Optimizing Queries over Video via Lightweight Keypoint-based Object Detection

Jiansheng Dong
Wuhan University of Technology
Wuhan, Hubei, China
1210090743@qq.com

Jingling Yuan
Wuhan University of Technology
Wuhan, Hubei, China
yjl@whut.edu.cn

Lin Li
Wuhan University of Technology
Wuhan, Hubei, China
cathylilin@whut.edu.cn

Xian Zhong
Wuhan University of Technology
Wuhan, Hubei, China
zhongx@whut.edu.cn

Weiru Liu
University of Bristol
Bristol, UK
weiru.liu@bristol.ac.uk

ABSTRACT

Recent advancements in convolutional neural networks based object detection have enabled analyzing the mounting video data with high accuracy. However, inference speed is a major drawback of these video analysis system because of the heavy object detectors. To address the computational and practicability challenges of video analysis, we propose FastQ, a system for efficient querying over video at scale. Given a target video, FastQ can automatically label the category and number of objects for each frame. We introduce a novel lightweight object detector named FDet to improve the efficiency of query system. First, a difference detector filters the frames whose difference is less than the threshold. Second, FDet is employed to efficiently label the remaining frames. To reduce inference time, FDet detects a center keypoint and a pair of corners from the feature map generated by a lightweight backbone to predict the bounding boxes. FDet completely avoid the complicated computation related to anchor boxes. Compared with state-of-the-art real-time detectors, FDet achieves superior performance with 29.1% AP on COCO benchmark at 25.3ms. Experiments show that FastQ achieves 150× to 300× speed-ups while maintaining more than 90% accuracy in video queries.

CCS CONCEPTS

• **Computing methodologies** → **Object detection**.

KEYWORDS

object detection, real-time, anchor-free, keypoint-based, high-resolution representation

ICMR '20, June 08–11, 2020, Dublin, Ireland

ACM ISBN 978-1-4503-XXXX-X/20/06…$15.00
https://doi.org/10.1145/1122445.1122456

ACM Reference Format:
Jiansheng Dong, Jingling Yuan, Lin Li, Xian Zhong, and Weiru Liu. 2020. Optimizing Queries over Video via Lightweight Keypoint-based Object Detection. In *ICMR '20: International Conference on Multimedia Retrieval, June 08–11, 2020, Dublin, Ireland*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Video is unstructured data with rich semantic information and is growing explosively at scale. For instance, YouTube claims over 400 hours of video uploaded every single minute. Therefore, low-cost video analysis technologies are gaining attention in many domains, including crowd counting, traffic camera analysis, visual data management,and so on. Analysts are increasingly interested in adopting object detection technologies to quickly understand higher-level feature and make statistics and queries on object-level information of video data. For example, traffic supervisors may need to count the types and numbers of vehicles appearing in the camera to optimize traffic control. In crowded places, it is necessary to monitor the number of people in real time to prevent accidents caused by rapid changes in the number of people.

In recent years, with the development of computer vision technology, the performance of video analysis has been greatly improved, with the accuracy close to human level. In particular, a common video analysis method is to adopt the object detection technologies to detect the object-level information of each frame in the target video, including the category, time, location and quantity of the object. These object-level information is an important video information that can answer the queries of analysts.

Unfortunately, it is slow and expensive to process video data with high accuracy CNN-based object detectors. For example, it takes more than 200ms to process an image using Mask R-CNN [6] with a state-of-the-art average precision (AP) of 39.8%, which means it takes about 10 GPU days to process the video data obtained by a single camera in a day. Moreover, state-of-the-art object detection models continue to get deeper and more costly to analyse. Therefore, enabling

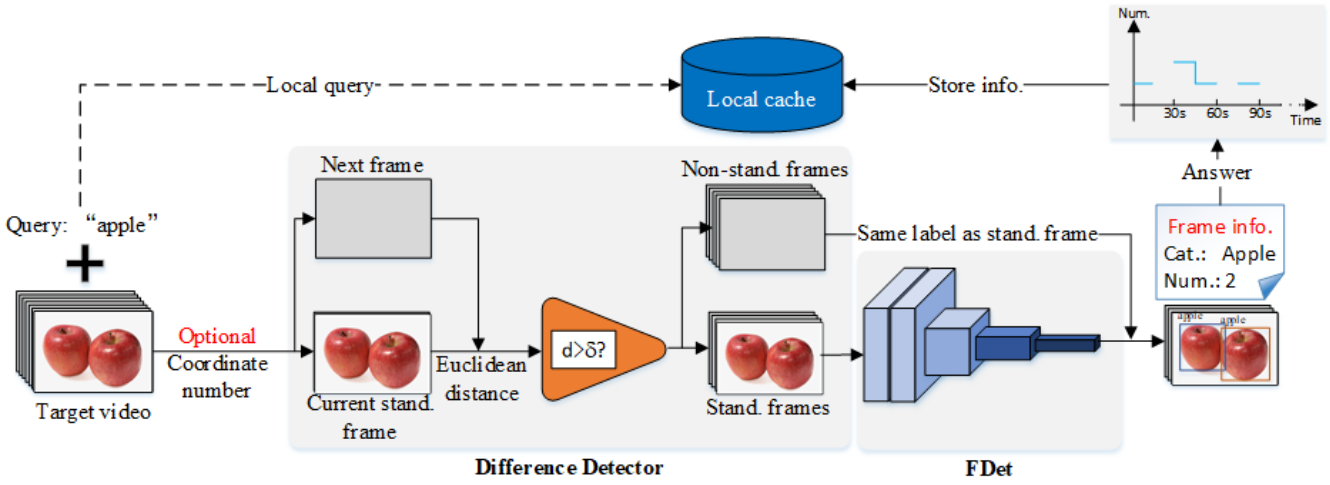

Figure 1: FastQ is a system for efficient querying over video at scale. Given a video, the category of object, and the optional input is the number of object and the coordinate O of the region to be detected. If the coordinate O is not none, FastQ will crop each frame according to O. We adopt a difference detector to filter standard frames. The Euclidean distance is used as metric of similarity. If the distance between adjacent frames is less than the threshold, the next frame will be labeled and excluded. Finally, FDet is utilize to detect the standard frames. Non-standard frames will be labeled with the same information as similar frames.

efficient queries over video is a crucial but challenging task in computer vision.

A common approach is to employ lightweight detectors to reduce inference time by sacrificing precision. Lightweight detectors such as SSD [12] and YOLOv2/v3 [16, 17] achieve real-time inference on GPU with very competitive accuracy. Concretely, for a 300×300 image, YOLOv3 achieves 28.2% AP at 41 frames per second (FPS). However, these mainstream lightweight detectors often place a set of pre-defined anchor boxes densely over an image to predict coordinates of objects. The extensive anchor boxes introduce complicated computation such as calculating the intersection-over-union (IoU) scores with ground-truth bounding boxes. Moreover, the performance of anchor-based detectors is sensitive to the manually designed sizes, aspect ratios and number of anchor boxes. Therefore, these hyper-parameters need to be carefully designed in anchor-based detectors.

To overcome the drawbacks of anchor-based detectors, some anchor-free detectors have emerged. Most anchor-free detectors detect keypoints to predict the bounding boxes, which is similar to the task of human pose estimation, also known as keypoint detection. For instance, CornerNet [11] detects each object by predicting the heatmaps of top-left and bottom-right corners of bounding boxes. These keypoint-based detectors avoid the complicated computation and hyper-parameters related to anchor boxes.

However, inference speed is still a major drawback of these detectors because of the heavy backbone network. For example, CenterNet [2] achieves an average precision (AP) of 47.0% on COCO at 961ms and the Hourglass-104 [13] backbone takes up more than 80% of the inference time. Although anchor-free detectors cost less than anchor-based detectors in theory, it still needs to simplify the backbone network to realize efficient query over video.

To efficiently query over video, we propose FastQ that can automatically label the category and number of objects in each frame of video. On the one hand, a low-cost difference detector is employed to filter out duplicate frames to avoid redundant detection. On the other hand, we introduce a novel lightweight object detector named FDet to improve the efficiency of query system. To summarize, we have the following contributions:

(1) We propose a system named FastQ for efficient querying over video that can automatically label the category and number of objects in each frame of video.

(2) To avoid redundant detection, we employ a low-cost difference detector to filter out duplicate frames.

(3) Combining keypoint-based detection and a novel lightweight backbone, we introduce a real-time object detector named FDet. Compared with state-of-the-art real-time detectors, FDet achieves superior performance with 29.7% AP on COCO benchmark at 35.7 FPS.

2 RELATED WORK

Video Querying Systems. Video querying is an important technology of video analysis, which is attracting more and more researchers' attention. NoScope [10] trains specialized detectors that gives up all generality, but greatly greatly simplifies the detector. NoScope can query over video with high-speed, but it can only predict whether there is an object in each frame of the video. Focus [8] achieves a trade-off

between accuracy and efficiency by adopting a highly compressed model to quickly filter frames without objects at ingest-time. BlazeIt [9] accepts queries via a SQL-like language for spatiotemporal information of objects in videos and present optimizations for three common classes of queries. Panorama [23] proposes a unified information system architecture for unbounded vocabulary queries over video and devise self short-circuiting and query cache to improve efficiency. In this work, our FastQ supports multi-object simultaneous query, and can query based on the number and coordinates of objects.

Anchor-based Detectors. In anchor-based detectors, a set of anchor boxes is place densely over an image and classified as positive or negative patches. And offsets regression is employed to refine the coordinates of the bounding boxes. R-CNN [5] uses a selective search method to locate RoIs in the input images. Faster-RCNN [3, 18] proposes region proposal network (RPN) which can generate RoIs by regressing the anchor boxes and generates two outputs for each ROI, a class label and a bounding box offset. Compared with Faster-RCNN, Mask-RCNN [6] adds a third branch of mask in parallel, which achieves pixel-level segmentation. In addition, some detectors such as SSD [12] and YOLOv2/v3 [16, 17], which balance accuracy and efficiency, have become the paradigms of modern detectors. In this work, our FDet avoids the complicated computation related to anchor boxes, which is helpful to improve the efficiency of the detector.

Anchor-free Detectors. YOLOv1 [15] might be the early anchor-free detector which predicts bounding boxes at points near the center of objects. CornerNet [11] is a recently proposed one-stage anchor-free detector, which detects each object by predicting the heatmaps of top-left and bottom-right corners of bounding boxes. Followed CornerNet, CenterNet [2] detects each object as a triplet of keypoints top-left, bottom-right and center), which improves both precision and recall. ExtermeNet [24] detect four extreme points and one center point of objects. In this work, instead of heavy backbones like Hourglass-104, we devise a lightweight backbone which can greatly improve the inference speed of the detector.

3 FASTQ SYSTEM

3.1 Overview

Given a video consisting of a sequence of frames $F = \{f_1, f_2, ..., f_m\}$. The input is the index C of category(e.g., index 1 indicates person, index 2 indicates car or index 48 indicates apple), and the optional input is the number N of object and the coordinate $O = \{x_1, x_2, y_1, y_2\}$ of the region to be detected. The overall system architecture is shown in Figure 1. FastQ queries video in three stages. First, if the coordinate O is not none, FastQ will crop each frame according to O. Second, FastQ adopts a difference detector to exclude similar frames. The Euclidean distance is used as metric of similarity. If the distance between adjacent frames is less than the threshold, the later frame will be labeled and excluded. Third, FDet is utilize to detect the remaining frames efficiently. Excluded frames will be labeled with the same information as similar frames. Assume that the prediction result of all frames is

$$Y = \{y_1, y_2, ..., y_m\}, \tag{1}$$

where

$$y_i = \{n_{i1}, n_{i2}, ..., n_{il}\}, i = 1, 2, ..., m, \tag{2}$$

l is the number of categories used in training FDet. n_{ij} indicates the number of objects of category j existing at frame i. If there is no object of category j in frame i, then $n_{ij} = 0$.

According to the prediction result Y, FastQ can calculate the index set of frames satisfying the query requirements as follows

$$F_{pre} = \{f_i | n_{iC} = N\}, \tag{3}$$

Finally, FastQ outputs the time periods that object C appear in the video according to F_{pre} and frame rate of input video. FastQ will save the detection information of each frame, including the category, confidence, quantity and coordinates of the object locally. So analysts can use more constraints to make fine-grained queries offline.

Practically, it is a very effective method to limit the region to be detected to improve the query efficiency and can be applied to many scenarios. For example, analysts can crop the video to the width of the road (assuming the video is shot at a fixed angle) when they only need to count cars on the road and don't care about pedestrians or trees on the sidewalk. In some cases, low-cost cropping can achieve several times the query speed.

Although some video query systems have achieved rapid speed, their query conditions are limited. For example, NoScope and Focus can only perform binary detection tasks of a single class (presence or absence of a target class). Compared with these methods based on image classification model, our

Algorithm 1: Difference detector algorithm

Input: $F = \{f_1, f_2, ..., f_m\}$: A set of frames
r: frame rate
t_{skip}: frame skipping time
δ: similarity threshold
Output: F_s: A set of standard frames

$f_s \leftarrow f_1$;
$f_{next} \leftarrow f_{s+r\times t_{skip}}$;
/* f_{next} is the next frame of f_s */
push f_s to F_s;
/* f_s is current standard frame */
while $f_{next} \neq none$ **do**
  $d = \|f_{next} - f_s\|$;
  /* the distance between f_s and f_{next} */
  if $d > \delta$ **then**
    $f_s \leftarrow f_{next}$;
    push f_s to F_s;
  end
  $f_{next} \leftarrow f_{s+r\times t_{skip}}$;
end

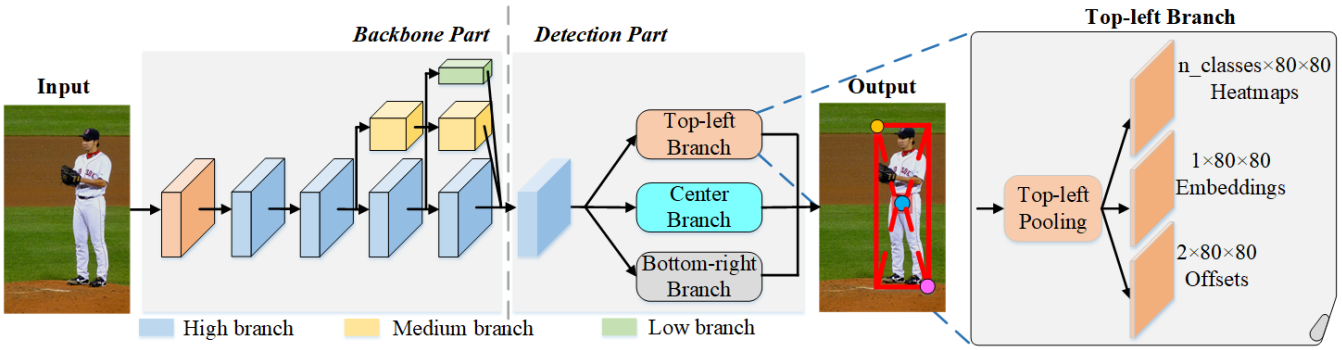

Figure 2: Architecture of FDet. The backbone part extracts feature map and feeds it to the detection part. The horizontal and vertical directions of the backbone part correspond to the depth of the network and the scale of the feature maps, respectively. The detection part consists of three branches, each of which outputs the corresponding heatmaps, embeddings and location offsets.

method supports multi-object query, quantity query and spatial information query.

3.2 Difference Detector

Videos have high temporal redundancy which means most consecutive frames are similar. Depending on the frame rates and shooting scenes, some objects can last from seconds to minutes without change. As a result, FastQ uses a standard frame to represent frames that do not change much over time.

The overall procedure of difference detector filtering standard frame is shown in Algorithm 1. Frame rate r is the number of frames per second (FPS). According to the different shooting conditions, the frame rate is also different. The commonly used frame rates are 25 and 30 FPS. We use a frame skip time t_{skip} to decide how often to perform difference detection. Therefore, the adjacent frames we defined are separated by $r \times t_{skip}$ frames. The rate of change between frames in different types of video is inconsistent. FastQ sets t_{skip} to 0.5 seconds, which basically does not miss the standard frame. The efficiency of query can be improved by increasing t_{skip}, but the accuracy will be lost in theory. FastQ's difference detector utilizes Euclidean distance to estimate the similarity between frames. If the Euclidean distance between the current standard frame and the adjacent frame is greater than the similarity threshold δ, the adjacent frame is taken as the next standard frame and pushed to the standard frame set. In addition, each non-standard frame will be labeled with an index of the nearest standard frame so that FastQ can label them with the same information as their corresponding standard frame.

Finally, the selected standard frames will be fed into FDet for object detection. FastQ's difference detector selects only 3% to 8% of the original frames as the standard frames, which is equivalent to accelerating video query up to 30 times.

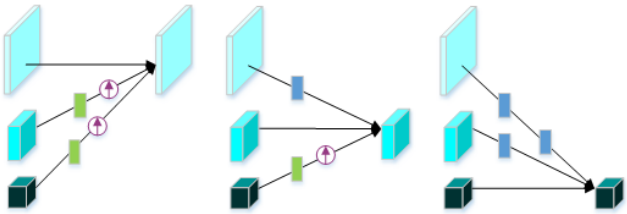

Figure 3: Structure of exchange unit. The arrows denote 2× up-sampling. The blue rectangles denote 3×3 convolution with stride 2 and green rectangles denote 1×1 convolution with stride 1. We employ exchange unit to aggregate the information for high, medium and low resolutions from the left to the right, respectively.

3.3 FDet: A Lightweight Keypoint-based Detector for Querying over Video

Backbone Part The backbone network extracts the feature map of the input image and has an important impact on the accuracy and efficiency. The backbone of FDet is carefully designed for two purposes: (1) design simple structure to achieve the tradeoff between accuracy and efficiency. (2) maintain high-resolution feature maps throughout the process and fusing feature maps across scales.

Inspired by the success of HRNet [20], we design a parallel multi-scale branches network architecture to obtain and integrate features across scales. As shown in Figure 2, FDet's backbone contains three parallel branches. We first generate a high-resolution branch, and then gradually add the other two high-to-low resolution branches one by one in parallel. In detail, The high, medium and low branches keep the feature maps of one quarter, one eighth and one sixteenth of the output size respectively. To exchange and integrate spatial information across scales, the exchange units are employed to ensure that each branch shares the spatial information with

Table 1: Evaluation results on COCO test-dev. Our network improves both the accuracy and efficiency by 0.9% AP and about 2× faster than the popular real-time detector YOLOv3.

Model	Backbone	Input	Times(ms)	AP	AP_{50}	AP_{75}
MobileNet-SSD [7]	MobileNet	300×300	29.1	19.3	-	-
MobileNetV2-SSDLite [19]	MobileNetV2	320×320	27.6	22.1	-	-
Pelee [21]	PeleeNet	304×304	30.1	22.4	38.3	22.9
SSD300 [12]	VGG-16	300×300	49.6	25.1	43.1	25.8
SSD321 [4]	ResNet-101	321×321	82.6	28.0	45.4	29.3
DSSD [4]	ResNet-101+FPN	321×321	98.8	28.0	46.1	29.2
ThunderNet [14]	SNet535	320×320	30.6	28.0	46.2	29.5
YOLOv3 [17]	Darknet-52	320×320	52.7	28.2	48.5	30.2
RefineDet512 [22]	ResNet-101	512×512	245.7	36.4	57.5	39.5
CornerNet (single-scale) [11]	Hourglass-104	511×511	1105.3	40.5	56.5	43.1
CenterNet (single-scale) [2]	Hourglass-104	511×511	1156.7	44.9	62.4	48.1
FDet	FDet's backbone	320×320	**25.3**	29.1	45.7	30.9

other branches. The exchange unit is shown in Figure 4. The feature map of different branches is aligned by the exchange unit and then fused by the addition operation.

At the end of the backbone, the medium and low resolution feature maps are rescaled to the shape of high-resolution, and then the low, medium and high resolution feature maps are added as the output of backbone. Compared with most of the 32× down-sampling, FDet's backbone can maintain and output the feature map with 4× down-sampling. Based on the above design, our backbone obtains multi-scale spatial information while maintaining high-resolution feature map.

In addition to designing a simple network structure, we also optimize the network modules to further reduce the complexity of the backbone. We adopt the cheap computing fire module proposed by SqueezeNet instead of residual block. Furthermore, inspired by the success of MobileNets, we replace the 3×3 standard convolution of the fire module with a 3×3 depth-wise separable convolution, which further reduces the calculation of the fire module.

Detection Part The backbone part extracts feature map and feeds it to the detection part. The detection part is motivated by CenterNet, we detects a center keypoint and a pair of corners of a bounding box. The detection part consists of three branches: the top-left branch, the bottom-right branch and the center branch. The top-left branch outputs heatmap, embedding and location offset of top-left corner by top-left pooling and a series of convolutions. In a similar way, the bottom-right branch outputs heatmap, embedding and location offset of bottom-right corner by bottom-right pooling and a series of convolutions. Note that the center branch only outputs heatmap and location offset of center keypoints by center pooling and a series of convolutions. The details of corner poolings and center pooling can be found in [2].

The heatmaps predict the probability that each pixel of the feature map belongs to a category. In our network, each set of heatmaps is of a quarter of the input image size and has C channels, where C is the number of categories (C=80 on COCO datasets). The embedding vectors are employed to determine if a pair of the top-left corner and bottom-right corner is from the same bounding box. The distance between the embedding vectors of two corners from the same object is small. To remap the corners and center keypoints from heatmaps to the input image, the location offsets recover part of the information lost in the down-sampling of the backbone.

3.4 Training Detail

We use the same training losses of CornerNet to train FDet. FDet is trained with no pre-training on any external dataset. We train the FDet with learning rate of 2.5×10^{-4} for the first 450K iterations and then continue training 30K iterations with learning rate of 2.5×10^{-5}. The network is trained on two 1080Ti (11GB) GPUs and use a batch size of 32.

4 EXPERIMENTS

4.1 FDet Results

Results on COCO We evaluate FDet on popular COCO benchmark, which has 80 objects categories. We use the trainval35k (union of 80k training set and 35k large-val set) for training and test the results on 5k mini-val set. We also test the final results of our network on COCO test-dev set, which has no public labels and requires use of the evaluation server. Figure 4 visualizes several examples on COCO validation dataset.

Table 1 shows the comparison with state-of-the-art lightweight detectors on COCO test-dev dataset. Experiments are tested on the same machine with a 1080Ti GPU and an Intel E5-2680v4 CPU. Our network achieves 29.1% AP at 25.3ms for a 320×320 input image. Compared with the state-of-the-art detectors, FDet achieves a better trade-off between accuracy and efficiency. In particular, we improve both the accuracy and efficiency by 0.9% AP and about 2× faster than the popular real-time detector YOLOv3. ThunderNet is one of the latest real-time detector, it also tried to design a lightweight backbone to improve the efficiency of backbone. However, it is limited to the post-processing related to the anchor boxes, and its speed is 14% lower than FDet. These

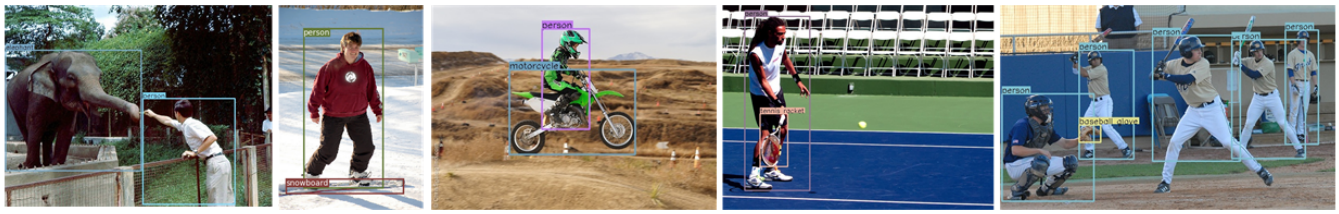

Figure 4: Some qualitative detection results of FDet on the MS-COCO validation dataset.

Table 2: Comparison of lightweight backbones. +SSD means to use the union of backbone and the detection part of SSD for experiment on COCO test-dev. +Kb means to use the union of backbone and the keypoint-based detection part for experiment on COCO test-dev.

Model	Top-1	Top-5	AP	Time
MobileNetV2 (+SSD)	67.1	84.6	25.7	35.6
ShuffleNetV2 (+SSD)	68.6	85.8	24.7	42.9
Xception [1]	65.9	83.1	-	40.5
FDet's backbone (+SSD)	**72.2**	**90.5**	**26.1**	**13.8**
MobileNetV2 (+Kb)	-	-	24.2	-
ShuffleNetV2 (+Kb)	-	-	23.5	-
FDet's backbone (+Kb)	-	-	**29.1**	-

results firmly demonstrate the effectiveness of our network. The best performance CenterNet is AP 44.9%, dramatically surpassing all the published one-stage approaches. CenterNet shows that the anchor-free method has some advantages in accuracy, but inference time is its major drawback.

Effectiveness of backbone network We compare FDet's backbone with other lightweight backbones on ImageNet classification dataset and COCO test-dev dataset. On ImageNet dataset, each network is trained with identical settings and tested at 256×256 and inference times are measured at 256×256 without detection part. Our backbone achieves 72.2% Top-1 accuracy and 90.5% Top-5 accuracy at 13.8ms. As shown in the Table 2, FDet's backbone is significantly faster and more accurate than other lightweight backbones. In detail, our backbone is has a 5.2% higher top-1 accuracy than ShuffleNetV2 and 3.1× faster. These results strongly show that FDet's backbone is superior to other popular lightweight networks in efficiency and performance.

An ablation study is also given in the Table 2 to analyze the contribution of the backbone part and the detection part. We test and compare the unions of {ShuffleNetV2, MobileNetV2, FDet's backbone}×{the detection part of SSD, the keypoint-based detection part} on COCO test-dev. By comparing ShuffleNetV2+SSD, MobileNetV2+SSD and FDet's backbone+SSD, the results report that when we use SSD as the detection part, we improve 0.4% AP(from 25.7% of MobileNetV2+SSD to 26.1% of FDet's backbone+SSD). This tells us that our backbone has certain advantages even in anchor-based detection. By comparing ShuffleNetV2+Kb, MobileNetV2+Kb and FDet's backbone+Kb, the results suggest that FDet's backbone improve the AP by 20.2% (from 24.2% of MobileNetV2+Kb to 29.1% of FDet's backbone+Kb) when we use keypoint-based detection part. The greater improvement than take SSD as detection part shows that FDet's backbone is more suitable for keypoint detection than other popular lightweight backbone. These results demonstrates that our backbone and the keypoint-based detection part are complementary to each other.

Table 3: Video datasets.

Video name	Object	Resol.	FPS	Time(mins)
traffic_1	car	720p	30	120
traffic_2	bus	720p	30	120
traffic_3	person	720p	30	120
Docu._1	elephant	720p	30	102
Docu._2	dog	720p	30	100
film	car	1024	23	90

4.2 FastQ Results

Video Datasets. We evaluate FastQ on six videos as shown in the Table 3, including three traffic cameras (Auburn_1, Auburn_2 and Jacksonh), two documentaries (*Elephant Family and Me* and *Secrets of Dogs*) and one film (*Flipped*). Traffic cameras are two hours of video we've captured from YouTube's live webcams.

Baseline. We take CenterNet as our ground-truth CNN (GT-CNN) and employ it to generate correct answers. We extract one frame of video every 0.5 seconds (extract one frame every 12 frames for *Flipped*, and one frame every 15 frames for others) for detection and save the detection results as the correct answer. In addition, the speed at which the GT-CNN generates the correct answers is taken as our baseline speed.

Metrics. We use two metrics to evaluate FastQ. The first metric is query speed-up ratio (QSR). Denoting that the GPU time taken by GT to query an object class over video is t_{gt}, and FastQ to query an object class over video is t_{pre}. The QSR is t_{gt}/t_{pre}. The second metric is accuracy. We evaluate every half second whether the prediction is correct. Denote the time of target video is $n \times 0.5$ seconds. For each of n, if the prediction results of more than 90% of the frames are the same as the ground truth, we consider that the prediction is

Table 4: QSR and accuracy results of FastQ. Acc.: accuracy. Dif-D: difference detector

Name	QSR	Acc.	QSR of Dif-D	QSR of FDet
traffic_1	225×	95.2%	15×	15×
traffic_2	300×	96.5%	20×	15×
traffic_3	150×	93.1%	10×	15×
Docu._1	255×	95.7%	17×	15×
Docu._2	270×	91.6%	18×	15×
film	255×	94.8%	17×	15×

correct. Denote the number of correct predictions is $n_{correct}$. We calculate the query accuracy as $n_{correct}/n$.

QSR and Accuracy Results. The QSR and accuracy results of FastQ are shown in the Table 4. FastQ achieves 150× to 300× QSR while maintaining more than 90% accuracy. The experimental results show that FDet contributes 15× QSR, which means that the processing speed of naive FDet is 15 times of GT-CNN. Difference detector contributes 10× to 20× QSR. Because the traffic cameras are fixed angle, the distance threshold should be adjusted according to the size of the object. For fixed angle videos, the QSR of the difference detector depends on the size of the object to be detected. For documentaries, films, which are non-fixed angle videos, the influence of object size on QSR of difference detector can be almost ignored.

In summary, the difference detector improves the query efficiency of FastQ by 10× to 20× by eliminating similar redundant frames. Furthermore, lightweight FDet improves the query speed by 15× by optimizing the inference speed of object detection.

5 CONCLUSION

Video is rich in semantic information and growing explosively at scale. Unfortunately, it is slow and expensive to process video data with high accuracy CNN-based object detectors. In this paper, we propose a system named FastQ for efficient querying over video at scale. Given a target video, FastQ can automatically label the category and number of objects for each frame. We adopt a difference detector to filter standard frames, which greatly reduces the number of frames. To improve the efficiency of video query, we introduce a lightweight object detector named FDet for efficient detection. Compared with state-of-the-art real-time detectors, FDet achieves superior performance with 29.1% AP on COCO benchmark at 39 FPS. Experiments show that FastQ achieves 150× to 300× speed-ups while maintaining more than 90% accuracy in video queries.

ACKNOWLEDGMENTS

This research project is supported by the National Natural Science Foundation of China (Grant No: 61303029), National Social Science Foundation of China(Grant No: 15BGL048) and Hubei Province Science and Technology Support Project(2015BAA072).

REFERENCES

[1] F. Chollet. 2017. Xception: Deep learning with depthwise separable convolutions. In *CVPR*. 1251–1258.
[2] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian. 2019. Centernet: Keypoint triplets for object detection. In *ICCV*. 6569–6578.
[3] C. Eggert, D. Zecha, S. Brehm, and R. Lienhart. 2017. Improving small object proposals for company logo detection. In *ICMR*. 167–174.
[4] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C Berg. 2017. Dssd: Deconvolutional single shot detector. *arXiv preprint arXiv:1701.06659* (2017).
[5] R. Girshick, J. Donahue, T. Darrell, and J. Malik. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*. 580–587.
[6] K. He, G. Gkioxari, P. Dollár, and R. Girshick. 2017. Mask r-cnn. In *CVPR*. 2961–2969.
[7] A. Howard, Me. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861* (2017).
[8] K. Hsieh, G. Ananthanarayanan, P. Bodik, S. Venkataraman, P. Bahl, M. Philipose, P. Gibbons, and O. Mutlu. 2018. Focus: Querying large video datasets with low latency and low cost. In *OSDI*. 269–286.
[9] D. Kang, P. Bailis, and M. Zaharia. 2018. Blazeit: Fast exploratory video queries using neural networks. *arXiv preprint arXiv:1805.01046* (2018).
[10] D. Kang, J. Emmons, F. Abuzaid, P. Bailis, and M. Zaharia. 2017. Noscope: optimizing neural network queries over video at scale. *arXiv preprint arXiv:1703.02529* (2017).
[11] H. Law and J. Deng. 2018. Cornernet: Detecting objects as paired keypoints. In *ECCV*. 734–750.
[12] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C Berg. 2016. Ssd: Single shot multibox detector. In *ECCV*. 21–37.
[13] A. Newell, K. Yang, and J. Deng. 2016. Stacked hourglass networks for human pose estimation. In *ECCV*. 483–499.
[14] Z. Qin, Z. Li, Z. Zhang, Y. Bao, G. Yu, Y. Peng, and J. Sun. 2019. Thundernet: Towards real-time generic object detection. *arXiv preprint arXiv:1903.11752* (2019).
[15] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. 2016. You only look once: Unified, real-time object detection. In *CVPR*. 779–788.
[16] J. Redmon and A. Farhadi. 2017. YOLO9000: better, faster, stronger. In *CVPR*. 7263–7271.
[17] J. Redmon and A. Farhadi. 2018. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767* (2018).
[18] S. Ren, K. He, R. Girshick, and J. Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*. 91–99.
[19] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*. 4510–4520.
[20] K. Sun, B. Xiao, D. Liu, and J. Wang. 2019. Deep high-resolution representation learning for human pose estimation. In *CVPR*. 5693–5703.
[21] R. Wang, X. Li, and C. Ling. 2018. Pelee: A real-time object detection system on mobile devices. In *NIPS*. 1963–1972.
[22] S. Zhang, L. Wen, X. Bian, Z. Lei, and S. Li. 2018. Single-shot refinement neural network for object detection. In *CVPR*. 4203–4212.
[23] Y. Zhang and A. Kumar. 2019. Panorama: a data system for unbounded vocabulary querying over video. *VLDB* 13, 4 (2019), 477–491.
[24] X. Zhou, J. Zhuo, and P. Krahenbuhl. 2019. Bottom-up object detection by grouping extreme and center points. In *CVPR*. 850–859.