we exclude M109s images without annotations because objects on irregular pages are not annotated.

We selected the test splits from images with publicly available annotations to reduce labor for submissions. Participants should not fine-tune hyperparameters based on the test splits to prevent overfitting.

C.3 Importance of Many Instances

Here, we highlight the importance of a larger number of instances than UODB [61]. We show that if we introduced scale-wise metrics to UODB, the results would be unreliable. Water-

color2k, one of the datasets adopted by UODB, has 6 classes and 27 bicycle instances [27]. If we equally divided the dataset for training and evaluation and they had the same number of small, medium, and large bicycles, the average number of bicycles of a particular scale in the evaluation split would be 4.5. Since the 4.5 bicycles affect $\frac{1}{6}$ of a scale-wise metric, a single error can change the results by 3.7%. Thus, randomness can easily reverse the ranking between methods, making the benchmark results unreliable.

C.4 Exceptions of Protocols

The rounding error of epochs between epoch- and iteration-based training can be ignored when calculating the maximum epochs. Small differences of eight pixels or less can be ignored when calculating the maximum resolutions. For example, DSSD513 [17] will be compared in Mini USB.

The number of additional images loaded for multi-image data augmentation techniques (*e.g.*, Between-Class Learning [58], mixup [65], RICAP [52], and Mosaic [4]) can be ignored when calculating the maximum epochs. Even in that case, the time per epoch is considered according to another provision in Sec. 3.5.

C.5 Constraints on Training Time

We do not adopt constraints on training time as the major constraints of the training protocols because they have the following issues.

- It is difficult to measure training time on unified hardware.
- It is complicated to measure training time, calculate allowable epochs, and set learning rate schedules for each model.
- It is difficult to compare with previous studies, which align the number of epochs.
- They will reduce the value of huge existing resources for standard training epochs (trained models, configuration files, and experimental results) provided by popular object detection libraries such as MMDetection [9].
- They overemphasize implementation optimization rather than the trial and error of novel methods.
- There are overlaps between the factors of training time and those of inference time.

The proposed constraints on training epochs are much easier to adopt and more reasonable. Furthermore, our protocols compensate for the shortcomings of the epoch constraints by defining the provisions for hyperparameter optimization and data augmentation.

C.6 Characteristics of Scale-Wise Metrics

ASAP and COCO-style scale-wise metrics are based on the absolute scale. It has a weakness that it changes with image resizing. To limit inference time and GPU memory consumption, and to ensure fair comparisons, input image scales are typically resized. If they are smaller than the original image scales, relative scales have direct effects on accuracy rather than absolute scales. Furthermore, objects with the same absolute scale in the original images may have different absolute scales in the input images. Fluctuating object scale thresholds is not desirable for scale-wise metrics.

In addition, ASAP is not suitable for evaluating accuracy for very large objects. It may be impossible to calculate ASAP for large absolute scales on some datasets. In the case of

Method	Head		Neck			Backbone			Input	FPS	COCO (1× schedule)					
	ATSS	GFL	PConv	DCN	iBN	Res2	DCN	SyncBN	MStrain		AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
RetinaNet [33]										33.9	36.5	55.4	39.1	20.4	40.3	48.1
ATSS [66]	✓									35.2	39.4	57.6	42.8	23.6	42.9	50.3
GFL [29]	✓	✓								37.2	40.2	58.4	43.3	23.3	44.0	52.2
ATSEPC [60, 66]	✓		✓	P, LC						25.0	42.1	59.9	45.5	24.6	46.1	55.0
UniverseNet	✓		✓	P, LC		✓	c3-c5		✓	17.3	46.7	65.0	50.7	29.2	50.6	61.4
UniverseNet+GFL	✓	✓	✓	P, LC		✓	c3-c5		✓	17.5	47.5	65.8	51.8	29.2	51.6	62.5
UniverseNet-20.08d	✓	✓	✓	P, LC	✓	✓	c3-c5	✓	✓	17.3	**48.6**	**67.1**	**52.7**	**30.1**	**53.0**	**63.8**
UniverseNet-20.08	✓	✓	✓	LC	✓	✓	c5	✓	✓	24.9	47.5	66.0	51.9	28.9	52.1	61.9
UniverseNet-20.08 w/o SEPC [60]	✓	✓				✓	c5	✓	✓	26.7	45.8	64.6	50.0	27.6	50.4	59.7
UniverseNet-20.08 w/o Res2Net-v1b [18]	✓	✓	✓	LC	✓		c5	✓	✓	32.8	44.7	62.8	48.4	27.1	48.8	59.5
UniverseNet-20.08 w/o DCN [13]	✓	✓	✓		✓	✓		✓	✓	27.8	45.9	64.5	49.8	28.9	49.9	59.0
UniverseNet-20.08 w/o iBN, SyncBN [40, 60]	✓	✓	✓	LC		✓	c5		✓	25.7	45.8	64.0	50.2	27.9	50.0	59.8
UniverseNet-20.08 w/o MStrain	✓	✓	✓	LC	✓	✓	c5	✓		24.8	45.9	64.5	49.6	27.4	50.5	60.1

Table 12: Architectures of UniverseNets with a summary of ablation studies on COCO `minival`. See Sec. E.7 for step-by-step improvements. All results are based on MMDetection [9] v2. The "Head" methods (ATSS and GFL) affect losses and training sample selection. Res2: Res2Net-v1b [18]. PConv (Pyramid Convolution) and iBN (integrated Batch Normalization) are the components of SEPC [60]. The DCN columns indicate where to apply DCN. "P": The PConv modules in the combined head of SEPC [60]. "LC": The extra head of SEPC for localization and classification [60]. "c3-c5": conv3_x, conv4_x, and conv5_x layers in ResNet-style backbones [22]. "c5": conv5_x layers in ResNet-style backbones [22]. ATSEPC: ATSS with SEPC (without iBN). MStrain: Multi-scale training. FPS: Frames per second on one V100 with mixed precision.

USB, we cannot calculate $ASAP_\infty$ on COCO because the absolute scales of COCO objects are smaller than 1024 (we filled $ASAP_\infty$ on COCO with zero in experiments). Furthermore, ASAP for large absolute scales may show unusual behavior. For example, in the evaluation of $ASAP_\infty$ on M109s, all predictions larger than 1024 of absolute scales have larger IoUs than 0.5 with an object of image resolution size (1654×1170).

We prefer RSAP to ASAP due to the above-mentioned weaknesses of ASAP. Absolute scales may be important depending on whether and how participants resize images. In that case, RSAP and ASAP can be used complementarily.

D Details of UniverseNets

For fast and accurate detectors for USOD, we designed UniverseNets. We adopted single-stage detectors for efficiency. We show the detailed architectures in Table 12.

As a baseline model, we used the RetinaNet [33] implemented in MMDetection [9]. Specifically, the backbone is ResNet-50-B [23] (a variant of ResNet-50 [22], also known as the PyTorch style). The neck is FPN [32]. We used focal loss [33], single-scale training, and single-scale testing.

Built on the RetinaNet baseline, we designed *UniverseNet* by collecting human wisdom about multi-scale object detection as of May 2020. We used ATSS [66] and SEPC without iBN [60] (hereafter referred to as *ATSEPC*). The backbone is Res2Net-50-v1b [18]. We adopted Deformable Convolutional Networks (DCN) [13] in the backbone and neck. We used multi-scale training. Unless otherwise stated, we used single-scale testing for efficiency.

By adding GFL [29], SyncBN [40], and iBN [60], we designed three variants of UniverseNet around August 2020. *UniverseNet-20.08d* heavily uses DCN [13]. *UniverseNet-20.08* speeds up inference (and training) by the light use of DCN [13, 60]. *UniverseNet-20.08s* further speeds up inference using the ResNet-50-C [23] backbone.

E Details of Experiments

Here, we show the details of experimental settings and results. See also the code to reproduce our settings including minor hyperparameters.

E.1 Common Settings

We follow the learning rate schedules of MMDetection [9], which are similar to those of Detectron [21]. Specifically, the learning rates are reduced by $10\times$ in two predefined epochs. Epochs for the first learning rate decay, the second decay, and ending training are $(8, 11, 12)$ for the $1\times$ schedule, $(16, 22, 24)$ for the $2\times$ schedule, and $(16, 19, 20)$ for the 20e schedule. To avoid overfitting by small learning rates [45], the 20e schedule is reasonable. We mainly used the $1\times$ schedule (12 epochs).

We mainly used ImageNet [44] pre-trained backbones that are standard in MMDetection [9]. Some pre-trained backbones not supported in MMDetection were downloaded from the repositories of Res2Net [18] and Swin Transformer [35]. We used the COCO pre-trained models of the MMDetection [9] repository for several existing methods (Faster R-CNN [42] with FPN [32], Cascade R-CNN [7], RetinaNet [33], ATSS [66], GFL [29], and Sparse R-CNN [49]). We trained most models with mixed precision and 4 GPUs ($\times$ 4 images per GPU). We mainly used NVIDIA T4 GPUs on the Google Cloud Platform. All results on USB and all results of UniverseNets are single model results without ensemble. We could not train each object detector multiple times with different random seeds to report error bars because training object detectors is too computationally expensive.

E.2 Settings for Specific Methods

Many recent detectors [8, 35] adopt the AdamW optimizer [37]. Since AdamW does not necessarily give better results than SGD, we follow the optimizer settings of the official implementations and MMDetection [9]. Specifically, for COCO and WOD, we used AdamW with an initial learning rate of 10^{-4} for DETR [8], Deformable DETR [68], ATSS [66] with Swin-T [35], and ATSS [66] with ConvNeXt-T [36], and 2.5×10^{-5} for Sparse R-CNN [49]. The learning rate of the backbone is 10^{-5} for DETR [8] and Deformable DETR [68]. For training ATSS [66] with ConvNeXt-T [36], we used layer-wise learning rate decay and a stochastic depth rate of 0.2 (see [36]).

We trained YOLOX-L [19] with SGD and an initial learning rate of 2.5×10^{-3} for COCO and WOD. The test scale on COCO is 1024×1024. Since YOLOX models are trained from scratch [19], we trained a COCO model for 36 epochs (USB 2.0) such that it achieves similar AP to a model trained for 12 epochs from an ImageNet pre-trained model [45]. We also trained a COCO model for 24 epochs (USB 1.0).

We used multi-scale training for YOLOX-L [19] and UniverseNets. The range of shorter side pixels for most models is 480–960, following prior work [60]. That for YOLOX-L [19] on COCO is 512–1024, and that for UniverseNets on WOD is 640–1280. Since we do not have sufficient computational resources, these hyperparameters have room for improvement.

For comparison with state-of-the-art methods on COCO, we used the $2\times$ schedule (24 epochs) for most models and the 20e schedule (20 epochs) for UniverseNet-20.08d due to overfitting with the $2\times$ schedule. For comparison with state-of-the-art methods on WOD, we trained UniverseNet on the WOD full training set for 7 epochs. We used a learning rate of 10^{-3} for 6 epochs and 10^{-4} for the last epoch.

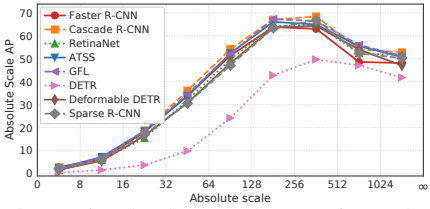


Figure 6: Absolute Scale AP of popular baseline methods.

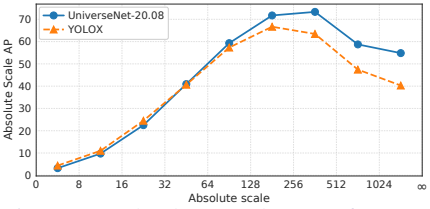


Figure 7: Absolute Scale AP of strong baseline methods.

For comparison with state-of-the-art methods with TTA on COCO, we used soft voting with 13-scale testing and horizontal flipping following the original implementation of ATSS [56]. Specifically, shorter side pixels are (400, 500, 600, 640, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1800), while longer side pixels are their 1.667×. For the 13 test scales, target objects are limited to corresponding 13 predefined ranges ((96, ∞), (96, ∞), (64, ∞), (64, ∞), (64, ∞), (0, ∞), (0, ∞), (0, ∞), (0, 256), (0, 256), (0, 192), (0, 192), (0, 96)), where each tuple denotes the minimum and maximum absolute scales. We also evaluated 5-scale TTA because the above-mentioned ATSS-style TTA is slow. We picked (400, 600, 800, 1000, 1200) for shorter side pixels, and ((96, ∞), (64, ∞), (0, ∞), (0, ∞), (0, 256)) for absolute scale ranges.

For M109s, we used learning rates 8× those of COCO and WOD. The value is roughly tuned based on a preliminary experiment with the RetinaNet [33] baseline model. For training ATSS [56] with different backbones (Swin-T [35] and ConvNeXt-T [36]), we used an initial learning rate of 4×10^{-4}, roughly tuned from choices $\{2\times10^{-4}, 4\times10^{-4}, 8\times10^{-4}\}$.

E.3 Evaluation with Scale-Wise Metrics

We show RSAP and ASAP of popular baseline methods on USB in Figures 4 and 6, respectively. They do not increase monotonically but rather decrease at relative scales greater than 1/4 or absolute scales greater than 512. The difficulty of very large objects may be caused by truncation or unusual viewpoints [24]. Except for the issues of ASAP_∞ discussed in Sec. C.6, ASAP shows similar changes to RSAP. We conjecture that this is because image resolutions do not change much in each dataset of USB. $\text{RSAP}_{\frac{1}{64}}$ and ASAP_{16} are less than 10%, which indicates the difficulty of tiny object detection [64]. RetinaNet [33] shows low AP for small objects, while Faster R-CNN [42] with FPN [32] shows low AP for large objects. These results are consistent with the benchmark results of previous work [25], which compares SSD [34] with Faster R-CNN without FPN on COCO. For further analysis, it will be worth comparing the design choice of pyramid levels [33, 64].

We show RSAP and ASAP of strong baseline methods on USB in Figures 5 and 7, respectively. UniverseNet-20.08 is more accurate for large objects, and YOLOX-L [19] is more accurate for small objects. We conjecture that the former is due to SEPC [60] and DCN [13] (see Sec. E.7 and E.8). For the latter, YOLOX may be biased toward small object detection due to Mosaic augmentation [4] and the absence of pre-training [45] on ImageNet that has larger objects than COCO [46].

E.4 Details on Each Dataset

Tables 13, 14, and 15 show the results on COCO, WOD, and M109s, respectively.

Method	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Faster R-CNN [40]	37.4	58.1	40.4	21.2	41.0	48.1
Cascade R-CNN [0]	**40.3**	**58.6**	**44.0**	22.5	43.8	52.9
RetinaNet [33]	36.5	55.4	39.1	20.4	40.3	48.1
ATSS [64]	39.4	57.6	42.8	**23.6**	42.9	50.3
GFL [29]	40.2	58.4	43.3	23.3	**44.0**	52.2
DETR [3]	22.2	39.7	22.0	6.7	22.4	36.8
Deformable DETR [63]	37.1	55.9	39.9	19.3	40.7	50.5
Sparse R-CNN [45]	37.9	56.0	40.5	20.7	40.0	**53.5**
ATSS [64]+Swin-T [35]	43.7	63.0	47.1	**28.2**	47.0	56.8
ATSS [64]+ConvNeXt-T [34]	**45.5**	**64.7**	**49.5**	28.0	**49.1**	**59.1**
ATSS [64]+SEPC [51]	42.1	59.9	45.5	24.6	46.1	55.0
ATSS [64]+DyHead [10]	**43.3**	**60.9**	**47.2**	**26.6**	**47.1**	**55.8**
YOLOX-L [19] (USB 1.0)	37.8	56.5	41.5	26.4	42.0	41.1
YOLOX-L [19] (USB 2.0)	41.1	60.2	45.3	29.1	45.8	45.5
UniverseNet	46.7	65.0	50.7	**29.2**	50.6	61.4
UniverseNet-20.08 w/o MStrain	45.9	64.5	49.6	27.4	50.5	60.1
UniverseNet-20.08	**47.5**	**66.0**	**51.9**	28.9	**52.1**	**61.9**

Table 13: Results on COCO `minival`.

Method	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	veh.	ped.	cyc.
Faster R-CNN [40]	34.5	55.3	36.3	6.0	35.8	67.4	42.7	34.6	26.1
Cascade R-CNN [0]	**36.4**	**56.3**	**38.6**	6.5	**38.1**	70.6	**44.5**	**36.3**	**28.5**
RetinaNet [33]	32.5	52.2	33.7	2.6	32.8	67.9	40.0	32.5	25.0
ATSS [64]	35.4	56.2	37.0	6.1	36.6	69.8	43.6	35.6	27.0
GFL [29]	35.7	56.0	37.1	6.2	36.7	**70.7**	44.0	36.0	27.1
DETR [3]	17.8	35.1	15.6	0.7	10.8	48.1	24.5	16.2	12.6
Deformable DETR [63]	32.7	55.1	34.1	6.0	32.9	66.2	39.5	33.7	24.9
Sparse R-CNN [45]	32.8	54.3	33.9	**7.2**	33.6	65.1	38.7	33.4	26.2
ATSS [64]+Swin-T [35]	37.2	58.6	38.9	7.2	39.3	71.1	45.4	36.9	29.3
ATSS [64]+ConvNeXt-T [34]	**38.3**	**59.9**	**40.1**	**7.4**	**40.1**	**72.9**	**46.4**	**37.8**	**30.6**
ATSS [64]+SEPC [51]	35.0	55.3	36.5	5.8	35.5	70.5	43.5	35.3	26.3
ATSS [64]+DyHead [10]	**37.1**	**57.8**	**39.1**	**6.8**	**39.1**	**72.0**	**45.4**	**37.1**	**28.9**
YOLOX-L [19] (USB 1.0)	41.0	63.4	43.2	11.5	43.9	72.8	49.0	40.9	33.1
YOLOX-L [19] (USB 2.0)	**41.6**	**64.1**	**43.9**	**12.0**	**44.4**	73.5	**49.3**	**41.3**	**34.1**
UniverseNet	38.6	59.8	40.9	7.4	41.0	**74.0**	46.0	37.6	32.3
UniverseNet-20.08 w/o MStrain	37.9	58.8	39.4	7.8	39.2	72.9	45.9	37.5	30.2
UniverseNet-20.08	39.0	60.2	40.4	8.3	41.7	73.3	47.1	38.7	31.0

Table 14: Results on WOD `f0val`.

Method	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	body	face	frame	text
Faster R-CNN [40]	65.8	**91.1**	70.6	**18.4**	39.9	72.1	58.3	47.5	90.1	67.1
Cascade R-CNN [0]	**67.6**	90.6	**72.0**	17.9	**41.9**	74.3	60.8	**48.2**	**92.5**	69.0
RetinaNet [33]	65.3	90.5	69.5	15.7	38.9	71.9	58.3	46.3	88.8	67.7
ATSS [64]	66.5	90.1	70.8	16.8	38.9	74.0	60.9	44.6	91.3	69.0
GFL [29]	67.3	90.6	71.5	17.9	38.9	**74.4**	**61.7**	45.7	92.2	**69.4**
DETR [3]	31.2	63.0	27.1	1.1	8.2	41.3	24.8	16.0	65.0	19.1
Deformable DETR [63]	64.1	90.1	67.9	16.1	34.8	71.0	57.0	45.6	89.0	64.8
Sparse R-CNN [45]	63.1	85.8	66.3	15.3	33.8	70.3	54.2	45.5	88.9	63.5
ATSS [64]+Swin-T [35]	66.2	90.1	70.1	16.1	39.1	73.8	60.5	44.0	91.7	68.6
ATSS [64]+ConvNeXt-T [34]	**67.4**	**90.8**	**71.5**	16.5	**39.8**	**75.0**	**62.9**	**45.3**	**92.1**	**69.2**
ATSS [64]+SEPC [51]	67.1	90.2	71.5	16.2	39.8	74.9	62.3	44.6	92.1	69.4
ATSS [64]+DyHead [10]	**67.9**	**90.6**	**72.3**	**17.1**	**42.9**	**75.5**	**63.4**	**45.3**	**92.7**	**70.1**
YOLOX-L [19] (USB 1.0)	70.1	**93.7**	**75.0**	22.3	**48.4**	76.1	65.8	50.9	93.0	70.8
YOLOX-L [19] (USB 2.0)	**70.2**	93.6	**75.0**	**22.6**	47.5	76.1	65.9	**51.0**	93.1	70.7
UniverseNet	68.9	91.4	73.7	18.7	43.4	76.6	65.8	46.6	93.0	70.3
UniverseNet-20.08 w/o MStrain	68.3	91.2	72.2	17.9	40.9	75.6	63.3	46.3	93.1	70.3
UniverseNet-20.08	69.9	92.5	74.3	20.5	43.6	**77.1**	**66.6**	48.0	**93.7**	**71.2**

Table 15: Results on Manga109-s `15test`.

Protocol	Method	Backbone	DCN	Epoch	Max test scale	TTA	FPS	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Standard USB 1.0	Faster R-CNN [53, 18]	ResNet-101		22	1333× 800		(14.2)	36.2	59.1	39.0	18.2	39.0	48.2
Standard USB 1.0	Cascade R-CNN [1]	ResNet-101		19	1312× 800		(11.9)	42.8	62.1	46.3	23.7	45.5	55.2
Standard USB 1.0	RetinaNet [33]	ResNet-101		18	1333× 800		(13.6)	39.1	59.1	42.3	21.8	42.7	50.2
Standard USB 1.0	FCOS [51]	X-101 (64×4d)	✓	24	1333× 800		(8.9)	44.7	64.1	48.4	27.6	47.5	55.6
Standard USB 1.0	ATSS [56]	X-101 (64×4d)	✓	24	1333× 800		10.6	47.7	66.5	51.9	29.7	50.8	59.4
Standard USB 1.0	FreeAnchor+SEPC [50]	X-101 (64×4d)	✓	24	1333× 800		—	50.1	69.8	54.3	31.3	53.3	63.7
Standard USB 1.0	PAA [25]	X-101 (64×4d)	✓	24	1333× 800		—	49.0	67.8	53.3	30.2	52.8	62.2
Standard USB 1.0	PAA [25]	X-152 (32×8d)	✓	24	1333× 800		—	50.8	69.7	55.1	31.4	54.7	65.2
Standard USB 1.0	RepPoints v2 [11]	X-101 (64×4d)	✓	24	1333× 800		(3.8)	49.4	68.9	53.4	30.3	52.1	62.3
Standard USB 1.0	RelationNet++ [12]	X-101 (64×4d)	✓	20	1333× 800		10.3	50.3	69.0	55.0	**32.8**	55.0	**65.8**
Standard USB 1.0	GFL [29]	ResNet-50		24	1333× 800		37.2	43.1	62.0	46.8	26.0	46.7	52.3
Standard USB 1.0	GFL [29]	ResNet-101		24	1333× 800		29.5	45.0	63.7	48.9	27.2	48.8	54.5
Standard USB 1.0	GFL [29]	ResNet-101	✓	24	1333× 800		22.8	47.3	66.3	51.4	28.0	51.1	59.2
Standard USB 1.0	GFL [29]	X-101 (32×4d)	✓	24	1333× 800		15.4	48.2	67.4	52.6	29.2	51.7	60.2
Standard USB 1.0	UniverseNet-20.08s	ResNet-50-C	✓	24	1333× 800		31.6	47.4	66.0	51.4	28.3	50.8	59.5
Standard USB 1.0	UniverseNet-20.08	Res2Net-50-v1b	✓	24	1333× 800		24.9	48.8	67.5	53.0	30.1	52.3	61.1
Standard USB 1.0	UniverseNet-20.08d	Res2Net-101-v1b	✓	20	1333× 800		11.7	**51.3**	**70.0**	**55.8**	31.7	**55.3**	64.9
Large USB 1.0	UniverseNet-20.08d	Res2Net-101-v1b	✓	20	1493× 896		11.6	51.5	70.2	56.0	32.8	55.5	63.7
Large USB 1.0	UniverseNet-20.08d	Res2Net-101-v1b	✓	20	2000×1200	5	—	**53.8**	**71.5**	**59.4**	**35.3**	**57.3**	**67.3**
Huge USB 1.0	ATSS [56]	X-101 (64×4d)	✓	24	3000×1800	13	—	50.7	68.9	56.3	33.2	52.9	62.4
Huge USB 1.0	PAA [25]	X-101 (64×4d)	✓	24	3000×1800	13	—	51.4	69.7	57.0	34.0	53.8	64.0
Huge USB 1.0	PAA [25]	X-152 (32×8d)	✓	24	3000×1800	13	—	53.5	**71.6**	59.1	**36.0**	56.3	66.9
Huge USB 1.0	RepPoints v2 [11]	X-101 (64×4d)	✓	24	3000×1800	13	—	52.1	70.1	57.5	34.5	54.6	63.6
Huge USB 1.0	RelationNet++ [12]	X-101 (64×4d)	✓	20	3000×1800	13	—	52.7	70.4	58.3	35.8	55.3	64.7
Huge USB 1.0	UniverseNet-20.08d	Res2Net-101-v1b	✓	20	3000×1800	13	—	**54.1**	**71.6**	**59.9**	35.8	57.2	**67.4**
Huge USB 2.0	TSD [40]	SENet-154	✓	34	2000×1400	4	—	51.2	**71.9**	56.0	33.8	54.8	64.2
Huge USB 2.5	DetectoRS [40]	X-101 (32×4d)	✓	40	2400×1600	5	—	**54.7**	**73.5**	**60.1**	**37.4**	**57.3**	66.4
Mini USB 3.0	EfficientDet-D0 [52]	EfficientNet-B0		300	512× 512		98.0	33.8	52.2	35.8	12.0	38.3	51.2
Mini USB 3.1	YOLOv4 [4]	CSPDarknet-53		273	512× 512		83	**43.0**	**64.9**	**46.5**	**24.3**	**46.1**	**55.2**
Standard USB 3.0	EfficientDet-D2 [52]	EfficientNet-B2		300	768× 768		56.5	43.0	62.3	46.2	22.5	47.0	58.4
Standard USB 3.0	EfficientDet-D4 [52]	EfficientNet-B4		300	1024×1024		23.4	49.4	69.0	53.4	30.3	53.2	63.2
Standard USB 3.1	YOLOv4 [4]	CSPDarknet-53		273	608× 608		62	43.5	65.7	47.3	26.7	46.7	53.3
Large USB 3.0	EfficientDet-D5 [52]	EfficientNet-B5		300	1280×1280		13.8	50.7	70.2	54.7	33.2	53.9	63.2
Large USB 3.0	EfficientDet-D6 [52]	EfficientNet-B6		300	1280×1280		10.8	51.7	71.2	56.0	34.1	55.2	64.1
Large USB 3.0	EfficientDet-D7 [52]	EfficientNet-B6		300	1536×1536		8.2	52.2	71.4	56.3	34.8	55.5	64.6
Freestyle	RetinaNet+SpineNet [16]	SpineNet-190		400	1280×1280		—	52.1	71.8	56.5	35.4	55.0	63.6
Freestyle	EfficientDet-D7x [52]	EfficientNet-B7		600	1536×1536		6.5	**55.1**	**74.3**	59.9	37.2	**57.9**	**68.0**

Table 16: State-of-the-art methods on COCO `test-dev`. We classify methods by the proposed protocols without compatibility. X in the Backbone column denotes ResNeXt [62]. See method papers for other backbones. TTA: Test-time augmentation including horizontal flip and multi-scale testing (numbers denote scales). FPS values without and with parentheses were measured on V100 with mixed precision and other environments, respectively. We measured the FPS of GFL [29] models in our environment and estimated those of ATSS [56] and RelationNet++ [12] based on the measured values and [12, 29]. Other methods' settings are based on conference papers, their arXiv versions, and authors' codes. Values shown in gray were estimated from descriptions in papers and codes. Some FPS values are from [29].

E.5 Rethinking COCO with USB Protocols

We classify state-of-the-art methods on COCO `test-dev` (as of November 14, 2020) by the proposed protocols without compatibility. The results are shown in Table 16. Although state-of-the-art detectors on the COCO benchmark were trained with various settings, the introduced divisions enable us to compare methods in each division. UniverseNet-20.08d achieves the highest AP (51.3%) in the Standard USB 1.0 protocol. Despite 12.5× fewer epochs, the speed-accuracy trade-offs of UniverseNets are comparable to those of EfficientDet [52]. With 13-scale TTA, UniverseNet-20.08d achieves the highest AP (54.1%) in the Huge USB 1.0 protocol. Results in higher protocols than USB 1.0 are scattered. If we ignore the difference of hyperparameter optimization, YOLOv4 [4] shows a better speed-accuracy trade-off than EfficientDet [52] in Standard USB 3.x.

Comparisons across different divisions are difficult. Especially, long training is problematic because it can secretly increase AP without decreasing FPS, unlike large test scales. Nevertheless, the EfficientDet [52], YOLOv4 [4], and SpineNet [16] papers compare meth-

Rank	Method	# Models		AP/L2
		Multi-stage	Single-stage	
Methods including multi-stage detector:				
1	RW-TSDet [26]	6+		**74.43**
2	HorizonDet [11]	4	8	70.28
3	SPNAS-Noah [53]	2		69.43
Single-stage detectors:				
7	**UniverseNet (Ours)**		**1**	**67.42**
13	YOLO V4 [3]		1+	58.08
14	ATSS-Efficientnet [54, 66]		1+	56.99

Table 17: Waymo Open Dataset Challenge 2020 2D detection [1].

ods in their tables without specifying the difference in training epochs. The compatibility of the USB training protocols resolves this disorder. We hope that many papers report results with the protocols for inclusive, healthy, and sustainable development of detectors.

To simulate the compatibility from Standard USB 3.0 to 1.0, we refer to the training log of the EfficientDet author. The AP of EfficientDet-D4 [55] on COCO `minival` is 43.8% at 23 epoch [53]. Although it could be improved by changing the learning rate schedule, EfficientDet's inference efficiency is not compatible with training efficiency.

E.6 Comparison with State-of-the-Art

WOD. For comparison with state-of-the-art methods on WOD, we submitted the detection results of UniverseNet to the Waymo Open Dataset Challenge 2020 2D detection, a competition held at a CVPR 2020 workshop. The primary metric is AP/L2, a KITTI-style AP evaluated with LEVEL_2 objects [1, 48]. We used multi-scale testing with soft-NMS [5]. The shorter side pixels of test scales are $(960, 1600, 2240)$, including 8 pixels of padding. These scales enable utilizing SEPC [60] (see Sec. E.8) and detecting small objects. Table 17 shows the top teams' results. UniverseNet achieves 67.42% AP/L2 without multi-stage detectors, ensembles, expert models, or heavy backbones, unlike other top methods. RW-TSDet [26] overwhelms other multi-stage detectors, whereas UniverseNet overwhelms other single-stage detectors. These two methods used light backbones and large test scales [3]. Interestingly, the maximum test scales are the same (3360×2240). We conjecture that this is not a coincidence but a convergence caused by searching the accuracy saturation point.

Manga109-s. To the best of our knowledge, no prior work has reported detection results on the *Manga109-s* dataset (87 volumes). Although many settings differ, the state-of-the-art method on the full *Manga109* dataset (109 volumes, non-public to commercial organizations) achieves 77.1–92.0% (mean: 84.2%) AP_{50} on ten test volumes [39]. The mean AP_{50} of UniverseNet-20.08 on the `15test` set (92.5%) is higher than those results.

E.7 Ablation Studies for UniverseNets

We show the results of ablation studies for UniverseNets on COCO in Table 18. As shown in Table 18a, ATSEPC (ATSS [66] with SEPC without iBN [60]) outperforms ATSS by a large margin. The effectiveness of SEPC for ATSS is consistent with those for other detectors reported in the SEPC paper [60]. As shown in Table 18b, UniverseNet further improves AP metrics by ∼5% by adopting Res2Net-v1b [18], DCN [13], and multi-scale training. As shown in Table 18c, adopting GFL [29] improves AP by 0.8%. There is room for improvement of AP_S in the Quality Focal Loss of GFL [29]. As shown in Table 18d, UniverseNet-20.08d achieves 48.6% AP by making more use of BatchNorm (SyncBN [40] and iBN [60]).

Method	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
ATSS [64]	39.4	57.6	42.8	23.6	42.9	50.3
ATSEPC [60, 66]	**42.1**	**59.9**	**45.5**	**24.6**	**46.1**	**55.0**

(a) AP improvements by SEPC without iBN [60].

Method	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
ATSEPC [60, 66]	42.1	59.9	45.5	24.6	46.1	55.0
UniverseNet	**46.7**	**65.0**	**50.7**	**29.2**	**50.6**	**61.4**

(b) AP improvements by Res2Net-v1b [18], DCN [13], and multi-scale training.

Method	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
UniverseNet	46.7	65.0	50.7	**29.2**	50.6	61.4
UniverseNet+GFL	**47.5**	**65.8**	**51.8**	**29.2**	**51.6**	**62.5**

(c) AP improvements by GFL [29].

Method	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
UniverseNet+GFL	47.5	65.8	51.8	29.2	51.6	62.5
UniverseNet-20.08d	**48.6**	**67.1**	**52.7**	**30.1**	**53.0**	**63.8**

(d) AP improvements by SyncBN [40] and iBN [60].

Method	DCN	FPS	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
UniverseNet-20.08d	heavy	17.3	**48.6**	**67.1**	**52.7**	**30.1**	**53.0**	**63.8**
UniverseNet-20.08	light	**24.9**	47.5	66.0	51.9	28.9	52.1	61.9

(e) Speeding up by the light use of DCN [13, 60].

Method	FPS	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
UniverseNet-20.08	24.9	**47.5**	**66.0**	**51.9**	**28.9**	**52.1**	**61.9**
w/o SEPC [60]	26.7	45.8	64.6	50.0	27.6	50.4	59.7
w/o Res2Net-v1b [18]	**32.8**	44.7	62.8	48.4	27.1	48.8	59.5
w/o DCN [13]	27.8	45.9	64.5	49.8	**28.9**	49.9	59.0
w/o multi-scale training	24.8	45.9	64.5	49.6	27.4	50.5	60.1
w/o SyncBN, iBN [40, 60]	25.7	45.8	64.0	50.2	27.9	50.0	59.8

(f) Ablation from UniverseNet-20.08. Replacing Res2Net-v1b backbone with ResNet-B [23] has the largest effects.

Backbone	FPS	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
ResNet-50-B [23]	**32.8**	44.7	62.8	48.4	27.1	48.8	59.5
ResNet-50-C [23]	32.4	45.8	64.2	50.0	28.8	50.1	60.0
Res2Net-50 [18]	25.0	46.3	64.7	50.3	28.2	50.6	60.8
Res2Net-50-v1b [18]	24.9	**47.5**	**66.0**	**51.9**	**28.9**	**52.1**	**61.9**

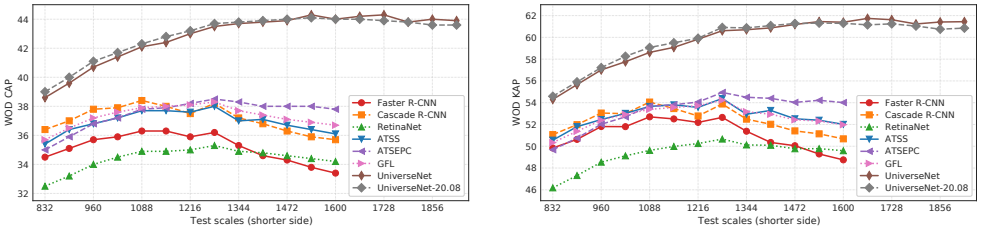
(g) UniverseNet-20.08 with different backbones.

Table 18: Ablation studies on COCO `minival`.

It is much more accurate than other models trained for 12 epochs using ResNet-50-level backbones (*e.g.*, ATSS: 39.4% [9, 66], GFL: 40.2% [9, 29]). On the other hand, the inference is not so fast (less than 20 FPS) due to the heavy use of DCN [13]. UniverseNet-20.08 speeds up inference by the light use of DCN [13, 60]. As shown in Table 18e, UniverseNet-20.08 is 1.4× faster than UniverseNet-20.08d at the cost of a ∼1% AP drop. To further verify the effectiveness of each technique, we conducted ablation from UniverseNet-20.08 shown in Table 18f. All techniques contribute to the high AP of UniverseNet-20.08. Ablating the Res2Net-v1b backbone (replacing Res2Net-50-v1b [18] with ResNet-50-B [23]) has the largest effects. Res2Net-v1b improves AP by 2.8% and increases the inference time by 1.3×. To further investigate the effectiveness of backbones, we trained variants of UniverseNet-20.08 as shown in Table 18g. Although the Res2Net module [18] makes inference slower, the deep stem used in ResNet-50-C [23] and Res2Net-50-v1b [18] improves AP metrics with similar speeds. UniverseNet-20.08s (the variant using the ResNet-50-C backbone) shows a good speed-accuracy trade-off by achieving 45.8% AP and over 30 FPS.

E.8 Effects of Test Scales

We show the results on WOD at different test scales in Figure 8a. Single-stage detectors require larger test scales than multi-stage detectors to achieve peak performance, probably because they cannot extract features from precisely localized region proposals. Although ATSEPC shows lower AP than ATSS at the default test scale (1248×832 in Standard USB), it outperforms ATSS at larger test scales (*e.g.*, 1920×1280 in Large USB). We conjecture that we should enlarge object scales in images to utilize SEPC [60] because its DCN [13] enlarges effective receptive fields. SEPC and DCN prefer large objects empirically (Tables 18a, 18f, [13, 60]), and DCN [13] cannot increase the sampling points for objects smaller than the kernel size in principle. By utilizing the characteristics of SEPC and multi-scale training, UniverseNets achieve the highest AP in a wide range of test scales.



(a) COCO-style AP (b) KITTI-style AP

Figure 8: Test scales *vs*. different AP metrics on WOD `f0val`.

Pre-training	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	body	face	frame	text
ImageNet	68.9	92.2	73.3	19.9	42.6	75.8	64.3	47.6	93.0	70.7
COCO 1×	**69.9**	**92.5**	**74.3**	**20.5**	**43.6**	**77.1**	**66.6**	**48.0**	93.7	**71.2**
COCO 2×	69.8	92.3	74.0	**20.5**	43.4	77.0	66.5	47.8	**93.8**	**71.2**

Table 19: UniverseNet-20.08 on Manga109-s `15test` with different pre-trained models.

E.9 Evaluation with KITTI-Style AP

We evaluated the KITTI-style AP (KAP) on WOD. KAP is a metric used in benchmarks for autonomous driving [20, 48]. Using different IoU thresholds (0.7 for vehicles, and 0.5 for pedestrians and cyclists), KAP is calculated as $\mathrm{KAP} = (\mathrm{AP}_{0.7,\mathrm{veh.}} + \mathrm{AP}_{0.5,\mathrm{ped.}} + \mathrm{AP}_{0.5,\mathrm{cyc.}})/3$. The results of KAP are shown in Figure 8b. GFL [29] and Cascade R-CNN [7], which focus on localization quality, are less effective for KAP.

E.10 Effects of COCO Pre-Training

To verify the effects of COCO pre-training, we trained UniverseNet-20.08 on M109s from different pre-trained models. Table 19 shows the results. COCO pre-training improves all the metrics, especially body AP.

References

[1] Waymo Open Dataset 2D detection leaderboard. https://waymo.com/open/challenges/2d-detection/, Accessed on June 18, 2020.

[2] Kiyoharu Aizawa, Azuma Fujimoto, Atsushi Otsubo, Toru Ogawa, Yusuke Matsui, Koki Tsubota, and Hikaru Ikuta. Building a manga dataset "Manga109" with annotations for multimedia applications. *IEEE MultiMedia*, 2020.

[3] Khalid Ashraf, Bichen Wu, Forrest N. Iandola, Mattthew W. Moskewicz, and Kurt Keutzer. Shallow networks for high-accuracy road object-detection. *arXiv:1606.01561*, 2016.

[4] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. YOLOv4: Optimal speed and accuracy of object detection. *arXiv:2004.10934*, 2020.

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

[6] Daniel Bolya, Sean Foley, James Hays, and Judy Hoffman. TIDE: A general toolbox for identifying object detection errors. In *ECCV*, 2020.

[7] Zhaowei Cai and Nuno Vasconcelos. Cascade R-CNN: Delving into high quality object detection. In *CVPR*, 2018.

[8] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020.

[9] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv:1906.07155*, 2019.

[10] Sijia Chen, Yu Wang, Li Huang, Runzhou Ge, Yihan Hu, Zhuangzhuang Ding, and Jie Liao. 2nd place solution for Waymo Open Dataset challenge – 2D object detection. *arXiv:2006.15507*, 2020.

[11] Yihong Chen, Zheng Zhang, Yue Cao, Liwei Wang, Stephen Lin, and Han Hu. RepPoints v2: Verification meets regression for object detection. In *NeurIPS*, 2020.

[12] Cheng Chi, Fangyun Wei, and Han Hu. RelationNet++: Bridging visual representations for object detection via transformer decoder. In *NeurIPS*, 2020.

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

[14] 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.

[15] Piotr Dollár, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: An evaluation of the state of the art. *TPAMI*, 2012.

[16] Xianzhi Du, Tsung-Yi Lin, Pengchong Jin, Golnaz Ghiasi, Mingxing Tan, Yin Cui, Quoc V. Le, and Xiaodan Song. SpineNet: Learning scale-permuted backbone for recognition and localization. In *CVPR*, 2020.

[17] Cheng-Yang Fu, Wei Liu, Ananth Ranga, Ambrish Tyagi, and Alexander C. Berg. DSSD : Deconvolutional single shot detector. *arXiv:1701.06659*, 2017.

[18] Shang-Hua Gao, Ming-Ming Cheng, Kai Zhao, Xin-Yu Zhang, Ming-Hsuan Yang, and Philip Torr. Res2Net: A new multi-scale backbone architecture. *TPAMI*, 2021.

[19] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. YOLOX: Exceeding YOLO series in 2021. *arXiv:2107.08430*, 2021.

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

[21] Ross Girshick, Ilija Radosavovic, Georgia Gkioxari, Piotr Dollár, and Kaiming He. Detectron. https://github.com/facebookresearch/detectron, 2018.

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

[23] Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of tricks for image classification with convolutional neural networks. In *CVPR*, 2019.

[24] Derek Hoiem, Yodsawalai Chodpathumwan, and Qieyun Dai. Diagnosing error in object detectors. In *ECCV*, 2012.

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

[26] Zehao Huang, Zehui Chen, Qiaofei Li, Hongkai Zhang, and Naiyan Wang. 1st place solutions of Waymo Open Dataset challenge 2020 – 2D object detection track. *arXiv:2008.01365*, 2020.

[27] Naoto Inoue, Ryosuke Furuta, Toshihiko Yamasaki, and Kiyoharu Aizawa. Cross-domain weakly-supervised object detection through progressive domain adaptation. In *CVPR*, 2018.

[28] Kang Kim and Hee Seok Lee. Probabilistic anchor assignment with IoU prediction for object detection. In *ECCV*, 2020.

[29] 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. In *NeurIPS*, 2020.

[30] Tsung-Yi Lin, Piotr Dollár, et al. COCO API. https://github.com/cocodataset/cocoapi, Accessed on Nov. 8, 2020.

[31] 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.

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

[33] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, 2017.

[34] 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.

[35] 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. In *ICCV*, 2021.

[36] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *CVPR*, 2022.

[37] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

[38] Yusuke Matsui, Kota Ito, Yuji Aramaki, Azuma Fujimoto, Toru Ogawa, Toshihiko Yamasaki, and Kiyoharu Aizawa. Sketch-based manga retrieval using Manga109 dataset. *Multimedia Tools and Applications*, 2017.

[39] Toru Ogawa, Atsushi Otsubo, Rei Narita, Yusuke Matsui, Toshihiko Yamasaki, and Kiyoharu Aizawa. Object detection for comics using Manga109 annotations. *arXiv:1803.08670*, 2018.

[40] Chao Peng, Tete Xiao, Zeming Li, Yuning Jiang, Xiangyu Zhang, Kai Jia, Gang Yu, and Jian Sun. MegDet: A large mini-batch object detector. In *CVPR*, 2018.

[41] Siyuan Qiao, Liang-Chieh Chen, and Alan Yuille. DetectoRS: Detecting objects with recursive feature pyramid and switchable atrous convolution. *arXiv:2006.02334*, 2020.

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

[43] Henry A. Rowley, Shumeet Baluja, and Takeo Kanade. Neural network-based face detection. *TPAMI*, 1998.

[44] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 2015.

[45] Yosuke Shinya, Edgar Simo-Serra, and Taiji Suzuki. Understanding the effects of pre-training for object detectors via eigenspectrum. In *ICCV Workshop on Neural Architects*, 2019.

[46] Bharat Singh and Larry S. Davis. An analysis of scale invariance in object detection – SNIP. In *CVPR*, 2018.

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

[48] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo Open Dataset. In *CVPR*, 2020.

[49] Peize Sun, Rufeng Zhang, Yi Jiang, Tao Kong, Chenfeng Xu, Wei Zhan, Masayoshi Tomizuka, Lei Li, Zehuan Yuan, Changhu Wang, and Ping Luo. Sparse R-CNN: End-to-end object detection with learnable proposals. In *CVPR*, 2021.

[50] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[51] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016.

[52] Ryo Takahashi, Takashi Matsubara, and Kuniaki Uehara. Data augmentation using random image cropping and patching for deep CNNs. *IEEE TCSVT*, 2020.

[53] Mingxing Tan. A comment about the training log of EfficientDet. https://github.com/google/automl/issues/380#issuecomment-800814557, Accessed on Aug. 23, 2021.

[54] Mingxing Tan and Quoc V. Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In *ICML*, 2019.

[55] Mingxing Tan, Ruoming Pang, and Quoc V. Le. EfficientDet: Scalable and efficient object detection. In *CVPR*, 2020.

[56] Mingxing Tan, Ruoming Pang, and Quoc V. Le. EfficientDet: Scalable and efficient object detection. *arXiv:1911.09070v7*, 2020.

[57] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. FCOS: Fully convolutional one-stage object detection. In *ICCV*, 2019.

[58] Yuji Tokozume, Yoshitaka Ushiku, and Tatsuya Harada. Between-class learning for image classification. In *CVPR*, 2018.

[59] Angelina Wang, Arvind Narayanan, and Olga Russakovsky. REVISE: A tool for measuring and mitigating bias in visual datasets. In *ECCV*, 2020.

[60] Xinjiang Wang, Shilong Zhang, Zhuoran Yu, Litong Feng, and Wayne Zhang. Scale-equalizing pyramid convolution for object detection. In *CVPR*, 2020.

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

[62] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017.

[63] Hang Xu, Chenhan Jiang, Dapeng Feng, Chaoqiang Ye, Rui Sun, and Xiaodan Liang. SPNAS-Noah: Single Cascade-RCNN with backbone architecture adaption for Waymo 2D detection. https://sites.google.com/view/cvpr20-scalability/wod-reports, Accessed on June 21, 2020.

[64] Xuehui Yu, Yuqi Gong, Nan Jiang, Qixiang Ye, and Zhenjun Han. Scale match for tiny person detection. In *WACV*, 2020.

[65] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2018.

[66] 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.

[67] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. Simple multi-dataset detection. In *CVPR*, 2022.

[68] 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.