

Bridging the Gap Between Object Detection and User Intent via Query-Modulation

Marco Fornoni¹, Chaochao Yan^{2,1}, Liangchen Luo¹, Kimberly Wilber¹, Alex Stark¹, Yin Cui¹, Boqing Gong¹, and Andrew Howard¹

¹Google Research, ²University of Texas at Arlington

Abstract

When interacting with objects through cameras, or pictures, users often have a specific intent. For example, they may want to perform a visual search. With most object detection models relying on image pixels as their sole input, undesired results are not uncommon. Most typically: lack of a high-confidence detection on the object of interest, or detection with a wrong class label. The issue is especially severe when operating capacity-constrained mobile object detectors on-device. In this paper we investigate techniques to modulate mobile detectors to explicitly account for the user intent, expressed as an embedding of a simple query. Compared to standard detectors, query-modulated detectors show superior performance at detecting objects for a given user query. Thanks to large-scale training data synthesized from standard object detection annotations, query-modulated detectors also outperform a specialized referring expression recognition system. Query-modulated detectors can also be trained to simultaneously solve for both localizing a user query and standard detection, even outperforming standard mobile detectors at the canonical COCO task.

1. Introduction

Convolutional neural networks (CNNs) have transformed computer vision, enabling new use cases, as research pushes the limits of accuracy and efficiency. Still, on capacity-constrained mobile devices even the best performing models can produce results mismatched to the user intent. An example scenario is illustrated in Fig. 1.

The goal of this paper is to present a formulation for building mobile object detectors that can leverage user queries to significantly improve detection accuracy. This new flavor of models and training procedures, named *Query-Modulated object Detectors (QMD)*, can even outperform traditional detectors when no query is available, thanks to multitask learning. Computationally, Query-Modulated Detectors are thus more efficient than traditional detection models.

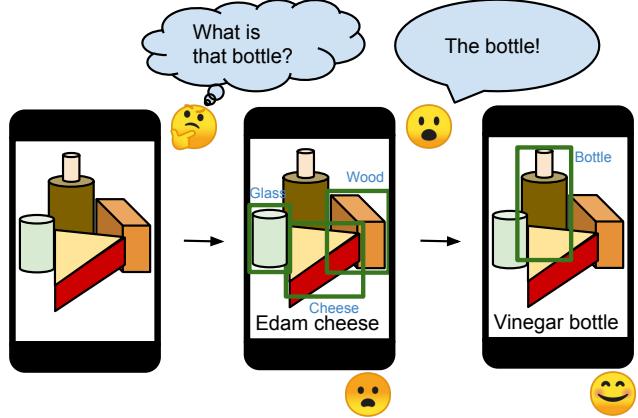


Figure 1. Example use case for query-modulated detection: a user points their phone at a group of objects, seeking information about one of them. The model may not detect the object of interest, or it may detect it with the wrong label. By explicitly taking into account the user input, and actively searching for the requested object, the model can correct the results, and detect the intended object. Further along the pipeline, the phone might follow up with nutrition information, or product reviews.

The main contributions of the paper are:

- 1) We introduce the notion of query-modulated detection, the task of adapting an object detector computation to take into account a user query.
 - 2) We propose a way to train query-modulated detectors directly using synthetic queries generated from large-scale object detection datasets.
 - 3) We present a multi-task training strategy to simultaneously solve for both the query-modulated detection task, and the traditional object detection task.
 - 4) We show that for query-modulated detection, query embeddings can be learned end-to-end, in a lightweight manner, significantly reducing computational and memory costs with respect to traditional visual grounding approaches, depending on large language models such as BERT [5].
- We validate our approach by comparing to standard object detection and referring expression recognition models.

2. Related Works

In this section we review prior works that most closely relate to our approach, arranging them into two classes: object detection, and referring expression recognition.

Object detection. Object detection is one of the fundamental tasks in computer vision, solving for both localizing the objects, and classifying them using a closed set of object labels. Modern object detection models are built as convolutional neural networks and can be broadly divided into two classes—one-stage and two-stage—according to the model architecture. Two-stage detection methods [1, 4, 6, 7, 14, 25, 26] decompose object detection into object localization and classification by first generating object region proposals, and then refining and classifying those proposals. One-stage methods [13, 15, 17, 22–24, 30, 38, 40] accomplish localization and classification simultaneously, achieving much lower inference times with some accuracy degradation. While some of the errors produced by capacity-constrained mobile detectors could be addressed by explicitly taking into account the user intent, architectures capable of doing so [33] have only been explored in the *referring expression recognition* and visual grounding literature.

Referring expression recognition. The goal of referring expression recognition is to unambiguously localize an object, or a region in an image referred to by a *natural language expression*, which can be a word, a phrase, a sentence or even a dialogue [3, 11, 18, 20, 21, 33, 36]. Similarly to object detection, existing referring-expression recognition systems employ either two-stage, or one-stage architectures. These are inspired by, or directly derived from, the object detection literature. Two-stage frameworks [18, 20, 31, 35–37] operate by first generating a set of region candidates, and subsequently ranking these regions according to the provided expression. The performance of two-stage frameworks is largely capped by the region proposal generation [29, 33], which is not trained simultaneously with the candidates ranking module in an end-to-end manner. One-stage methods [2, 27, 32–34, 39] recently emerged as the preferred approach. These methods directly regress a bounding box in accordance with the user query. Such solutions typically rely on one-stage detection architectures, such as YOLO [22–24] or SSD [17].

Referring Expression recognition is currently considered expensive. First, data acquisition is costly: for each given object of interest, multiple referring expressions need to be collected and validated by different annotators [11]. Second, computationally expensive NLP models such as BERT [5] need to be used to process the verbal expressions. On the other hand, [11] reports that 50% of the referring expressions in their study is composed only of a

noun, while 82% of the remaining ones uses only one additional attribute, most typically the *object coarse location* (e.g. left, right, bottom). In other words, 91% of the objects can be correctly localized with referring expressions as simple as an object label, plus an optional coarse location. A basic referring expression recognition system could thus be directly trained on existing large-scale object detection datasets, by synthesizing the referring expressions from the groundtruth bounding-box annotations, and stripping all NLP processing. Such an approach could be built on top of high-performance object-detection training pipelines, and scaled through inexpensive data annotation pipelines. In other words, it'd come almost for free.

3. Approach

In this paper we formulate the problem of referring-expression recognition in the context of object detection, and refer to it as *Query-Modulated Detection*. We focus on simple yet effective queries, consisting of an object label, plus an optional coarse location (e.g. “top-right”). Similarly to referring expression recognition, we assume that when a user provides a given query, the target object must be present in the image. In contrast to referring expression recognition models: 1) We directly train on *synthetic queries* from common object detection datasets. 2) We employ a *multi-task training* strategy to optimize simultaneously for both standard object detection and query-modulated detection. 3) Since we focus on a closed vocabulary with a limited set of words (the set of labels supported by the detector), we optionally replace BERT embeddings with *inexpensive K-Hot binary encodings*, and learn the query embedding end-to-end. Furthermore, while visual referring expression recognition is traditionally a single-object localization problem (there exists one and only one image region corresponding to the expression), our setting is slightly more challenging, as we ask the model to detect *all* the objects corresponding to the given query

The architecture for our model is shown in Fig. 2. It features two inputs: the image (in blue), and the query encoding (in red). Similarly to [33], an embedding for the query is computed by two fully-connected layers (w_1 and w_2), followed by a ℓ_2 -normalization. The query embedding is then tiled and concatenated to the last convolutional feature maps (also ℓ_2 -normalized), before the box predictor. A 1×1 convolution is then employed to fuse the query with the image features. Finally, the fused feature maps are passed to a canonical SSD box predictor [17].

3.1. Query Synthesis

As discussed in Sec. 2, 91% of the objects in the ReferIt dataset [11] can be correctly localized with referring expressions as simple as the object label, plus an optional coarse location. We thus propose to generate simple syn-

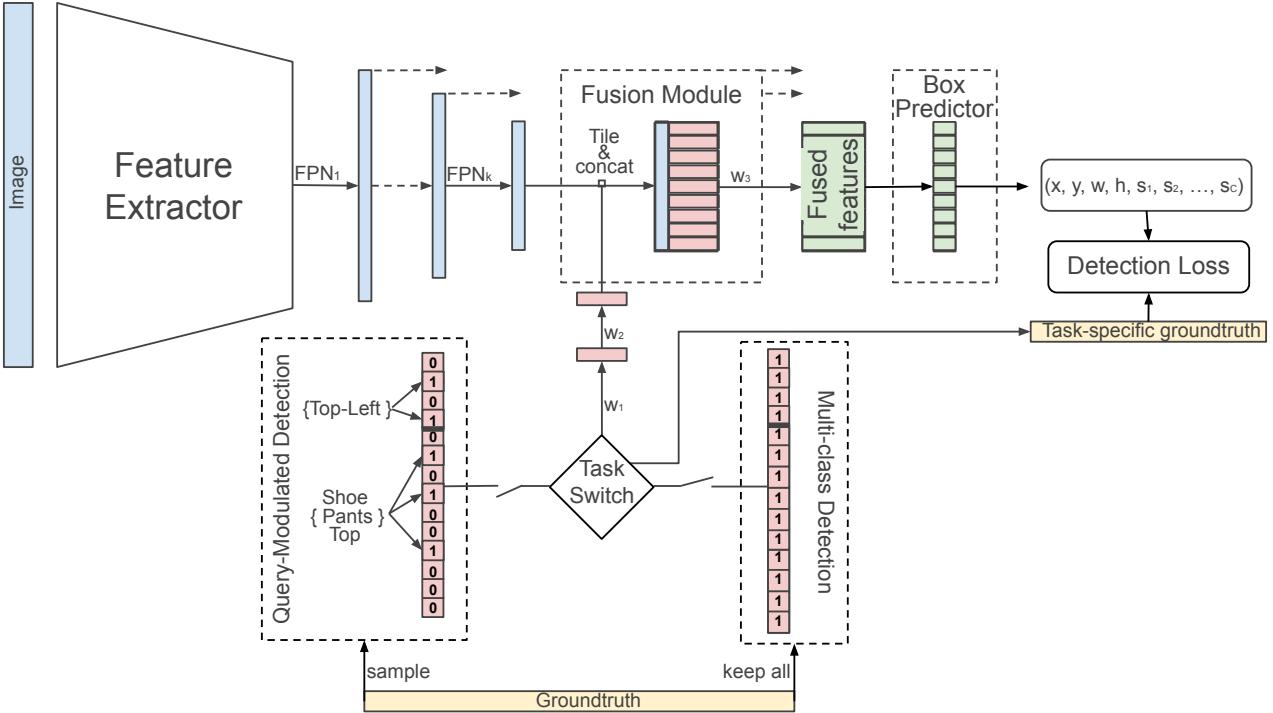


Figure 2. The Query-Modulated Detector (QMD) architecture, and its multi-task training pipeline. The reserved $[1, 1, \dots, 1]$ query triggers standard object detection. All other values are treated as detection queries. A query is represented by a k -hot encoding, and is divided in two parts: the label encoding, and the location encoding. During training the model alternates between detecting the objects referred by randomly synthesized queries, and standard object detection.

thetic referring expressions from standard object detection groundtruth annotations, provided by large-scale datasets such as Open Images [12] and COCO [16]. Specifically, we focus on referring expressions containing the categorical label of the object, plus an optional coarse spatial location (e.g. “on the right”). Differently from the referring expression recognition literature, we ask our models to detect *all* objects matching a given expression, as in our case more than one object may be associated to the expression. We also allow for queries containing multiple object labels at the same time, e.g. all clothing labels, or all vehicles labels. For brevity, we use the term *query* to indicate such loose referring expressions, and use the term “*query-modulated detection*” to indicate the task. We focus on the following three types of queries, which can be directly synthesized and evaluated on object-detection datasets:

- *Single-Label Detection (SLD)*. Detect all objects for a single label, sampled from the image groundtruth.
- *K-Label Detection (KLD)*. Detect all objects for K random labels, obtained by independently sampling (with probability 0.5) each label from the image groundtruth.
- *Localized-Label Detection (LLD)*. Detect all objects for a single random label, and a coarse spatial loca-

tion. Both the label and the coarse location are sampled from the image groundtruth.

3.1.1 Localized-Label Detection Query Synthesis

In this section we provide the details of the approach used to synthesize LLD queries. Based on the spatial distribution in [11] (Fig. 3), and to reflect the way users typically refer to spatial locations, we employ a few predefined slices for each axis of a standard 4×4 grid. As shown in Fig. 3, we use 3 reference overlapping slices for the y -axis: $\{top, center, bottom\}$, and 5 partially overlapping slices for the x -axis: $\{far-left, left, center, right, far-right\}$. For brevity, we refer to those as y -slices and x -slices. Furthermore, for each axis we also employ an additional *all* slice, to indicate a lack of constraint on that axis. A location constraint is finally defined as the product of a y -slice constraint, and a x -slice constraint. E.g. (top, right), (bottom, left), (all, left).

As in [11], we assume the user would provide a location constraint only if it is necessary to correctly identify the target object. For example, if the user wants to indicate the car in the image, and the image contains only one car instance, location constraints are not necessary to correctly detect the desired object. Furthermore, we assume that the user will

select the spatial constraint that most clearly identify the object of interest. For example, if both *far-right*, *right*, and *all* are valid *x*-slices, we assume the user would pick the most informative one, namely: *far-right*.

More formally, we adopt the following approach to synthesize LLD queries from the image groundtruth:

1. Randomly select a target label, among the labels in the image groundtruth, and prune out all boxes not from this label.
2. If the target label contains more than one object instance in the image:
 - (a) Randomly select one of the instances as the final target.
 - (b) Select the tightest (*y*-slice, *x*-slice) constraint containing the target instance. In case of multiple possible solutions, select the one containing the least number of groundtruth boxes. E.g. if the target object lies in the overlap between the *center*, and *right* *x*-slices, select the one containing the least number of groundtruth boxes.
3. Finally, prune all remaining boxes not contained in the selected (*y*-slice, *x*-slice) constraint .

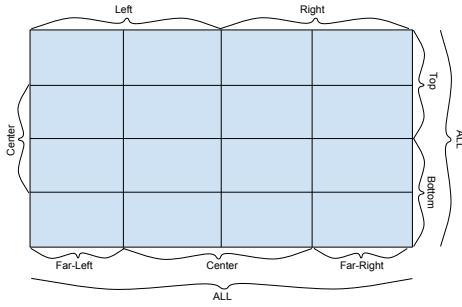


Figure 3. Grid and reference *x*-slices (horizontal axis), and *y*-slices (vertical axis). A 4×4 grid is divided into 3 overlapping slices on the *y* axis, and 5 overlapping slices on the *x* axis. Additional two slices fully covering each axis are used to denote lack of constraints on that axis.

We use $\text{IntersectionOverArea} \geq 0.9$ as the criteria for considering a box as contained in a cell grid. Namely the intersection between the box and the grid, divided by the box area should be at least 0.9.

3.2. Query encoding

We represent user queries using *k-hot encodings*. For label encoding, given an object detection problem with C classes, we use a C -dimensional binary vector, where each bit indicates a given class label, and a value of 1 indicates that all objects from that class need to be localized.

For location encoding, we again use a binary representation. We use the first 3 bits to represent the *y*-slice, and the next 5 bits to represent the *x*-slice, and the all-ones encoding to represent the lack of constraints on the *y*-axis, or the *x*-axis. For example, (top, right) is represented by the [1,0,0,0,0,1,0] encoding, (all, far-right) is represented by the encoding [1,1,1,0,0,0,1], and the complete lack of constraints (all, all) is represented by the encoding: [1,1,1,1,1,1,1]. The final *k*-hot encoding is obtained by concatenating the location encoding, to the label encoding.

With respect to BERT embeddings [5], *k*-hot encodings are extremely efficient to compute and store, and do not introduce any bias in how they model classes. BERT embeddings, on the other hand, incorporate extra prior knowledge, since similar class names are supposed to have a closer distance in embedding space. We experimentally compare *k*-hot encodings with BERT [5] embeddings in Sec. 4.9.

3.3. Multi-Task Training

A model trained solely to perform query-modulated detection is not suitable as a stand-alone replacement of a standard object detector (see Sec. 4.5). The main reason is that the model is trained only using queries for objects actually present in the image. For example, when presented with an image not containing the queried label, the modulated feature maps may “hallucinate” the desired object, returning high-confidence boxes on unrelated objects. To address the above issue, we introduce a new multi-task training strategy. Specifically, for each training image we randomly switch, with a given task switching ratio, between:

1. Query-modulated detection. Using a query synthesized and encoded as described in 3.1.
2. Standard object detection. In this case the model is provided the fixed [1, 1, 1, ..., 1] query, activating all labels regardless of the image groundtruth.

We call this approach the *Query-Modulated object Detector (QMD)*. The full architecture is depicted in Fig. 2. Experimental results are provided in Sec. 4.

4. Experiments

4.1. Implementation details

We implemented our models in TensorFlow Object Detection API [10], using a RetinaNet [15] architecture with both mobile and server-side backbones. For the mobile-friendly models we employed a MobileNet-v2 [28], with 1.0 width multiplier, 320px resolution, L3-L7 128-d FPN-Lite, and a fully convolutional box predictor with: weight-sharing across scales, four 128-d layers, and depthwise-separable convolutions. For the server models we employed a ResNet-101 [8], with 1024px resolution, L3-L7 256-d

FPN, and a fully convolutional box predictor with four 256-d layers per scale. For QMD, the query encoding is first passed through 2 fully connected layers with 512 neurons each, then ℓ_2 -normalized, tiled and concatenated across the spatial dimension of the ℓ_2 -normalized FPN feature maps. Finally, the concatenated features are fused together using a 1×1 convolution, and fed to the box predictor.

4.2. Datasets

The ReferIt dataset [11] comprises 19,894 images, annotated with 130,525 expressions spanning 96,654 distinct objects. We employ the standard 9,000, 1,000 and 10,000 split from [9] for training, validation, and test respectively. The COCO dataset [16] is a 80-class common object detection dataset. We use the 2017 version of the dataset. Since the test annotations are not released and our evaluation protocol differs from the official one, we perform all our evaluations on the validation set. Open Images Detection (OID) v4 [12] is a large-scale, large-vocabulary dataset, composed of more than 1.7M training, 40K validation, and 125K test images, spanning 600 different classes. Since the test set annotations are made available, for this dataset we report metrics as measured on the test set (except where specified).

4.3. Metrics

To evaluate accuracy we adopt the following metrics:

1. Detection (DET): standard mAP for object detection.
2. Query-modulated detection. In this setting the model is asked to detect all objects for a given query. The image is guaranteed to contain at least one object matching the query. We report the AP measured when solving for SLD, KLD, or LLD.

Since for query-modulated detection the target label is provided as an input to the model, it is superfluous to evaluate the model labeling accuracy. After pruning the groundtruth with the desired label, we thus assign the remaining groundtruth and all detected boxes to one single class-agnostic label, and evaluate the predictions in a class-agnostic fashion. Following the standard protocol, COCO (m)AP is computed using the standard set of [.5:.95] IOUs, while Open-Images (m)AP is computed with the Open Image V2 metric [19] using 0.5 IOU.

4.4. ReferIt and post-processing baselines for SLD

Why not just using a referring expression model? As a first experiment to motivate our approach, we use the COCO 2017 validation set as the target dataset, and compare two obvious baselines to perform query-modulated detection. Namely: 1) Evaluating a model specifically trained for general referring expression recognition [33]; 2) Post-processing a standard object detector to retain only the boxes specified by the user query.

Baseline	SLD AP	SLD AR@1
Object Detector + Post-processing	47.72	26.32
ReferIt Model [33]	12.19	12.23

Table 1. We compare the performance of a One-Stage BERT Referring Expressions model [33] trained on ReferIt [11] to a simple post-processed object detector, on the subset of the COCO validation that only contains ReferIt entities (labels). See text for details.

For this experiment, we focus on the single label detection (SLD) metric, as object labels are the most important and common form of referring expressions [11], and correctly recognizing them is a necessary condition for recognizing more complex ones.

As a referring expression recognizer baseline, we reimplement the One-Stage BERT referring expression recognition model [33], and train it on the ReferIt [11] dataset. Specifically, we use a one-stage SSD architecture with a ResNet-101 backbone at 320px, with query fusion and spatial encoding features, as in [33]. For query embedding we use a 768-d BERT embedding from the CLS token. Also, similarly to [33] we employ several augmentations to avoid over-fitting on this small training set: random brightness, hue, saturation, contrast, gray-scale and random crop augmentations. We use a COCO pre-trained feature extractor and froze all layers, only training the fusion and box predictor modules. Since we are aiming at evaluating the model on detection data, we neither add a softmax non-linearity across all anchors, nor do we optimize the anchors on the ReferIt dataset (as done by [33]). This model achieves 57.1% top-1 accuracy on ReferIt [11]. This is slightly inferior¹ to the 59.3% reported in [33], but far above the other approaches benchmarked in [33], so we feel it is a comparable reproduction.

For the second baseline, we train a standard ResNet101-SSD detector on COCO (Row 5 in Table 2) and post-process its outputs by pruning all detections except for that of the query class.

For a fair evaluation, we only use the subset of the COCO validation set containing the 73 COCO labels that also appear in the ReferIt vocabulary (everything except *baseball bat/glove*, *fire hydrant*, *hair drier*, *hot dog*, *parking meter*, *tennis racket*). This corresponds to 4,926 validation images.

Results are shown in Table 1. The post-processed object detector handily outperforms the ReferIt model in both SLD AP and Recall@1. Besides the ReferIt training set being much smaller than COCO (9,000 images vs 118,287) and the dataset shift, this could possibly due also to the ReferIt model being trained with only a single box associated to each query. This last factor can be excluded by noting that

¹The delta is possibly explained by the absence of soft-max across anchors, the lack of anchors customization, as well as the lack of fine-tuning of the feature extractor.

Average Recall @ 1 is also much lower with respect to the COCO model. This metric considers only one single top-confident box for each (query, image) pair.²

To summarize, ReferIt models have poor generalization abilities due to the very limited number of training samples, and severely under-perform a simple post-processing baseline. This motivates developing query-modulated detectors that can directly be trained on large-scale detection datasets.

4.5. Single Task Query-Modulated Detector

In this Section we discuss the results achieved by QMD trained solely to solve for query-modulated detection. We consider both mobile and server models, and perform experiments on both COCO-17 and Open Images Detection v4, for both object-detection, and query-modulated detection. In Table 2 we report the SLD AP, KLD AP and detection

Dataset	Backbone	Model	SLD AP	KLD AP	DET mAP
OID	MobileNetV2	Detector	51.3	45.4	27.3
OID	MobileNetV2	QMD	67.6	57.2	1.4
COCO	MobileNetV2	Detector	28.9	25.6	22.6
COCO	MobileNetV2	QMD	33.4	28.0	9.8
COCO	ResNet101	Detector	47.7	44.0	38.9
COCO	ResNet	QMD	49.6	46.1	22.7

Table 2. **Single-Task results on OID and COCO.** For SLD / KLD models are asked to put boxes on objects belonging to specific label(s) in each image. Labels are selected among those appearing in the groundtruth. For the standard detector this is obtained by pruning all detections from labels other than the requested one(s).

mAP achieved by these models on the the OID test set, and the COCO validation set. As a baseline, we report the performance of a standard object detector using the same architecture hyper-parameters (feature extractor, input resolution, FPN layers, etc.). To solve for SLD, and KLD using a standard object detector, predictions are pruned to retain only those matching the classes specified in the query. All detections are then mapped to a single class.

To evaluate the ability of QMD to operate also as a standard object detector we also compare the COCO mAP achieved on the task of detecting all objects in the image. For QMD this is achieved by activating all labels on all images, using a $[1, 1, 1, \dots, 1]$ query. By modulating the FPN activations to focus on the queried objects, the MobileNet-based QMD is able to outperform the object detector by +16.3% SLD AP on OID, and +4.5% SLD AP on COCO. The ResNet model still improves COCO SLD AP by +1.9%. On the other hand, in the standard detection set-

²For example, if the dataset contained only one image with three ground-truth “car” boxes, and for the “car” query the ReferIt model predicted only one box on one single car, while the COCO model predicted three boxes on three cars, the AR@1 for the two models would be identical.

ting, triggered by the $[1, 1, 1, \dots, 1]$ query, the single-task QMD severly underperforms the standard object detector.

The main advantage of QMD with respect to post-processing a standard object detector is that QMD is trained for actively “searching” the desired object(s) in the image. This is particularly important for large-vocabulary datasets like OID, where the box classification problem is considerably harder compared to COCO, and post-processing might not be enough to confidently and stably surface the objects belonging to the user-requested class.

To summarize, QMD significantly outperforms the post-processing baseline for SLD and KLD. This is especially true for large-vocabulary problems, where class-confusion is a more important issue. However, single-task training is not directly suitable for solving standard object detection.

4.6. Multi-Task Query-Modulated Detector

In this Section we analyze the performance achieved by QMD trained for both query-modulated detection and standard object detection, as described in Section 3.3.

Dataset	Backbone	Model	SLD AP	KLD AP	DET mAP
OID	MobileNetV2	Detector	51.3	45.4	27.3
OID	MobileNetV2	QMD	63.1	51.5	27.4
COCO	MobileNetV2	Detector	28.9	25.6	22.6
COCO	MobileNetV2	QMD	32.0	27.8	23.3
COCO	ResNet101	Detector	47.7	44.0	38.9
COCO	ResNet	QMD	50.0	46.5	38.6

Table 3. **Multi-Task results on OID and COCO.** QMD matches Detector DET mAP, while outperforming on SLD and KLD AP.

Results for OID-v4 and COCO-17 are summarized in Table 3. With multi-task training QMD is able to match the standard detector DET mAP, while still providing large gains on SLD / KLD (+11.8% on OID). As shown in Fig. 4, by conditioning on the desired labels, QMD is able to detect the desired objects even when they are missed, or wrongly labelled when using the model as a multi-class detector. For a per-class analysis of the results, please refer to the supplementary material.

To summarize, it is possible to augment an object detector with the ability of leveraging user queries to significantly improve SLD and KLD AP for the desired labels, while preserving the DET mAP when no query is specified.

4.7. Efficiency Analysis

In this Section we provide an analysis of QMD efficiency compared to post-processing a standard object detector. Similarly to [30], for both QMD and the standard detector baseline we simultaneously scale up the model width, depth, and resolution. When scaling the model width, we increase the width multiplier for backbone, FPN, and box predictor. When scaling the model depth we increase the number of layers in the box predictor. Specifically, with $D \in \{0, 1, 2, 3, 4, 5, 6, 7, 8\}$, we set:

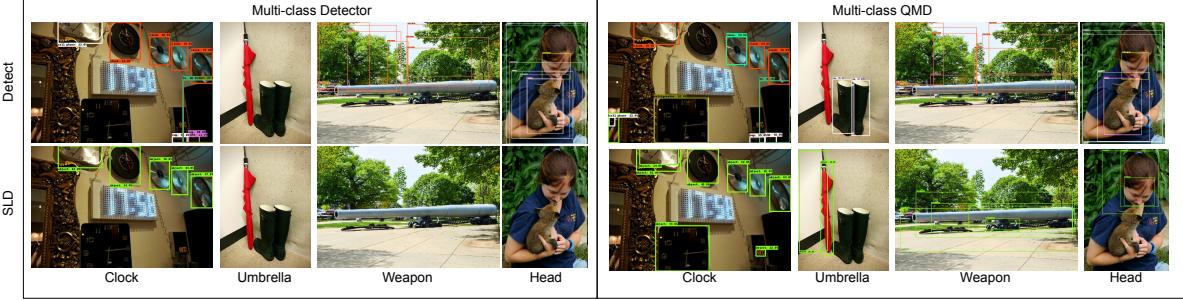


Figure 4. Eight top-scoring detections above 0.2 confidence, on COCO validation (clock, umbrella) and OID test (weapon, head) images. Left: results using a standard object detector. Right: results with Multi-Task QMD. First row: object detection results. Second row: SLD results, with the corresponding query.

- $width_multiplier = 1.22^D$
- $layers(box_predictor) = 4 + D$
- $resolution = 320 + 64 * D$

$D = 0$ corresponds to models in Table 3. For each subsequent D value, FLOPs are approximately doubled (Fig. 5).

Accuracy VS FLOPs (COCO)

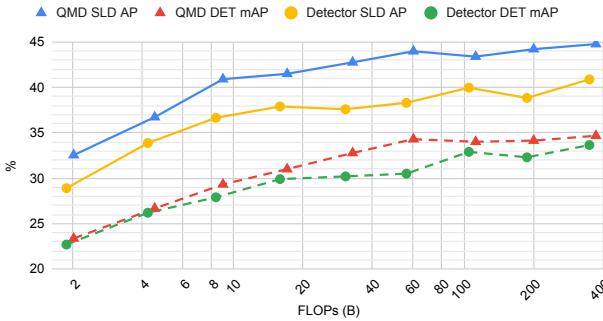


Figure 5. COCO SLD AP and DET mAP for QMD and a standard Detector, when varying FLOPs. The FLOPs axis is in log-scale.

We observe that: 1) QMD FLOPs overhead WRT standard detectors is between 6.8% (372.7B vs 349B - slowest model) and 7.4% (2.02B vs 1.88B - fastest model). 2) For all considered computational budgets, QMD outperforms the vanilla-detector on both SLD AP and DET mAP. We attribute the DET task efficiency gains to the SLD task positively affecting the DET task during multi-task training. 3) QMD achieves the same SLD AP of the largest vanilla-detector (40.9%), with 39x less FLOPS (9B vs 349B).

To summarize, QMD is largely more efficient compared to traditional object detectors for the SLD task. Thanks to multi-task training, it is also equally or more efficient for the standard COCO DET task.

4.8. Localized Label Detection

In this Section we report the performance of QMD when trained to perform Localized Label Detection (LLD). For

this experiment we only use MobileNetV2 backbones and, as for the ReferIt model in Section 4.4, we use the spatial encoding from [33]. As reported in Table 4, adding loca-

Dataset Model	LLD AP	SLD AP	KLD AP	DET mAP
OID Detector	55.4	51.3	45.4	27.3
OID QMD	68.1	62.5	50.0	27.4
COCO Detector	31.1	28.9	25.6	22.6
COCO QMD	35.9	32.4	28.0	23.5

Table 4. Localized Label Detection results on OID and COCO. All models use a MobileNetV2 backbone. QMD is trained in a multi-task fashion, to solve for both LLD and DET.

tion constraints further improves the results for all models with respect to SLD. This is expected, as we are making the problem even simpler by constraining the area where the objects of interest are searched. Still, similarly to Section 4.6, the improvement from SLD to LLD is larger for QMD than for Detector (+5.6% | +3.5% on OID | COCO for QMD, vs +4.1% | +2.2% for Detector), even though QMD SLD baseline is higher (62.5% | 32.4% on OID | COCO, vs 51.3% | 28.9% for Detector). Visualizations for the localized queries and the corresponding results are provided in the supplementary material.

To summarize, while coarse location constraints can be used to further improve accuracy for all models, QMD is more effective at leveraging them.

4.9. Ablation studies

We provide here several ablation studies, analyzing the impact of model and training hyper-parameters. All experiments in this section are performed on the COCO and OID validation sets.

SLD vs KLD training. We consider two different QMD training policies, corresponding to the SLD, and KLD metrics. In the SLD case, training queries are built by sampling one single label from the image groundtruth, with embeddings build as 1-hot vectors. In the KLD case, training

queries are built by sampling each groundtruth label with a probability of 0.5, with embeddings built as k -hot vectors. All experiments for this study are performed using only the MobileNetV2 backbone, and only on the COCO 2017 validation set. To minimize noise, we only perform this ablation on the single-task QMD model.

Training	SLD AP	KLD AP
SLD (1-hot)	33.3	20.0
KLD (k -hot)	33.4	28.7

Table 5. Single-Task MobileNetV2 QMD AP, training with 1-hot and k -hot, on COCO 2017. All evaluation results are reported on the validation set.

Results in Table 5 show that KLD-training largely outperforms SLD for the KLD task, while also matching or outperforming SLD-training for the SLD task. We thus adopt KLD as the standard training for all experiments in Sections 4.6 to 4.8.

BERT embeddings. We analyze how BERT [5] embeddings compare to binary 1-hot / k -hot embeddings. As in Section 4.4, BERT embeddings are computed by passing the textual label to BERT, and extracting the CLS token. KLD BERT embeddings are obtained by *averaging* the BERT embeddings of the k labels.

BERT Type	Training/Evaluation			
	SLD/SLD	SLD/KLD	KLD/SLD	KLD/KLD
Mobile (192)	33.4	13.1	33.0	28.0
Base (768)	33.4	18.6	32.7	27.9
Large (1024)	33.3	18.4	32.8	28.3

Table 6. Single-Task QMD results using BERT embeddings. For KLD, BERT embeddings of the k labels are simply averaged.

In Table 6 we provide detailed results for different mobile and server BERT embeddings. Increasing the embedding size from 192 to 768, or 1024 does not significantly improve results. Furthermore, comparing results in Table 6 with those in Table 5 shows that in our closed-world object-detection settings, binary 1-hot / k -hot embeddings achieve similar or better performance than BERT embeddings. Please note that *binary 1-hot / k -hot embeddings are much cheaper to compute and store* with respect to their BERT counterpart. Based on the above observations we employ binary 1-hot / k -hot embeddings throughout the paper.

Detection Task Sampling Ratio. Figure 6 shows the effect of varying the detection task sampling ratio during training, for both COCO and OID validation sets. The optimal value is dataset dependent. For both datasets and most choices of the task ratio QMD SLD AP is largely better than Detector SLD AP. On the COCO dataset, multi-task training improves also the DET mAP for most values.

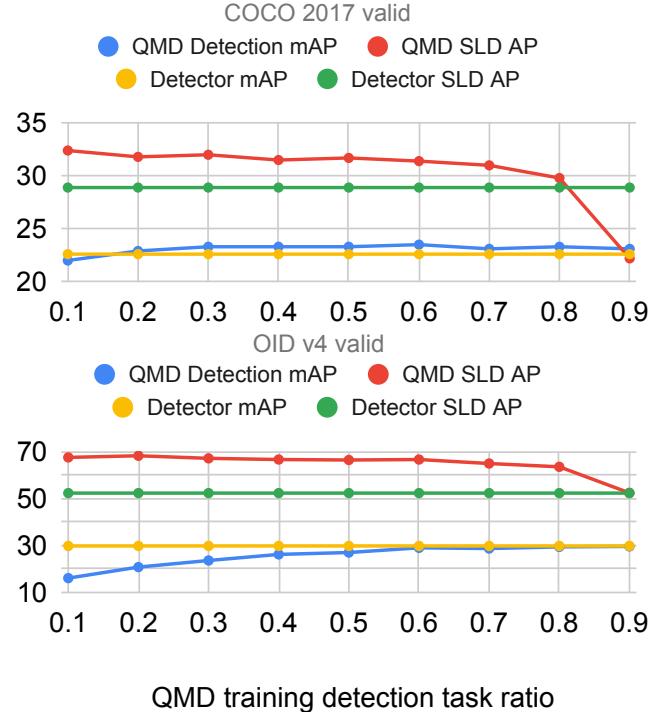


Figure 6. Effect of varying the training detection task sampling ratio on the COCO 2017 and OID v4 validation sets.

5. Conclusions

In this work we presented a formulation to build object detectors that can be queried for detecting specific objects of interest. We focused on simple queries containing only the class label(s) of the object(s) of interest, and optionally a coarse location. We described how to synthesize and encode such queries from standard object detection annotations. We demonstrated that a ReferIt model does not generalize well to large-scale object detection problems, and is outperformed by a simple detector plus post-processing baseline. We showed how the post-processing baseline is in turn largely outperformed by Query-Modulated Detectors. This is particularly true on large-vocabulary datasets, where class-confusion is a more severe issue. We also showed how by jointly training for both standard object-detection and query-modulated detection, one can efficiently and simultaneously solve both problems. Thanks to multi-task training, QMD even improves performance on the original COCO object-detection task. Finally, we showed how for QMD, a simple k -hot query encoding performs equally to BERT, while being much cheaper to compute and store. Our formulation is generic and can potentially support other types of queries, and query embeddings. In the future we plan to investigate training QMD on more complex queries, still synthesized from large-scale object detection datasets.

References

- [1] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6154–6162, 2018. [2](#)
- [2] Xinpeng Chen, Lin Ma, Jingyuan Chen, Zequn Jie, Wei Liu, and Jiebo Luo. Real-time referring expression comprehension by single-stage grounding network. *arXiv preprint arXiv:1812.03426*, 2018. [2](#)
- [3] Volkan Cirik, Louis-Philippe Morency, and Taylor Berg-Kirkpatrick. Visual referring expression recognition: What do systems actually learn? In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 781–787, 2018. [2](#)
- [4] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems*, pages 379–387, 2016. [2](#)
- [5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. [1, 2, 4, 8](#)
- [6] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014. [2](#)
- [7] K He, G Gkioxari, P Dollar, and R Girshick. Mask r-cnn. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017. [2](#)
- [8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2016. [4](#)
- [9] Ronghang Hu, Huazhe Xu, Marcus Rohrbach, Jiashi Feng, Kate Saenko, and Trevor Darrell. Natural language object retrieval. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2016. [5](#)
- [10] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, et al. Speed/accuracy trade-offs for modern convolutional object detectors. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7310–7311, 2017. [4](#)
- [11] Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara Berg. Referitgame: Referring to objects in photographs of natural scenes. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 787–798, 2014. [2, 3, 5](#)
- [12] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Alexander Kolesnikov, Tom Duerig, and Vittorio Ferrari. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *IJCV*, 2020. [3, 5](#)
- [13] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 734–750, 2018. [2](#)
- [14] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 936–944. IEEE, 2017. [2](#)
- [15] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. [2, 4](#)
- [16] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. [3, 5](#)
- [17] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016. [2](#)
- [18] Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan L Yuille, and Kevin Murphy. Generation and comprehension of unambiguous object descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 11–20, 2016. [2](#)
- [19] Open images v2 detection metric. [online] available at: https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/evaluation_protocols.md#open-images-v2-detection-metric, 2017. [5](#)
- [20] Bryan A Plummer, Liwei Wang, Chris M Cervantes, Juan C Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. In *Proceedings of the IEEE international conference on computer vision*, pages 2641–2649, 2015. [2](#)
- [21] Jordi Pont-Tuset, Jasper Uijlings, Soravit Changpinyo, Radu Soricut, and Vittorio Ferrari. Connecting vision and language with localized narratives. In *European Conference on Computer Vision*, pages 647–664. Springer, 2020. [2](#)
- [22] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016. [2](#)
- [23] Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. [2](#)
- [24] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018. [2](#)
- [25] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. [2](#)
- [26] S Ren, K He, R Girshick, and J Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*, 39(6):1137, 2017. [2](#)

- [27] Arka Sadhu, Kan Chen, and Ram Nevatia. Zero-shot grounding of objects from natural language queries. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4694–4703, 2019. 2
- [28] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted residuals and linear bottlenecks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, jun 2018. 4
- [29] Amar Shrestha, Krittaphat Pugdeethosapol, Haowen Fang, and Qinru Qiu. Magnet: Multi-region attention-assisted grounding of natural language queries at phrase level. *arXiv preprint arXiv:2006.03776*, 2020. 2
- [30] Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10781–10790, 2020. 2, 6
- [31] Liwei Wang, Yin Li, Jing Huang, and Svetlana Lazebnik. Learning two-branch neural networks for image-text matching tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(2):394–407, 2018. 2
- [32] Zhengyuan Yang, Tianlang Chen, Liwei Wang, and Jiebo Luo. Improving one-stage visual grounding by recursive subquery construction. In *ECCV*, 2020. 2
- [33] Zhengyuan Yang, Boqing Gong, Liwei Wang, Wenbing Huang, Dong Yu, and Jiebo Luo. A fast and accurate one-stage approach to visual grounding. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, oct 2019. 2, 5, 7
- [34] Raymond Yeh, Jinjun Xiong, Wen-Mei Hwu, Minh Do, and Alexander Schwing. Interpretable and globally optimal prediction for textual grounding using image concepts. In *Advances in Neural Information Processing Systems*, pages 1912–1922, 2017. 2
- [35] Licheng Yu, Zhe Lin, Xiaohui Shen, Jimei Yang, Xin Lu, Mohit Bansal, and Tamara L Berg. Mattnet: Modular attention network for referring expression comprehension. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1307–1315, 2018. 2
- [36] Licheng Yu, Patrick Poirson, Shan Yang, Alexander C Berg, and Tamara L Berg. Modeling context in referring expressions. In *European Conference on Computer Vision*, pages 69–85. Springer, 2016. 2
- [37] Hanwang Zhang, Yulei Niu, and Shih-Fu Chang. Grounding referring expressions in images by variational context. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4158–4166, 2018. 2
- [38] Shifeng Zhang, Longyin Wen, Xiao Bian, Zhen Lei, and Stan Z Li. Single-shot refinement neural network for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4203–4212, 2018. 2
- [39] Fang Zhao, Jianshu Li, Jian Zhao, and Jiashi Feng. Weakly supervised phrase localization with multi-scale anchored transformer network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5696–5705, 2018. 2
- [40] Qijie Zhao, Tao Sheng, Yongtao Wang, Zhi Tang, Ying Chen, Ling Cai, and Haibin Ling. M2det: A single-shot object detector based on multi-level feature pyramid network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 9259–9266, 2019. 2

Supplementary Material

1. Visualizations for Localized-Label Detection

In Fig. 7 we provide visualizations for the COCO models reported in Tab. 4 of the paper. By modulating the detection using both the label and the coarse location, QMD can better localize difficult objects with respect to the pipeline with a standard detector followed by post-processing. This results in an improvement of +12.7% LLD AP@0.5 for OID, and +4.8% LLD AP@[.5:.95] for COCO (Sec. 4.8 of the paper).

One limitation of QMD is that it can also seldom output boxes outside the region of interest. I.e. different from post-processing, QMD does not always 100% suppress the confidence scores for boxes outside the region of interest. The confidence for such boxes is normally low. This can be seen for the “Right Cow” query in Fig. 7.

2. Class-agnostic QMD

In this Section we provide additional results for a class-agnostic version of multi-task QMD. When using the $[1, 1, 1, \dots, 1]$ query, class-agnostic QMD behaves as a standard class-agnostic detector, and is thus unable to tell apart objects from different classes. On the other hand, by requesting a given label, it is possible to condition the model to produce “class-agnostic” boxes only for the desired label.

Dataset	Backbone	Model	SLD AP	KLD AP	DET AP
OID	MobileNetV2	Detector	51.3	45.4	41.7
OID	MobileNetV2	QMD	67.2	60.1	46.4
COCO	MobileNetV2	Detector	28.9	25.6	25.7
COCO	MobileNetV2	QMD	33.3	28.7	27.3
COCO	ResNet101	Detector	47.7	44.0	43.4
COCO	ResNet	QMD	50.5	46.7	44.5

Table 7. **Class-agnostic QMD on OID and COCO.** For SLD / KLD the model is asked to put boxes on objects belonging to specific label(s) in each image. DET AP refers to the class-agnostic detection AP.

In Tab. 7 we report the results for class-agnostic QMD on COCO and OID v4. By adopting a simple multi-task training approach, the class-agnostic detection AP achieved by QMD is always the highest in the benchmark. Similarly to the multi-class results (Sec. 4.6 in the submitted manuscript), we observe much larger gains on OID v4 (Tab. 7 above) WRT COCO (Tab. 2 on the submitted manuscript). We believe this is due to the larger number of classes in OID.

To summarize, the experiment in Tab. 7 shows that query-modulation could also be used to strongly condition the class-agnostic box-proposal stage for two-stage object detectors.

2.1. Visualizations on COCO

In Fig. 8 we visualize outputs of class-agnostic QMD, on COCO-valid. Standard class-agnostic detection results are visible in the first row. The second row shows how, by requesting a given label, it is possible to condition class-agnostic QMD to produce “class-agnostic” boxes only for the desired label.

In Fig. 9 we provide some visualizations for class-agnostic QMD, compared to multiclass detection with post processing. In many cases (Refrigerator, Teddy Bear, Horse, Bed) the object cannot be obtained by post-processing the multi-class detector, as the latter does not provide a detection for the desired class. On the other hand, in cases where objects similar to the queried label are present in the image, QMD can provide false-positive detections on the distractor objects.

2.2. Per-class results on COCO

In Fig. 10 and Tab. 8 we report the COCO-valid class-by-class AP achieved by:

- Class-agnostic QMD, with each class queried only on images actually containing that class.
- Multi-class detector post-processed by pruning out detections for classes not appearing in a given image.
- Multi-class detector as is, reported for reference.

Overall, QMD improves over the post-processing by 4.1%. QMD also provides the highest AP for any given class, except Snowboard. QMD provides the highest average improvement WRT the multi-class detector (6.8% = 4.1% + 2.7%).

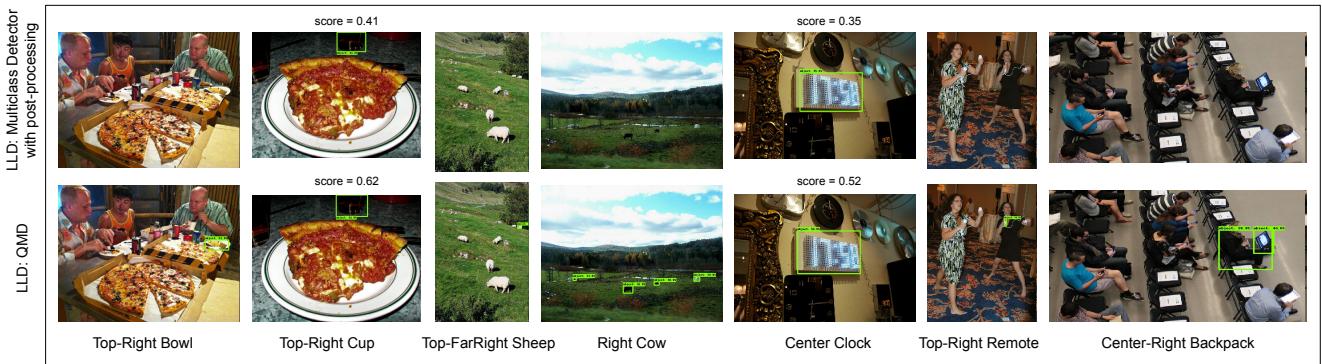


Figure 7. Visualization of results for the MobileNetV2 models with Localized Label Detection (LLD) queries. Top: results obtained by post-processing a standard object detector. Bottom: results obtained by QMD. For each example is reported the LLD query used to generate it. Visualizations include top 5 boxes above 0.3 confidence. For images where both QMD and the object detector provide a detection we report the detection score for both models.

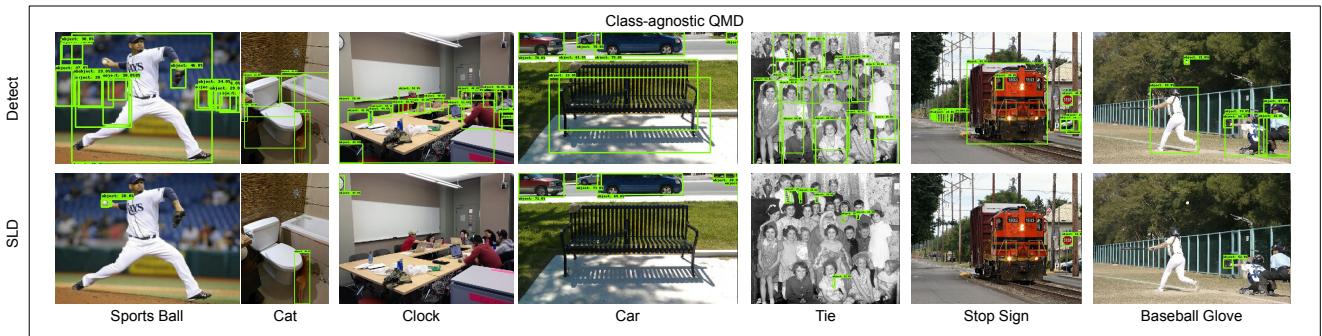


Figure 8. Visualization of results for the MobileNetV2 class-agnostic QMD on COCO. First row: class-agnostic detection results. Second row: SLD results, with the corresponding query. Both rows are obtained using class-agnostic QMD.

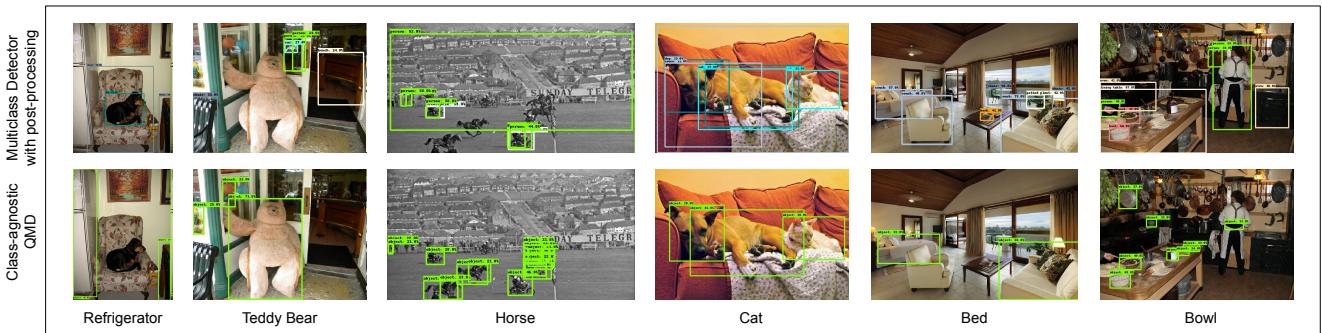


Figure 9. Eight top-scoring detections above 0.2 confidence, on COCO-valid. First row: multiclass detector with post-proc, pruning out all detections for classes not in the image. Second row class-agnostic QMD with the associated query. All models use a MobileNetV2 feature extractor.

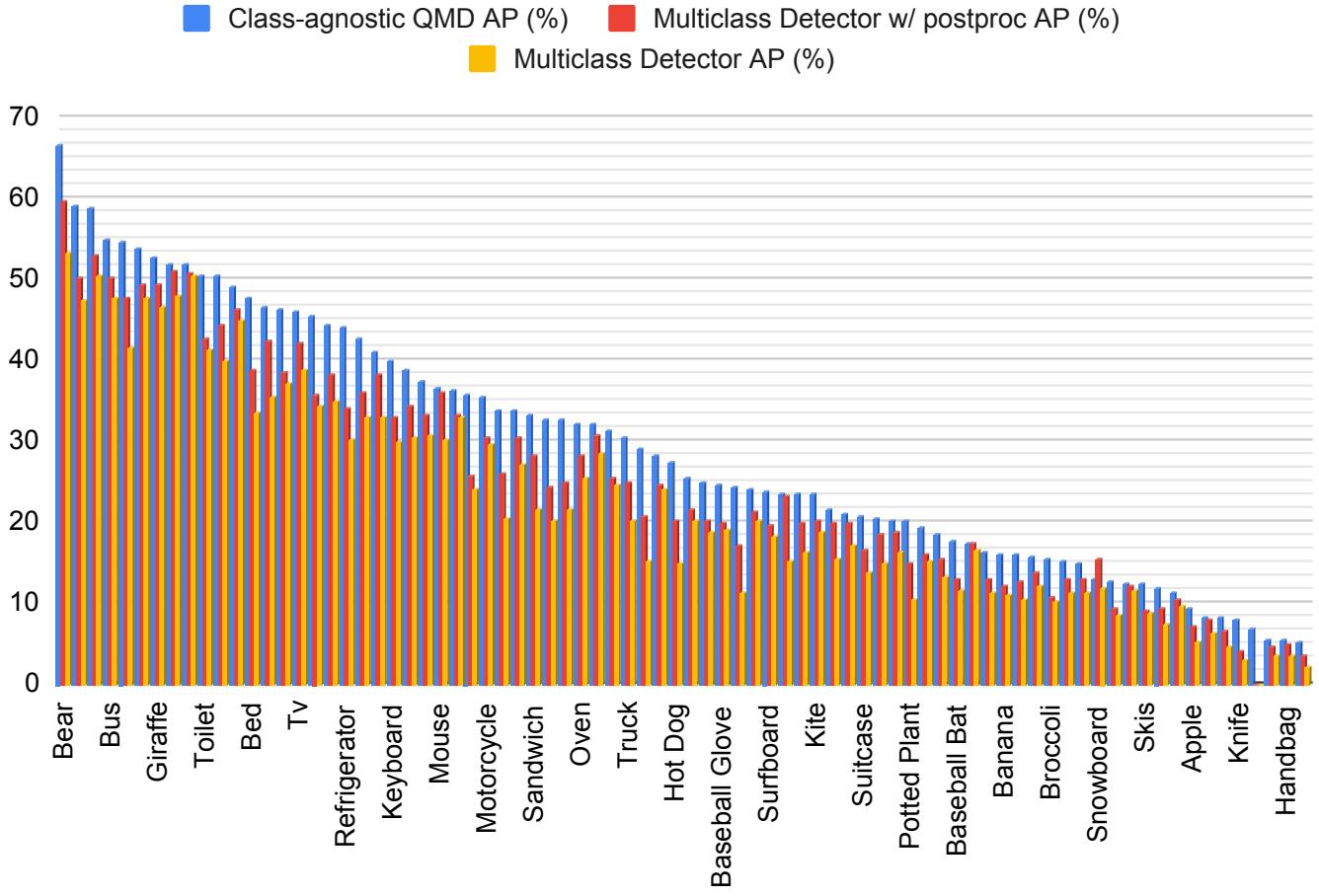


Figure 10. Visualizations of per-class AP obtained by, respectively: class-agnostic QMD; the multi-class detector post-processed to retain only detections for classes that do appear in the image; the multi-class detector as is. Classes are sorted by the AP obtained by QMD. On average QMD improves multi-class detection results by 6.8%, while post-processing only improves AP by 2.7%. All models use a MobileNetV2 feature extractor.

Class	Class-agnostic QMD AP (%)	Multiclass Detector w/ postproc AP (%)	Multiclass Detector AP (%)	(QMD - Multiclass w/ postproc) AP Delta (%)	(Multiclass w/ postproc - Multiclass) AP Delta (%)
Refrigerator	44	34	30.1	10	3.9
Teddy Bear	35.7	25.8	24	9.9	1.8
Horse	45.6	35.8	34.4	9.8	1.4
Cat	59.2	50.1	47.3	9.1	2.8
Bed	47.7	38.9	33.5	8.8	5.4
Bowl	32.8	24.4	20.2	8.4	4.2
Cake	29.1	20.9	15.1	8.2	5.8
Toilet	50.6	42.8	41.4	7.8	1.4
Laptop	46.4	38.6	37.2	7.8	1.4
Donut	32.6	24.9	21.6	7.7	3.3
Dining Table	33.8	26.1	20.5	7.7	5.6
Hot Dog	27.5	20.2	14.8	7.3	5.4
Scissors	24.5	17.2	11.3	7.3	5.9
Dog	54.6	47.6	41.7	7	5.9
Hair Drier	7	0	0	7	0
Keyboard	39.9	32.9	29.9	7	3
Bear	66.6	59.7	53.2	6.9	6.5
Couch	42.7	36.1	32.9	6.6	3.2
Fire Hydrant	50.4	44.3	39.8	6.1	4.5
Pizza	44.4	38.4	34.9	6	3.5
Train	58.9	52.9	50.6	6	2.3
Skateboard	31.4	25.6	24.7	5.8	0.9
Truck	30.4	25	20.3	5.4	4.7
Potted Plant	20.1	14.8	10.5	5.3	4.3
Sandwich	33.3	28.2	21.6	5.1	6.6
Motorcycle	35.4	30.4	29.7	5	0.7
Baseball Glove	24.7	19.8	19	4.9	0.8
Baseball Bat	17.8	13	11.5	4.8	1.5
Broccoli	15.4	10.7	10.1	4.7	0.6
Umbrella	25	20.3	18.9	4.7	1.4
Bus	54.9	50.2	47.7	4.7	2.5
Cow	38.9	34.3	30.6	4.6	3.7
Airplane	53.7	49.4	47.8	4.3	1.6
Suitcase	20.9	16.6	13.8	4.3	2.8
Frisbie	46.5	42.3	35.5	4.2	6.8
Clock	37.3	33.2	30.9	4.1	2.3
Surboard	23.7	19.7	18.2	4	1.5
Oven	32.2	28.3	25.5	3.9	2.8
Tv	46	42.1	38.8	3.9	3.3
Banana	16.1	12.3	11.1	3.8	1.2
Knife	8	4.2	3	3.8	1.2
Sink	25.4	21.6	20.3	3.8	1.3
Cup	23.5	19.8	16.2	3.7	3.6
Tennis Racket	28.4	24.7	24	3.7	0.7
Bottle	16.1	12.6	10.4	3.5	2.2
Sheep	33.8	30.4	27.1	3.4	3.3
Giraffe	52.6	49.3	46.7	3.3	2.6
Chair	16.2	12.9	11.3	3.3	1.6
Bicycle	19.4	16.1	15.2	3.3	0.9
Bench	18.6	15.4	13.3	3.2	2.1
Skis	12.4	9.2	8.8	3.2	0.4
Kite	23.5	20.3	18.9	3.2	1.4
Person	36.4	33.3	33	3.1	0.3
Fork	12.6	9.5	8.5	3.1	1
Microwave	41.1	38.3	33	2.8	5.3
Elephant	49	46.3	44.9	2.7	1.4
Car	24	21.3	20.3	2.7	1
Remote	11.8	9.3	7.5	2.5	1.8
Apple	9.3	7.2	5.2	2.1	2
Wine Glass	15.1	13	11.3	2.1	1.7
Tie	15.7	13.7	12.2	2	1.5
Sports Ball	20.5	18.6	14.9	1.9	3.7
Carrot	14.9	13.1	11.3	1.8	1.8
Vase	21.6	19.9	15.6	1.7	4.3
Spoon	5.2	3.6	2.1	1.6	1.5
Backpack	8.2	6.7	4.7	1.5	2
Bird	20.1	18.7	16.4	1.4	2.3
Parking Meter	32.1	30.9	28.6	1.2	2.3
Cellphone	21	19.9	17.2	1.1	2.7
Zebra	51.9	50.8	50.4	1.1	0.4
Book	5.6	4.6	3.6	1	1
Stop Sign	51.9	50.9	48	1	2.9
Traffic Light	11.2	10.5	9.7	0.7	0.8
Mouse	36.6	36	30.1	0.6	5.9
Toaster	23.6	23.2	15.1	0.4	8.1
Handbag	5.4	5	3.5	0.4	1.5
Toothbrush	8.4	8.1	6.2	0.3	1.9
Boat	12.5	12.3	11.5	0.2	0.8
Orange	17.5	17.4	16.6	0.1	0.8
Snowboard	13	15.5	11.9	-2.5	3.6
Mean	29.4	25.3	22.6	4.1	2.7

Table 8. Per-class localization AP on COCO-valid. For QMD, each class is queried only on images actually containing it. For the multiclass detector with post-processing, predictions for each class are retained only on images actually containing it. All models use a MobileNetV2 feature extractor.