CaTDet: Cascaded Tracked Detector for Efficient Object Detection from Video

Anonymous Authors[1]

Abstract

Detecting objects in a video is a compute-intensive task. In this paper we propose CaTDet, a system to speedup object detection by leveraging the temporal correlation in video. CaTDet consists of two DNN models that form a cascaded detector, and an additional tracker to predict regions of interests based on historic detections. We also propose a new metric, mean Delay(mD), which is designed for latency-critical video applications. Experiments on the KITTI dataset show that CaTDet reduces operation count by 5.1-8.7x with the same mean Average Precision(mAP) as the single-model Faster R-CNN detector and incurs additional delay of 0.3 frame. On CityPersons dataset, CaTDet achieves 13.0x reduction in operations with 0.8% mAP loss.

1 Introduction

We consider the task of detecting objects in a video. Video is an important data source for real-world vision tasks such as surveillance analysis and autonomous driving. Such tasks require detecting objects in an accurate and efficient manner.

Processing video data is compute-intensive. A $50 camera can generate 1080p video stream at 25fps, while a $1000 Maxwell Titan X with SSD512 algorithm can only run object detection at 19 fps (Liu et al., 2016a).

One approach to reducing the computational workload of video processing is to exploit the temporal and spatial locality of video: treating it as a sequence rather than running detection on each image separately. For most videos, similar objects appear in adjacent frames(temporal locality) and in the nearby locations(spatial locality). We can exploit this locality to make detection more efficient.

We propose CaTDet, a framework for video detection. CaTDet stands for Cascaded Tracking Detector, which incorporates the tracker into a cascaded system to exploit locality in video. It is designed for, but not limited to, moving-camera delay-sensitive scenarios such as autonomous driving. As shown in Figure 1, CaTDet is a detector cascade with temporal feedback. The tracker and the inexpensive proposal network extract the interesting regions in an image, which reduces the workload on the expensive refinement network.

For single-image object detection algorithms, many detection accuracy metrics have been proposed, e.g., Average Precision (Everingham et al., 2010). These metrics do not account for the temporal characteristics of a video. For many video applications, such as autonomous driving, the metric that matters is *delay*, the time from when an object first appears in a video to when it is detected.

Our contribution in this work is two-fold: the delay metric for object detection in video and CaTDet, a detection system to efficiently detect objects with the aid of temporal information. We evaluate CaTDet on KITTI dataset, with both the traditional mAP metric and our delay metric. The results show that CaTDet is able to achieve 5.1-8.7x speed-up with no loss of mAP and only a small increase in delay.

2 Related Work

Object detection from video. Others have also exploited the temporal properties of video to improve object detection. T-CNN (Kang et al., 2016) regularizes detection results with tracking output. In Detect and Track (Feichtenhofer et al., 2017), an end-to-end architecture is proposed which incorporates detection and tracking to improve accuracy. Both methods require future frames to predict current frame, therefore they are non-causal. Deep feature flow (Zhu et al., 2017c) exploits temporal redundancy between neighboring frames by estimating feature flow with FlowNet (Ilg et al., 2017). It achieves high speed-up with minor loss of accuracy by skipping feature extraction for non-key frames. Flow-guided feature aggregation (Zhu et al., 2017b), on the other hand, tries to improve accuracy by aggregating features from nearby frames. In (Zhu et al., 2017a), a unified approach of exploiting feature flow is presented, which improves both the detection accuracy and the speed.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the Systems and Machine Learning (SysML) Conference. Do not distribute.

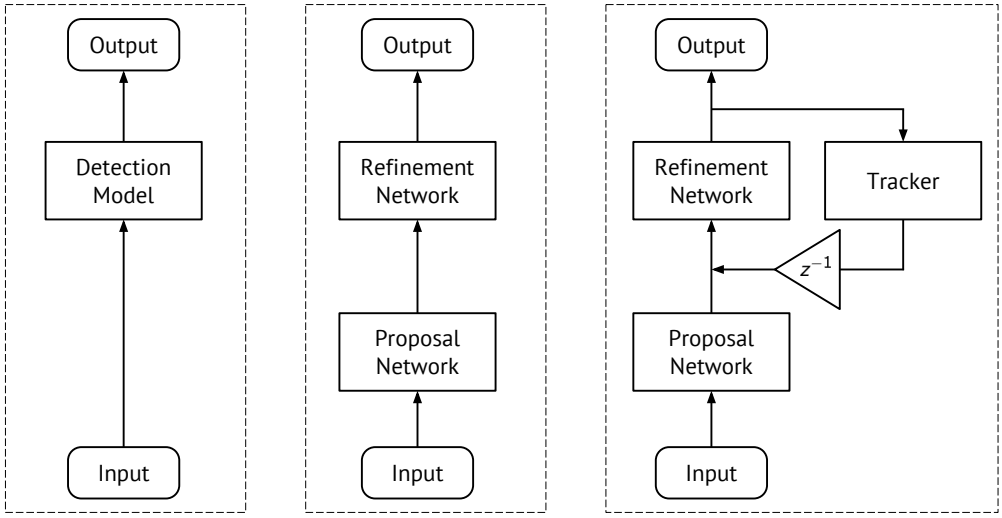


Figure 1. (a) Single-model detector system: the detection network could either be Faster R-CNN, SSD, etc. (b) Cascaded detector system: A cheap proposal network scans the whole images and produces potential object candidates to be calibrated by refinement network. Both the proposal network and refinement network are detection models, although technically the proposal network does not need to be. (c) CaTDet: a tracker is added to incorporate previous frames' information to aid detection and save computations.

Region extraction and selected execution. For videos with static backgrounds, only the detections on the changing regions require updates. CBinfer (Cavigelli et al., 2017) filters out moving pixels by a change-based thresholding. In (Zhang et al., 2017b), foreground extraction methods are employed to extract regions of interest. After that, both methods run CNN models only on the selected regions. However, they do not work in the general case of a moving camera.

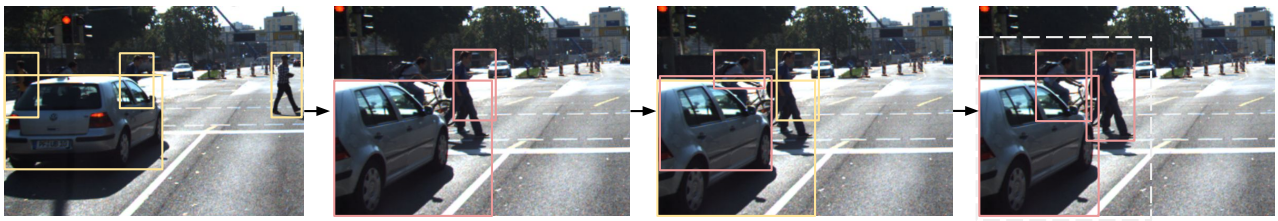
Adaptive/cascaded neural network. Others also use a system-level approach to improve computation efficiency of Deep Neural Network. In Adaptive Neural Networks (Bolukbasi et al., 2017), a system is proposed that adaptively chooses the parts of a neural network model according to the input. SACT (Figurnov et al., 2017) further develops this work (Bolukbasi et al., 2017) by adaptively extracting regions with higher possibility to contain objects and selectively feeding these regions into a deeper network. While these works improve computational efficiency, they are designed for still image object detection which does not take advantage of temporal locality. NoScope (Kang et al., 2017) proposes a framework with selective processing and cascaded models that can speedup frame classification in video up to 1000x. However, the task of NoScope is relatively simple and the input is limited to static background videos.

3 Proposed Detection System

Figure 1 illustrates a single-model detection system, a cascaded system, and CaTDet. The single-model system is a single detection DNN such as Faster R-CNN (Ren et al., 2015) or SSD (Liu et al., 2016b). The cascaded model consists of two detection DNNs, a lightweight "proposal network" followed by a higher-precision "refinement network". The proposal network selects regions of interest for the refinement network to reduce computation. Our proposed system, CaTDet, improves on the cascaded system by using a tracker to exploit the temporal correlation of a video stream.

CaTDet consists of a proposal network, a refinement network, and a tracker. The proposal network, which may be small and inaccurate, takes every full video frame and outputs the potential locations of objects within the frame. The tracker tracks the history of high-confidence objects in the current frame predicting their location in the next frame. For each frame, the outputs of the proposal network and the tracker are combined to select the regions of interest examined by the refinement network, which is more accurate and complex, to obtain calibrated bounding boxes and classes.

CaTDet is based on the concept that validation and calibration are easier than re-detection. An object that has been detected in one video frame is very likely to appear in the next frame at a nearby location. Based on this prior, running a high-precision detector on regions around previous objects saves workload while preserving accuracy. This



(a) (b) (c) (d)

Figure 2. A clip of CaTDet's execution loop. Each step's new output is in red, while the output of previous steps is in yellow. (a) Last frames detections. (b) Locations of the pre-existing objects are predicted by the tracker. Objects leaving the current frame are deleted. (c) New detections from the proposal network are added. (d) Refinement network performs selected calibration, as shown in the dotted box. Duplicated detections will be deleted the NMS step.

simple concept, however, faces two problems in practice:

New objects. It is difficult to predict where new objects will appear in a new video frame. To detect them requires scanning the entire image. We use a fast, inaccurate detector, the proposal network, to propose potential objects. As we have a more accurate detector to refine the results, a high false positive rate can be tolerated. In functionality, the proposal network is very similar to the region proposal network(RPN) in Faster R-CNN.

Motion and occlusion. The movement of objects or the camera can add to the difficulty for the refinement network. In some cases due to occlusion, objects may be temporally invisible. If the refinement network simply looks at previous locations, minor mismatches or temporal misses could lead to permanent loss. Therefore, we use a tracker to predict the future locations of objects based on history. For the tracker, standard tracking algorithms can be directly applied, except that the output is predicted locations instead of tracklets.

An example of CaTDet's workflow is given in Figure 2. Starting with the last frame's detection results in stage *a*, the tracker updates its internal state and predicts the corresponding locations/proposals for the next frame as in stage *b*. Together with the proposal net's outputs in stage *c*, the proposals are fed into the refinement network for selected calibration. As shown in stage *d*, the refinement network only operates in the regions of interest.

4 Implementation Details

In this section we present a detailed description of each module of CaTDet.

4.1 Tracker

Here the tracker behaves almost the same as the ones for object tracking, except that we are interested in its predicted objects for the next frame. In short, typical tracking algorithms output tracked sequences of detected objects, while our tracker outputs location predictions which were usually intermediate results. The location predictions are then fed into the refinement network to aid object detection.

Our tracking algorithm is inspired by SORT(Simple Online and Realtime Tracking) (Bewley et al., 2016). As in SORT, two major components of the algorithm are object association and motion prediction. Object association matches the objects in two(or more) adjacent frames. Motion prediction uses the states of existing tracked objects to predict their locations in future frames. These two steps are executed iteratively as new frames come.

For object association, our tracker adopts the modified Hungarian algorithm, just as SORT does, to match objects in two adjacent frames. For an N-to-M matching problem, we construct an N-by-M cost matrix of which the elements are minus IoUs of bounding boxes. For IoUs that are smaller than a threshold β, the two bounding boxes are set as non-relevant regardless of Hungarian algorithm's results. The output of object association is matched objects, lost objects (unmatched objects in the previous frame) and emerging objects (unmatched objects in the new frame). Notice that this process is performed one time per class.

To make the tracker robust to different frame rates and resolutions, our motion prediction module adopts the simple exponential decay instead of the Kalman Filter algorithm in SORT. For Kalman Filter, the parameters need to be carefully tuned on the training dataset (Bewley et al., 2016), while in our system they are set as constants even at different frame rates.

In our motion prediction module, the state of an object is represented by two vectors $\mathbf{x} = [x, y, s]$, $\dot{\mathbf{x}} = \dot{x}, \dot{y}, \dot{s}]$ and a scalar r. Here x and y are the center coordinates, s is the width of the bounding box and r is the aspect ratio(height to width). The following update rule is adopted in the tracker:

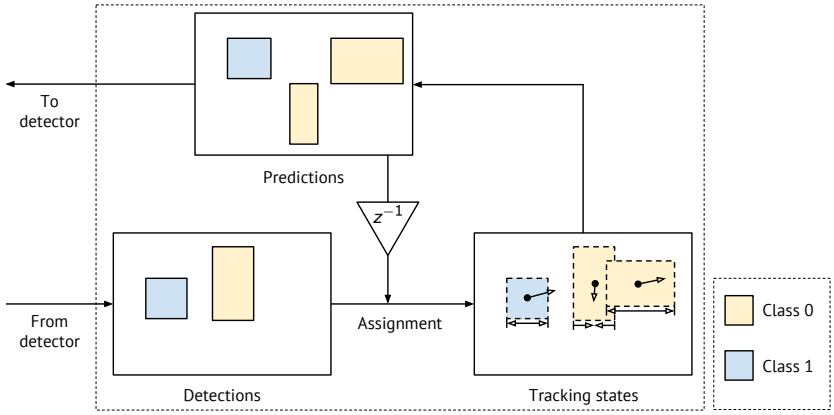


Figure 3. Tracker work flow. The tracker takes as input the current frame's detected objects and outputs the predicted locations in the next frame. The predicted objects and detected objects of the current frame are matched to update the internal state of the tracker, which includes the coordinates, dimensions and motions of objects per class.

$$\dot{\mathbf{x}}_{n+1} = \eta\dot{\mathbf{x}}_n + (1-\eta)(\mathbf{x}_{n+1} - \mathbf{x}_n) \quad (1)$$

$$\mathbf{x}'_{n+1} = \mathbf{x}_n + \dot{\mathbf{x}}_n \quad (2)$$

$$r'_{n+1} = r_n \quad (3)$$

Here $\mathbf{x}'$ and r' are the predicted values, which are later converted into bounding box coordinates and fed into the refinement network. If the object is not matched in the new frame (missing object), the motion is kept constant for several frames until it is discarded. For emerging objects, the motion vector is initialized as $\mathbf{0}$.

In our implementation, the IoU threshold β is set to 0 and decay coefficient η to 0.7, although the tracker is robust to a wide range of η.

The tracker operates fairly efficient. Our experiments on KITTI dataset show that it is able to reach 1082fps with single-thread Intel E5-2620 v4. On the other hand, the number of predicted objects from tracker largely affects the workload of refinement network. We filter out the objects that are too small(width smaller than 10 pixels) or have been largely chopped by the boundary. Instead of tracking the missing objects for a fixed number of frames, we use an adaptive scheme that every match adds to confidence with a upper limit and every miss reduces confidence. Once the confidence value goes below zero, the object is discarded. Compared with the original tracking algorithm, our tracker is optimized to reduce the number of predictions to save total operations of the system.

4.2 Proposal network

Proposal network and refinement network are basically detectors of different sizes and accuracy. Proposal network tends to be much smaller than refinement network, thus they together form a cascaded detector to save operations. For simplicity, we use Faster R-CNN (Ren et al., 2015) for both proposal network and refinement network in CaTDet. Faster R-CNN was proposed in 2015, yet still serves as a good baseline detection framework for object detection.

For proposal network, we adopt the standard Faster R-CNN settings. The image is first processed by Feature Extractor which shares the same convolutional layers as the classifier model. Region Proposal Network(RPN) predicts 3 types of anchors with 4 different scales for each location. After Non-Maximum Suppression(NMS), 300 proposals are selected to fed into the classifier. Also, proposal network does not necessarily predict class labels, though in our case the labels are produced but unused.

Four different types of simple ResNet models are used in our experiments, including the standard ResNet-18, which in our case is the largest model.

Table 1. Model specifications for proposal nets. In ResNet-18, all blocks are repeated 2 times. The number of arithmetic operations is measured with Faster R-CNN on KITTI dataset, of which the input image size is 1242x375 and the number of proposals is 300.

	ResNet-18	ResNet-10a	ResNet-10b	ResNet-10c
conv1	64	48	32	24
block1	64(x2)	48	32	24
block2	128(x2)	96	64	48
block3	256(x2)	168	128	96
block4	512(x2)	512	256	192
ops	138.3G	20.7G	7.5G	4.5G

4.3 Refinement network

The refinement network runs a modified version of Faster R-CNN. Two major differences are listed as follows:

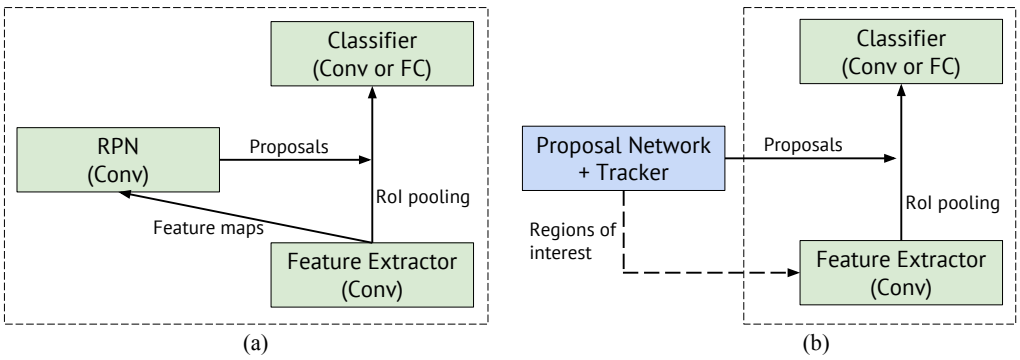


Figure 4. Inference-time work flows of Faster R-CNN and refinement network. (a) Standard Faster R-CNN detector: the RPN takes the feature maps from the feature extractor and . (b) Faster R-CNN detector for the refinement network: the proposals from the proposal network and the tracker instruct the feature extractor to only compute features on regions of interest. Regions of interest is a mask of all proposals over the frame.

Selected regions of features. The proposal network and the tracker together provide a high-recall region selection, so the refinement network only needs the feature maps that are corresponding to the proposals. A margin of 30 pixels is appended around the proposals to maintain enough information for the ConvNet. Here we are interested in the real number of operations needed to extract required features, which is platform-independent, therefore the regions-of-interest are not required to be rectangular. Some works (Zhang et al., 2017b) merge regions into rectangles to maximize the computational efficiency on GPU.

Reduced number of proposals. The typical number of proposals from RPN is 300 (Ren et al., 2015). In CaTDet, however, it is usually much smaller, as we observed that the proposal network and tracker combined are much more accurate than RPN. That helps to save computations for models such as ResNet-50, of which the classifier is a relatively heavy ConvNet.

Apart from its own hyper-parameters, the workload of the refinement net is also affected by the hyper-parameters of the tracker and the proposal net. There are two that significantly affect the inference speed – T-thresh and C-thresh, the confidence thresholds for the tracker's input and proposal network's output, respectively. Due to the fact that proposal net and tracker predict overlapping proposals, their impact on total system operations is coupled. Increasing either threshold will decrease the total number of operations of CaTDet, at a potential risk of hurting accuracy.

5 Evaluation Metrics

We selected two metrics to evaluate our detection results, mean Average Precision (mAP) and mean Delay(mD).

Average Precision (AP) is a common metric to measure the overall accuracy of a detector. It is defined as the integral of precision over recall. For convenience, precisions at discrete recall values are usually averaged to approximate the integral. As an example, in Pascal VOC (Everingham et al., 2010), 11 recall values ranging from 0 to 1.0 are selected to calculate AP. Mean Average Precision is simply the arithmetic mean of all different classes' AP values. mAP is comprehensive enough to capture a detector's accuracy regarding the precision-recall trade-off, therefore makes it the standard evaluation metric in most detection benchmarks (Everingham et al., 2010; Deng et al., 2009; Lin et al., 2014).

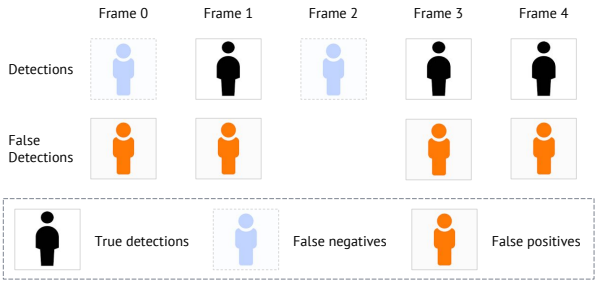


Figure 5. Illustration of how recall, precision and delay correspond to groudtruths and detections. One ground-truth object spans 5 frames in this example. There are in total 7 detections (3 true detections and 4 false positives) and 2 false negatives. Only the single false negative in Frame 0 counts towards delay. Recall = 3/5. Precision = 3/7. Delay = 1.

mAP is designed in such a way that all detections are weighed equally, which is the typical case for single image detection. However, we argue that not all detections are equal in the video detection problem. Given detections of a single object in a video sequence, the first detection is particularly important as it determines the response time.

For latency-critical scenarios like autonomous driving, it is essential to have a low response time. For applications that are not sensitive to detection latency, in-sequence detection errors can be fixed with post-processing, while detection errors in the beginning are much more difficult to fix.

The delay metric is proposed to emphasize early detection of an object. By its definition, delay simply means the number of frames from the beginning frame of an object sequence to the first frame the object is detected. An example is given in Figure 5, where the relationship between recall, precision and delay is illustrated. By averaging the delay per class, we obtain mean Delay(mD) which is still measured in number of frames. Notice that for simplicity, the delay metric does not consider the frame rate of the video steam.

It is easy to define and measure the delay, however, it is difficult to compare the average delay across different methods. The reason is that one can always reduce the delay by detecting as many objects as possible. The AP metric makes a trade-off between false negatives in recall and false positives in precision, while the delay metric only penalizes false negatives as shown in Figure 5.

To fairly compare the average delay of different methods, we measure the delay at the same precision level. To be specific, given a target precision value β, we select the confidence threshold so that the mean precision of all classes matches β. With one threshold for a specific method, we then measure the average delay.

$$mD@\beta := \frac{1}{|C|} \sum_{c \in C} Delay_c(t_\beta) \,, \tag{4}$$

where C is the set of classes and t_β is chosen as

$$\frac{1}{|C|} \sum_{c \in C} Prec_c(t_\beta) = \beta \tag{5}$$

The above formula only describes the entry latency. Some applications may also be sensitive to exit latency, which is defined as the actual exit frame minus the predicted exit frame. Due to the fact that entry frame is harder to predict as no prior knowledge exists, we are focusing on entry latency throughout this paper.

6 Experiments on KITTI

6.1 Dataset overview

KITTI is a comprehensive dataset with multiple computer vision benchmarks related with the autonomous driving task. Out of the available benchmarks, we use the data from the 2d object benchmark and the tracking benchmark for training and evaluation of CaTDet, respectively. KITTI's 2d object (detection) benchmark contains 7481 training images. The tracking benchmark contains 21 training sequences with a total number of 8008 frames at a frame rate of 10 fps. Both the detection and tracking datasets contain a relatively small number of images, compared with ImageNet (128k images) and MS-COCO (83k images).

In our experiments the models are trained on the 2d object dataset and evaluated on tracking dataset. We found that overlapping images exist in the 2d object dataset and tracking dataset, therefore we filtered out the duplicated images in the training set. After that there are 4383 remained images.

We evaluate our results on KITTI dataset using the official evaluation codes and procedure as described in the KITTI paper (Geiger et al., 2012). While KITTI's object detection subset evaluates three classes (*Car*, *Pedestrian*, and *Cyclist*) among 8 types of objects, we only evaluate *Car* and *Pedestrian* classes in accordance to the tracking subset. An overlap of 50% is required for a valid detection of *Pedestrian*, while the class *Car* requires at least 70%. The open-sourced KITTI development kit can be found via the following link[1].

It should also be noticed that in the official KITTI evaluation protocol, there are three difficulty levels (*Easy*, *Moderate*, *Hard*). Each difficulty level sets a specific threshold of bounding box size, occlusion level and truncation for a valid groundtruth. For example, in *Easy* mode only groundtruth objects that are wider than 40 pixels and fully visible are evaluated, otherwise they do not count towards false negatives. In our experiments, we find that the *Easy* mode does not distinguish different methods, therefore we do not list its results in the following sections.

6.2 Training procedure

Due to the fact that the training set is relatively small compared with the common datasets used for Deep Neural Networks training, we adopt a 3-stage training pipeline for the Faster R-CNN detector models. Similar to other detection frameworks, our model is first pretrained on the ImageNet classification dataset. The classification model (all layers minus the last classifier layer), together with the RPN module, is then trained with Faster R-CNN framework on MS-COCO detection dataset. Finally we finetune the Faster R-CNN model on the target dataset.

Such a long training pipeline comes with a number of hyperparameters. In our experiments, we aim to keep the settings as simple as possible and ensure that all the models are fairlycompared, therefore we use the default settings for both ImageNet pretraining and MS-COCO detection training. The ImageNet pretraining follows the 90-epoch training

[1] KITTI development kit

Table 2. Comparison on KITTI dataset with two difficulty modes, *Moderate* and *Hard*. (Res10-a, Res50) stands for ResNet-10a as the proposal net and ResNet-50 as the refinement net. mD@0.8 indicates that the delay is tested at an average precision of 0.8.

System	ops(G)	mAP(*Moderate*)	mAP(*Hard*)	mD@0.8(*Moderate*)	mD@0.8(*Hard*)
Res50, Faster R-CNN	254.3	0.812	0.740	2.6	3.3
Res10a, Res50, Cascaded	43.2	0.807	0.733	3.2	3.8
Res10a, Res50, CaTDet	49.3	0.814	0.740	2.9	3.7
Res10b, Res50, Cascaded	23.5	0.787	0.730	4.7	5.7
Res10b, Res50, CaTDet	29.3	0.815	0.741	3.3	4.1

scheme in PyTorch examples[2]. The MS-COCO detection training follows the open-sourced PyTorch implementation of Faster R-CNN[3].

For KITTI finetuning, the same settings as for MS-COCO are adopted except that we set the number of iterations in proportion to the size of the training set. All models are trained for 25k iterations.

6.3 Overall results

Table 2 compares the mAP, mean Delay and operations between a single-model Faster R-CNN detector and a cascaded detector and two CaTDet systems with different configurations. Both CaTDet systems achieve no mAP loss compared with the single-model detector, while saves 5.15x and 8.67x operations. The cascaded system, though saves slightly more operations compared with its counterpart, loses 0.5% to 0.7% mAP. In our later experiments it is shown that this gap cannot be mitigated even with further increasing the number of proposals as well as operations.

Here we only consider the arithmetic operations in convolutional layers and fully-connected layers. The tracker and the other layers in DNN models are relatively negligible in terms of either time or operations.

In terms of delay, CaTDet is slightly worse than the single-model detector. CaTDet-A(proposal net: ResNet-10a; refinement net: ResNet-50) incurs an additional delay of 0.3-0.5 frame compared with the single-model Faster R-CNN detector. An interesting observation is that, CaTDet-B(proposal net: ResNet-10b; refinement net: ResNet-50) with a smaller proposal network and fewer operations can still match the original mAP results. Its delay results, however, are much worse. It indicates that the delay statistics is more sensitive to a bad proposal network.

6.4 System analysis

Number of operations break-down. We counted the number of arithmetic operations and present the results in Table 3. Notice that we also show a further break-down of operations in refinement network. Due to the overlaps of proposals from tracker and proposal net, the sum of refinement net's break-downs is larger than its actual number of operations.

It is also noticed that with ResNet-10b as proposal net, the refinement network is already dominant. Therefore, further reducing the size of proposal network, like using ResNet-10c, is not meaningful.

Analysis of proposal network. We provide an analysis of the proposal network's role in CaTDet. Table 4 shows different choices of proposal network with their single-model accuracy and system accuracy. Here ResNet-18 is listed just for comparison purpose. Due to its large size, it is never a serious system design choice.

Considering the great differences of the single-model Faster R-CNN mAP (ranging from 0.542 to 0.687), these four models provide almost identical mAP (0.740-0.742) when acting as the proposal net in CaTDet. On the other hand, a better proposal net makes CaTDet substantially better in the delay metric. CaTDet with ResNet-18 achieves 16% less delay compared with ResNet-10c. These observations suggest that mAP is not sensitive to the choice of the proposal net, but delay is.

Importance of refinement network. Refinement network's accuracy largely determines the overall accuracy of CaTDet. As shown in Table 5, CaTDet's mAP and delay are very close to the single-model accuracy of the refinement network. It is also noticed that for a less accurate model like ResNet-18, the CaTDet system slightly surpasses the single-model Faster R-CNN, probably due to a better source of proposals.

Ablation study of the tracker. We study how important the tracker is in the CaTDet system. The overall results on KITTI listed in Table 2 seem to indicate that eliminating the tracker only incurs minor accuracy loss. However, our experiments show that without the tracker, even if we further increase the number of proposals from proposal net, the accuracy drop cannot be compensated.

As shown in Figure 6, in the cases where the tracker exists, varying the C-thresh makes little difference to mAP. In the cases without the tracker, none of the cascaded system can

[2] https://github.com/pytorch/examples
[3] https://github.com/ruotianluo/pytorch-faster-rcnn

Table 3. Operation break-down, including further break-down of refinement network. Refinement network only operates on proposed regions from tracker and proposal net. Due to the fact that there are overlaps among these two sources, the refinement net's operation break-down does not sum up to the total operations. Unit: Gops

System	Total	Total break-down		Refinement break-down	
		Proposal	Refinement	From tracker	From proposal net
Res10a, Res50, Cascaded	43.2	20.7	22.5	/	/
Res10a, Res50, CaTDet	49.3	20.7	28.6	11.9	22.5
Res10b, Res50, Cascaded	23.5	7.5	16.0	/	/
Res10b, Res50, CaTDet	29.1	7.5	21.8	11.4	16.0

Table 4. Comparison of the accuracy of DNN model as (a) single Faster R-CNN model (b) proposal network inside CaTDet system. The refinement net is ResNet-50. mAP and delay are both tested in KITTI's *Hard* mode.

Model	Setting	mAP	mD@0.8	ops(G)
ResNet-18	FR-CNN	0.687	5.9	138
	CaTDet(P)	0.742	3.5	163
ResNet-10a	FR-CNN	0.606	10.9	20.7
	CaTDet(P)	0.740	3.7	49.3
ResNet-10b	FR-CNN	0.564	13.4	7.5
	CaTDet(P)	0.741	4.0	29.3
ResNet-10c	FR-CNN	0.542	15.4	4.5
	CaTDet(P)	0.741	4.1	27.3

Table 5. Comparison of the accuracy of DNN model as (a) single Faster R-CNN model (b) refinement network inside CaTDet system. The proposal net is ResNet-10b. mAP and delay are both evaluated in KITTI's *Hard* mode.

Model	Setting	mAP	mD@0.8	ops(G)
ResNet-18	FR-CNN	0.687	5.9	138
	CaTDet(R)	0.696	6.0	24.4
ResNet-50	FR-CNN	0.74	3.3	254
	CaTDet(R)	0.741	4.0	39.8
VGG-16	FR-CNN	0.742	4.2	179
	CaTDet(R)	0.743	4.4	63.9

match original mAP except Res-18. Also, the cascaded system becomes more sensitive to the choices of the proposal network and the output threshold than CaTDet. Here C-thresh is the output threshold for proposal net. A higher C-thresh means fewer proposals to the refinement net, at the cost of more operations for refinement network.

For the delay metric, both cascaded system and CaTDet are sensitive to the choice of proposal network and the C-threshold. Figure 6 shows that as C-thresh increases, the average delay gradually increases as well. The main reason is that not enough proposals are fed into the refinement work, causing delayed detection of an object sequence.

Additional results in Section 7 further show that removing the tracker greatly harms the performance of the system.

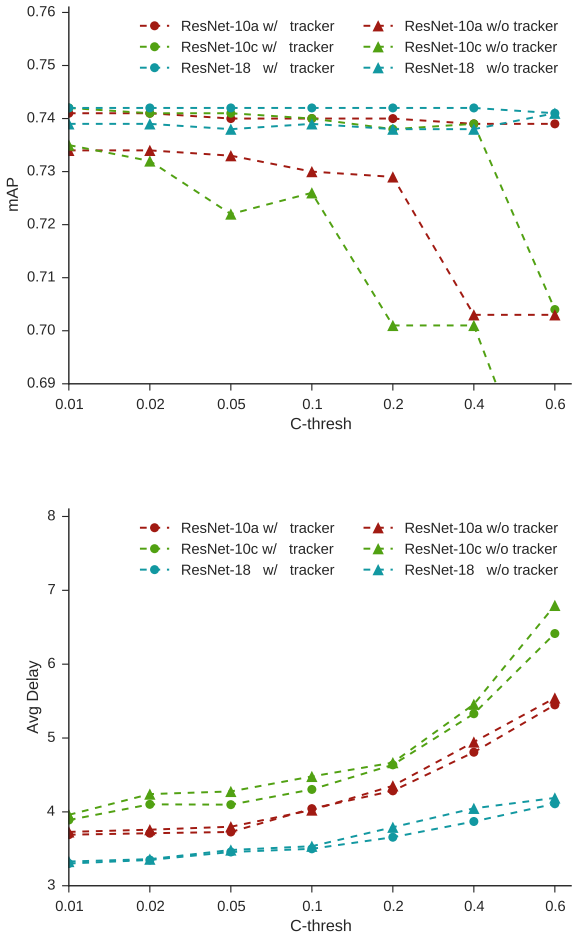


Figure 6. Upper: mean Average Precision(mAP) with varying thresholds for proposal network. Lower: mean Delay at a precision of 0.8(mD@0.8) with varying output thresholds for proposal network(C-thresh). In the case without a tracker, the system is a typical cascaded system.

Visualization of delay-recall correlation Figure 7 illustrates the trade-offs of recall vs. precision and delay vs. precision. Recall and delay have a strong correlation as the precision changes. Due to fewer number of instances in-

volved in delay evaluation, the delay curve is not as smooth as the recall curve. In this case, pedestrians usually have smaller bounding boxes, which makes them harder to be detected.

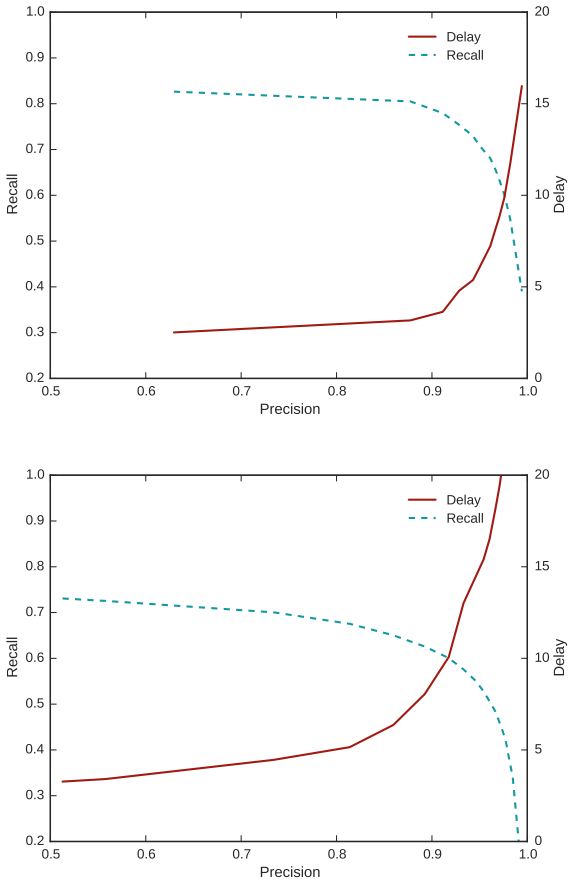


Figure 7. Illustration of how delay, recall correlate with precision. Upper: class *Car*; Lower: class *Pedestrian*. Delay is in unit of frames.

7 Experiments on CityPersons

7.1 Dataset and training

CityPersons dataset (Zhang et al., 2017a) provides bounding-boxes level annotations over a subset of the popular semantic segmentation dataset CityScapes (Cordts et al., 2016). As the name suggested, only the class *Person* is annotated. A total of 35016 bounding boxes are annotated in 5000 images.

CityPersons consists of 30-frame sequences at a frame rate of 30 fps and a resolution of 2048x1024. The 20th frame of every sequence is labelled. In the training time, we follow the 3-stage pipeline for KITTI dataset. In the inference time, the sparse annotation makes it impossible to evaluate the detection delay. Therefore only mAP is evaluated for CityPersons dataset. The detection system is run on the full sequence, and the output of the labelled frames are compared against the groudtruths.

We compute AP for *Person* class following the evaluation protocol of Pascal VOC (Everingham et al., 2010) in accordance to KITTI, where 11 recall values are selected to calculate AP. It should be noted that the official CityPersons benchmark follows the protocol of MS-COCO (Lin et al., 2014), which measures mAP under 10 different IoUs(Intersection Over Union) ranging from 0.5 to 0.95.

7.2 Results

As the CityPersons dataset is sparsely annotated, we only list the mAP in Table 6. Due to the fact that frames in CityPersons has much higher resolution than KITTI (2048x1024 vs. 1242x375), the baseline ResNet-50 model has a higher operation count(597Gops vs. 254Gops).

Table 6 compares the baseline Faster R-CNN model, cascaded system and CaTDet. A significant difference between the results of KITTI and CityPersons is that the cascaded system performs substantially worse on CityPersons. For both configurations shown in the table, eliminating the tracker reduces the mAP by more than 5%.

On CityPersons, our proposed system is not able to fully match the original accuracy but provides very close mAP to the baseline model. CaTDet with ResNet-10b as proposal net and ResNet-50 as refinement net achieves 13x operation saving while only causes 0.8% mAP loss.

Table 6. mAP and number of operations on CityPerson dataset. All the hyper-parameters are kept the same as in KITTI experiments to ensure that CaTDet systems are robust across different scenarios.

System	mAP	ops(G)
Res50 Faster R-CNN	0.674	597
Res10a, Res50, Cascaded	0.611	79.5
Res10a, Res50, CaTDet	0.662	87.4
Res10b, Res50, Cascaded	0.607	39.0
Res10b, Res50, CaTDet	0.666	46.0

8 Conclusion

In this paper we proposed a new detection system for detection from video and a new delay metric for latency-critical video applications. On KITTI dataset, the proposed system CaTDet is able to save arithmetic operations of object detection by 5.1-8.7x with no mAP loss and minor delay overhead. On CityPersons dataset, CaTDet achieves 13.0x saving with 0.8% mAP loss. In addition, our analysis showed that the delay correlates with, but not behaves exactly the same as the recall, therefore requires additional attention for latency-critical detection system design.

REFERENCES

Bewley, A., Ge, Z., Ott, L., Ramos, F., and Upcroft, B. Simple online and realtime tracking. In *Image Processing (ICIP), 2016 IEEE International Conference on*, pp. 3464–3468. IEEE, 2016.

Bolukbasi, T., Wang, J., Dekel, O., and Saligrama, V. Adaptive neural networks for efficient inference. In *International Conference on Machine Learning*, pp. 527–536, 2017.

Cavigelli, L., Degen, P., and Benini, L. Cbinfer: Change-based inference for convolutional neural networks on video data. In *Proceedings of the 11th International Conference on Distributed Smart Cameras*, pp. 1–8. ACM, 2017.

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

Everingham, M., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88 (2):303–338, 2010.

Feichtenhofer, C., Pinz, A., and Zisserman, A. Detect to track and track to detect. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3038–3046, 2017.

Figurnov, M., Collins, M. D., Zhu, Y., Zhang, L., Huang, J., Vetrov, D., and Salakhutdinov, R. Spatially adaptive computation time for residual networks. *arXiv preprint*, 2017.

Geiger, A., Lenz, P., and Urtasun, R. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., and Brox, T. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *IEEE conference on computer vision and pattern recognition (CVPR)*, volume 2, pp. 6, 2017.

Kang, D., Emmons, J., Abuzaid, F., Bailis, P., and Zaharia, M. Noscope: Optimizing neural network queries over video at scale. *arXiv preprint arXiv:1703.02529*, 2017.

Kang, K., Li, H., Yan, J., Zeng, X., Yang, B., Xiao, T., Zhang, C., Wang, Z., Wang, R., Wang, X., et al. T-cnn: Tubelets with convolutional neural networks for object detection from videos. *arXiv preprint arXiv:1604.02532*, 2016.

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 Fleet, D., Pajdla, T., Schiele, B., and Tuytelaars, T. (eds.), *Computer Vision – ECCV 2014*, pp. 740–755, Cham, 2014. Springer International Publishing. ISBN 978-3-319-10602-1.

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, 2016a.

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, 2016b.

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

Zhang, S., Benenson, R., and Schiele, B. Citypersons: A diverse dataset for pedestrian detection. *arXiv preprint arXiv:1702.05693*, 2017a.

Zhang, S., Lin, W., Lu, P., Li, W., and Deng, S. Kill two birds with one stone: Boosting both object detection accuracy and speed with adaptive patch-of-interest composition. In *Multimedia & Expo Workshops (ICMEW), 2017 IEEE International Conference on*, pp. 447–452. IEEE, 2017b.

Zhu, X., Dai, J., Yuan, L., and Wei, Y. Towards high performance video object detection. *arXiv preprint arXiv:1711.11577*, 2017a.

Zhu, X., Wang, Y., Dai, J., Yuan, L., and Wei, Y. Flow-guided feature aggregation for video object detection. *arXiv preprint arXiv:1703.10025*, 2017b.

Zhu, X., Xiong, Y., Dai, J., Yuan, L., and Wei, Y. Deep feature flow for video recognition. In *Proc. CVPR*, volume 2, pp. 7, 2017c.