End-to-End Multi-Object Detection with a Regularized Mixture Model

Jaeyoung Yoo [*1] **Hojun Lee** [*2] **Seunghyeon Seo** [3] **Inseop Chung** [2] **Nojun Kwak** [2 3]

Abstract

Recent end-to-end multi-object detectors simplify the inference pipeline by removing hand-crafted processes such as non-maximum suppression (NMS). However, during training, they still heavily rely on heuristics and hand-crafted processes which deteriorate the reliability of the predicted confidence score. In this paper, we propose a novel framework to train an end-to-end multi-object detector consisting of only two terms: negative log-likelihood (NLL) and a regularization term. In doing so, the multi-object detection problem is treated as density estimation of the ground truth bounding boxes utilizing a regularized mixture density model. The proposed *end-to-end multi-object Detection with a Regularized Mixture Model* (D-RMM) is trained by minimizing the NLL with the proposed regularization term, maximum component maximization (MCM) loss, preventing duplicate predictions. Our method reduces the heuristics of the training process and improves the reliability of the predicted confidence score. Moreover, our D-RMM outperforms the previous end-to-end detectors on MS COCO dataset. Code is available at https://github.com/lhj815/D-RMM.

1. Introduction

"How can we train a detector to learn a variable number of ground truth bounding boxes for an input image without duplicate predictions?" This is a fundamental question in the training of end-to-end multi-object detectors (Carion et al., 2020). In multi-object detection, each training image has a different number of bounding box coordinates and the corresponding class labels. Thus, the network output is challenging to match one-to-one with the ground truth. Conventional methods (Ren et al., 2015; Liu et al., 2016; Redmon et al., 2016) train the detector by assigning a ground truth to many duplicate predictions, but they cannot directly obtain the final predictions without relying on non-maximum suppression (NMS) in the inference phase.

As an answer to the opening question, recent end-to-end multi-object detection methods address the training of detector by searching for unique assignments between the predictions and the ground truth via bipartite matching (Figure 1). The assigned positive prediction by bipartite matching learns the corresponding ground truth by a pre-designed objective function. On the other hand, the unassigned negative predictions do not care about ground truth information but are trained as backgrounds. Unlike conventional detectors, end-to-end methods can obtain final predictions from detector networks through bipartite matching-based training without relying on NMS in the inference phase.

However, the training of the current end-to-end multi-object detectors has several drawbacks as follows:

Immoderate heuristics. In the training process, the current end-to-end methods use heuristically designed objective functions and matching criteria between the ground truth and a prediction. For instance, DETR, the representative end-to-end method, uses the following combination of losses as the training objective and matching criterion:

$$\mathcal{L} = w_1 \cdot \mathcal{L}_{CE} + w_2 \cdot \mathcal{L}_{L1} + w_3 \cdot \mathcal{L}_{GIoU}, \qquad (1)$$

where, $\mathcal{L}_{CE}$, $\mathcal{L}_{L1}$, and $\mathcal{L}_{GIoU}$ are cross-entropy, L1 and GIoU (Rezatofighi et al., 2019) loss with its balancing hyperparameters (w_1, w_2, and w_3) respectively. Deformable DETR (Zhu et al., 2020) and Sparse R-CNN (Sun et al., 2021), other popular end-to-end methods, replace the cross-entropy loss with the focal loss (Lin et al., 2017b).

Hand-crafted assignment. Since there are many possible pairs between ground truths and predictions, the end-to-end methods need to find an optimal set of pairs among them. To solve this assignment problem, most end-to-end detectors utilize hand-crafted algorithm such as the Hungarian method (Kuhn, 1955) in the training pipeline to find a good bipartite matching.

Unreliable confidence. In the aspect of the nature of hu-

[*]Equal contribution [1]NAVER WEBTOON AI [2]Department of Intelligence and Information Science, Seoul National University [3]Interdisciplinary Program in Artificial Intelligence, Seoul National University. Correspondence to: Jaeyoung Yoo <yoojy31@webtoonscorp.com>, Hojun Lee <hojun815@snu.ac.kr>, Nojun Kwak <nojunk@snu.ac.kr>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

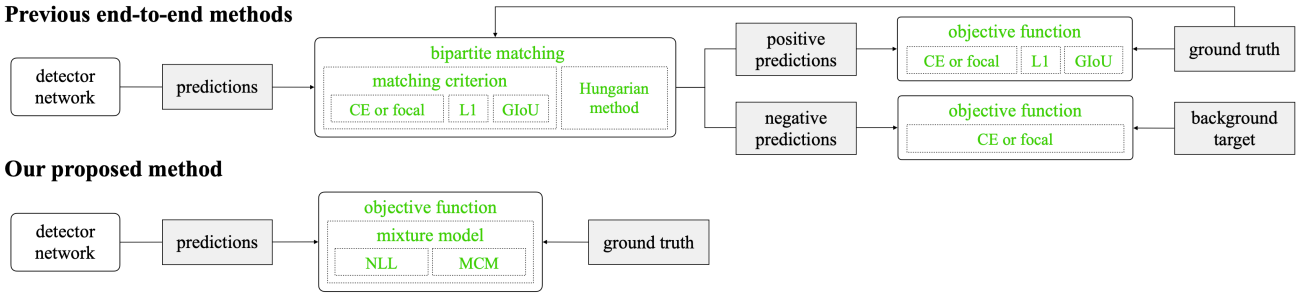

Figure 1. Training pipeline of end-to-end multi-object detectors. First row : previous end-to-end method such as (Carion et al., 2020; Sun et al., 2021). Second row: our proposed method. The green text denotes the manually designed training components.

man recognition, the confidence score of the prediction is regarded as an estimate of the accuracy. For the confidence score to be reliable, which means that the confidence score is to be used directly as an accuracy estimate, it should have a probabilistic meaning (Guo et al., 2017b). However, in the training process based on bipartite matching, the predictions are discretely classified as either positive or negative by a hand-crafted assignment process and a heuristic matching criterion rather than from a probabilistic point of view. In addition, the weight in the focal loss, which is a general loss term to learn class probability, is controlled by hyperparameters and does not provide a clear probabilistic basis. These heuristics and hand-designed training processes lead to gaps between predicted confidence scores and actual accuracy. Figure 2 shows the difference between the confidence score and the actual accuracy in the prediction of previous end-to-end detectors (DETR, Deformable DETR, Sparse R-CNN) and our probabilistic model (D-RMM).

In this study, we aim to overcome the aforementioned limitations and answer the opening question better. To this end, we propose a novel end-to-end multi-object detection framework, D-RMM, where we reformulate the end-to-end multi-object detection problem as a parametric density estimation problem. Our detector estimates the distribution of bounding boxes and object class using a mixture model. The proposed training loss function consists of the following two terms: the negative log-likelihood (NLL) and the maximum component maximization (MCM) loss. The NLL loss is a simple density estimation term of a mixture model. The MCM loss is the regularization term of the mixture model to achieve non-duplicate predictions. The NLL loss is calculated without any matching process, and the MCM loss only uses a simple maximum operation as matching. The contributions of our study are summarized as follows:

- We approach the end-to-end multi-object detection as a mixture model-based density estimation. To this end, we introduce an intuitive training objective function and the corresponding network architecture.
- We replace the heuristic objective function consisting of several losses with a simple negative log-likelihood (NLL) and the regularization (MCM) terms of a mixture model from a probabilistic point of view.
- Thanks to the simplicity of the NLL and MCM loss, they are directly calculated from the network outputs without any additional process, such as bipartite matching.
- As can be seen in Figure 2, the predictions of our method provide more reliable confidence scores.
- Our work outperforms the structural baselines (Sparse R-CNN, AdaMixer) and other state-of-the-art end-to-end multi-object detectors on MS COCO dataset.

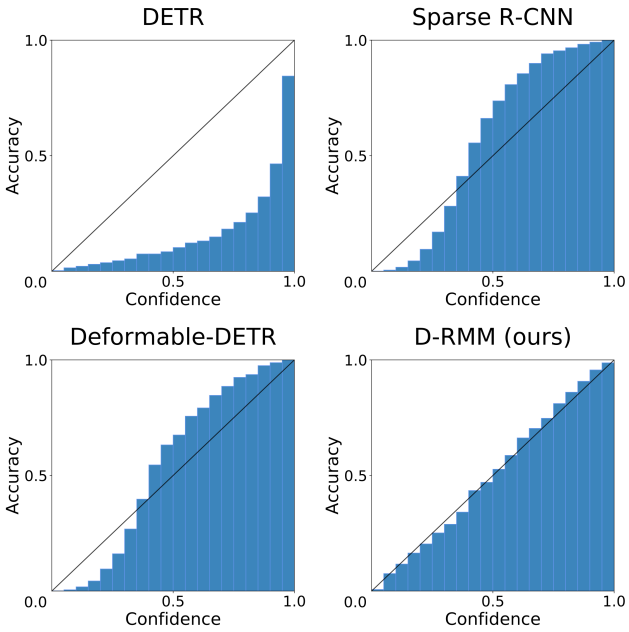

Figure 2. Confidence score *vs.* accuracy of various end-to-end detectors. Accuracy is measured for the predicted bounding boxes with the confidence score in the corresponding confidence bin.

2. Related Works

Most modern deep-learning-based object detectors require post-processing to remove redundant predictions (e.g. NMS) from dense candidates in estimating final bounding boxes (Redmon et al., 2016; Liu et al., 2016; Ren et al., 2015). Instead of depending on manually-designed post-processing, a line of recent works (Hu et al., 2018; Carion et al., 2020; Sun et al., 2021) 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 (Hu et al., 2018; Carion et al., 2020; Zhu et al., 2020) that do not use NMS-based post-processing have been proposed. DETR (Carion et al., 2020) proposes the training process for end-to-end detectors using the Hungarian algorithm (Kuhn, 1955), which yields an optimal bipartite matching between $N \times K$ samples. This training process has become a standard for end-to-end detectors. Among them, Sparse R-CNN (Sun et al., 2021) is one of the representative methods in which a fixed set of learned proposal boxes and features are used. Sparse R-CNN argues that it has a simple, but it still uses Hungarian-algorithm-based bipartite matching for training.

However, the training of end-to-end detectors based on bipartite matching relies on heuristic objective functions, matching criteria, and hand-assigned algorithms. It is also known that the efficiency and stability of training are impaired due to the limited supervision by bipartite matching. (Jia et al., 2022; Li et al., 2022)

Another line of research has focused on removing the heuristics of the ground truth assignment process. Among them, Mixture Density Object Detector (MDOD) (Yoo et al., 2021) 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 truths. 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.

In this paper, we extend the density-estimation-based multi-object detector to an end-to-end method that does not need the deduplication process for the predictions. In addition, our D-RMM 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 method for bipartite matching. Our work greatly simplifies the training process of the end-to-end multi-object detector by using a straightforward strategy.

3. The D-RMM Framework

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. The D-RMM network conditionally estimates the distribution of the g for an image X using a mixture model.

The mixture model consists of two types of probability distribution: Cauchy (continuous) for bounding box coordinates and categorical (discrete) for class estimation. 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 (Yoo et al., 2021). We use the 4-dimensional Cauchy to represent the distribution of the object's location coordinates. Also, a categorical distribution is used to estimate the object's class probabilities for the one-hot class representation. The probability density function of our mixture model is defined as follows:

$$p(g_i|X) = \sum_{k}^{K} \pi_k \mathcal{F}(b_i; \mu_k, \gamma_k)\mathcal{P}(c_i; p_k). \tag{2}$$

Here, the k is the index for the K mixture components and the corresponding mixing coefficient is denoted by π_k. $\mathcal{F}$ and $\mathcal{P}$ denote the probability density function of the Cauchy and the probability mass function of the categorical distribution respectively. The parameters $\mu_k = \{\mu_{k,l}, \mu_{k,t}, \mu_{k,r}, \mu_{k,b}\}$ and $\gamma_k = \{\gamma_{k,l}, \gamma_{k,t}, \gamma_{k,r}, \gamma_{k,b}\}$ are the location and scale parameters of a Cauchy distribution, while $p_k = \{p_1, ..., p_C\}$ is the class probability of a categorical distribution. Here, C is the number of possible classes for an object excluding the 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

$$\mathcal{F}(b_i; \mu_k, \gamma_k) = \prod_{j \in D} \mathcal{F}(b_{i,j}; \mu_{k,j}, \gamma_{k,j}), \quad D = \{l, t, r, b\}. \tag{3}$$

3.2. Architecture

For the implementation of our D-RMM, we adopt the overall architecture of Sparse R-CNN (Sun et al., 2021) and its network characteristics such as learnable proposal box, dynamic head, and iteration structure due to the intuitive structure and fast training compared to the DETR (Carion et al., 2020) and Deformable DETR (Zhu et al., 2020). Also, we applied D-RMM to AdaMixer (Gao et al., 2022) while maintaining its structural characteristics.

Figure 3 shows the overview of our D-RMM network when

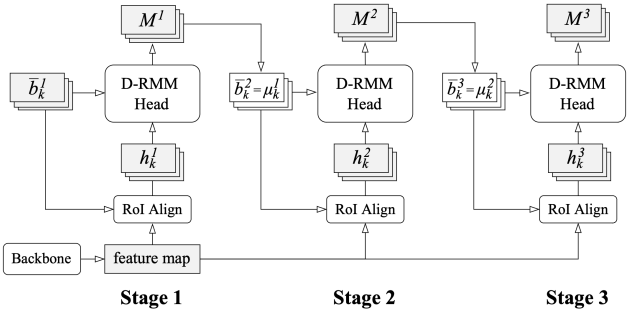

Figure 3. 3-stage example of D-RMM architecture. D-RMM head predicts the mixture model's parameters M^s from the proposal boxes $\bar{b}_k^s$ and an input image X.

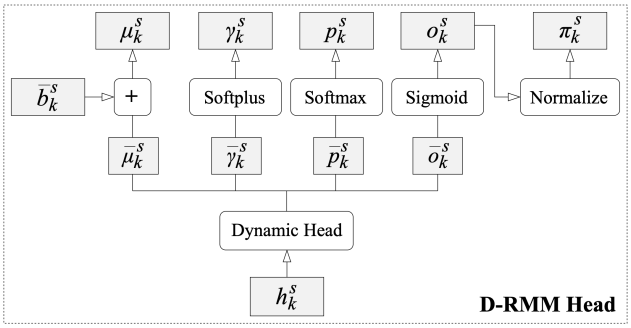

Figure 4. Structure of D-RMM head. D-RMM head predicts the parameters of the mixture model (π_k^s, μ_k^s γ_k^s and p_k^s) and the objectness score (o_k^s) from a proposal box $\bar{b}_k^s$ and a RoI feature h_k^s.

the 3-stage iteration structure is used. First, the backbone network outputs the feature map from the input image X. In the first stage, a set of RoI features $h^1 = \{h_1^1, ..., h_K^1\}$ is obtained through RoI align process from the predefined learnable proposal boxes $\bar{b}^1 = \{\bar{b}_1^1, ..., \bar{b}_K^1\}$ and the feature map. Then, D-RMM head predicts $M^1 = \{\pi^1, \mu^1, \gamma^1, p^1, o^1\}$, the parameters of the mixture model $(\pi^1, \mu^1, \gamma^1, p^1)$ and the objectness score (o^1), from h^1. Here, the number of mixture components K equals the number of proposal boxes. In the s-th stage ($s \geq 2$), the process from RoI align to D-RMM head is repeated. $\mu^{s-1} \in \mathbb{R}^4$ which is the predicted location vector in the previous stage, is used as the proposal boxes $\bar{b}^s$ for the current stage s. Following Sparse R-CNN (Sun et al., 2021), we adopt the 6-stage iteration structure.

The details of D-RMM head are illustrated in Figure 4. The dynamic head outputs $\bar{\mu}_k^s, \bar{\gamma}_k^s, \bar{p}_k^s$ and $\bar{o}_k^s$ from h_k^s. The location parameter $\mu_k^s \in \mathbb{R}^4$ represents the coordinates of a mixture component and is produced by adding $\bar{b}_k^s$ to $\bar{\mu}_k^s$. The positive scale parameter $\gamma_k^s \in \mathbb{R}^4$ is obtained by applying the softplus activation (Dugas et al., 2000) that always converts $\bar{\gamma}_k^s$ into a positive value. The object class probability $p_k^s \in \mathbb{R}^C$ is calculated by applying softmax function to $\bar{p}_k^s$ along the class dimension.

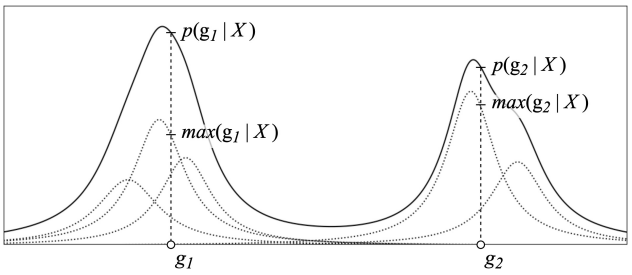

Figure 5. Illustration of 1-D example for the MCM loss ($\mathcal{L}_{MCM}$). By reducing the difference between $max(g_i|X)$ and $p(g_i|X)$, it is trained to require only one mixture component to represent one g_i. Hence it restrains from having multiple components to represent the same single g_i.

Note that the probability of whether it is an object or not is not computed through p_k^s but computed using an alternative way we propose to learn objectness score. 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 π_k^s. In other words, the mixture component that is likely to belong to an object area has a higher π_k^s value. In this aspect, we assume that π could be regarded as the scaled objectness score such that $\sum_k^K \pi_k^s$ equals 1. From this assumption, we propose to express the mixture coefficient π using the objectness score o. As shown in Figure 4, the sigmoid activation outputs o_k^s from $\bar{o}_k^s$. And then, π_k^s is calculated by normalizing o_k^s as $\pi_k^s = \frac{o_k^s}{\sum_{k'=1}^K o_{k'}^s}$.

3.3. Training

The D-RMM network is trained to maximize the likelihood of g for the input image X through the mixture model. The loss function is simply defined as the negative log-likelihood (NLL) of the probability density function as follows:

$$\mathcal{L}_{NLL} = -\log p(g_i|X) \tag{4}$$

$$= -\log \sum_k^K \pi_k^s \mathcal{F}(b_i; \mu_k^s, \gamma_k^s) \mathcal{P}(c_i; p_k^s) \tag{5}$$

The D-RMM network learns the coordinates of the bounding box and the probability of the object class as μ and p by minimizing the NLL loss ($\mathcal{L}_{NLL}$). The mixture coefficient π learns the probability of a mixture component $\mathcal{F}(b_i)\mathcal{P}(c_i)$ that represents the joint probability for both box coordinates and object class. The objectness score o is not directly used to calculate the NLL loss, but it is trained through π (see Figure 4).

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:

$$\mathcal{L}_{MCM} = -\log \frac{\max(g_i|X)}{p(g_i|X)} \tag{6}$$

$$= -\log \frac{\max(g_i|X)}{\sum_{k=1}^{K} \pi_k^s \mathcal{F}(b_i; \mu_k^s, \gamma_k^s)\mathcal{P}(c_i; p_k^s)}, \tag{7}$$

$$\max(g_i|X) = \max_{k \in \{1,\dots,K\}} (\pi_k^s \mathcal{F}(b_i; \mu_k^s, \gamma_k^s)\mathcal{P}(c_i; p_k^s)) \tag{8}$$

Figure 5 shows 1-D example for the MCM loss ($\mathcal{L}_{MCM}$). 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: $\mathcal{L} = \mathcal{L}_{NLL} + \beta \times \mathcal{L}_{MCM}$, where β balances between the NLL and the MCM loss. The total loss ($\mathcal{L}$) is computed for all stages of D-RMM and all ground truth boxes, then summed together and back-propagated. To calculate the total loss, we do not need any additional process such as bipartite matching.

3.4. Inference

In the inference, μ of the last stage is used as the coordinates of the predicted bounding boxes. The class probability $p_k \in \mathbb{R}^C$ of k-th mixture component 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 objectness score o learned though the mixing coefficient π is used with p. The confidence score of an k-th output prediction for class c is calculated as $p_{k,c} \times o_k$ where $p_{k,c}$ is the c-th element of p_k. In the same manner as other end-to-end multi-object detectors, D-RMM also obtains final predictions without any duplicate bounding box removal process such as NMS.

4. Experiments

4.1. Experimental details

Dataset. We evaluate D-RMM on MS COCO 2017 (Lin et al., 2014). Following the common practice, we split the dataset into 118K images for the training set, 5K for the validation set, and 20K for the 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 validation set and compare with other methods on the test-dev and validation set.

Training. As mentioned in Section 1 and 3.1, we model bounding box coordinates as Cauchy distributions and class probability as categorical distributions. We applied D-RMM to Sparse R-CNN (S-RCNN) and AdaMixer. For analysis, we adopt Sparse R-CNN architecture with 300 proposals. Unless specified, we followed the hyperparameters of each original paper. As backbones, ResNet50 (R50), ResNet101 (R101) (He et al., 2016) and Swin Transformer-Tiny (Swin-T) (Liu et al., 2021) with Feature Pyramid Network (FPN) (Lin et al., 2017a) are adopted, which are pretrained on ImageNet-1K (Deng et al., 2009). The parameter β for balancing the losses $\mathcal{L}_{NLL}$ and $\mathcal{L}_{MCM}$ is set to 0.5. β was found experimentally, and related experiments are in Appendix Section C.1. Synchronized batch normalization (Peng et al., 2018) is applied for consistent learning behavior regardless of the number of GPUs. More details like batch size, data augmentation, optimizer and training schedule for reproducibility are in Appendix A.

Inference. We select top-100 bounding boxes among the last head output according to their confidence scores without any further post-processing, such as NMS.

Table 1. Comparison with Sparse R-CNN (Sun et al., 2021). FPS is measured as a network inference time excluding data loading on a single NVIDIA TITAN RTX using MMDet (Chen et al., 2019) with batch size 1.

Method	Backbone	AP	AP_{50}	AP_{75}	FPS
S-RCNN	R50 FPN	45.0	64.1	49.0	22.7
D-RMM		47.0	64.8	51.6	22.8
S-RCNN	R101 FPN	46.4	65.6	50.7	17.3
D-RMM		48.0	65.7	52.6	17.3
S-RCNN	Swin-T FPN	47.4	67.1	52.0	16.4
D-RMM		49.9	68.1	55.1	16.5

Table 2. B denotes Bipartite Matching, M denotes Max. Both the matching cost and objective function of the 2nd row are the NLL loss. And in the case of the objective function, the NLL loss is computed only for matched predictions. In other words, unlike the 3rd-5th rows, the 2nd row is not a mixture model.

Method	REG	CLS	NLL	MCM	AP
S-RCNN	✓(B)	✓(B)			45.0
			✓(B)		46.3
			✓		35.6
			✓	✓(B)	46.9
D-RMM			✓	✓(M)	47.0

4.2. Comparison with Sparse R-CNN

Table 1 presents a comparison between Sparse R-CNN and D-RMM with different backbone networks on COCO validation set. We achieved significant AP improvement over Sparse R-CNN while maintaining the FPS on a similar level. There exists a slight gain ($\leq$ 0.1 FPS) in inference speed due to implementation details. The analysis of performance

(a) S-RCNN (45.0 / 63.5) (b) NLL (35.6 / 65.1)

(c) NLL+MCM (47.0 / 65.3) (d) NLL+WTA (45.8 / 65.0)

Figure 6. Qualitative result. Red/Green boxes indicate confidence 0.1-0.5 and 0.5-1.0, respectively. '/' separates AP/AR

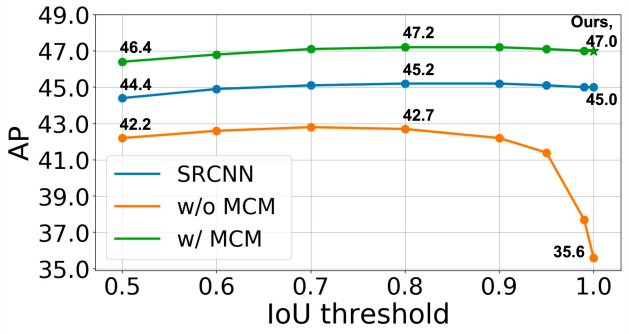

Figure 7. NMS result. The IoU threshold=1.0 means that NMS is not performed.

improvement is as follows.

Objective function. In Table 2, we compared the APs by varying the objective function and observed the following. First, in Sparse R-CNN, modeling the objective function and the matching cost jointly with the NLL loss is more effective in performance than a combination of semantically different regression and classification losses (45.0→46.3). Second, the performance of the NLL loss with a mixture model is relatively low because there is no means to remove duplicate bounding boxes (35.6). Third, the MCM loss to remove duplicates significantly improves the AP (35.6→46.9). Fourth, the AP is slightly increased even if the matching method for the MCM loss is changed to a simple MAX function (46.9→47.0). Also, D-RMM significantly outperforms Sparse R-CNN.

Visualization. Figure 6 (a), (b), (c) visualize the 1st, 3rd, and last rows of Table 2. As shown in (a) and (c), the inference results with confidence scores of 0.5 or higher are quite similar. However, with the confidence scores below 0.5, S-RCNN 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 many overlapping boxes exist. For example, in (b), there are 3-overlapping boxes with low confidence scores of $\{0.40, 0.40, 0.13\}$. Within the mixture model framework, the likelihood of (b) might be similar to (c). It will be further discussed in Section 4.3. More examples are in Appendix J.

Table 3. Validation loss. $exp(-\mathcal{L}_{MCM})$ is $\max(g_i|X)/p(g_i|X)$, which means the ratio of maximum mixture component to likelihood for each instance.

	$w/o\ \mathcal{L}_{MCM}$	$w/\ \mathcal{L}_{MCM}$
$\mathcal{L}_{NLL}$	15.374	15.391
$exp(-\mathcal{L}_{MCM})$	0.590	0.892

Table 4. Handicraft method for merging confidences by applying the winner-take-all strategy (WTA).

Method	AP	AR
(I) $\mathcal{L}_{NLL}$ with Mixture model	35.6	65.1
(II) (I)+NMS	42.7	65.0
(III) (I)+WTA	45.8	65.0
(IV) $\mathcal{L}_{NLL}$ only for MAX	45.7	63.7
(V) $\mathcal{L}_{NLL}$+$\mathcal{L}_{MCM}$	47.0	65.3

As shown in the figure, (b) has many duplicates. Interestingly, (b) and (c) has a large AP gap, but AR is similar. We conjectured the MCM loss contributed significantly to deduplication. Therefore, we conducted related experiments.

4.3. Analysis of the MCM loss and deduplication

NMS. Figure 7 is the result of applying NMS post-processing, although the models do not actually have the NMS post-processing at inference time. The MCM loss eliminates duplication as effectively as S-RCNN, 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 S-RCNN 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. This indicates that the MCM loss plays a key role for the deduplication without the NMS post-processing.

Table 3 reports the validation loss of models with or without L_{MCM}. Obviously, there was a large difference in the MCM loss, but the NLL loss was similar. In other words, without L_{MCM}, the performance is low because duplication cannot be avoided, and the failure to estimate the density is not the cause of the low performance. Detailed experimental results are in Appendix D.

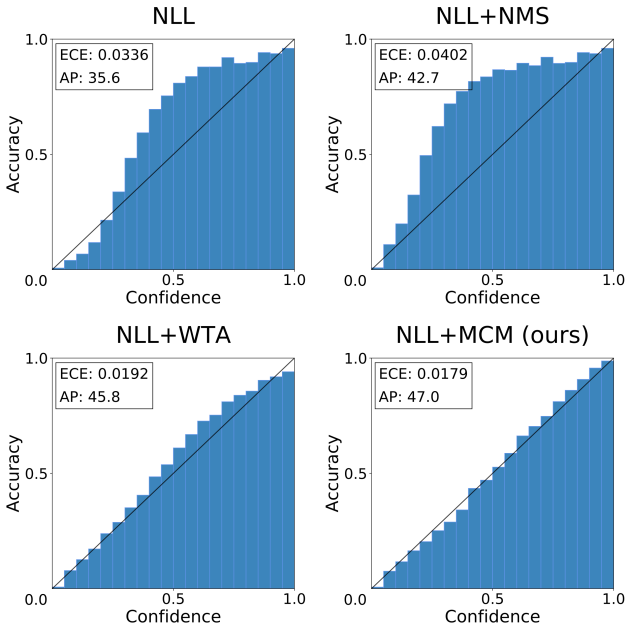

Figure 8. Confidence score and accuracy for model reliability with Expected Confidence Error (ECE) and AP.

Winner-Take-All method for the dispersed confidences. In Figure 7, NMS improved the AP of the model trained only with NLL from 35.6% to 42.7%. However, it still does not reach 47.0% of the model with the MCM loss. In order to reduce the AP gap, we applied a handicraft process called winner-take-all strategy (WTA), which is a way that the confidences of the boxes removed by NMS were added to those of the remaining boxes (Figure 6 (d)). In Table 4, we compared (III) and (IV) to see the effect of WTA, and there is an AP improvement from 42.7% to 45.8%. However, it was still below 47.0% AP of (V). Rather than learning only with the NLL loss and merging it with manual rules, it is more effective to learn with the MCM loss to predict without overlapping. Furthermore, the MCM loss is straightforward and efficient in that there is no need for additional post-processing in the inference phase.

4.4. Reliabilty and deduplication.

Figure 8 shows the relationship between deduplication method and confidence reliability through confidence score and Expected Calibration Error (ECE), which calculates the difference between the confidence score and accuracy of each bin (Guo et al., 2017a). ECE is defined as,

$$ECE = \sum_{m} \frac{|B_m|}{n} \left| acc(B_m) - conf(B_m) \right|, \qquad (9)$$

where B_m is m-th bin, $|B_m|$ is the m-th bin size, and n is the total number of predictions.

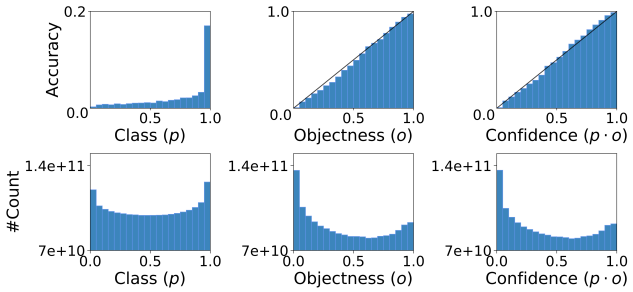

Figure 9. From left, illustrating histogram for class probability (p), objectness score (o), and confidence score ($p \cdot o$). Top: confidence histogram. Bottom: the number of predictions. The left and center histograms are plotted from the same bounding box predictions of the right.

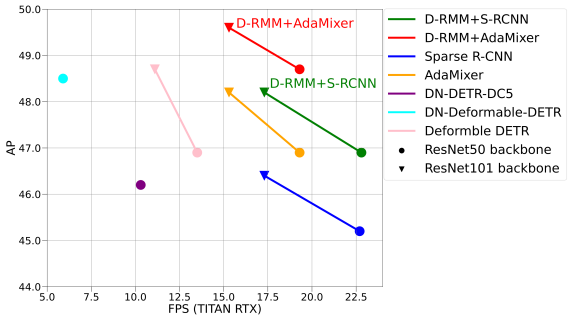

Figure 10. COCO test-dev results.

Although the NLL model is statistically trained, it is generally under-confident because the confidences are dispersed to multiple redundancies. Since the NMS removed duplicates, each bin's accuracy increased, but the ECE deteriorated. WTA, which was also influential in improving AP, was also effective in ECE. As the confidence of duplicates gathered together, the under-confidence effect was resolved similarly to ours.

Since D-RMM has an overall statistical training pipeline, including deduplication, confidence reliability is high. Besides, the model learned the mechanics of the WTA during the training process. Therefore, our model outperforms the WTA model for both AP and ECE and does not require a separate post-processing like WTA. From 'NLL' to 'NLL + MCM', AP increases while ECE decreases.

In addition, we further analyzed the reliability of our model in Figure 9. If only class probability excluding the objectness score is considered (the left of Figure 9), there is a lot of duplication and it tends to be over-confident. On the other hand, the objectness score showed a similar histogram to the confidence score (the center and right of Figure 9). In other words, objectness score plays a key role in model reliability.

Table 5. Comparision results with end-to-end Detectors on COCO validation. † uses 100 queries.

Method	Backbone	#epochs	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
DETR (Carion et al., 2020) †	R50	500	42.0	62.4	44.2	20.5	45.8	61.1
DETR (Carion et al., 2020) †	R101	500	43.5	63.8	46.4	21.9	48.0	61.8
Conditional-DETR (Meng et al., 2021)	R50	108	43.0	64.0	45.7	22.7	46.7	61.5
Conditional-DETR (Meng et al., 2021)	R101	108	44.5	65.6	47.5	23.6	48.4	63.6
Deformable DETR (Zhu et al., 2020)	R50	50	46.9	65.6	51.0	29.6	50.1	61.6
Anchor DETR (Wang et al., 2022)	R50	50	42.1	63.1	44.9	22.3	46.2	60.0
Anchor DETR (Wang et al., 2022)	R101	50	43.5	64.3	46.6	23.2	47.7	61.4
DAB-Deformable-DETR (Liu et al., 2022)	R50	50	46.9	66.0	50.8	30.1	50.4	62.5
DN-DETR (Li et al., 2022)	R50	50	44.1	64.4	46.7	22.9	48.0	63.4
DN-Deformable-DETR (Li et al., 2022)	R50	50	48.6	67.4	52.7	31.0	52.0	63.7
DN-DETR (Li et al., 2022)	R101	50	45.2	65.5	48.3	24.1	49.1	65.1
Sparse DETR (Roh et al., 2022)	R50	50	46.3	66.0	50.1	29.0	49.5	60.8
Sparse DETR (Roh et al., 2022)	Swin-Tiny	50	49.3	69.5	53.3	32.0	52.7	64.9
ViDT (Song et al., 2022)	Swin-Tiny	150	47.2	66.7	51.3	28.4	50.2	64.7
ViDT (Song et al., 2022)	Swin-Small	150	48.8	68.8	53.0	30.7	52.0	65.9
Sparse R-CNN (Sun et al., 2021)	R50	36	45.0	64.1	49.0	27.8	47.6	59.7
Sparse R-CNN (Sun et al., 2021)	R101	36	46.4	65.6	50.7	28.6	49.4	61.3
Sparse R-CNN (Sun et al., 2021)	Swin-Tiny	36	47.4	67.1	52.0	30.1	50.3	63.1
AdaMixer (Gao et al., 2022)	R50	36	47.0	66.0	51.1	30.1	50.2	61.8
AdaMixer (Gao et al., 2022)	R101	36	48.0	67.0	52.4	30.0	51.2	63.7
AdaMixer (Gao et al., 2022)	Swin-Tiny	36	48.9	68.5	53.5	31.5	52.0	64.2
AdaMixer (Gao et al., 2022)	Swin-Small	36	51.3	**71.2**	55.7	34.2	54.6	67.3
D-RMM + Sparse R-CNN	R50	36	47.0 (+2.0)	64.8	51.6	30.5	50.4	61.1
D-RMM + Sparse R-CNN	R101	36	48.0 (+1.6)	65.7	52.6	30.4	51.4	63.5
D-RMM + Sparse R-CNN	Swin-Tiny	36	49.9 (+2.5)	68.1	55.1	32.1	53.6	64.6
D-RMM + AdaMixer	R50	36	48.4 (+1.4)	66.3	52.8	31.0	52.1	64.0
D-RMM + AdaMixer	R101	36	49.2 (+1.2)	67.3	53.5	31.4	53.1	65.4
D-RMM + AdaMixer	Swin-Tiny	36	50.7 (+1.8)	69.0	55.5	33.5	54.4	66.3
D-RMM + AdaMixer	Swin-Small	36	**52.4 (+1.1)**	70.8	**57.5**	**34.9**	**56.6**	**68.5**

Table 6. COCO test-dev result. FPS is measured on a single NVIDIA TITAN RTX with batch size 1, excluding the data loading time. $*$ excluded the denoising task when measuring FPS.

Method	Backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	FPS
Deformable DETR (Zhu et al., 2020)	R50	46.9	66.4	50.8	27.7	49.7	59.9	13.5
Deformable DETR (Zhu et al., 2020)	R101	48.7	**68.1**	52.9	29.1	51.5	62.0	11.1
Dynamic DETR (Dai et al., 2021)	R50	47.2	65.9	51.1	28.6	49.3	59.1	-
DN-DETR-DC5 (Li et al., 2022) $*$	R50	46.2	66.4	49.6	24.5	49.3	62.7	10.3
DN-Deformbale-DETR (Li et al., 2022) $*$	R50	48.5	67.5	52.7	28.7	51.6	62.0	5.9
Sparse R-CNN (Sun et al., 2021)	R50	45.2	64.6	49.1	27.0	47.2	57.4	22.7
Sparse R-CNN (Sun et al., 2021)	R101	46.4	65.8	50.4	27.0	48.7	59.5	17.3
AdaMixer (Gao et al., 2022)	R50	46.9	66.1	51.1	28.3	49.1	60.5	19.3
AdaMixer (Gao et al., 2022)	R101	48.2	67.5	52.5	28.9	50.8	62.0	15.3
D-RMM + Sparse R-CNN	R50	46.9 (+1.7)	65.0	51.5	28.1	49.4	59.3	22.8
D-RMM + Sparse R-CNN	R101	48.2 (+1.8)	66.2	52.9	28.7	51.2	61.4	17.3
D-RMM + AdaMixer	R50	48.7 (+1.8)	66.8	53.1	29.0	51.6	62.1	19.3
D-RMM + AdaMixer	R101	**49.6 (+1.4)**	67.7	**54.1**	**29.8**	**52.5**	**64.0**	15.3

4.5. Comparison with others on COCO validation

Table 5 shows the comparison with previous end-to-end detectors. Note that D-RMM has 300 proposal boxes, and we only evaluated the top-100 boxes. Compared to Sparse R-CNN and AdaMixer, from which we borrowed the structure, all of ours have significantly improved performance. Furthermore, D-RMM is effective not only in the ResNet backbone but also in the Swin Transformer-Tiny backbone.

Compared with the DETR variants, ours generally has higher performance. DN-Deformable-DETR, which denoising technique applied to DAB-Deformable-DETR, is similar to ours in R50 (48.6 and 48.4). We compared them by considering the speed in the next section.

4.6. Comparison with others on COCO test-dev

Figure 10 and Table 6 compare ours with other state-of-the-art methods on COCO test-dev. Compared to Sparse R-CNN, ours have a significant AP improvement by +1.7%p in ResNet50 and +1.8%p in ResNet101. When we applied D-RMM to AdaMixer (Gao et al., 2022), the performance improved over AdaMixer and showed the state-of-the-art performance. In the figure, ours stand out from the others regarding speed and performance.

5. Conclusion

Our D-RMM, 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 D-RMM has the new pipeline and loss function that is not much deformed from the structure of Sparse R-CNN and AdaMixer, the performance is comparable to or better than other state-of-the-art methods. Furthermore, 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.

Acknowledgements

The researchers at Seoul National University were funded by the Korean Government through the NRF grants 2021R1A2C3006659 and 2022R1A5A7026673 as well as IITP grant 2021-0-01343.

References

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. End-to-end object detection with transformers. In *European conference on computer vision*, pp. 213–229. Springer, 2020.

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., and Lin, D. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019.

Chen, Q., Chen, X., Zeng, G., and Wang, J. Group detr: Fast training convergence with decoupled one-to-many label assignment. *arXiv preprint arXiv:2207.13085*, 2022.

Dai, X., Chen, Y., Yang, J., Zhang, P., Yuan, L., and 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.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009.

Dugas, C., Bengio, Y., Bélisle, F., Nadeau, C., and Garcia, R. Incorporating second-order functional knowledge for better option pricing. *Advances in neural information processing systems*, 13, 2000.

Gao, Z., Wang, L., Han, B., and Guo, S. Adamixer: A fast-converging query-based object detector. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5364–5373, 2022.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. On calibration of modern neural networks. In *International conference on machine learning*, pp. 1321–1330. PMLR, 2017a.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. On calibration of modern neural networks. In *International conference on machine learning*, pp. 1321–1330. PMLR, 2017b.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *CVPR*, pp. 770–778, 2016.

He, Y., Zhu, C., Wang, J., Savvides, M., and Zhang, X. Bounding box regression with uncertainty for accurate object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2888–2897, 2019.

Hu, H., Gu, J., Zhang, Z., Dai, J., and Wei, Y. Relation networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3588–3597, 2018.

Jia, D., Yuan, Y., He, H., Wu, X., Yu, H., Lin, W., Sun, L., Zhang, C., and Hu, H. Detrs with hybrid matching. *arXiv preprint arXiv:2207.13080*, 2022.

Kuhn, H. W. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

Li, F., Zhang, H., Liu, S., Guo, J., Ni, L. M., and Zhang, L. Dn-detr: Accelerate detr training by introducing query denoising. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13619–13627, 2022.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. Microsoft coco: Common objects in context. In *European conference on computer vision*, pp. 740–755. Springer, 2014.

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2117–2125, 2017a.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988, 2017b.

Liu, S., Li, F., Zhang, H., Yang, X., Qi, X., Su, H., Zhu, J., and Zhang, L. Dab-detr: Dynamic anchor boxes are better queries for detr. *arXiv preprint arXiv:2201.12329*, 2022.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. Ssd: Single shot multibox detector. In *European conference on computer vision*, pp. 21–37. Springer, 2016.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and 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.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Meng, D., Chen, X., Fan, Z., Zeng, G., Li, H., Yuan, Y., Sun, L., and Wang, J. Conditional detr for fast training convergence. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3651–3660, 2021.

Peng, C., Xiao, T., Li, Z., Jiang, Y., Zhang, X., Jia, K., Yu, G., and 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.

Redmon, J., Divvala, S., Girshick, R., and 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.

Ren, S., He, K., Girshick, R., and Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, pp. 91–99, 2015.

Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I., and 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.

Roh, B., Shin, J., Shin, W., and Kim, S. Sparse DETR: Efficient end-to-end object detection with learnable sparsity. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=RRGVCN8kjim.

Song, H., Sun, D., Chun, S., Jampani, V., Han, D., Heo, B., Kim, W., and Yang, M.-H. ViDT: An efficient and effective fully transformer-based object detector. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=w4cXZDDib1H.

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.

Wang, Y., Zhang, X., Yang, T., and Sun, J. Anchor detr: Query design for transformer-based detector. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pp. 2567–2575, 2022.

Yoo, J., Lee, H., Chung, I., Seo, G., and 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.

Zhu, X., Su, W., Lu, L., Li, B., Wang, X., and Dai, J. Deformable detr: Deformable transformers for end-to-end object detection. In *International Conference on Learning Representations*, 2020.

A. Training details

In training based on Sparse R-CNN, the batch size is 16. The identical data augmentations used in Deformable DETR (Zhu et al., 2020) are used for multi-scale training, where the input image size is 480 $\sim$ 800 with random crop and random horizontal flip. We use AdamW (Loshchilov & Hutter, 2017) 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. In training based on AdaMixer, following the AdaMixer schedule, we changed the learning rate decay epoch from (27, 33) to (24, 33).

B. Stop-gradient of the MCM loss

Table 7 shows the results when the stop-gradient scheme is applied to the MCM loss where the likelihood Eq. 2 of the main paper is calculated with three elements: the mixing coefficient π, the Cauchy $\mathcal{F}$ and the categorical distribution $\mathcal{P}$. 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 $\mathcal{F}$. The back-propagation toward the $\mathcal{F}$ 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 $\mathcal{F}$. Since it is an unintended phenomenon, applying stop-gradient toward the $\mathcal{F}$ is reasonable. However, there is no dramatic performance change with or without the stop-gradient.

C. Analysis of the number of Proposals

C.1. The MCM loss weight, β

Table 8 is an experiment on the number of proposals and the weight of the MCM loss β. 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.

C.2. Top-k ratio for proposals

Figure 11 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 * $0.7^5 \approx 336$. 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.

Figure 12 shows the trade-off between speed and performance when top-k is applied. Then, D-RMM shows higher APs than

Table 7. Stop-gradient of likelihood of the MCM loss.

Stop-gradient			Metric		
π	Cauchy ($\mathcal{F}$)	categorical ($\mathcal{P}$)	AP	AP_{50}	AP_{75}
-	-	-	46.8	64.7	51.3
-	stop	-	**47.0**	**64.8**	**51.6**
-	stop	stop	46.7	64.5	51.3

Table 8. The MCM loss weight β. '/' 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**

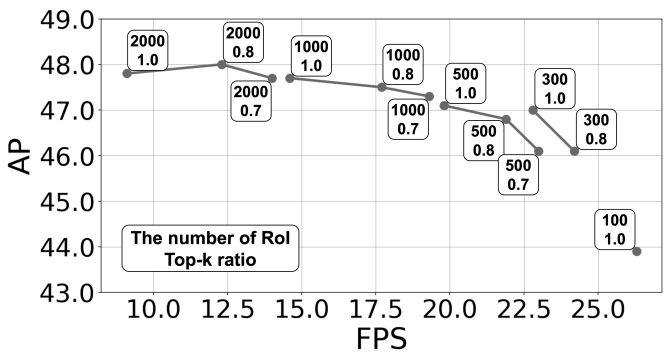

Figure 11. Top-k ratio for proposals. Top-k ratio denotes the percentage of remaining proposals after each stage.

Figure 12. COCO test-dev results. # is the number of proposals.

Table 9. Validation loss, the NLL loss and the MCM loss. $exp(-\mathcal{L}_{MCM})$ is $\max(g_i|X)/p(g_i|X)$, which means the ratio of maximum mixture component to likelihood for each instance.

Stage	$\mathcal{L}_{NLL}$			$\exp(-\mathcal{L}_{MCM})$		
	$w/\ \mathcal{L}_{MCM}$ (AP 47.0)	$w/o\ \mathcal{L}_{MCM}$ (AP 35.6)	$\frac{w/o\ \mathcal{L}_{MCM}}{w/\ \mathcal{L}_{MCM}}$	$w/\ \mathcal{L}_{MCM}$ (AP 47.0)	$w/o\ \mathcal{L}_{MCM}$ (AP 35.6)	$\frac{w/o\ \mathcal{L}_{MCM}}{w/\ \mathcal{L}_{MCM}}$
1	17.904	17.607	0.983	0.804	0.759	0.944
2	16.185	15.961	0.986	0.794	0.685	0.863
3	15.715	15.576	0.991	0.828	0.636	0.768
4	15.528	15.449	0.995	0.874	0.609	0.697
5	15.434	15.398	0.998	0.895	0.596	0.666
6	15.391	15.374	0.999	0.892	0.590	0.661

Deformable DETR (+1.2%p and +0.7%p) at similar speeds.

D. Validation loss

Table 9 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 $\exp(-\mathcal{L}_{MCM}) = \max(g_i|X)/p(g_i|X)$, 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 stages. On the other hand, the model without the MCM loss has lower $exp(-\mathcal{L}_{MCM})$ 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 Section 4.3 of the main paper, this result is consistent with that of Table 4 of the main paper in that the AP is significantly different depending on the presence of the MCM loss, whereas the AR is similar as 65.1% and 65.3% respectively.

E. Limited supervision

Current end-to-end methods learn ground truth information only from predictions selected by bipartite matching. The inherent ambiguity in the bounding box arising from occlusion, uncertain boundary, and inaccurate labeling (He et al., 2019) permits multiple plausible candidate bounding boxes for some objects. Hence, some objects have multiple possibilities for the bounding box representing them. However, due to bipartite matching, most end-to-end detectors learn one ground truth

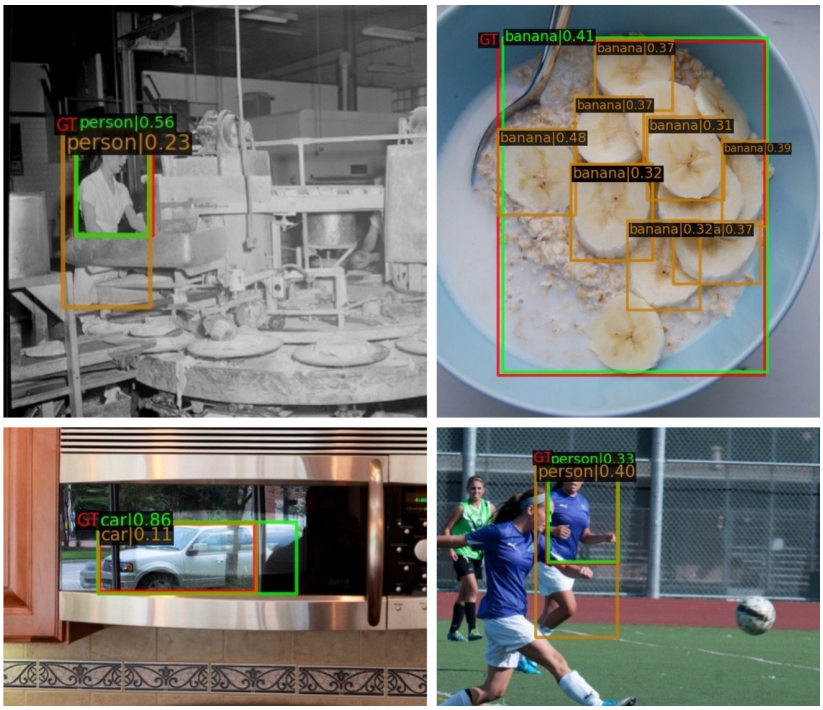

Figure 13. Limited supervision. Red: GT, Green: Positive prediction, Orange: Negative prediction. Negative Predictions are plausible predictions, but the class confidences are trained as zero.

information through only one positive prediction. Therefore, other adjacent negative predictions are trained to be classified as background and cannot learn ground truth information at all, even if this prediction adequately represents another possible bounding box of the object.

Figure 13 shows limited supervision examples. We take the banana class as an example (Figure 13 top right). The banana pieces showed high confidence, but each piece is assigned 'negative' by the annotation rule, and the banana class confidences are learned as zero. That is, it learns that it is not a banana. On the other hand, the NLL loss can learn that various banana predictions are bananas in the categorical distribution of NLL (Eq. 4). Whereas bipartite matching methods lose consistency when learning classification, ours is less prone to that with the NLL loss. The following question may arise: The NLL loss is calculated together with multiple predictions. Then, isn't it possible that the predictions far from ground truth greatly influence the weight update, which can cause learning problems? The answer is in Section F.2.2.

F. The number of positive assignments and interpretation in terms of the gradient

Recently, studies on the relationship between the number of positive assignments and performance have been published, such as DN-DETR (Li et al., 2022), Hybrid-DETR (Jia et al., 2022) and Group-DETR (Chen et al., 2022). We will interpret the NLL loss of our mixture model in terms of the number of positive assignments.

With the bipartite matcher, the number of positives in an image equals the number of ground truth boxes in that image. However, DN-DETR, Hybrid-DETR and Group-DETR improved performance by increasing the number of assigned positives during training. In common, the three papers created additional Transformer Decoder branches used only during training and increased positives with the new query sets of the additional branches. DN-DETR increased the number of queries and positive assignments as a denoising task, and the other two papers performed additional bipartite matching on additional queries of new branches. Therefore, computation cost and GPU usage significantly increase during training.

The NLL loss of the mixture model can be viewed from the perspective of increasing positives with only one query set without additional branches. Unlike the others, computation cost and GPU usage hardly increase during training. For each ground truth, all predictions are learned as if they were positives. However, when calculating the gradient, the effect of each prediction is automatically adjusted.

F.1. Copy-Paste ground truth for additional positive assignment

Table 10 shows the correlation between performance and the number of positives of Sparse R-CNN. We increased positives by copying and pasting the ground truth. Note that we did not create any additional branches like Group-DETR in Table 10. We just copied and pasted the ground truths to simulate many-to-one matching. And, we applied NMS because it is many-to-one matching.

As seen in the table, AP increased only by adequately increasing the positive. However, at some point, the increase in performance decreases, and even AP becomes similar to the default. On the other hand, Ours shows more significant performance improvements and no need for NMS, although the model learned all predictions as positives by the NLL loss.

Table 10. The relationship between AP and the number of Copy-Paste of ground truth. The number of positive per ground truth is the same as the number of copies.

NMS	The number of Copy-Paste per ground truth				
	1	2	3	4	Ours (NLL+MCM)
w/o	45.0	28.5	21.1	17.4	47.0
w/	45.2	46.1	45.6	45.3	47.2

F.2. Interpretation in terms of the gradient

F.2.1. Why does performance improvement decrease after Copy-Paste with bipartite matching?

Let us assume three times Copy-Paste of the ground truths. The classification loss of positive boxes becomes Eq. (10a) because the bipartite matching frameworks calculate the loss independently for each prediction. Then, the gradient becomes Eq. (10b). Therefore, a prediction with an enormous loss significantly affects the total gradient.

$$\mathcal{L}_{cls} = -\log(p_i) - \log(p_j) - \log(p_k) \quad where, (i, j, k : assigned \quad index) \tag{10a}$$

$$\nabla_i \mathcal{L}_{cls} + \nabla_j \mathcal{L}_{cls} + \nabla_k \mathcal{L}_{cls} = -\frac{1}{p_i} - \frac{1}{p_j} - \frac{1}{p_k} \tag{10b}$$

This can negatively affect learning. The loss will induce relatively less well-fitting predictions to be learned strongly rather than inducing learning better to fit the predictions closest to the ground truth.

Figure 14 shows examples. Even if it cannot be regarded as 'person' and the confidence is relatively low, it is learned as 'person' class. In this example, if the model learns with NLL, the effect of misassigned predictions on parameter update is reduced by the prediction that finds ground truth well. On the other hand, if learned with bipartite matching, the effect of incorrectly assigned prediction is relatively more significant by Eq. (10b).

F.2.2. Ours' NLL effectively increases performance even though it is all-to-one matching.

Eq (11) is a simplified representation of Eq (5) of the main paper. p_k is the likelihood of the k-th prediction. The gradient for the k-th prediction is Eq (11b).

$$\mathcal{L}_{NLL} = -\log(p_1 + p_2 + ... + p_K) \tag{11a}$$

$$\nabla_k \mathcal{L}_{NLL} = -\frac{1}{(p_1 + p_2 + ... + p_K)} \tag{11b}$$

In Eq. (11), what has a more significant effect on the gradient is that the likelihood is more prominent, that is, closer to ground truth. Note that the likelihood of D-RMM is the joint of Categorical and Cauchy. In other words, if a prediction fits a certain ground truth well, other predictions that are wrong a lot have little effect on learning.

Figure 14. Examples of incorrect assignments when copied three times. For effective visualization, only one object was expressed. The numbers next to the boxes are the confidence scores of the 'person' class. Red: GT, Green: best assignment out of 3, Orange: the other two. Columns 2 and 3 are also learned as a 'person' class.

Table 11. Experiments utilizing Group-DETR (Chen et al., 2022). The additional group size is 11. The GPU memory is reported on 2-batch size per GPU using MMDet (Chen et al., 2019). FLOPs are measured for a sample of size 800x1280.

Method		AP	Param (M)	Training GPU memory (GB)	Inference FLOPs (G)
Sparse R-CNN	default	45.0	106.1	7	165.5
	w/ Group	45.9	107.0	21	165.5
	w/ ours	47.0	106.1	7	165.5
AdaMixer	default	46.6	134.6	9	124.6
	w/ Group	47.4	135.5	35	124.6
	w/ ours	48.4	134.6	9	124.6

F.2.3. MULTI-BRANCH METHOD

Table 11 is the result of applying the Group-DETR method (Chen et al., 2022) to Sparse R-CNN and AdaMixer. Note that Table 10 is the result of many-to-one matching on just one branch without additional branches, and Table 11 has several branches and each branch performs one-to-one matching. Group-DETR has the effect of increasing the number of query groups, which independently performs bipartite matching. We experimented by adding 11-groups.

In Sparse R-CNN and AdaMixer, the AP increased significantly by additional groups but less than ours. Even if the number of positives is increased by applying the Group-DETR method, it is difficult to fundamentally avoid the hand-crafted assignment and inaccurate confidence score problem mentioned in Section 1 because bipartite matching is performed in each group. Furthermore, due to additional groups, the GPU memory increases significantly.

G. Discussion for objective function and confidence

We will discuss the reliability in terms of the cross-entropy loss (DETR) and the focal loss (Deformable DETR, Sparse R-CNN). We describe in Section 1 that it is a key for reliability that the entire pipeline has a probabilistic point of view. In DETR, the classification with cross-entropy has a probabilistic basis, but bipartite matching and regression are not. Since cross-entropy is calculated based on hand-crafted bipartite matching, classification is not trained from the probabilistic point of view.

The focal loss is relatively well-calibrated compared to the cross-entropy. The focal loss adjusts the loss scale according to confidence; a weak loss for high confidence (easy example) and a substantial loss for low confidence (hard example). Therefore, the high confidence prediction is learned relatively weakly, and the low confidence is learned relatively strongly. So it becomes an S shape histogram. The low confidence area becomes overconfident, and the high confidence area becomes underconfident. Note that below the diagonal line is called overconfident, and the opposite is called underconfident.

Table 12. Comparison of the costs to calculate the loss of Sparse R-CNN and D-RMM in a single training iteration.

	bipartitie matching	loss function	total
Sparse R-CNN	28.7 ms	8.3 ms	37.0 ms
D-RMM (ours)	-	6.7 ms	6.7 ms

Table 13. Comparison of the training components between D-RMM and MDOD. NLL is negative log-likelihood. MoC is a mixture of Cauchy. MM is a mixture model of Cauchy and categorical distribution. Also, GT and RoI mean ground truth bounding box and RoI bounding box.

	D-RMM (ours)	**MDOD**
Localization	NLL of MM_{GT}	NLL of MoC_{GT}
Classification		NLL of MM_{RoI}
Matching for localization	-	-
Matching for classification	-	many-to-one matching
BBox de-duplication	MCM loss in training	NMS in inference
Objectiveness or background probability	un-normalized mixing coefficient	background probability of softmax output

H. Training cost

We compare the training costs for loss calculation between D-RMM and Sparse R-CNN in a single training iteration. While Sparse R-CNN follows a bipartite-matching-based training process, our method D-RMM does not require bipartite matching for loss calculation. As shown in Table 12, the time required for loss calculation is longer in Sparse R-CNN compared to D-RMM. The major factor contributing to this difference is the presence of bipartite matching. However, in overall training, this difference is not significant since the ratio of network forward and backward times during training is considerably higher (approximately 97 %) than the time required for loss calculation.

I. D-RMM vs. MDOD

In terms of methodology, our D-RMM and MDOD utilize mixture models in entirely different ways in the training of the detector network. As shown in Table 13, D-RMM performs joint training of localization and classification without matching, using the negative log-likelihood of a mixture of Cauchy and categorical distribution with ground truth boxes. On the other hand, MDOD trains localization without matching using the negative log-likelihood of a mixture of Cauchy with ground truth boxes, and trains classification using the negative log-likelihood of a mixture of Cauchy and categorical distribution with region of interest (RoI) boxes sampled from a mixture model and employing many-to-one matching (referred to as RoI sampling in MDOD). Additionally, D-RMM performs box deduplication during training using the MCM loss, while MDOD performs deduplication during inference using non-maximum suppression (NMS). Regarding the objectiveness of the boxes, D-RMM obtains an objectiveness score using mixing coefficients, whereas MDOD computes the background probability through the softmax output.

J. More examples for analysis

Figure 6 of the main paper shows qualitative result of Sparse R-CNN, model for ablation study and D-RMM. Figure 15 shows more examples for these models.

Figure 15. Qualitative result. From above, 1st row:Sparse R-CNN (AP:45.0%), 2nd row:NLL (AP:35.6%), 3rd row:NLL+NMS (AP:42.7%), 4th row:NLL+WTA (AP:45.8%), 5th row:NLL+MCM (ours) (AP:47.0%). Red/Green colors indicate confidence 0.1-0.5 and 0.5-1.0, respectively.