Sparse MDOD: Training End-to-End Multi-Object Detector without Bipartite Matching

Jaeyoung Yoo\textsuperscript{1,*} Hojun Lee\textsuperscript{2,*} Seunghyeon Seo\textsuperscript{2} Inseop Chung\textsuperscript{2} Nojun Kwak\textsuperscript{2}

\textsuperscript{1}NAVER WEBTOON AI \textsuperscript{2}Seoul National University
yoojy31@webtoonscorp.com \{hojun815|zzzishh|jis3613|nojunk\}@snu.ac.kr

\textbf{Abstract.} Recent end-to-end multi-object detectors simplify the inference pipeline by removing the hand-crafted process such as the duplicate bounding box removal using non-maximum suppression (NMS). However, in the training, they require bipartite matching to calculate the loss from the output of the detector. Contrary to the directivity, which is at the heart of end-to-end learning, the bipartite matching makes the training of the end-to-end detector complex, heuristic, and reliant. In this paper, we propose a method to train an end-to-end multi-object detector without bipartite matching. To this end, we approach end-to-end multi-object detection as a density estimation problem using a mixture model. Our proposed detector, called Sparse Mixture Density Object Detector (Sparse MDOD), estimates the distribution of bounding boxes using a mixture model. Sparse MDOD is trained by minimizing the negative log-likelihood and our proposed regularization term, maximum component maximization (MCM) loss that prevents duplicated predictions. During training, no additional procedure such as bipartite matching is needed, and the loss is directly computed from the network outputs. Moreover, our Sparse MDOD outperforms the existing detectors on MS-COCO, a renowned multi-object detection benchmark. Code will be available.

\textbf{Keywords:} End-to-End Multi-Object Detection, Bipartite Matching, Density Estimation, Mixture Model

1 Introduction

Training of multi-object detectors based on deep neural networks relies on a matching process between the predicted bounding boxes and the ground truth bounding boxes \cite{25,20}. This matching process not only needs several hyper-parameters but also requires a sophisticated design for the matching algorithm, and matching criterion. It highly affects the detection performance and makes the training of the multi-object detector complex, heuristic, and reliant \cite{31,30}.

Meanwhile, the conventional methods \cite{25,31} are trained by a many-to-one matching process that assigns a ground truth to many predictions. Their detector

\textsuperscript{*}J. Yoo and H. Lee equally contributed to this work.
networks output the various predicted bounding boxes in a dense prediction scheme. Then, they need to remove the duplicated bounding boxes using Non-Maximum Suppression (NMS) to obtain final predictions (left of Figure 1).

Recent end-to-end detection methods, following a sparse prediction scheme, address this duplication problem by searching for unique assignment between the predictions and the ground truth via bipartite matching during the training process [2][28]. It enables efficiently eliminating the duplicated bounding box removal step in the inference pipeline.

However, due to bipartite matching, the end-to-end methods require an additional optimization algorithm. Since there are many possible pairs between ground truths and predictions, the end-to-end methods need to find an optimal solution among them. Most end-to-end detectors utilize Hungarian method to find an optimal bipartite matching [2]. Hungarian method [15] is an optimization algorithm that handles assignment problems, and is known to have a time complexity of $O(N^3)$ for $N$ ground truths and predictions [14]. This makes the training of the end-to-end detector complex and not straightforward.

Also, in the bipartite matching, a heuristic loss function is used as a matching criterion between ground truth and prediction. The conventional methods generally utilize Intersection over Union (IoU) as a matching criterion. However, in DETR [2], which is the representative end-to-end method, the matching criterion $L_{\text{criterion}}$ is manually calculated as the weighted sum of cross-entropy (CE) loss, L1 loss, and GIoU loss $[26]$ with empirically found hyper-parameters as follows:

$$L_{\text{criterion}} = w_1 \cdot L_{\text{CE}} + w_2 \cdot L_{\text{L1}} + w_3 \cdot L_{\text{GIoU}}$$  \hspace{1cm} (1)$$

where $w_1$, $w_2$ and $w_3$ are balancing hyper-parameters. In Deformable DETR [32], another representative end-to-end method, CE loss was changed to focal loss [18] with more hyper-parameters such as focal coefficient and ratio of positive and negative. Furthermore, $w_1$ was also changed from 1 to 2. The hyper-parameters and the combination of the objective function used as the matching criterion are designed empirically. In other words, the necessity of bipartite matching makes the training of the end-to-end detector heuristic.
A previous study simplified the matching process while training a multi-object detector. MDOD\cite{30} eliminated the matching process for the localization loss through density estimation based on the mixture model. However, in order to learn the classification, MDOD still relies on a matching process called RoI sampling. In addition, since it is a conventional method that does not handle duplicate boxes in training, it relies on post-processing such as NMS for inference. Therefore, it cannot be applied as a recent end-to-end method.

Our goal is to reduce the heuristics and simplify the training process of end-to-end multi-object detection. To this end, we need to eliminate or replace bipartite matching with a more straightforward way and prevent the problem of duplicate predictions.

In this paper, we propose a training method for a new end-to-end multi-object detector, called Sparse Mixture Density Object Detector (Sparse MDOD), by utilizing a mixture model-based detector that does not depend on bipartite matching. First, we remove the matching process for learning information about bounding boxes by approaching localization and classification tasks into a density estimation of bounding boxes. In our method, a simple negative log-likelihood (NLL) of a mixture model is used as the objective for density estimation. Second, to achieve non-overlapping predictions, we propose the maximum component maximization (MCM) loss, the regularization term for the mixture model. The MCM loss only requires a simple max operation as matching for a ground truth. Thanks to its simplicity, the learning process for deduplication can be defined with only one loss term. The NLL and MCM loss are directly calculated from the network outputs without any separate process such as bipartite matching.

Moreover, we evaluate Sparse MDOD on the representative multi-object detection datasets MS-COCO\cite{19}, and it outperforms the baseline Sparse R-CNN\cite{28} as well as other multi-object detectors. Sparse MDOD, an end-to-end detector, achieves SOTA detection performance without using bipartite matching.

## 2 Related Works

Most modern deep-learning-based object detectors require post-processing to remove redundant predictions (e.g. NMS) from dense candidates to estimate final bounding boxes\cite{24,20,25}. Instead of depending on the manually designed post-processing, a line of recent works\cite{12,2,28} has proposed end-to-end object detection methods which output final bounding boxes directly without any post-processing in both the training and inference phase.

Recently, end-to-end methods\cite{12,2,32} that do not use NMS-based post-processing have been proposed. DETR\cite{2} proposes the training process for end-to-end detectors using the Hungarian algorithm\cite{15}, which yields an optimal bipartite matching between $N \times K$ samples. This training process has become the standard for end-to-end detectors. Among them, Sparse R-CNN\cite{28} is one of the end-to-end object detection methods in which a fixed set of learned proposal boxes and features are used. Sparse R-CNN argues that it has a simpler frame-
work than other end-to-end detectors \cite{2,32}. Likewise, it uses bipartite matching based on the Hungarian algorithm for training.

However, the Hungarian algorithm makes the training of end-to-end detectors heuristic and complex in that matching criterion between every pair of $N$ ground truths and $K$ predictions should be calculated. The calculation of this matching criterion in \cite{1} for all the $N \times K$ pairs is burdensome and incurs non-negligible computational overhead.

Another line of research has focused on removing the heuristics of the ground truth assignment process. Among them, Mixture Density Object Detector (MDOD) \cite{30} reformulated the multi-object detection task as a density estimation problem of bounding box distributions with a mixture model. This enabled MDOD to perform regression without an explicit matching process with ground truth bounding boxes. However, MDOD still requires the matching process for training a classification task. Furthermore, it is not an end-to-end method and cannot replace the training process based on bipartite matching.

Through our proposed Sparse MDOD and the MCM loss, we extend the density-estimation-based multi-object detector from dense prediction to an end-to-end method that does not need the deduplication process for the predictions. In addition, our Sparse MDOD is trained as an end-to-end detector by directly calculating the loss from the network outputs, unlike other end-to-end methods that rely on an additional process such as the Hungarian algorithm for bipartite matching. Our work makes the training process of the end-to-end multi-object detector much more straightforward.

3 Sparse Mixture Density Object Detector

3.1 Mixture model

For the multiple ground truths $g = \{g_1, ..., g_N\}$ on an image $X$, each ground truth $g_i$ contains the coordinates of an object’s location $b_i = \{b_{i,l}, b_{i,t}, b_{i,r}, b_{i,b}\}$ (left, top, right, and bottom) and a one-hot class information $c_i$. Our proposed Sparse MDOD conditionally estimates the distribution of the $g$ for an image $X$ using a mixture model.

We follow the design of the mixture model used in MDOD \cite{30}, which is the previous study dealing with the multi-object detection task as a mixture-model-based density estimation problem. The mixture model consists of two types of probability distribution: Cauchy (continuous) and categorical (discrete) distributions. The Cauchy distribution is a continuous probability distribution that has a shape similar to the Gaussian distribution. However, it has heavier tails than the Gaussian, and is known to be less likely to incur underflow problems due to floating-point precision \cite{30}. We use the 4-dimensional Cauchy to represent the distribution of object’s location coordinates. Also, a categorical distribution is used to estimate the object’s class probabilities for the one-hot representation.
Fig. 2: 3-stage example of Sparse MDOD architecture. Sparse MDOD has multi-stage structure, and Sparse MDOD head predicts the mixture model’s parameters $M^s$ from the proposal boxes $\bar{b}^s_k$ and an input image $X$.

The probability density function of our mixture model is defined as follows:

$$p(g_i|X) = \sum_k^K \pi_k F(b_i; \mu_k, \gamma_k) P(c_i; p_k)$$

(2)

Here, the $k$ is the index for the $K$ mixture components and the corresponding mixing coefficient is denoted by $\pi_k$. $F$ and $P$ denote the probability density function of the Cauchy and the probability mass function of the categorical distribution, respectively. The $\mu_k = \{\mu_{k,l}, \mu_{k,t}, \mu_{k,r}, \mu_{k,b}\}$ and $\gamma_k = \{\gamma_{k,l}, \gamma_{k,t}, \gamma_{k,r}, \gamma_{k,b}\}$ are the parameters of a Cauchy distribution. The $p_k = \{p_1, ..., p_C\}$ is the class probability of a categorical distribution. Here, the $C$ is the number of possible classes for an object excluding background class. To avoid over-complicating the mixture model, each element of $b_i$ is assumed to be independent of others. Thus, the probability density function of the Cauchy is factorized as follows:

$$F(b_i; \mu_k, \gamma_k) = \prod_{d \in D} F(b_{i,d}; \mu_{k,d}, \gamma_{k,d}), \quad D = \{l, t, r, b\}.$$ 

(3)

Here, $d$ is an element of the set of bounding box coordinates $D$.

### 3.2 Architecture

For our Sparse MDOD, we adopt the overall architecture of Sparse R-CNN [28] and its network characteristics such as learnable proposal box, dynamic head, and multi-stage structure.

Figure 2 shows the overview of our Sparse MDOD network when the 3-stage structure is used. First, the backbone network outputs the feature map from the input image $X$. In the first stage, the RoI features $h^1 = \{h_{1,1}^1, ..., h_{1,K}^1\}$ is obtained through RoI align process from the predefined learnable proposal boxes $\bar{b}^1 = \{\bar{b}_{1,1}^1, ..., \bar{b}_{1,K}^1\}$ and the feature map. And then, Sparse MDOD head predicts $M^1 = \{\pi^1, \mu^1, \gamma^1, p^1, o^1\}$, the parameters of the mixture model and the foreground probability, from the $h^1$. Here, the number of mixture components $K$
Fig. 3: Structure of Sparse MDOD head. Sparse MDOD head predicts the parameters of the mixture model ($\pi^s_k$, $\mu^s_k$, $\gamma^s_k$, and $p^s_k$) and the foreground probability ($o^s_k$) from a proposal box $\bar{b}^s_k$ and a RoI feature $h^s_k$.

equals the number of proposal boxes. In the s-th stage ($s \geq 2$), the process from RoI align to Sparse MDOD head is repeated. $\mu^{s-1} \in \mathbb{R}^4$ which is the predicted location parameters in the previous stage, is used as the proposal boxes $\bar{b}^s_k$ for the current stage. Following [28], we use the 6-stage structure.

The details of Sparse MDOD head is illustrated in Figure 3. Dynamic head outputs $\bar{\mu}^s_k$, $\bar{\gamma}^s_k$, $\bar{p}^s_k$ and $\bar{o}^s_k$ from $h^s_k$. The location parameter $\mu^s_k \in \mathbb{R}^4$ represents the coordinates of a mixture component and it is produced by adding $\bar{b}^s_k$ to $\bar{\mu}^s_k$. The positive scale parameter $\gamma^s_k \in \mathbb{R}^4$ is obtained by applying the softplus activation [7] that always converts $\bar{\gamma}^s_k$ into a positive value. The object class probability $p^s_k \in \mathbb{R}^C$ is calculated by applying softmax function to $\bar{p}^s_k$ along the class dimension.

Note that, the probability of whether it is an object or not is not computed through $p^s_k$ but computed using an alternative way we proposed to learn foreground probability. Returning to the nature of the probability distribution, we utilize the properties of the mixture model. In the mixture model, the probability of a mixture component is expressed as a mixture coefficient $\pi^s_k$. In other words, the mixture component that is likely to belong to a foreground area has a higher $\pi^s_k$ value. In this aspect, we assume that $\pi$ could be regarded as the scaled foreground probability such that $\sum_{k}^K \pi^s_k$ equals 1. From this assumption, we propose to express the mixture coefficient $\pi$ using the foreground probability $o$. As can be seen in Figure 3, the sigmoid activation outputs $o^s_k$ from $\bar{o}^s_k$. And then, $\pi^s_k$ is calculated by normalizing $o^s_k$ as $\pi^s_k = \frac{o^s_k}{\sum_{k'=1}^C o^s_{k'}}$.

3.3 Inference

In the inference, $\mu$ of the last stage is used as the coordinates of the predicted bounding boxes. The class probability $p$ of Sparse MDOD is the softmax output but, just the probability for the class of an object without background probability. Thus, we do not directly use $p$ as a confidence score of our prediction. Instead, the foreground probability $o$ learned though the mixing coefficient $\pi$ is used with $p$. The confidence score of an output prediction is calculated as follows: $p_c \times o$. In the same manner as other end-to-end multi-object detectors,
Sparse MDOD is trained to maximize the likelihood of \( g \) for the input image \( X \) through the mixed model. The loss function is simply defined as the negative log-likelihood (NLL) of the probability density function as follows:

\[
\mathcal{L}_{\text{NLL}} = -\log p(g_i | X) = -\log \sum_{k=1}^{K} \pi_k \mathcal{F}(b_i; \mu_k, \gamma_k)\mathcal{P}(c_i; p_k^s).
\]

Figure 4 shows 1-D example for the MCM loss (\( \mathcal{L}_{\text{MCM}} \)). By reducing the difference between \( \max(g_i | X) \) and \( p(g_i | X) \), it is trained to require only one mixture component to represent one \( g_i \). Hence it restrains from having multiple components to represent the same single \( g_i \).

Sparse MDOD also obtains final predictions without any duplicated bounding box removal process such as NMS.

### 3.4 Training

Sparse MDOD is trained to maximize the likelihood of \( g \) for the input image \( X \) through the mixed model. The loss function is simply defined as the negative log-likelihood (NLL) of the probability density function as follows:

\[
\mathcal{L}_{\text{NLL}} = -\log p(g_i | X) = -\log \sum_{k=1}^{K} \pi_k F(b_i; \mu_k, \gamma_k)\mathcal{P}(c_i; p_k^s).
\]

Sparse MDOD learns the coordinates of the bounding box and the probability of the object class as \( \mu \) and \( p \) by minimizing the NLL loss (\( \mathcal{L}_{\text{NLL}} \)). The foreground probability \( o \) is not directly used to calculate the NLL loss, but it is trained through \( \pi \) representing the probability of a mixture component (see Figure 2). Here, we need to consider that the NLL loss does not restrict the distributional redundancy between multiple mixture components for single ground truth. This problem could lead to duplication of the predicted bounding boxes, as well as dispersion of the probability for one object to several mixture components. Thus, we introduce the maximum component maximization (MCM) loss which is the regularization term to the density estimation of the mixture model:

\[
\mathcal{L}_{\text{MCM}} = -\log \frac{\max(g_i | X)}{p(g_i | X)} = -\log \frac{\max(g_i | X)}{\sum_{k=1}^{K} \pi_k \mathcal{F}(b_i; \mu_k, \gamma_k)\mathcal{P}(c_i; p_k^s)},
\]

\[
\max(g_i | X) = \max_{k \in \{1, ..., K\}} (\pi_k \mathcal{F}(b_i; \mu_k, \gamma_k)\mathcal{P}(c_i; p_k^s))
\]

Figure 4 shows 1-D example for the MCM loss (\( \mathcal{L}_{\text{MCM}} \)). Minimizing the MCM loss reduces the difference of likelihood between \( \max(g_i | X) \) and \( p(g_i | X) \). Through this, the mixture model is trained to maximize the probability of only one mixture component for one ground truth while reducing the probability of other...
adjacent components. The total loss function is defined as follows: $L = L_{NLL} + \beta \times L_{MCM}$, where $\beta$ is used to adjust the balance between the NLL and the MCM loss. The total loss ($L$) is computed for all stages of Sparse MDOD, then summed together and back-propagated. To calculate the total loss, we do not need any additional process such as bipartite matching.

4 Experiments

4.1 Experimental Details

Dataset. We evaluate Sparse MDOD on the standard object detection dataset, MS COCO [19], consisting of 80 categories. Following the common practice, we split the dataset into 118K images for training set, 5K images for validation set, and 20K images for test-dev set. We adopt the standard COCO AP (Average Precision) and AR (Average Recall) at most 100 top-scoring detections per image as the evaluation metrics. We report analysis results and comparison with a baseline on the COCO validation set and compare with other methods on the COCO test-dev set and COCO validation set.

Training. As mentioned in Section 3 and 3.1, we model bounding box coordinates as Cauchy distributions and class probability as categorical distributions following MDOD [30]. Unless specified, we adopt the hyper-parameters following Sparse R-CNN with 300 proposals as a default setting. Extensive experiments are conducted with different backbone networks, including ResNet50, ResNet101 [11] and Swin Transformer-Tiny [21] with Feature Pyramid Network (FPN) [17], which are pretrained on ImageNet-1K [6]. The identical data augmentations as used in Deformable DETR [32] are used for multi-scale training, where the input image size is $480 \sim 800$ with random crop and random horizontal flip. The parameter $\beta$ for balancing the losses $L_{NLL}$ and $L_{MCM}$ is set to 0.5. The batch size is 16, and synchronized batch normalization [23] is applied for consistent learning behavior regardless of the number of GPUs. We use AdamW [22] optimizer with a weight decay of 5e-5 and a gradient clipping with an L2 norm of 1.0. We adopt the training schedule of 36 epochs with an initial learning rate of 5e-5, divided by a factor of 10 at the 27th and 33rd epoch, respectively.

Inference. In the inference phase, we select top-100 bounding boxes among the output of the last Sparse MDOD head according to their confidence scores without any further post-processing, such as NMS.

4.2 Comparison with Baseline (Sparse R-CNN)

Table 1 presents a comparison between Sparse R-CNN and our Sparse MDOD with different backbone networks on COCO validation set. We achieve AP improvement over Sparse R-CNN while maintaining the FPS on a similar level. There exists a slight gap ($\leq 0.1$ FPS) in inference speed due to implementation details. The analysis of performance improvement is as follows.
Table 1: Comparison with Sparse R-CNN (S-RCNN) [28]. FPS is measured as a network inference time excluding data loading on a single NVIDIA TITAN RTX using MMDet [3] with batch size 1.

| Method  | Backbone        | AP  | AP$_{50}$ | AP$_{75}$ | AP$_S$ | AP$_M$ | AP$_L$ | FPS |
|---------|-----------------|-----|-----------|-----------|--------|--------|--------|-----|
| S-RCNN  | ResNet50 FPN    | 45.0| 64.1      | 49.0      | 27.8   | 47.6   | 59.7   | 22.7|
| Ours    |                 | 47.0| 64.8      | 51.6      | 30.5   | 50.4   | 61.1   | 22.8|
| S-RCNN  | ResNet101 FPN   | 46.4| 65.6      | 50.7      | 28.6   | 49.4   | 61.3   | 17.3|
| Ours    |                 | 48.0| 65.7      | 52.6      | 30.4   | 51.4   | 63.5   | 17.3|
| S-RCNN  | Swin-T FPN      | 47.9| 67.3      | 52.3      | -      | -      | -      | 16.4|
| Ours    |                 | 49.9| 68.1      | 55.1      | 32.1   | 53.6   | 64.6   | 16.5|

Table 2: From baseline to Sparse MDOD. AR is the maximum recall given the maximum number of detections=100 per image.

| Method                          | AP  | AP$_{50}$ | AP$_{75}$ | AP$_S$ | AP$_M$ | AP$_L$ | AR  |
|--------------------------------|-----|-----------|-----------|--------|--------|--------|-----|
| Baseline (Sparse R-CNN)         | 45.0| 64.1      | 49.0      | 27.8   | 47.6   | 59.7   | -   |
| + Synchronized BatchNorm [23]   | 45.2| 64.4      | 49.4      | 27.8   | 47.8   | 60.1   | 63.6|
| + Sparse MDOD and $\mathcal{L}_{NLL}$ | 35.6| 49.3      | 38.3      | 23.9   | 37.0   | 47.9   | 65.1|
| + $\mathcal{L}_{MCM}$           | 47.0| 64.8      | 51.6      | 30.5   | 50.4   | 64.1   | 65.3|

From baseline to Sparse MDOD. Table 2 shows performance changes from the baseline to ours. For fair comparison, we changed the Batchnorm [13] of the baseline to the synchronized Batchnorm [23] and there was an AP improvement of 0.2%p. With replacing the output format and loss function of the baseline with MDOD’s and a negative log-likelihood (NLL) loss, respectively, the performance drops from 45.2% AP to 35.6% AP. However, we achieve 47.0% AP, which improves 1.8%p AP over the baseline, by adding only the MCM loss. It shows that the MCM loss is the key to the performance gain. Interestingly, even without the MCM loss, the Average Recall (AR) is similar to that with the MCM loss (65.1% and 65.3%). We conjecture that many false positives exist under the setting without the MCM loss. Since it might result from a large number of duplicates, we conducted the related experiment. First, we demonstrate that the MCM loss contributes significantly to eliminating duplicates through qualitative results. A series of experiments with regard to the MCM loss and duplicates removal are elaborated in Section 4.3.

Visualization. Figure 5 is a visualization of Table 2. From baseline to Sparse MDOD. As shown in (b) and (c), the inference results with confidence scores of 0.5 or higher are quite similar. However, with the confidence scores below 0.5, the baseline results are rather noisy. Although (b) seems to show comparable results to (c), the confidence scores of (b) tend to be lower than those of (c) and a considerable number of overlapping boxes exist. For example, from the top in the second column, there are 2, 6 and 3 overlapping boxes with low confidence scores.
scores of \{0.42, 0.38\}, \{0.26, 0.17, 0.16, 0.15, 0.12, 0.12\} and \{0.40, 0.40, 0.13\}, respectively. Within the framework of a mixture model density estimation, the likelihood of (b) might be similar to (c). It will be further discussed in Section 4.3.

### 4.3 Analysis of the MCM loss and deduplication

**Validation loss.** Table 3 shows the average validation loss of two models, which are trained with and without the MCM loss. This experiment demonstrates that the NLL and MCM loss contribute to the density estimation and deduplication, respectively.

The model without the MCM loss performs density estimation comparably well in terms of likelihood in that the NLL losses are similar in both cases. In other words, a lack of density estimation capability is not the root cause of the sharp drop of performance (AP 47.0% → 35.6%). However, the two models show opposite tendencies in terms of \(\exp(-L_{MCM}) = \max(g_i|X)/p(g_i|X)\), which means the ratio of the maximum mixture component to likelihood. In the case of the model with the MCM loss, the ratio tends to increase as it goes through the Sparse MDOD stages. On the other hand, the model without the MCM loss has lower \(\exp(-L_{MCM})\) for each stage than those of the model with the MCM loss. That is, the model without the MCM loss performs density estimation resulting in several overlapping predictions with low confidence in the final stage. As mentioned in Sec. 4.2, this result is consistent with that of Table 2 in that the
Table 3: Validation loss, the NLL loss and the MCM loss. \( \exp(-L_{MCM}) \) is \( \max(g_i|X)/p(g_i|X) \), which means the ratio of maximum mixture component to likelihood for each instance.

| stage | \( L_{NLL} \) w/ \( L_{MCM} \) (AP 47.0) | \( L_{NLL} \) w/o \( L_{MCM} \) (AP 35.6) | \( \exp(-L_{MCM}) \) w/ \( L_{MCM} \) (AP 47.0) | \( \exp(-L_{MCM}) \) w/o \( L_{MCM} \) (AP 35.6) |
|-------|-----------------|-----------------|-----------------|-----------------|
| 1     | 17.904          | 17.607          | 0.983           | 0.804           | 0.759           |
| 2     | 16.185          | 15.961          | 0.986           | 0.794           | 0.685           |
| 3     | 15.715          | 15.576          | 0.991           | 0.828           | 0.663           |
| 4     | 15.528          | 15.449          | 0.995           | 0.874           | 0.699           |
| 5     | 15.434          | 15.398          | 0.998           | 0.895           | 0.596           |
| 6     | 15.391          | 15.374          | 0.999           | 0.892           | 0.590           |

Fig. 6: NMS result. The IoU threshold=1.0 means that NMS is not performed.

Fig. 7: Top-k ratio for proposals. Top-k ratio denotes the percentage of remaining proposals after each stage.

AP is significantly different depending on the presence of the MCM loss, whereas the AR is similar as 65.1% and 65.3% respectively.

**NMS.** Figure 6 is the result of applying NMS post-processing to the baseline and ours, although both of them do not actually have the NMS post-processing at inference time. The MCM loss eliminates duplication as effectively as the baseline, showing little performance change even after applying NMS. The NMS with thresholds \( \in \{0.7, 0.8, 0.9\} \) achieves a slight gain of performance for both the baseline and ours by eliminating a few overlapping boxes still remaining. However, in the case of the model without the MCM loss, it achieves a significant AP improvement through the NMS, which implies that a lot of duplicates exist. It indicates that the MCM loss plays a key role for the deduplication without the NMS post-processing. As described earlier in Table 3, the model’s confidence for overlapping predictions is dispersed without the MCM loss. Therefore, even if the model without the MCM loss is post-processed with NMS, the performance cannot be superior to that of the model with the MCM loss.
Table 4: Stop-gradient of likelihood of the MCM loss.

| stop-gradient | Metric               | AP  | AP50 | AP75 |
|---------------|----------------------|-----|------|------|
|               | π Cauchy (F) categorical (P) |     |      |      |
| -             | -                    | 46.8| 64.7 | 51.3 |
| -             | stop                 | 47.0| 64.8 | 51.6 |
| -             | stop stop            | 46.7| 64.5 | 51.3 |

Table 5: The MCM loss weight $\beta$. ‘/’ separates AP/AP$_{50}$

| MCM weight | The number of proposals | 100  | 300  | 500  | 1000 |
|------------|-------------------------|------|------|------|------|
| 0.4        |                         | 43.9 | 61.8 | 46.8 | 64.2 | 47.1 | 64.5 | 47.5 | 64.7 |
| 0.5        |                         | 43.9 | 61.9 | 47.0 | 64.8 | 47.1 | 64.7 | 47.7 | 65.0 |
| 0.6        |                         | 43.9 | 61.8 | 46.8 | 64.8 | 47.3 | 65.2 | 47.7 | 65.2 |
| 0.7        |                         | 43.5 | 61.6 | 46.7 | 64.7 | 47.2 | 65.1 | 47.6 | 65.4 |

Stop-gradient of the MCM loss. Table 4 shows the results when the stop-gradient scheme is applied to the MCM loss where the likelihood (Eq. 2) is calculated with three elements: the mixing coefficient $\pi$, the Cauchy $F$ and the categorical distribution $P$. Note that there is no stop-gradient when calculating the NLL loss. It is the best when the stop-gradient is applied only to the $F$. The back-propagation toward the $F$ might make the box’s coordinates deviate from the optimal to suppress the likelihoods of the overlapping boxes unless the stop-gradient is applied to the $F$. Since it is an unintended phenomenon, applying stop-gradient toward the $F$ is reasonable.

4.4 Analysis of the number of Proposals

MCM loss weight. Table 5 is an experiment on the number of proposals and the weight of the MCM loss $\beta$. When the number of RoIs is 100 and 300, the AP is the highest at $\beta = 0.5$. And when it is 500 and 1000, the AP is the highest at $\beta = 0.6$. It is assumed that the MCM weight for deduplication had to be a little stronger because the higher the number of proposals (500, 1000), the more room for duplication. The MCM weight was fixed at 0.5 for simplification because the AP was not sensitively changed.

Top-k ratio for proposals. Figure 7 shows how the performance changes when the number of proposals is reduced at each stage. The ‘top-k ratio’ in the legend of the figure denotes the percentage of remaining proposals after every stage, and the number of remaining proposals in the final stage can be calculated by \# of proposals $\times$ (top-k ratio)$^5$. For example, if there are 2000 proposals in the 1st stage and the top-k ratio is 0.7, the number of final proposals is $2000 \times 0.7^5 \approx 336$. As a result of the experiment, there is a trade-off between performance and speed. Furthermore, the more the number of proposals, the less the performance dropped even with a low top-k ratio. We conjecture that it is due to the enough number of final boxes remaining.

4.5 Comparison with others on COCO test-dev

Figure 8 and Table 6 compare ours with other state-of-the-art methods on COCO test-dev. FPS is measured on a single NVIDIA TITAN RTX with batch size 1,
excluding the data loading time. Compared to Sparse R-CNN, ours have a significant AP improvement by +1.7%p and +1.8%p in ResNet50 and ResNet101, respectively. We compared ours with Query Inst [8] which is a model that adds segmentation mask head to Sparse R-CNN and learns bounding boxes and segmentation masks together. Although Query Inst learned segmentation mask together, our Sparse MDOD shows almost the same APs in both ResNet50 and ResNet101. Ours’ FPS is higher than Deformable DETR [32] using Transformer Encoder [29]. In order to compare performance at similar speeds, we applied RoI-2000 and Top-0.7 to ResNet50 and ResNet101. Then, Sparse MDOD shows not only higher APs than Deformable DETR (+1.2%p and +0.7%p) but also outperforms all the other methods in this table.

There is no publicly available model weight, so FPS is measured by randomly initialized weight.

When we checked with Deformable DETR-R50, there was a difference in FPS below 0.2.

### Table 6: COCO test-dev result. R50/R101 denote ResNet50/ResNet101. Deform/Dynam Encoder are based on Transformer Encoder [29]. # is the number of proposals or queries. † learned bounding boxes and segmentation masks together, and excluded mask inference time when measuring FPS.

| Method                        | Feature Extractor | AP 50 | AP 75 | AP S | AP M | AP L | FPS  |
|-------------------------------|-------------------|-------|-------|------|------|------|------|
| Query Inst-R50 #300 †         | FPN               | 46.9  | 66.0  | 51.5 | 28.4 | 49.0 | 59.3 | 22.7 |
| Query Inst-R101 #300 †        | FPN               | 48.1  | 67.3  | 52.8 | 28.7 | 50.3 | 61.0 | 17.3 |
| Deformable DETR-R50 #300 [32]  | DeformEncoder     | 46.9  | 66.4  | 50.8 | 27.7 | 49.7 | 59.9 | 13.5 |
| Deformable DETR-R101 #300 [32] | DeformEncoder     | 48.7  | 68.1  | 52.9 | 29.1 | 51.5 | 62.0 | 11.1 |
| Dynamic DETR-R50 #300 [32]    | DynamEncoder      | 47.2  | 65.9  | 51.1 | 28.6 | 49.3 | 59.1 | -    |
| RepPoints V2-R50 [3]          | FPN               | 44.4  | 63.5  | 47.7 | 26.6 | 47.0 | 54.6 | 16.2 |
| RepPoints V2-R101 [3]         | FPN               | 46.0  | 65.3  | 49.5 | 27.4 | 48.9 | 57.3 | 13.1 |
| ATSS-R101 [31]                | FPN               | 43.6  | 62.1  | 47.4 | 26.1 | 47.0 | 53.6 | 15.5 |
| TOOD-R101 [31]                | FPN               | 46.7  | 64.6  | 50.7 | 28.9 | 49.6 | 57.0 | 16.7 |
| GFocalV2-R50 [16]             | FPN               | 44.3  | 62.3  | 48.5 | 26.8 | 47.7 | 54.1 | 21.2 |
| GFocalV2-R101 [16]            | FPN               | 46.2  | 64.3  | 50.5 | 27.8 | 49.9 | 57.0 | 16.3 |
| Sparse R-CNN-R50 #300 [28]    | FPN               | 45.2  | 64.6  | 49.1 | 27.0 | 47.2 | 57.4 | 22.7 |
| Sparse R-CNN-R101 #300 [28]   | FPN               | 46.4  | 65.8  | 50.4 | 27.0 | 48.7 | 59.5 | 17.3 |
| Sparse MDOD-R50 #300           | FPN               | 46.9  | 65.0  | 51.5 | 28.1 | 49.4 | 59.3 | 22.8 |
| Sparse MDOD-R50 #2000 Top-0.7  | FPN               | 48.1  | 65.8  | 53.0 | 29.8 | 50.7 | 60.0 | 14.0 |
| Sparse MDOD-R101 #300          | FPN               | 48.2  | 66.2  | 52.9 | 28.7 | 51.2 | 61.4 | 17.3 |
| Sparse MDOD-R101 #2000 Top-0.7 | FPN               | 49.4  | 67.2  | 54.4 | 30.5 | 52.1 | 61.6 | 11.7 |
Table 7: COCO validation results with Swin Transformer-Tiny [21]. † learned bounding boxes and segmentation masks together, and excluded mask inference time when measuring FPS.

| Method                  | Feature Extractor | AP  | AP$_{50}$ | AP$_{75}$ | AP$_{S}$ | AP$_{M}$ | AP$_{L}$ | FPS  |
|-------------------------|-------------------|-----|-----------|-----------|----------|----------|----------|------|
| Mask R-CNN [10] †       | FPN               | 46.0| 68.1      | 50.3      | 31.2     | 49.2     | 60.1     | 16.1 |
| Cascade Mask R-CNN [1] †| FPN               | 50.4| 69.2      | 54.7      | 33.8     | 54.1     | 65.2     | 5.9  |
| DETR [2]                | Encoder           | 45.4| 66.2      | 48.1      | 22.9     | 49.5     | 65.9     | -    |
| Sparse DETR-10% [27]    | Sparse Encoder    | 48.2| 69.2      | 52.3      | 29.8     | 51.2     | 64.5     | 15.4 |
| Sparse DETR-50% [27]    | Sparse Encoder    | 49.3| 69.5      | 53.3      | 32.0     | 52.7     | 64.9     | 13.1 |
| ATSS [41]               | FPN               | 47.2| 66.5      | 51.3      | -        | -        | -        | -    |
| RepPoints v2 [3]        | FPN               | 50.0| 68.5      | 54.2      | -        | -        | -        | -    |
| Sparse R-CNN [28]       | FPN               | 47.9| 67.3      | 52.3      | -        | -        | -        | 16.4 |
| Sparse MDOD             | FPN               | 49.9| 68.1      | 55.1      | 32.1     | 53.6     | 64.6     | 16.5 |

4.6 Comparison on Swin Transformer

To show that Sparse MDOD is also effective when using other backbone networks, we apply Sparse MDOD to the Swin Transformer [21] which is the representative transformer-based network. Table 7 is the result of comparing our Sparse MDOD with other methods using Swin Transformer-Tiny backbone on COCO validation set. Compared to Sparse R-CNN, ours shows an AP improvement of +2.0%p at a similar FPS. Compared to the instance segmentation methods [10,1], there was a significant difference in performance and speed. The AP of Cascade Mask R-CNN, which was trained with segmentation masks and bounding boxes, is +0.5%p higher than ours. However, its FPS is about 35.8% of our level. We also compared with the DETR series [2,27]. Our AP is +4.5%p higher than that of DETR. In addition, our AP is +1.7%p and +0.6%p higher than the two versions (10% and 50%) of Sparse DETR [27], which are faster than DETR. Furthermore, our FPS is higher than those of the two versions. This result shows that our Sparse MDOD is effective not only in the ResNet backbone but also in the Swin Transformer-Tiny backbone.

5 Conclusion

Our Sparse MDOD, an end-to-end object detector, has a simple pipeline because there is neither heuristic post-processing for duplication removal such as NMS at inference time nor heuristic box matching process at training time. The proposed MCM loss induces the detector to be trained to predict only one box with a high confidence score without duplication in each instance. Although Sparse MDOD has the new pipeline and loss function that is not much deformed from the structure of Sparse R-CNN, the performance is comparable to or better than other state-of-the-art methods. Our Sparse MDOD has scalability because it has not changed the network structure. Not only can it be used with the latest backbone such as Swin Transformer, but it also has a large room to be applied to the improved network structure by changing only the output format and the loss function.
References

1. Cai, Z., Vasconcelos, N.: Cascade r-cnn: high quality object detection and instance segmentation. IEEE transactions on pattern analysis and machine intelligence 43(5), 1483–1498 (2019)
2. 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)
3. Chen, K., Wang, J., Pang, J., Cao, Y., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Xu, J., Zhang, Z., Cheng, D., Zhu, C., Cheng, T., Zhao, Q., Li, B., Lu, X., Zhu, R., Wu, Y., Dai, J., Wang, J., Shi, J., Ouyang, W., Loy, C.C., Lin, D.: MMDetection: Open mmlab detection toolbox and benchmark. arXiv preprint arXiv:1906.07155 (2019)
4. Chen, Y., Zhang, Z., Cao, Y., Wang, L., Lin, S., Hu, H.: Reppoints v2: Verification meets regression for object detection. Advances in Neural Information Processing Systems 33, 5621–5631 (2020)
5. Dai, X., Chen, Y., Yang, J., Zhang, P., Yuan, L., Zhang, L.: Dynamic detr: End-to-end object detection with dynamic attention. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 2988–2997 (2021)
6. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: A Large-Scale Hierarchical Image Database. In: CVPR (2009)
7. Dugas, C., Bengio, Y., Bélisle, F., Nadeau, C., Garcia, R.: Incorporating second-order functional knowledge for better option pricing. Advances in neural information processing systems 13 (2000)
8. Fang, Y., Yang, S., Wang, X., Li, Y., Fang, C., Shan, Y., Feng, B., Liu, W.: Instances as queries. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 6910–6919 (2021)
9. Feng, C., Zhong, Y., Gao, Y., Scott, M.R., Huang, W.: Tood: Task-aligned one-stage object detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 3510–3519 (2021)
10. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: Proceedings of the IEEE international conference on computer vision. pp. 2961–2969 (2017)
11. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. pp. 770–778 (2016)
12. 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)
13. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 (2015)
14. Jonker, R., Volgenant, A.: A shortest augmenting path algorithm for dense and sparse linear assignment problems. Computing 38(4), 325–340 (1987)
15. Kuhn, H.W.: The hungarian method for the assignment problem. Naval research logistics quarterly 2(1-2), 83–97 (1955)
16. Li, X., Wang, W., Hu, X., Li, J., Tang, J., Yang, J.: Generalized focal loss v2: Learning reliable localization quality estimation for dense object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11632–11641 (2021)
17. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2117–2125 (2017)
18. 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)
19. 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)
20. 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)
21. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 10012–10022 (2021)
22. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101 (2017)
23. Peng, C., Xiao, T., Li, Z., Jiang, Y., Zhang, X., Jia, K., Yu, G., Sun, J.: Megdet: A large mini-batch object detector. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition. pp. 6181–6189 (2018)
24. 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)
25. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: NIPS. pp. 91–99 (2015)
26. Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I., Savarese, S.: Generalized intersection over union: A metric and a loss for bounding box regression. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 658–666 (2019)
27. 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)
28. Sun, P., Zhang, R., Jiang, Y., Kong, T., Xu, C., Zhan, W., Tomizuka, M., Li, L., Yuan, Z., Wang, C., et al.: Sparse r-cnn: End-to-end object detection with learnable proposals. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 14454–14463 (2021)
29. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems 30 (2017)
30. Yoo, J., Lee, H., Chung, I., Seo, G., Kwak, N.: Training multi-object detector by estimating bounding box distribution for input image. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 3437–3446 (October 2021)
31. Zhang, S., Chi, C., Yao, Y., Lei, Z., Li, S.Z.: Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 9759–9768 (2020)
32. Zhu, X., Su, W., Lu, L., Li, B., Wang, X., Dai, J.: Deformable detr: Deformable transformers for end-to-end object detection. In: International Conference on Learning Representations (2020)