End-to-End Single Shot Detector using Graph-based Learnable Duplicate Removal

Shuxiao Ding[1,2][0000−0002−4040−5585], Eike Rehder[1][0000−0002−6255−0724], Lukas Schneider[1][0000−0001−9708−9248], Marius Cordts[1][0000−0001−8729−9233], and Juergen Gall[2][0000−0002−9447−3399]

[1] Mercedes-Benz AG, Stuttgart, Germany
{shuxiao.ding,eike.rehder,lukas.schneider,marius.cordts}@mercedes-benz.com
[2] University of Bonn, Bonn, Germany
gall@iai.uni-bonn.de

Abstract. Non-Maximum Suppression (NMS) is widely used to remove duplicates in object detection. In contrast to the end-to-end learning paradigm, NMS often remains as the only heuristic step that is not learned. While approaches for learning NMS have been proposed, they are either designed for the two-stage detector Faster-RCNN or rely on a separate network. In contrast, learning NMS for one-stage detectors like SSD is not well investigated. In this paper, we show that even a very simple rescoring network can be trained end-to-end with an underlying one-stage detector to solve the duplicate removal problem efficiently. For this, detection scores and bounding boxes are refined from image features by modeling relations between detections in a Graph Neural Network (GNN). To deal with the large number of object proposals of one-stage detectors, we propose a pre-filtering head, which can easily be employed in arbitrary SSD-like models with a weight-shared box predictor. Experiments on MS-COCO and KITTI show that our method improves accuracy compared with other duplicate removal methods at significantly lower inference time.

Keywords: Non-maximum suppression, object detection, duplicate removal, graph neural network

1 Introduction

Object detection is a fundamental and crucial task in computer vision. In the deep learning era, most object detection algorithms can be divided into two different categories: one-stage detectors such as YOLO [17] and Single Shot Detector (SSD) [15] or two-stage detectors such as Faster-RCNN [18], Mask-RCNN [5] or R-FCN [2]. In real-time applications, SSD and its variants MobileNet [8], RetinaNet [13], EfficientDet [22] have become more and more popular due to their efficiency, e.g. in robotics or automated driving. Regardless of the detection pipeline, the object detectors usually generate multiple detections for a single object. Therefore, Non-Maximum Suppression (NMS) is employed to remove these duplicates. NMS is a heuristic algorithm that solely relies on the

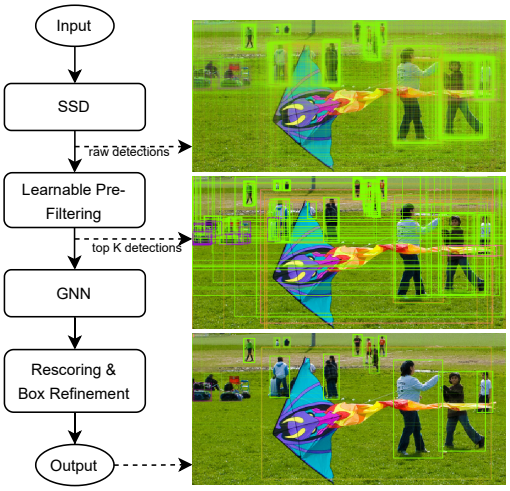

Fig. 1. We propose an end-to-end duplicate removal network for SSD. To deal with the large amount of raw detections (*top*), the network pre-filters all detections (*middle*). These are fed into a GNN to pass information between overlapping detections. Finally, the network performs a rescoring and a box refinement for each candidate, aiming to produce only one high-scoring and precise box for each instance (*bottom*). The opacity of boxes is proportional to the score. Different categories are depicted in different colors.

score and overlap between detections as criterion for removal. In order to make use of the meaningful image features of the detection system, it is desirable to replace the hand-engineered NMS by a learnable component.

Following the deep learning paradigm of the end-to-end optimization, learning-based duplicate removal approaches have been proposed. Most of them use a separate network which processes raw detections with handcrafted features from an existing object detector [6], [7], [16]. Usually, these duplicate removal methods are not directly applicable for SSD-like architectures and, thus, can hardly be used in real time applications. The reasons for this are mainly due to two challenges: First, SSD provides significantly more raw detections (*top* image of Fig. 1) compared to its two-stage counterpart. Since all existing approaches model relations between detections, the computational complexity increases quadratically with the number of input detections. Second, image features in SSD are of lower dimensionality. Thus, they are less discriminative when embedding them into the input of a separate duplicate removal network. Since especially real time applications demand for the use of SSD architectures, these challenges have to be overcome in order to employ true end-to-end detectors.

In this paper, we propose to learn the duplicate removal for SSD architectures. To cope with the large amount of candidates, we first introduce a learnable pre-filtering module that filters the highly overlapping boxes from the raw detections of SSD in an early stage (*middle* image of Fig. 1). We then model interactions between the top-K filtered detection candidates to eliminate those corresponding to the same object. The set of detections is regarded as an undi-

rected graph, where an *edge* exists between two detections (*nodes*) if they overlap. In our work, node features are propagated along the edges in the graph using a Graph Convolutional Network (GCN) [11] in order to obtain a single refined and rescored detection per object (*bottom* image of Figure 1). Although existing approaches for duplicate removal also model these relationships, they either rely on a deep network architecture with a long inference time [6], [7], [16] or treat the detections as a fully-connected graph [9], [16]. In contrast, our approach is much more efficient, and it is thus applicable to real-time sensitive applications. To train the network, we propose to use a bipartite matching that uniquely assigns ground-truth annotations to detections. It considers the localization and classification quality simultaneously instead of using a greedy matching as in existing related works.

We showcase the effectiveness of our proposed duplicate removal approach with two popular SSD models: EfficientDet [22] and RetinaNet [13]. Extensive experiments on COCO [14] and KITTI [4] show that our efficient approach achieves still a generally higher accuracy than more expensive duplicate removal methods. While maintaining the accuracy, our approach has an extremely small complexity and does not require any post-processing other than thresholding. Consequently, it runs significantly faster: the inference of the entire model is 24.5% and 19.5% faster than NMS for EfficientDet-D0 and RetinaNet-ResNet50, respectively, while achieving better mAP.

2 Related Work

As of today, standard detection networks still employ manually engineered duplicate removal such as NMS [15], [18], [22]. In order to remove it, two categories of approaches exist. Either, the detector is designed to produce duplicate-free detections or they are removed by an additional network component.

NMS-free object detection Networks that can operate without an explicit NMS need to suppress duplicates within their detection pipeline. In the field of human detection in crowded situations, [20] proposed a pipeline with a CNN backbone as encoder for extracting visual features and an LSTM controller as decoder for generating the set of detections iteratively. Inspired by [20] and [25], DETR [1] and its successors [28] [19] generate the detection set directly by implicit relationship modeling using a Transformer encoder and decoder after a CNN feature extractor. CenterNet [27] identifies local maxima in the centerness score, which effectively is a NMS for box centers. Our approach also learns an additional score, but it identifies non-duplicate boxes using a global loss based on a one-to-one target assignment. OneNet [21] uses a combination of a classification and localization loss as matching cost and a one-to-one matching for target assignment. While we also consider both classification and localization for target assignment, the features in our approach are updated by explicitly modeling the relationship between detections.

Learning duplicate removal Instead of the classical NMS, learnable duplicate removal techniques have been proposed. They can be divided into two groups. The first group employed a separate network that processes detection results from an object detector. GossipNet [7] uses handcrafted detection pair features as input and updates the representation of every detection by iterative communication between every detection pair. [16] also encodes the handcrafted features from raw detections as input and rescores detections with bidirectional RNN and self-attention. However, the network architectures of above methods are complicated and not shown to be applicable to SSD-like architectures such as RetinaNet [13] or EfficientDet [23]. A second line of these works proposed end-to-end trainable networks. For two-stage detectors, Relation Networks [9] allow learning duplicate removal using an object relation module based on scaled dot-product self-attention [25], which is easily attached to the refinement network in Faster-RCNN. For one-stage detectors, [26] proposed a positive sample selector by attaching a new head in FCOS [24] for eliminating the NMS. In this work, we show that a learnable duplicate removal network can also be added to SSD-like architectures and trained end-to-end.

3 Method

3.1 System overview

An overview of our proposed end-to-end learned duplicate removal approach for a Single Shot Detector (SSD) is shown in Figure 2. As a central concept, we treat detections as nodes in a graph and modify their predictions through a Graph Neural Network (GNN). For this, we start off with SSD as the detection architecture. To reduce the number of detection candidates, we attach a pre-filtering head to the classification branch, which produces a score offset for all detections followed by a top-K sampler. The filtered detections are then fed into a Graph Convolutional Network (GCN), which updates the node representations by message-passing and finally outputs a score offset and box refinement parameters. During training, the final detections are matched one-to-one with the ground-truth annotations using the Hungarian algorithm.

3.2 Local candidate pre-filtering

Given a standard SSD, detections are generated based on an evenly distributed grid of prior boxes with different locations, sizes, and aspect ratios for different feature map scales. Therefore, the number of detection candidates in SSD is by orders of magnitude larger than in Faster-RCNN, which pre-filters object proposals in the region proposal network using NMS. As the learned duplicate removal relies on the relationship of overlapping detections and the computational overhead explodes when the number of proposals increases, a candidate sampling for downstream duplicate removal is necessary. A straight-forward filtering is a top-K algorithm which keeps the K detections with highest scores

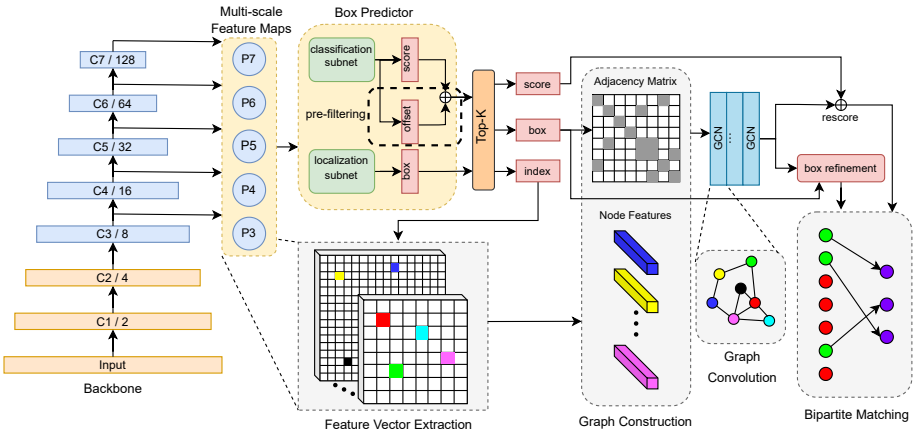

Fig. 2. An overview of our proposed network architecture. First, the base SSD model produces multi-scale feature maps using a backbone network including some feature pyramid network (FPN). Raw detections are generated by a weight-shared box predictor. The pre-filtering is done by a rescoring using an additional head and a top-K selection. The graph adjacency is constructed based on the overlap between detections, while initial node features are extracted from the feature map. The graph data is then processed by GCN layers that enable message passing between neighboring detections. The updated node representations are projected into a score offset and four box parameters. For training, the rescored and refined detection sets are matched uniquely with ground truth using a bipartite matcher.

because most detections in SSD contain only background. However, since high scoring detections may be highly redundant, this might discard positives with relatively low confidence. Classical NMS, on the other hand, is able to keep these by processing all detection candidates. For this reason, we aim to reduce the candidates to a reasonable amount while keeping as many potential true positives as possible.

Pre-filtering head Similar to previous learned NMS approaches [7], we consider the filtering as a learnable rescoring problem. To this end, we add a network which produces a score offset to indicate whether a detection candidate should be removed. Inspired by [26] for FCOS [24], we attach an additional head to the classification subnet. Parallel to the last convolutional layer that generates C binary classifications z_c^{cls} for every anchor in SSD, the new head predicts a class-agnostic confidence Δz^{pre} by a single 3×3 convolutional layer. C corresponds to the number of categories. Given a feature map $F_s \in \mathbb{R}^{H_s \times W_s \times C_F}$ with downsampling rate s, the output dimension of the pre-filtering head is $H_s \times W_s \times A$ where A is the number of anchors on every pixel. The rescoring is done by an addition in the logit space $z_c^{\text{pre}} = z_c^{\text{cls}} + \Delta z^{\text{pre}}$. All the C classes share the same offset for simplicity and efficiency. A top-K algorithm selects the best K detections for each class based on the new score after sigmoid activation $s_c^{\text{pre}} = \sigma(z_c^{\text{pre}})$.

Training objective The pre-filtering head is trained in order to remove the most obvious duplicates. For this, the convolutional layers in the box predictor and the pre-filtering head have only a local receptive field for identifying objects. In this case, the highly overlapping duplicated detections (as observed in the *top* image of Fig. 1) that stem from surrounding locations can be easily recognized by the pre-filtering network. To generate its training targets, we use a class-agnostic NMS with a high IoU threshold $T_1 = 0.9$. By setting a high threshold T_1, the outputs of the NMS, which are used as supervision to train the pre-filtering head, are more likely to cover all true positives while only discarding highly overlapping duplicates. Let $y_i \in \{0, 1\}$ indicate whether a certain box i shall be kept, defined by the class-agnostic NMS. The pre-filtering head is then trained with a Binary Cross Entropy loss $\mathcal{L}_{\text{pre}} = \sum_i \text{BCE}(\Delta z_i^{\text{pre}}, y_i)$. This training objective produces a sparser set of detections, which lead to a sparser adjacency matrix with less locally fully-connected clusters when constructing the graph for the proposed approach for NMS. This is helpful to mitigate the problem caused by over-smoothing [12] in GCNs.

3.3 Learning NMS based on GCN

Graph construction Let $G = (V, E)$ be a graph that we construct from detection candidates, where V is the set of nodes and E is the set of edges between them. The nodes V correspond to the top K detections after pre-filtering. An adjacency matrix $\mathbf{A} \in \{0, 1\}^{K \times K}$ represents the edges between nodes. For a pair of detection boxes (b_i, b_j) from V, an edge exists (i.e. $A_{ij} = 1$) if $\text{IoU}(b_i, b_j) > 0$ and otherwise $A_{ij} = 0$. Based on every filtered detection, we trace the location (x_s, y_s) in its corresponding feature map F_s at scale s and extract the feature vector $\mathbf{f}_A \in \mathbb{R}^{C_F}$ at this location as image feature. Since SSD generates multiple detections on every feature map pixel, detections that stem from the same pixel share the same image feature. For a more discriminative feature vector, we follow Relational Network [9] to use a rank feature. Concretely, all the K detections are sorted in descending order and each detection is given a rank r in $[1, K]$. This scalar is then embedded into a rank feature $\mathbf{f}_R \in \mathbb{R}^{C_F}$ using a positional encoding as demonstrated in [25]. Both feature vectors are projected into a C_N-dimensional space and then added to build the initial node feature $\mathbf{f} = \mathbf{W}_R \mathbf{f}_R + \mathbf{W}_A \mathbf{f}_A$, where $\mathbf{W}_R$ and $\mathbf{W}_A$ are learnable weights. We found that additional handcrafted features like box coordinates, score or box overlap did not improve the accuracy of our approach. Because image features are extracted from different feature maps, an SSD architecture with a weight-shared box predictor is necessary, which is able to produce semantically equal feature maps so that the extracted feature vectors are comparable in the same graph.

Network architecture We employ the simple Graph Convolutional Network [11] for updating detection representations by propagating messages between overlapping detections. Given the adjacency matrix $\mathbf{A} \in \mathbb{R}^{K \times K}$ and an input feature matrix $\mathbf{F} \in \mathbb{R}^{K \times C_N}$, a Graph Convolutional Layer generates the output

feature matrix $\mathbf{F}' \in \mathbb{R}^{K \times C_N^{'}}$ by:

$$\mathbf{F}' = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\mathbf{F}\mathbf{W} + \mathbf{b}, \tag{1}$$

where $\mathbf{D}$ denotes the degree matrix of $\mathbf{A}$ with $D_{ii} = \sum_j A_{ij}$. $\mathbf{W}$ and $\mathbf{b}$ are the learnable weights and bias term, respectively. Note that we do not add the self-loop by $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ because our IoU-based adjacency matrix includes it naturally. After message propagation from stacked Graph Convolutional Layers in Equ. (1), the node representations contain information on its neighborhood.

We then use a linear classifier to predict a score offset based on the updated feature representations. In a multi-class setting ($C > 1$), the same network is operated on all classes independently and thus generates the class-specific score offset Δz_c^{gcn} for class c. Together with the class-agnostic offset from pre-filtering, the final score of a detection for category c is generated by adding the score offsets to the score of SSD (z_c^{cls}) in logit space followed by a sigmoid function: $s_c = \sigma(z_c^{\text{cls}} + \Delta z^{\text{pre}} + \Delta z_c^{\text{gcn}})$. The scores of duplicated boxes are strongly decreased after rescoring so that the remaining high-scoring boxes are identified as true-positives using a score threshold.

When solely applying a rescoring, a detection box with highest updated score might have an imperfect localization. Combined with our matching strategy (Sec. 3.4), a confident prediction with insufficient precise localization will be less likely to be matched. To solve this problem, a box refinement can be applied which benefits from the information of neighboring duplicated detections in message passing. Therefore, we add an additional fully-connected layer to predict, parallel to the score offset, four box coordinate offsets $\{\Delta x, \Delta y, \Delta w, \Delta h\}$ according to the box encoding in Faster-RCNN [18]:

$$\Delta x = (x' - x)/w,\ \Delta y = (y' - y)/h,\ \Delta w = \log(w'/w),\ \Delta h = \log(h'/h), \tag{2}$$

where $\{x, y, w, h\}$ are bounding box parameters from SSD and $\{x', y', w', h'\}$ represent the refined bounding box.

3.4 Training

Target assignment with bipartite matching The message passing network outputs a set of detections with new scores and refined boxes but without changing the number of detections K, which is typically much larger than the number of ground truth objects N. For training, we seek an optimal assignment of a ground-truth object to the best detection determined by a combination of classification and localization accuracy. Most previous works [7], [9], [16] follow the simpler evaluation pipeline of Pascal VOC [3] or MS-COCO [14] for target assignment. This matching algorithm first sorts all detections in descending order and then greedily picks the detection with highest score for a ground-truth instance, when its IoU with this ground-truth box exceeds a pre-defined threshold. However, the greedy matching selects boxes based on the classification score at an unchanged IoU level without further consideration of localization.

Following the works with set prediction [20], [1], we introduce a pairwise matching cost which consists of a classification and a localization cost balanced with weights α and β:

$$L_{ij}(d_i, y_j) = -\mathbb{1}\left[\mathrm{IoU}(\hat{b}_i, b_j) > T_2\right]\left(\alpha\hat{s}_i + \beta\mathrm{IoU}(\hat{b}_i, b_j)\right), \tag{3}$$

given the classification score $\hat{s}_i$ and box $\hat{b}_i$ of detection d_i as well as a ground-truth annotation y_j with a bounding box b_j. The parameter T_2 is an IoU threshold that controls the positive and negative samples, similar to the one used for the anchor matching in SSD. The optimal matching between the set of detections $\{d_i\}^K$ and ground truths $\{y_j\}^N$ can be represented with an assignment matrix $X \in \{0,1\}^{K\times N}$, where $X_{ij} = 1$ indicates a matching between d_i and y_j. In a bipartite matching, each row of X is assigned at most to one column and vice versa. The optimal assignment X which minimizes the overall cost

$$\mathcal{L}_{\mathrm{match}} = \min_X \sum_i \sum_j L_{ij} X_{ij} \tag{4}$$

can be efficiently found by the Hungarian algorithm. With the matching cost in Equ. (3), we prefer to select a box with both high score and precise localization.

Overall training objective We train our proposed network together with the classification loss $\mathcal{L}_{\mathrm{cls}}$ and the localization loss $\mathcal{L}_{\mathrm{loc}}$ of SSD. We use the focal loss [13] and smooth-L1 loss as classification loss $\mathcal{L}_{\mathrm{nms,cls}}$ and localization loss $\mathcal{L}_{\mathrm{nms,loc}}$ in our learnable NMS network, respectively. Similar to the localization loss in SSD, the loss term $\mathcal{L}_{\mathrm{nms,loc}}$ is only calculated for the positive (matched) samples. Note that we use the final score s_c after rescoring as the prediction in the focal loss, thus the gradient from $\mathcal{L}_{\mathrm{nms,cls}}$ also flows into the pre-filtering head and classification subnet. The overall training objective is

$$\mathcal{L} = \lambda_1(\mathcal{L}_{\mathrm{cls}} + \mathcal{L}_{\mathrm{loc}}) + \lambda_2\mathcal{L}_{\mathrm{pre}} + \lambda_3\mathcal{L}_{\mathrm{nms,cls}} + \lambda_4\mathcal{L}_{\mathrm{nms,loc}}, \tag{5}$$

where $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ are weights for each loss term.

4 Experiments

4.1 Implementation Details

We show detection results on the MS-COCO [14] and KITTI 2D Object Detection [4] datasets. To avoid training instability caused by gradient domination of the duplication removal components at the beginning of the training, we train our model in two stages: first, the vanilla SSD model is trained alone. In the second stage, our learnable NMS network is attached to the base SSD network and the whole network is trained jointly according to Equ. (5) with a smaller learning rate. For a fair comparison, we compare our approach to a vanilla SSD that is trained for the same amount of epochs as the two stages together.

Table 1. Results on MS-COCO *val* and *test-dev*. We compare our approach with NMS and GossipNet [7]. AP_{50} and AP_{75} are AP at 50% IoU and 75% IoU. The inference time corresponds to the entire process including pre- and post-processing and we also show the inference time of the GossipNet alone in brackets.

base model	method	*val*			*test-dev*			
		AP	AP_{50}	AP_{75}	AP	AP_{50}	AP_{75}	time [ms]
EfficientDet-D0	NMS	31.6	**50.0**	33.3	32.0	**50.5**	33.8	47.3
	GossipNet	31.6	49.7	33.7	31.5	49.9	33.7	166.1(131.1)
	ours	**32.7**	49.9	**35.9**	**32.7**	49.8	**36.1**	**35.7**
RetinaNet-R50	NMS	34.4	52.1	36.8	34.2	**51.7**	37.2	59.7
	GossipNet	**34.7**	**52.2**	37.7	32.0	49.9	34.3	154.2(107.1)
	ours	**34.7**	50.8	**38.7**	**34.6**	51.0	**38.5**	**48.0**

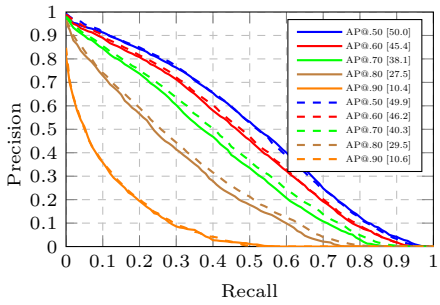

Fig. 3. Comparison of Precision-Recall Curves at different IoUs. Classical NMS is depicted in solid line, our approach in dashed line.

Fig. 4. Qualitative comparisons between our approach and classical NMS. We keep the top 100 detections for visualization. Box opacity is proportional to the score.

To validate the generalization ability, we use two different SSD models for evaluation: EfficientDet-D0 [22] and RetinaNet-ResNet50 [13]. The numbers of initial node feature channels C_N and GCN output channels $C_N^{'}$ are always set to $C_N = C_N^{'} = 4C_F$, where C_F corresponds to the number of channels of the feature map in the base SSD model. We stack only one GCN layer for EfficientDet and two GCN layers for RetinaNet. Furthermore, we use $K = 300$, $T_1 = 0.9$, $T_2 = 0.7$ and $\lambda_1 = \lambda_2 = \lambda_3 = \lambda_4 = 1.0$ for all baseline models.

4.2 MS-COCO Experiments

MS-COCO is a challenging object detection dataset with 80 different categories. There are about 118k images in the *train* set, 5k images in the *val* set and 20k images in the *test-dev* set. All models are trained on 4 NVIDIA Tesla V100-PCIE-16GB GPUs with SGD. The batch sizes are 48 for EfficientDet-D0 and 16 for RetinaNet-ResNet50. A momentum of 0.9 and cosine learning rate de-

cay schedule are used for both stages. For EfficientDet, we train the backbone detection network for 300k steps with base learning rate 0.08 in the first stage and a further 100k steps with a base learning rate 0.015 in the second stage. For RetinaNet, the base learning rate is 0.01 for the first 200k steps and 5×10^{-4} for the next 120k steps. Unless mentioned otherwise, ablation studies are performed on the *val* set with the EfficientDet-D0 backbone.

Comparison with baselines

Choice of baselines We first compare our approach with other duplicate removal methods using the same backbone network architectures. For a fair comparison, we train all networks with the same setup as explained above. To ensure validity of the baseline performance, we built our method based on the TensorFlow Object Detection API [10] and used their EfficientDet and RetinaNet implementation trained with the same hyperparameters as our baselines. Note that our best EfficientDet baseline falls 2.6% short on *test-dev* compared to the numbers reported in the original paper [22]. This is caused by our smaller batch size (48 vs. 128) and less training epochs (160 epochs vs. 300 epochs). Since we aim to evaluate the performance improvements of different NMS approaches, this does not limit the comparison since all approaches are trained in the same way.

We select the standard NMS with an IoU threshold of 0.5 as a baseline. We also compare to GossipNet [7] using the code provided by the authors. As input, GossipNet takes the same number of raw detections from SSD as our approach ($K = 300$) since it performed better than providing more detections to GossipNet. Other hyperparameters remain unchanged. The comparison on the COCO *val* and *test-dev* sets are shown in Table 1.

Performance For EfficientDet, our approach improves 1.1 percentage points on *val* and 0.7 on *test-dev* compared with NMS. It is worth mentioning that our approach achieves a significant improvement of AP_{75}, which can be attributed to our localization refinement loss and position-aware matching cost for the target assignment. We can also observe significant improvements at relatively high IoU thresholds (AP@0.70 and AP@0.80) in the precision-recall curves in Figure 3, which validates the better localization accuracy of our approach. While GossipNet achieves an improvement of AP for Faster-RCNN by 0.8% on COCO *test-dev* in our reproduced results, it does not outperform the standard NMS when applied to EfficientDet. The performance gain of our approach for RetinaNet is modest compared to the gains for EfficientDet. We argue that RetinaNet has already achieved a better localization quality than EfficientDet (3.4% AP_{75} vs. 1.2% AP_{50} improvement on *test-dev*) and thus the potential of AP improvement at a higher IoU threshold is smaller. However, our approach still performs better than the NMS baseline on both sets. It is noticed that, although GossipNet achieves the same AP as our method on *val*, the performance decreases by a large margin on *test-dev*, which indicates limited generalization. A qualitative comparison with classical NMS is shown in Figure 4. In the first example, classical NMS produces a duplicate box between the second and third person from

Table 2. Impact of number of GCN layers (L) and channels of initial features (C_N). The number of parameters and FLOPs include the node feature generation and final dense layers. We also show the number of parameters and FLOPs of the base model EfficientDet-D0.

L	C_N	AP	AP_{50}	AP_{75}	Params	FLOPs
0	256	30.2	47.4	33.2	83.72k	0.050B
1	256	**32.7**	**49.9**	**35.9**	149.51k	0.136B
2	256	32.4	49.4	35.3	215.30k	0.221B
3	256	32.3	49.4	35.4	281.09k	0.307B
1	64	32.5	49.8	35.6	29.09k	0.027B
1	128	32.4	49.6	35.6	58.37k	0.058B
1	512	32.5	49.8	35.6	430.09k	0.350B
EfficientDet-D0					3.9M	2.5B

Table 3. Impact of the weights for the loss terms and the modules for pre-filtering and localization refinement. λ_1, λ_2, λ_3 and λ_4 are the weights of the different loss terms in Equ. (5). − indicates that the corresponding module is not used.

Ablation	λ_1	λ_2	λ_3	λ_4	AP	AP_{50}	AP_{75}
full model	1.0	1.0	1.0	1.0	**32.7**	**49.9**	**35.9**
influence of pre-filtering	1.0	**0.5**	1.0	1.0	32.4	49.3	35.6
	1.0	**0.3**	1.0	1.0	32.5	49.7	35.8
	1.0	**0.1**	1.0	1.0	32.3	49.0	35.5
	1.0	−	1.0	1.0	31.4	48.0	34.4
influence of loc. refinement	1.0	1.0	1.0	**0.5**	32.2	49.6	35.5
	1.0	1.0	1.0	**0.3**	31.9	49.4	35.0
	1.0	1.0	1.0	**0.1**	31.4	48.9	34.2
	1.0	1.0	1.0	−	30.7	48.4	33.6

right, while our approach is able to suppress that box in the crowded scene. In the second example, our method is able to recover low-scoring detections after rescoring.

Runtime The last column of Table 1 shows the inference time of the entire model including pre- and post-processing on a NVIDIA RTX 2080Ti GPU. Our approach reduces inference time by 24.5% and 19.5% for both models compared to using NMS, thus further accelerates the efficient SSD model and increases the ability to be embedded into real-time applications. The inference time of the pure GossipNet model without considering SSD is shown in the bracket which is significant longer ($>$ 100ms) due to its complex architecture.

Ablation study

Impact of the GCN architecture An ablation study of the number of layers in the GCN architecture for the EfficientDet-D0 base model is shown in Table 2. We observe that the network does not need a deep architecture. One GCN layer, $L = 1$, provides the best performance. This may be caused by the over-smoothing in GCNs if too many layers are stacked. In addition, the message from one-hop neighbors are most important for duplicate removal. Without GCN layers ($L = 0$), the rescoring and refinement are made using the initial node features. This leads to a significant AP drop by 2.5%, which validates the importance of modeling the relations of detections for duplicate removal. For the number of channels, 256 shows the best performance. More learnable parameters do not lead to a better performance. According to the number of parameters and FLOPs, our model performs the duplicate removal and refinement with a minimal time and memory overhead compared to the EfficientDet-D0 model.

Table 4. Results on KITTI test set. The subscription E, M, H corresponds to Easy, Moderate and Hard, respectively.

	Car			Pedestrian			Cyclist		
	AP_E	AP_M	AP_H	AP_E	AP_M	AP_H	AP_E	AP_M	AP_H
NMS	76.21	52.92	43.38	**52.32**	**39.46**	**35.60**	**35.72**	**23.07**	**20.54**
ours	**78.55**	**61.93**	**53.72**	51.24	38.40	35.24	34.13	22.43	19.65

Proposed modules Table 3 shows an evaluation of the pre-filtering and localization refinement by decreasing its corresponding loss weights until it is fully disabled. The performance drop is relatively small when decreasing the pre-filtering weight λ_2, as shown in rows 2-5. This can be attributed to the gradient flow from the NMS classification loss $\mathcal{L}_{\text{nms,cls}}$ into the pre-filtering head, which jointly optimizes the pre-filtering ability. By disabling the candidate pre-filter, the mAP drops by 1.3%. We see that the AP of our approach after disabling pre-filtering (31.4%) is close to GossipNet (31.6% in Table 1), while both setting simply selects the top-K boxes with highest SSD scores as input. This shows an interpretation of the performance gap between GossipNet and our approach and indicates the importance of our pre-filtering method. In rows 6-9, we gradually decrease the weight of the NMS localization refinement loss λ_4 which also decreases performance. A reason for the performance drop is the matching cost that requires a high localization quality. It also reveals that rescoring alone cannot guarantee that high-scoring detections also have the best localization.

4.3 KITTI Experiments

For the KITTI 2D Object Detection Benchmark, we use EfficientDet-D0 trained on one single GPU with a batch size of 8 as baseline. The training schedule consists of first 120k iterations with a base learning rate of 0.005 for training the vanilla SSD and then 40k iterations with a base learning rate of 0.0005 for training the entire model. All the other settings are the same as for the experiments on MS-COCO. The results on the KITTI test set are shown in Table 4. For car detection, we observe a dramatic performance improvement against classical NMS: For Moderate and Hard, our approach improves the AP by around 10 percentage points. This can be attributed to the frequent occlusion and truncation of these objects in the traffic scenarios (e.g. closely parking cars), where our learning-based approach is able to deal with the crowded situation. For pedestrians and cyclists, however, the detection performance drops slightly when applying our approach. The reason lies in KITTI's evaluation protocols for different categories: AP_{70} is used for car but AP_{50} for pedestrian and cyclist. As shown in Fig. 3, our method outperforms NMS at AP_{70} but not at AP_{50}. Note that AP_{70} is a more accurate and difficult measure than AP_{50}.

5 Conclusion

In this paper, we proposed a novel learned duplicate removal network which can be easily embedded into SSD-like models and trained end-to-end. For duplicate removal, detection candidates are treated as nodes in a graph. Their relationship is processed by a Graph Convolutional Network (GCN) to reason about dependencies and suppression. In order to reduce the amount of candidates within this graph, a simple learnable pre-filtering head was introduced. Experimental evaluations showed that the proposed network outperforms classical NMS and other duplicate removal methods while at the same time reducing inference time significantly. Consequently, our work removes the last hand engineered component from SSD. The resulting architecture forms an entirely learned object detection pipeline within realtime constraints.

6 Acknowledgement

References

1. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: European Conference on Computer Vision. pp. 213–229. Springer (2020)
2. Dai, J., Li, Y., He, K., Sun, J.: R-fcn: Object detection via region-based fully convolutional networks. In: Advances in neural information processing systems. pp. 379–387 (2016)
3. Everingham, M., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes (voc) challenge. International journal of computer vision **88**(2), 303–338 (2010)
4. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition. pp. 3354–3361. IEEE (2012)
5. He, K., Gkioxari, G., Dollar, P., Girshick, R.: Mask r-cnn. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV) (Oct 2017)
6. Hosang, J., Benenson, R., Schiele, B.: A convnet for non-maximum suppression. In: German Conference on Pattern Recognition. pp. 192–204. Springer (2016)
7. Hosang, J., Benenson, R., Schiele, B.: Learning non-maximum suppression. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4507–4515 (2017)
8. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017)
9. Hu, H., Gu, J., Zhang, Z., Dai, J., Wei, Y.: Relation networks for object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3588–3597 (2018)
10. Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S., et al.: Speed/accuracy trade-offs for modern convolutional object detectors. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 7310–7311 (2017)

11. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
12. Li, Q., Han, Z., Wu, X.M.: Deeper insights into graph convolutional networks for semi-supervised learning. In: Thirty-Second AAAI conference on artificial intelligence (2018)
13. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: Proceedings of the IEEE international conference on computer vision. pp. 2980–2988 (2017)
14. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: European conference on computer vision. pp. 740–755. Springer (2014)
15. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: Ssd: Single shot multibox detector. In: European conference on computer vision. pp. 21–37. Springer (2016)
16. Qi, L., Liu, S., Shi, J., Jia, J.: Sequential context encoding for duplicate removal. Advances in Neural Information Processing Systems **31** (2018)
17. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 779–788 (2016)
18. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. Advances in neural information processing systems **28** (2015)
19. Roh, B., Shin, J., Shin, W., Kim, S.: Sparse detr: Efficient end-to-end object detection with learnable sparsity. arXiv preprint arXiv:2111.14330 (2021)
20. Stewart, R., Andriluka, M., Ng, A.Y.: End-to-end people detection in crowded scenes. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2325–2333 (2016)
21. Sun, P., Jiang, Y., Xie, E., Shao, W., Yuan, Z., Wang, C., Luo, P.: What makes for end-to-end object detection? In: Proceedings of the 38th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 139, pp. 9934–9944. PMLR (2021)
22. Tan, M., Pang, R., Le, Q.V.: Efficientdet: Scalable and efficient object detection. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 10781–10790 (2020)
23. Tan, Z., Nie, X., Qian, Q., Li, N., Li, H.: Learning to rank proposals for object detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 8273–8281 (2019)
24. Tian, Z., Shen, C., Chen, H., He, T.: FCOS: Fully convolutional one-stage object detection. In: Proc. Int. Conf. Computer Vision (ICCV) (2019)
25. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
26. Zhou, Q., Yu, C., Shen, C., Wang, Z., Li, H.: Object detection made simpler by eliminating heuristic nms. arXiv preprint arXiv:2101.11782 (2021)
27. Zhou, X., Wang, D., Krähenbühl, P.: Objects as points. arXiv preprint arXiv:1904.07850 (2019)
28. Zhu, X., Su, W., Lu, L., Li, B., Wang, X., Dai, J.: Deformable detr: Deformable transformers for end-to-end object detection. arXiv preprint arXiv:2010.04159 (2020)