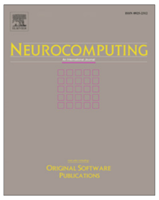
Neurocomputing 477 (2022) 14–24

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom



Towards real-time object detection in GigaPixel-level video

Kai Chen, Zerun Wang, Xueyang Wang, Dahan Gong, Longlong Yu, Yuchen Guo, Guiguang Ding

Tsinghua University, Haidian, Beijing 100084, China

ARTICLE INFO

Article history:
Received 24 August 2021
Revised 2 November 2021
Accepted 12 December 2021
Available online 17 December 2021

Keywords:
Object detection
GigaPixel
Deep learning
Real-time

ABSTRACT

Object detection aims to locate and recognize objects in images or videos, which contributes to many downstream intelligent applications. Recently, emerging gigapixel videography has attracted considerable attention from computer vision, microscopy, telescopy and many other communities. Its large field of view and high spatial resolution provide sufficient global and local information simultaneously. Although state-of-the-art detection methods have achieved success in common images, they can not be transferred to gigapixel images with both effectiveness and efficiency. To solve this problem, we make the first attempt towards accurate and real-time object detection in giga-pixel video. In this paper we propose a novel framework, termed as GigaDet, which adopts an efficient global-to-local strategy, following the principle of human vision system. Based on the spatial sparsity of objects, a patch generation network (PGN) is introduced to globally locate possible regions containing objects and determine the proper resize ratio of each patch. Then the collected multi-scale patches are fed into a decorated detector (DecDet) in parallel to perform accurate and fast detection in a local way. We carry out extensive experiments on PANDA dataset and GigaDet yields 76.2% AP and 5 FPS on a single 2080ti GPU, which is comparably accurate but 50x faster than Faster RCNN. We believe this research can inspire new applications based on gigapixel video for a large range of fields.

1. Introduction

Object detection is a basic, common, but challenging task which aims at locating and recognizing objects in images or videos. For most artificial intelligent applications, such as surveillance video analysis and person re-identification retrieval, their performance significantly depends on the output of the preset object detection methods. In some special scenarios such as crowd flow analysis and anomaly detection on surveillance video, detection efficiency is a key issue and (near) real-time execution is required. In recent years, object detectors based on deep learning have been improved significantly to achieve precise and efficient detection performance on common images. For example, on popular benchmarks like Pascal VOC [1] and MS COCO [2], satisfactory performances have been reported in two-stage detectors [3]45. Towards the faster speed, several single-stage detectors have achieved over 25 FPS on GPU with mega-pixel input images [6]789. Besides, some anchor-free methods like [10]1112 have also been proposed to overcome the drawbacks introduced by preset anchors.

Although the state-of-the-art object detectors have achieved great success, there are still shortcomings remaining in newly emerging scenarios such as GigaPixel-level videos. The emergence of GigaPixel-level videos benefits from the development of photography. They are usually captured by the giga camera in outdoor scene and the field of view often covers a fairly wide range from near to far. GigaPixel-level surveillance videos recently become the important materials to analyze group behavior in public space [13]. Most of existing object detectors are designed to process images with normal resolution from 640*px* to 2000*px* containing only dozens of objects[14], and they can not be directly applied to large images used in gigapixel videography or telescopy.

There are some preliminary attempts towards object detection in GigaPixel-level image and video. Since it is impossible to load the whole image as input for the deep model due to the limit of GPU memory, they used a strategy of sliding windows to scan the whole image block by block, and then combined the results of each grid to get final detection results. They achieved barely satisfactory performance compared to conventional detectors. By using Faster-RCNN [3] as a basic detector, only 75.5% AP.50 for large visible body and 0.10 FPS on a GigaPixel-level dataset PANDA [15] are achieved, which is far lower than the performance on benchmarks like Pascal VOC [1]. Even if using a single-stage detector like YOLO[16], the inference procedure takes several seconds as well, since scanning all grids needs a large expenditure of time. It is

https://doi.org/10.1016/j.neucom.2021.12.049
0925-2312/© 2021 Elsevier B.V. All rights reserved.

also a challenge to select the proper size of the grids, which may result in the cases that tiny objects exist in large grids or out-of-range objects occupy the whole small grids.

In this paper, we focus on the task of object detection for GigaPixel-level images. To overcome the problems of existing object detectors mentioned above, we propose a progressive strategy named GigaDet to perform efficient and accurate detection on the GigaPixel-level images. GigaDet consists of two key components: PGN (Patch Generation Network) and DecDet (Decorated Detector). PGN executes a fast analysis for the thumbnail of the image to obtain proper patches which have the highest probability to contain valid objects. The DecDet is equipped to execute precise detection within patches in parallel. The local detection outputs will be remapped into original image according to the offset information for each patch, and then the final outputs are combined and produced. Based on the progressive framework, we can obtain 76.2% AP and 5 FPS on a single 2080ti GPU, which is comparably accurate but 50x faster than the baseline method. Besides, benefiting from the logic of patch generation, the performance for small objects like heads of the persons is improved significantly both for efficiency and accuracy. The intuitive illustration for performance vs. efficiency can be seen in Fig. 1.

Acceleration for object detection has been extensively studied recently [18]1920. Our intuitive idea is inspired by the human visual behavior of scanning [21], which would be a rough look at first and then the meticulous inspection if necessary. The PGN module is designed to perform the scan action, which reduces large amount of regions with little possibility to contain objects.

To summarize, the contributions of this paper include:

- We propose a novel framework called GigaDet for real-time object detection in GigaPixel-level videos. To our best knowledge, this is the first work ever focusing on this challenging task.
- We design a preparatory module named Patch Generation Network (PGN) for large-scale GigaPixel-level input, to filter out irrelevant regions which are not tightly linked with the main task. The utilization of PGN module is critical for improving the detection efficiency.
- The proposed method is evaluated on PANDA dataset and achieves nearly real-time inference speed while maintaining state-of-the-art accuracy.

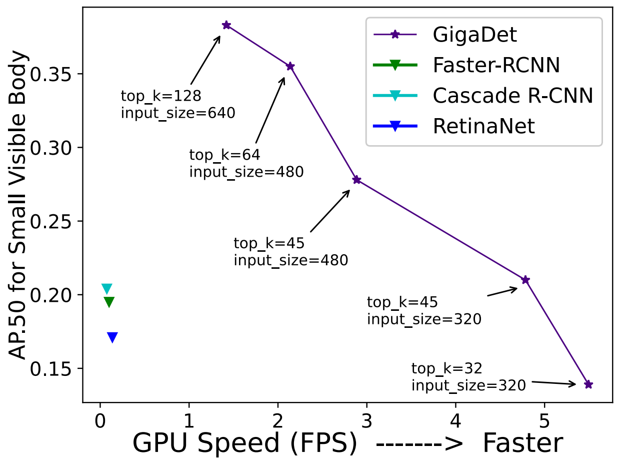


Fig. 1. Performance vs. Speed: Comparison of the proposed GigaDet framework and the baseline model. The results of Faster-RCNN [3], Cascade R-CNN [17] and RetinaNet [5] from [15] are denoted by del marker. The results for GigaDet (Ours) are denoted by the broken lines at different experimental settings. Our GigaDet boosts the performance both in efficiency and accuracy especially on small targets like sma.ll visible body.

2. Related work

2.1. Object detection

Object detection task is a common task in computer vision [22] 23. From the pioneer Viola-Jone detector [24], an intuitive idea to resolve object detection task is to scan the entire picture using sliding windows of different sizes. Early detectors filter out the windows that do not look like an object [25]26, converting the detection procedure into a binary-classification task. However, large amount of sliding windows severely slow down the inference speed, which prevents these technologies from practical application. With the development of deep learning technology, some methods have achieved a great promotion based on large annotated datasets like ImageNet [27] and MS COCO [2]. Modern object detectors are usually categorized into two-stage or one-stage by whether they own a RoI proposal step [28]. Two-stage detectors represented by Faster-RCNN [3] include operations about region proposal and target category regression, which help the detector to be more flexible and obtain higher accuracy. Single-stage detectors like SSD [8] and YOLO [6] family directly detect objects by presetting anchors, achieving higher speed compared to two-stage detectors. Recently, anchor-free methods [29,11,12,10] have attracted more attentions and they are proposed to avoid prior knowledge of hand-designed anchors, which is more compatible and faster. Anchor-free detectors like RepPoints [30,31] are proposed to optimize the representation for objects, which is traditionally presented by bounding boxes and corresponding labels.

In object detection task, one of the key challenges is the scale variation of objects within the image [32]. Image Pyramids [33] and Feature Pyramids [34] are proposed by researchers to resolve the problem of scale variation. In the case of coexistence of large and small object targets, how to perform effective detection for tiny objects has always troubled researchers. Kisantal et al. [35] proposed a method to randomly copy-paste small objects in images, which optimizes the proportion of small targets in the training sample and improves the precision. ClusterNet [36] utilizes spatial and temporal information to predict the location of multiple objects simultaneously, but it only regards position as one point regardless of bounding boxes. Yang et al. [37] analyze aerial images(e.g., 2000x1500 pixels) and find that the objects are generally small and usually unevenly distributed throughout the image. They propose a cluster model to recognize the group of objects, and then perform the further detection.

2.2. Object detection in large-scale scene

Images which contain wide FoV and high spatial resolution simultaneously to describe the crowd behavior and interaction are usually captured by a gigacamera [38,17,15]. They usually have over 25000 $\times$ 15000 resolutions so they are called GigaPixel-level images. The wider range of scale variation of objects in GigaPixel-level images increases the difficulty to detect both large and tiny objects. While current popular detectors [3]165 have achieved great success in nature pictures (e.g., 600x400) [37], their performance may deteriorate sharply when detecting objects under large-scale scene. The authors of PANDA dataset [15] have proved that directly applying popular detectors like Faster-RCNN [3] on GigaPixel-level images can not obtain desired results. They took strategy to split one image into several grids and then applied detectors on each grid. The combined detection results reach an acceptable level while the detection took a long time. The framework we proposed in this paper aims at accelerating the detection speed on GigaPixel-level images, which can be regarded as an improvement for the above methods.

As far as we know, there are some researches which promote detectors on images of large-scale scenes [39]. Zhou et al. [40] propose a scale adaptive data enhancement method for handling severe scale challenges in UAV object detection. Gao et al. [41] propose a regression network based on attention mechanism to estimate the number of people in the large-range pixel-wise crowd scenes. Li et al. [42] resolve domain adaptation problem for object detection in medical images. AutoFocus[43] proposes an efficient multi-scale inference strategy FocusPixels, by predicting category agnostic segmentation maps for small objects at coarser scales. AutoFocus zooms and crops only interesting regions when applying the detector at a specific scale. Sniper[44] uses context info to generate chips which contain ground-truth instances. Gao et al.[45] build a Q-net by reinforcement learning to compute accuracy gain map, and then zoom in the specific regions. Our proposed GigaDet has difference on strategy and specific processes with the methods mentioned above. Sniper[44] tries to find chips which are likely to cover a ground-truth object, but our PGN module aims at finding the blocks that contain as many valid objects as possible. Their goal is to detect objects more accurately by finding the appropriate scale, while our purpose is to collect valuable regions for subsequent detection steps to reduce the time cost. Methods in [45] obtain regions of interest by an accuracy gain map which is calculated by a Q-Net, while our patches generation procedure is based on region proposal, which is a different logic to generate the result, utilizing the annotations of objects as synthesized labels rather than individual one.

3. Proposed GigaDet

3.1. GigaDet framework

The overall framework of GigaDet is shown in Fig. 2. GigaDet is made up of a PGN (Patch Generation Network) module and a DecDet (Decorated Detector) module. PGN is used to extract the patches of interest from the thumbnail built by the high-resolution input. For the input image, meaningful regions are covered by the extracted patches and the rest of regions with few objects distributed are dropped, which significantly reduces the pixels to be processed in the subsequent tasks. DecDet is then used to detect objects in the extracted patches. The patches are organized into appropriate sizes and the tasks of detection are executed in parallel. The outputs are remapped to the original coordinates and necessary post-process operations are used to produce the comprehensive detection results. In such a coarse-to-fine procedure, a large number of objects in the GigaPixel-level image can be located and recognized correctly and promptly.

3.2. Patch generation

PGN (Patch Generation Network) is the key component of the proposed GigaDet, which aims at extracting patches that contain valid objects as many as possible. In order to achieve this goal, PGN presets patch candidates spread all over the input image and estimates the quality for each patch candidate. A conditional strategy is applied to select the most valuable patches and generate the outputs, which serve as the input for the subsequent DecDet module.

We define the term *Patch* as a scale-conscious sub-regions in an image. Suppose we have an image *GI* with width *W* and height *H*, we can formally define a *Patch* as:

$$Patch_i = Crop(GI, l_i, t_i, r_i, b_i) \tag{1}$$

Here $Crop()$ is the cropping operation to an image. The parameters l_i, t_i, r_i, b_i represent the coordinates of two vertices in $Patch_i$, which implies the width of the patch is $w_i = r_i - l_i$ and height of the patch is $h_i = b_i - t_i$. The values above should satisfy the conditions for basic geometric constraint:

$$0 \leqslant l_i < r_i < W \text{ and } 0 \leqslant t_i < b_i < H \tag{2}$$

Based on this definition, any sub-region $Patch_i$ within the image *GI* can be fetched by operation $Crop(GI, l_i, t_i, r_i, b_i)$. Meanwhile, according to the object annotation information of the image *GI*, there may be several objects located entirely within the $Patch_i$. We define the O_i as the set of the objects that are located entirely within the $Patch_i$:

$$O_i = \{\bigcup obj_c \mid obj_c \in Patch_i\} \tag{3}$$

The meaning of $\in$ is that the object obj_c is surrounded completely by the patch $Patch_i$, which means, no part of the object is out of the range. Equivalently this case can be called that the

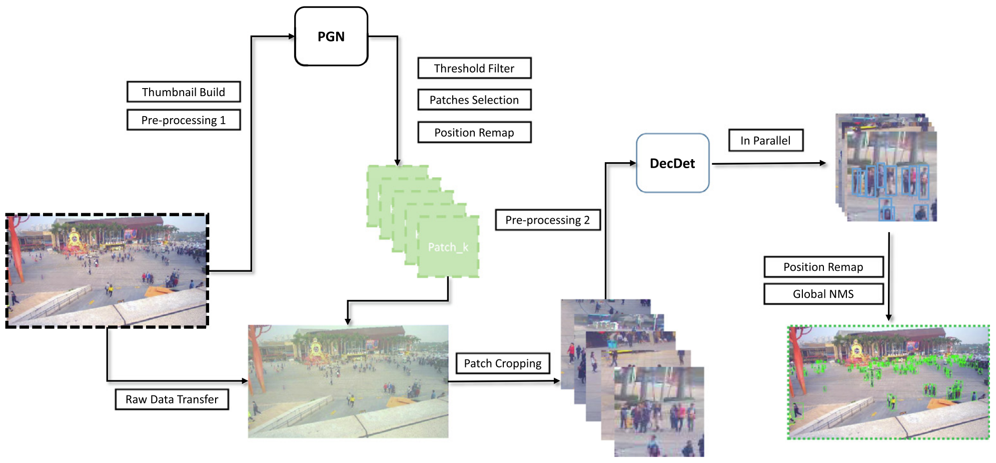


Fig. 2. Framework of GigaDet: The original GigaPixel-level image is loaded into memory and then fed into two branches. It will be firstly processed into a thumbnail and fed into the PGN module to obtain patches of interest, and then the patches along with original raw data make up the cropped images that will be fed into the DecDet module. Concurrent tasks of detection are executed and multiple outputs are remapped to the global coordinates. Post-processing steps such as Non-Maximum Suppression are applied and the detection results for the whole GigaPixel-level image are produced. The GigaDet accelerates the inference since the PGN model only processes thumbnail of a resized GigaPixel-level image, which would not slow down the inference, and the DecDet module receives a batch of images without dependency on each other, whose pixels are much fewer than the original image. This progressive mechanism contributes to the acceleration of inference on GigaPixel-level images meanwhile maintaining acceptable accuracy.

$Patch_i$ contains the obj_c. Notation $\bigcup$ means the union operation. To be more specific, we define O'_i as the subset of O_i whose objects are all valid. A object is valid when its size satisfies a certain range

$$O'_i = \{\bigcup obj_c \mid obj_c \in Patch_i \text{ and } obj_c \sim Range_i\} \tag{4}$$

The $Range_i$ is decided by the size of $Patch_i$, and the symbol $\sim$ means that the size of obj_c satisfies the condition of $Range_i$. We define $CountO_i$ as the length of the set O_i, which also represents the amount of objects the $Patch_i$ contains:

$$CountO_i = \|O_i\| \tag{5}$$

PGN presets multiple patch candidates of different scales in the image, and it is trained to estimate the property $CountO_i$ for each patch candidate $Patch_i$. The estimations are based on the feature learning procedure, where the input comes from the features of input image, and the corresponding label is calculated through the aggregation of annotated object information. Multi-scale patch candidates are used to make sure that objects of different sizes can be contained by at least one patch. An object may be located in one or more patches. The restriction for valid objects ensures the objects are neither too small nor too large compared to the patches they belong to. In the patches selection procedure, the $CountO_i$ property can be used as the assessment criteria for each patch candidate. An illustration of patch candidates in a GigaPixel-level image can be viewed in Fig. 3.

For a GigaPixel-level image, a hypothesis can be accepted that different regions have distinguished semantic information, and a minority of regions can possibly contain a large proportion of the objects in the image. The proposed PGN is designed to generate patches to cover these important regions. In consideration of the computation efficiency for subsequent tasks, the number of patches generated by PGN should be limited. We use $S_P = \{\bigcup_{k=1}^{K} Patch_k\}$ to represent a set of patches, which is a subset of all preset patch candidates, and the value K is much fewer than the number of them.

Based on the estimations for patch candidates, PGN use a conditional strategy to select the most valuable patches. All patch candidates are firstly sorted by their $CountO$ value. Since the patch candidates have overlapped regions with each other, a conditional removal operation is applied to avoid redundancy in patch selection. Finally, PGN retrieves the top K patches and organizes them into the output.

3.3. DecDet

DecDet (Decorated Detector) is used to detect specific objects within each patches. One of the important properties needed in DecDet module is the concurrent execution. The patches generated by PGN component have no dependencies on each other, and their detection results can be obtained in parallel, which saves a lot of time compared to the serial computation. The task processing flow for DecDet is similar to commonly used routines for big data analysis, where the tasks are dispatched simultaneously and then the outputs of tasks are collected to form the final results.

Along with the location information of patches themselves, the bounding boxes of objects can be remapped to global positions in the coordinates of the original GigaPixel-level image. After the global post-processing operations like NMS (Non Maximum Suppression) [46], the outputs from patches are arranged into final results of the overall detection.

Theoretically, DecDet could be made up from one of any existing detectors. Because this part is not about the basic architecture of GigaDet, we adopt YOLO [7] based model for its features of agility and simplicity.

3.4. Implementation details

We build the PGN module using the PyTorch toolkit[47]. It is constructed based on a common VGGNet[48] backbone, and the input image is pre-processed into a thumbnail to be fed into the CNN network. Patch candidates are preset on the final layer of

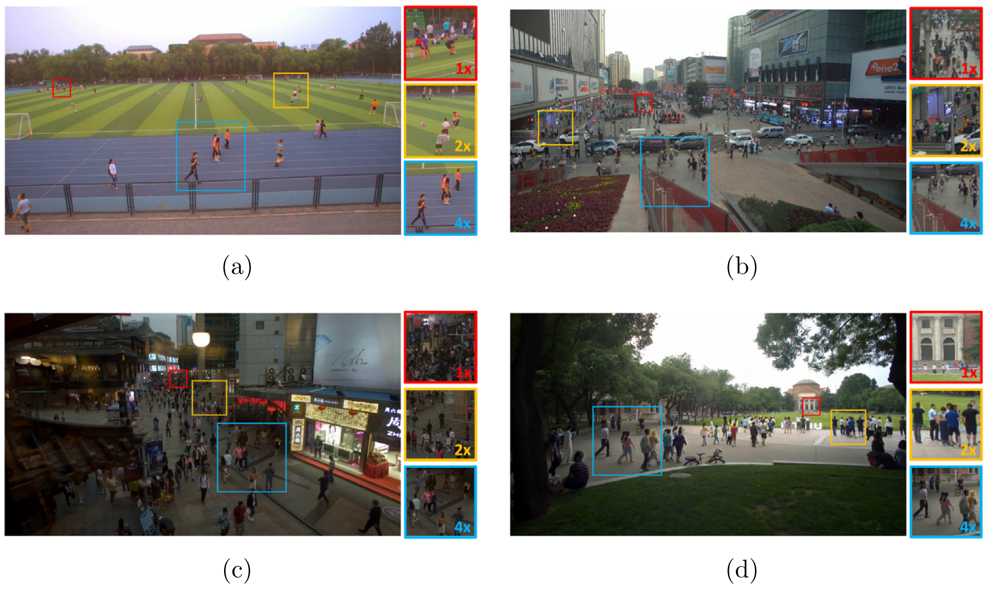


Fig. 3. Illustration of patch candidates in an image. A patch candidate is the output of cropping operation at specific parameters for a large image. 3 different sizes of patches are visualized in the image. "4x" means the side length ratio is 4 times as the unit one, which means the patch candidate is 4 times larger than patch candidate with "1x" notation. Objects may have different sizes in the distant/close view, so it is critical to select a patch candidate with proper size to contain plenty of valid objects. PGN presets different sizes of patch candidates uniformly distributed all over the image, which serve as the candidates of the final generated patches. As shown in the figure, an "4x" patch candidate in close shot area contains several persons with proper size, meanwhile the "1x" patch in distant shot area also contains many persons. A larger patch candidate does not always contain more valid persons than a smaller patch candidate, since it depends on the size of persons in that region. The size of persons which appear in "4x" patch in distant shot area may be too small.

VGGNet, and each patch candidate has an attribute *CountO*, which is initialized to zero. Patch candidates of different scales are preset for dealing with various objects. Patch candidates that go beyond the boundary of the image are just ignored in subsequent processing. For each patch candidate, a simple 1x1 convolution layer followed by a linear layer is used to regress its output value.

The main training procedure is to optimize parameters in PGN to estimate the $CountO_i$ for each $Patch_i$, making $CountO_i$ as close as possible to the ground-truth value. The corresponding label is calculated by the annotated object information. The input image is resized in proportion to build the thumbnail. Then it is fed into the model and features are extracted. We use *smooth l*1 *loss* to regress the property *CountO* of each patch candidate. Due to the large capacity of a GigaPixel-level image, the *batch_size* is set to 1 and the initial learning rate is set to $1e-3$. We use *lr_decay* as 0.1 and *weight_decay* as 0.0005 during the training. Weights of the backbone are initialized with ImageNet pre-trained model. we train the PGN model for 9 epochs, which need about 21 h in a single Titan 2080 Ti GPU.

At the inference step, the $CountO_i$ value for each patch candidate is calculated after the model forward execution, and the patch candidates with position out of boundary are removed firstly. A threshold $\xi_C \equiv 1$ is used to filter out patch candidates whose $CountO_i$ is below the threshold. This operation is meaningful to save a lot of computation complexity for later sort operation. Remained patch candidates are sorted by their $CountO_i$ value. Since the patch candidates have dense distribution on the spatial plane, we apply the NMS (Non-Maximum Suppression) [46] operation to remove redundant items. The value K is set to slice top patches according to realistic requirements, in our case we set it to 45. *iou_threshold* for NMS is set to 0.2, which can be adjusted to control the density of generated patches. The final generated patches are obtained by remapping the position information into the original image. The whole procedure for patch generation is visualized in Fig. 4.

In the GigaDet framework, DecDet (Decorated detector) is used to receive the patches generated from PGN module as the input. We adopt detector model based on YOLO[7] architecture. We choose this model to take full advantage of its agility. In fact the decorated detector can be replaced with any practical detection models, as long as they support batch detection and meet the speed requirement. To train the DecDet model, we prepare the training data from the generated patches. Since the relative size of objects in GigaPixel-level images are quite small, it would not bring benefits from training models using the raw annotations. Patches are extracted using pre-trained PGN model and the objects which are located entirely within the patches are regarded as valid annotations. Offsets are calculated according to the position of the patch respectively. Local annotations are produced based on raw GigaPixel-level images and corresponding annotations information. Then pairs of images and annotation files are saved to be organized as the training data. To balance amount of objects in different sizes and promote the ability of detecting small objects, we generate extra images surrounding tiny objects which are small than 288×288 pixels.

During the inference, patches are resized into the same size and normalized in pre-process stage. Patches are all square uniformly, so the extra operations in this stage like building shape for batch and adding letterbox for images can be omitted. After forwarding and fetching the inference outputs, the local NMS post-process

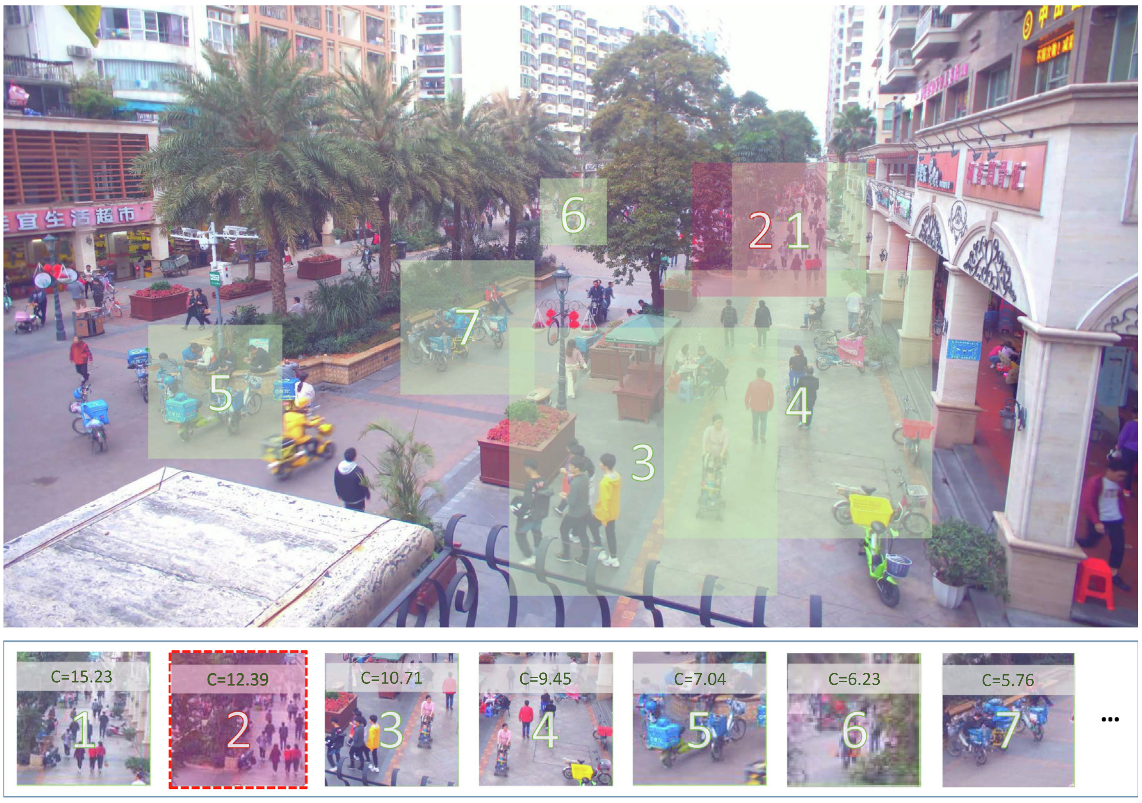


Fig. 4. Illustration of the procedure of patches selection strategy. Given an GigaPixel-level image, PGN module firstly builds the thumbnail, and uses the features extracted from backbone to estimate the property $CountO_i$, the number of valid objects, for each patch candidate. All patches candidates of different scales are sorted by the property $CountO_i$ and filtered through a specific threshold ξ_C. To avoid getting patches that are too close to each other, some post-processing operations such as NMS are applied and the redundant elements are removed. The squares colored in image represent the selected patches candidates, and the patch candidate marked in red is thrown during patch selection. Hyper-parameter K is used to intercept the top K elements and generate the output patches.

operation is applied to generate local detection results for each patch. The size of input image can be set dynamically according to the size of patches, here we simply set input size to 320 for all patches. Threshold of confidence is set to 0.001, and the *iou_threshold* for NMS is set to 0.6.

When PGN module and DecDet model are ready for use, we organize the overall process to an end-to-end inference procedure as follows. A GigaPixel-level image is loaded into memory, and it is pre-processed to build a thumbnail to be fed into the PGN module. List of patches are generated according to the inference logic described above, and then input images for DecDet are cropped from the raw GigaPixel-level image by the generated patches. Concurrent tasks of detection are executed, and some post-processing operations are applied. Direct detection for the original image can be an optional choice, which may benefit the accuracy in some situations. All the detection results from patches are applied by a global NMS operation, to remove redundant elements with high overlap ratio. A visualization result for final detection can be viewed in Fig. 5.

4. Experiments

4.1. Dataset and settings

4.1.1. Dataset

PANDA[15] is a GigaPixel-level human-centric dataset, which contains gigabit levels of pixels per image with extremely high resolution details for large-scale visual analysis. It provides 18 scenarios and over 15,974.6 k bounding box annotations, offering a completely new challenge for human detection task. For each person, PANDA provides bounding box annotations for its visible body, full body and head. Raw dataset includes IMAGE and VIDEO parts, and we use the IMAGE part to evaluate our method. The IMAGE part is split into 13 scenarios as train set and 5 scenarios for test. In our experiments, we use the subset of IMAGE part about persons in PANDA dataset to verify our proposed methods.

4.1.2. Evaluation metric

In spite of the challenging difficulty of processing a GigaPixel-level image, it can be viewed as a common detection task whatever. The popular evaluation metric *AP*@0.5 will be used as the primary performance measurement. Refer to the PANDA baseline[15], we also report *AR* value for comparison. We take inference speed into account as an important assessment criteria for our proposed method, which aims at improving the efficiency of processing detection task towards real-time performance.

4.1.3. Experimental setup

As the basis, we trained the PGN model using PyTorch toolkit [47]. We use the pre-trained PGN model to crop patches from original PANDA images, and generate common format detection labels with mapping of global annotation to local locations in the patches. The generated data are used to train the decorated detector, which we follow the YOLO[7] architecture. With PGN and DecDet modules, we organize the overall procedure to detect objects from GigaPixel-level images. The inference process has been integrated to an end-to-end style, which can be used like a black-box detector. The whole training procedure has not been combined together, which remains to be integrated and optimized in the future.

4.2. Inference speedup

We choose annotations of persons in PANDA dataset as our detection target. We evaluate our GigaDet by the annotated labels

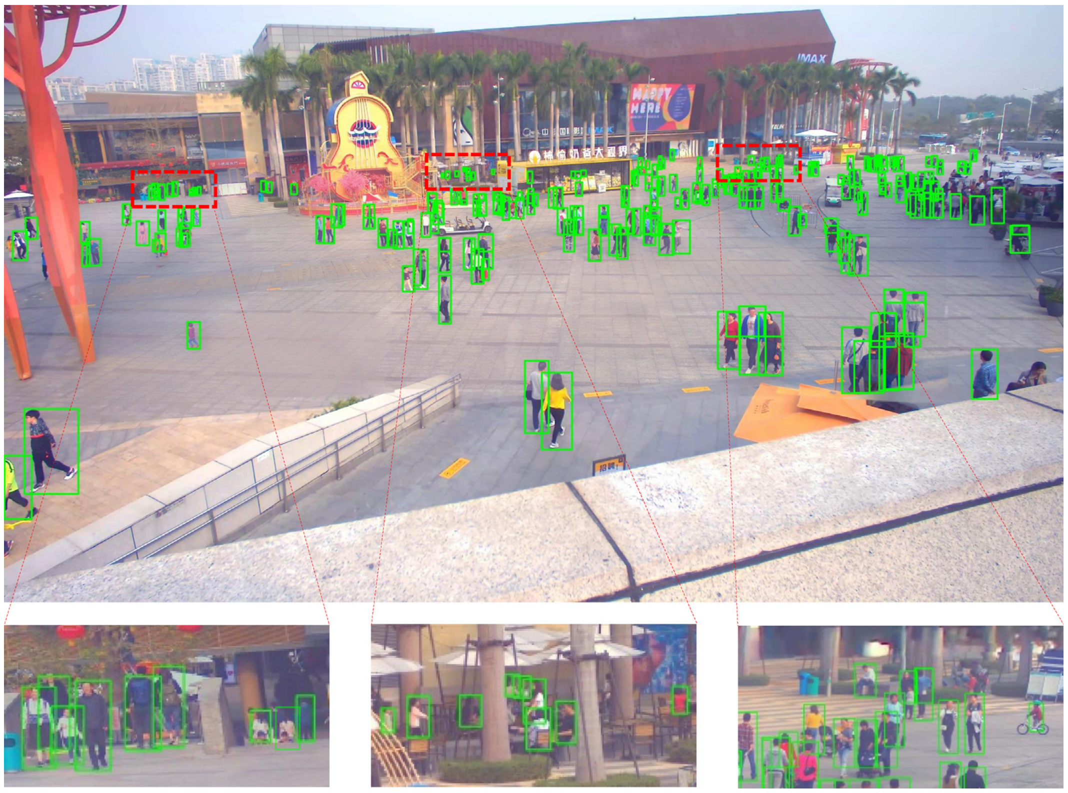


Fig. 5. Example of the detection results on a GigaPixel-level image. The sparse distribution and large variation in scales of objects are serious challenges for the detector. Using a progressive framework, GigaDet successfully fetches the most valuable patches and then uses the decorated detector to focus on fine inspection within the patches. Tiny objects in distant view are also detected correctly even if they are barely visible to the naked eye without zooming in. At the same time, large objects in close view are also detected without introducing too many false positive samples. Zoom in the figure for a better observation.

including visible body, full body and head. We use the method and results in [15] as our baseline. With our proposed framework, we can improve the inference speed up to ~ 5 FPS, at least 50x faster than baseline method, meanwhile maintaining almost the same precision performance. The *AP*.50 and *AR* results can be viewed in Table 1.

Many hyper-parameter settings in the evaluation procedure can affect the trade-off between precision and efficiency. We set the hyper-parameter values as follows. The scales of patch candidates are set to [1,2,4,8]. The *K* value for patch generation is set to 45. The IoU threshold for patch generation is set to 0.2. The input size for DecDet is set to 320. The confidence threshold for DecDet is set to 0.001. The IoU threshold for NMS is set to 0.6.

In maintaining almost the same AP.50 performance, GigaDet can achieve nearly 5 FPS inference speed for the detection task on GigaPixel-level images, which accelerates almost 50x than original baseline method. Through the coarse-to-fine progressive detection, many invalid operations on empty regions are eliminated and the time cost is reduced. The framework towards real-time object detection has been developed, and we believe it can be optimized to achieve real time analysis for GigaPixel-level videos in the future.

4.3. Necessity of patch generation

The key module for accelerating detecting objects on GigaPixel-level images is to select important patches. We explore the necessity of patch generation and verify that the PGN module can help to select effective information.

The intention of patch generation is to extract high-valued information of GigaPixel-level images. During patch generation, all patch candidates whose *CountO* is below the threshold $\xi_C = 1$ are filtered out, and the remaining ones are sorted by its *CountO* property. In principle, the larger the threshold ξ_C is, the fewer satisfied patches are left in next step. Due to the fact that patches with lower *CountO* will be ranked to the end in the sorting step, we set the threshold as 1 so that patches that contains at least one object will be taken into account in the sorting operation, which reserves the information to the maximum extent and the time cost of sorting is negligible.

According to different actual application scenarios, we can use the hyper-parameter *K* to slice part of the sorted patches list. More patches help achieving the higher *AP* metric but cost more computing time. The experimental results in Table 2 show the effects on the value *K* in patch generation.

Table 2
Study on the K value in patch generation procedure. The larger the K value is, the more patches are generated and fed into subsequent decorated detection process, which means the increment of expenditure of time. Specific setting of the value K can be decided by the practical scenario.

K	speed	*AP*.50 for Visible Body			
		S	M	L	Total
8	0.69×	0.0283	0.343	0.364	0.346
16	0.83×	0.0578	0.448	0.559	0.498
32	0.91×	0.139	0.556	0.715	0.633
45	1×	0.210	0.599	0.762	0.684
64	1.38×	0.275	0.607	0.784	0.707
128	2.47×	0.281	0.607	0.785	0.708

We use the fixed scales of patches candidates as $[1, 2, 4, 8]$ and the input size as 320 without option of detecting directly. As the results show, with more top K selected patches, the *AP*.50 value meliorates together with the increase of inference speed. We find that when *K* increases from 64 to 128, increasing *K* produces very little effect, which means it is close to saturation by adding patches.

For each *K* setting, we also conduct another experiment to verify the necessity of patch generation. We directly split the image into K grids, and we can observe that, the performance is seriously deteriorated when using the grids directly split from the image, which verifies the necessity of the proposed PGN module. Results are listed in Table 3 for comparison.

4.4. Ablation study on GigaDet settings

There are many hyper-parameters and experimental settings in GigaDet framework. We conducte a lot of experiments to explore their effects on the trade-off between accuracy and efficiency to get better performance.

4.4.1. Scales of patch candidates

Patch generation is the critical premise of obtaining good performance. In PGN module, we use preset patch candidates for patches selection. We conduct experiments to find the optimal scales settings. Consider that the generated patches will be used as input into decorated detector, they will be inevitably resized and cropped to square at that step, so there is no need to set multifarious irregular shapes for preset candidates. We fix the aspect ratio to 1. Different sizes are necessary, because they correspond to patches of different sizes which should contain objects of proper

Table 1
Performance of GigaDet on PANDA dataset, compared with original baseline in [15]. FR, CR, and RN denote Faster R-CNN, Cascade R-CNN and RetinaNet respectively. The speed is not introduced in [15], so we reproduced the methods according to the author's guidance and obtained the conclusion. Sub means subset of different target sizes, where Small, Middle, and Large indicate object size being $< 96 \times 96$, $96 \times 96 - 288 \times 288$, and $> 288 \times 288$, which is also not introduced in [15] but confirmed by the author. We obtain the best speed while maintaining the comparable performance with the baseline.

	Sub	Visible Body		Full Body		Head		Speed
		$AP_{.50}$	AR	$AP_{.50}$	AR	$AP^{.50}$	AR	
FR [3]	S	0.201	0.137	0.190	0.128	0.031	0.023	
	M	0.560	0.381	0.552	**0.376**	0.157	0.088	$\sim$ 0.10 FPS
	L	0.755	0.523	**0.744**	0.512	0.202	0.105	
CR [17]	S	0.204	**0.140**	0.227	**0.160**	0.028	0.018	
	M	0.561	**0.388**	0.579	0.384	0.168	0.091	$\sim$ 0.07 FPS
	L	0.747	**0.532**	0.765	**0.518**	0.241	0.116	
RN [5]	S	0.171	0.121	0.221	0.150	0.023	0.018	
	M	0.547	0.370	0.561	0.360	0.143	0.081	$\sim$ 0.13 FPS
	L	0.725	0.482	0.740	0.479	0.259	0.149	
Ours	S	**0.210**	0.093	**0.236**	0.100	**0.575**	**0.289**	
	M	**0.599**	0.339	**0.605**	0.326	**0.769**	**0.443**	$\sim$ 5 FPS
	L	**0.762**	0.467	0.728	0.418	**0.596**	**0.462**	

Table 3
Results of split image into K* grids directly. Instead of using PGN module to generate patches, we did control experiment about directly splitting the image into K* grids. For example, for K*=32, we split the image into 8x4 grids. Obviously this rough method can not obtain satisfactory result.

K*	speed	*AP*.50 for Visible Body			
		S	M	L	Total
8	0.25×	0.000	0.011	0.424	0.231
16	0.28×	0.000	0.0389	0.574	0.331
32	0.55×	0.005	0.305	0.692	0.511
45	0.76×	0.004	0.427	0.700	0.561
64	0.79×	0.003	0.360	0.554	0.461
128	1.61×	0.007	0.550	0.638	0.589

sizes. The multi-scale settings for patch candidates play a vital role in capturing objects from a near or distant view.

We explore many settings which can be viewed at the Table 4. We can find that different scales of patch candidates are responsible for the objects of different scales. For example, the setting [1,2,4,8] means four sizes of patches candidates are preset, and they are $1, 2, 4$ and 8 times the unit size respectively. Unit size is decided by the down-sampling ratio of the PGN model, and in our case it's $\frac{1}{16}$ of the shorter size of input image. Patch candidates with larger size are supposed to capture valid objects with larger scale, and vice versa. Richer scales help to capture more objects in the images, but also introduce difficult selection dilemma during patch generation. Visualization for the logic of patches selection can be viewed in Fig. 6.

4.4.2. Size of input for DecDet

After patch generation, the selected patches are resized to dynamic sizes and executed detection in parallel. Here the sizes of input for detector represent the uniform size which all the patches should be resized to. We fix other settings and observe the impact of the input size. The results can be seen in Table 5.

Table 4
Study on PGN anchors settings. The array of digits in Anchors column means the sizes of anchor bases. For example, [1,2,4] means three sizes of anchor base are preset, and they are 1, 2 and 4 times the unit size respectively. Anchors of larger size are supposed to capture valid objects with larger scale and vise versa.

Anchor Setting	*AP*.50 for Visible Body			
	S	M	L	Total
[1,2,4]	0.137	0.453	0.534	0.434
[1,2,4,8]	0.210	0.599	0.762	0.684
[1,2,4,8,16]	0.197	0.566	0.765	0.645

Table 5
Study on input sizes for DecDet. The size of input for DecDet model will affect the speed and accuracy. We did ablation study to find the most appropriate setting.

Input	speed	*AP*.50 for Visible Body			
		S	M	L	Total
320	1×	0.210	0.599	0.762	0.684
480	1.26×	0.278	0.638	0.764	0.707
640	1.83×	0.293	0.649	0.760	0.711
768	2.38×	0.316	0.647	0.751	0.705

Table 6
Study on optional direct detection. The first line means the GigaPixel-level images are directly resized and fed into DecDet, whose detection outputs are awful. w/o means the standard GigaDet workflow, and w means combining the outputs of the above two methods.

Type	speed	*AP*.50 for Visible Body			
		S	M	L	Total
only	0.14×	0.000	0.000	0.012	0.006
w/o	1×	0.210	0.599	0.762	0.684
w	1.28×	0.209	0.595	0.764	0.679

Table 7
Study on patch IoU threshold. Large patch IoU threshold means the greater tolerance for overlapped patches.

IoU T	speed	*AP*.50 for Visible Body			
		S	M	L	Total
0.0	0.78×	0.169	0.552	0.667	0.616
0.1	0.85×	0.172	0.568	0.729	0.655
0.2	1×	0.210	0.599	0.762	0.684
0.3	1.09×	0.133	0.540	0.701	0.617
0.5	1.12	0.09	0.472	0.547	0.505

Table 8
Statistics on expenditure of time. The operations of pre-process and non-maximum suppression make up the majority of inference time cost.

Modules	**stage**	**percentage**
PGN	pre-process	14.7%
PGN	forward	13.8%
PGN	get top k	2.0%
PGN	remap	0.3%
DecDet	pre-process	29.6%
DecDet	forward	15.0%
DecDet	nms	17.0%
DecDet	remap	1.7%
DecDet	final nms	5.9%
Total	–	100%

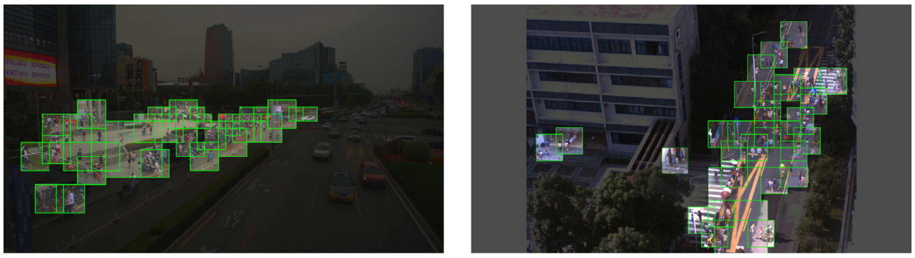

(a) (b)

Fig. 6. Example of anchor setting and patches selection. Different sizes of anchors are placed on the whole image. Each anchor is responsible for one value *CountO* that represents the number of valid objects contained. From the visualized schematic diagram we can find that, anchors in large area of the road (which is not marked due to its low *CountO* value) are not able to be selected, since there is really no person there. For the areas around streets and buildings where crowds gather, anchors get high *CountO* value. We notice that in the distant view anchors with large size do not get higher score since objects of extremely small size are not valid objects.

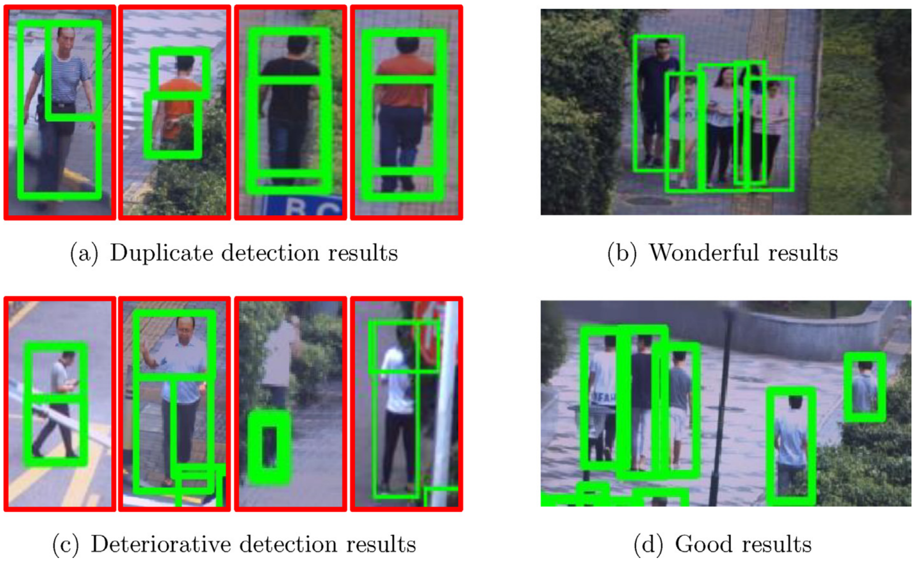

(a) Duplicate detection results
(b) Wonderful results
(c) Deteriorative detection results
(d) Good results

Fig. 7. Examples of fragmentary results. In the visualized images, we can find some misshapen boxes, which are caused by the boundaries of generated patches. To resolve the issue of occlusion and make the detector more robust, classical detectors often regard objects as a positive case if the IoU of them and the ground truth is greater than 0.5. This implicit logic introduces fragmentary detection results for objects located at the edge of two patches.

Larger input size requires longer inference time cost, and helps improving *AP*.50 performance especially in small objects, which is accord with the common sense. When the input size reaches a certain range like 480 or 640, the *AP*.50 value will not increase.

4.4.3. w/o direct detection

One simple optional action can be added is the direct detection for original GigaPixel-level image. The image can be resized into a size which can be loaded into the GPU memory, and then fed to the decorated detector directly. Usually the sizes of objects in original image are quite small and it is difficult to get effective results. We compare the results with or without option of direct detection, and demonstrate that a progressive detection procedure is inevitable. Results are listed in Table 6. The notation speed means the inference time cost per image.

4.4.4. patch IoU threshold

We test different IoU thresholds during the conditional selection strategy for patch generation. Results are shown in Table 7. The notation speed means the inference time cost per image. IoU T represents the hyper-parameter threshold for patches used in patch generation. The lower the threshold is, the denser the distribution of patches will be. It is possible for several selected patches to cover the same objects simultaneously. On the contrary, objects may be missed for later detection process at a strict IoU threshold.

4.5. Meticulous study on inference time

We construct various models via GigaDet with different experimental settings, and their performance and efficiency can be viewed in Fig. 1. Besides, to make it clear the bottleneck of the acceleration, we record elaborate statistics on expenditure of time on each step during the inference procedure. The details can be viewed in Table 8.

From the data in Table 8 we can find that the cost for image preprocessing is dominant, and it makes sense to process the high-capacity pixels a GigaPixel-level image contains. Model forward steps for PGN and DecDet are no longer the bottleneck that slows down the inference. If we want it to be faster, maybe the steps on normal image processing should be paid attention to in the future.

4.6. Disadvantages analysis

Although the GigaDet framework achieves a great improvement about detection speed on GigaPixel-level images, there are still many problems left. Above all, the training procedure has not been combined in an end-to-end framework, which introduces many susceptible factors which need to be designed carefully. The weakness of detecting small targets has not been overcome thoroughly, and that leaves a lot of room for improvement. A more serious problem is that the bounding boxes of detection are constrained by the boundary of patches, which yields many fragmentary results. There are many negative examples in Fig. 5 and Fig. 7. Besides, it is not enough to excavate and utilize the relationship between abundant annotations, such as the labels for full body and head of one person are logically dependent on each other. These disadvantages can be addressed in the future, and we hope the research will be helpful to the real-time detection for GigaPixel-level videos.

5. Conclusion

In this paper, we address the problem of time-consuming object detection in GigaPixel-level images. We propose a fast framework, named GigaDet, to detect objects with large scale variation in the images. The procedure of scanning, detection, and collection is used to handle the dilemma normal detectors will encounter and GigaDet can be adjusted flexibly to make a trade-off between precision and efficiency. We boost the inference speed significantly in PANDA dataset while maintaining acceptable precision performance. Various experiments for components in the GigaDet have been studied. The inference costs at different steps are discussed in detail, which may help the optimization in future work.

CRediT authorship contribution statement

Kai Chen: Conceptualization, Methodology, Software, Writing - original draft, Visualization. **Zerun Wang:** Resources, Writing - review & editing, Data curation, Formal analysis. **Xueyang Wang:** Resources, Writing - review & editing. **Dahan Gong:** Validation. **Longlong Yu:** Investigation. **Yuchen Guo:** Writing - review & editing, Resources, Supervision, Project administration, Funding acquisition. **Guiguang Ding:** Resources, Supervision, Project administration, Funding acquisition.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] M. Everingham, L. Van Gool, C.K. Williams, J. Winn, A. Zisserman, The pascal visual object classes (voc) challenge, Int. J. Comput. Vis. 88 (2) (2010) 303–338.

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

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

[4] J. Dai, Y. Li, K. He, J. Sun, R-fcn: Object detection via region-based fully convolutional networks, arXiv preprint arXiv:1605.06409 (2016). .

[5] T.-Y. Lin, P. Goyal, R. Girshick, K. He, P. Dollár, Focal loss for dense object detection, in: Proceedings of the IEEE international conference on computer vision, 2017, pp. 2980–2988.

[6] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: Unified, real-time object detection, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 779–788.

[7] A. Bochkovskiy, C.-Y. Wang, H.-Y.M. Liao, Yolov4: Optimal speed and accuracy of object detection, arXiv preprint arXiv:2004.10934 (2020). .

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

[9] B. Wu, F. Iandola, P.H. Jin, K. Keutzer, Squeezedet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2017, pp. 129–137. .

[10] Z. Tian, C. Shen, H. Chen, T. He, Fcos: Fully convolutional one-stage object detection, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 9627–9636.

[11] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, Q. Tian, Centernet: Keypoint triplets for object detection, in: Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 6569–6578.

[12] H. Law, J. Deng, Cornernet: Detecting objects as paired keypoints, in: Proceedings of the European conference on computer vision (ECCV), 2018, pp. 734–750..

[13] M. Cristani, R. Raghavendra, A. Del Bue, V. Murino, Human behavior analysis in video surveillance: a social signal processing perspective, Neurocomputing 100 (2013) 86–97.

[14] Y. Liu, D. Zhang, Q. Zhang, J. Han, Part-object relational visual saliency, IEEE Trans. Pattern Anal. Mach. Intell. (2021).

[15] X. Wang, X. Zhang, Y. Zhu, Y. Guo, X. Yuan, L. Xiang, Z. Wang, G. Ding, D. Brady, Q. Dai, et al., Panda: A gigapixel-level human-centric video dataset, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 3268–3278. .

[16] J. Redmon, A. Farhadi, Yolov3: An incremental improvement, arXiv preprint arXiv:1804.02767 (2018). .

[17] Z. Cai, N. Vasconcelos, Cascade r-cnn: Delving into high quality object detection, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 6154–6162.

[18] Q. Zhang, M. Zhang, T. Chen, Z. Sun, Y. Ma, B. Yu, Recent advances in convolutional neural network acceleration, Neurocomputing 323 (2019) 37–51.

[19] Y. Miao, Z. Lin, X. Ma, G. Ding, J. Han, Learning transformation-invariant local descriptors with low-coupling binary codes, IEEE Trans. Image Process. (2021).

[20] Q. Zhang, N. Huang, L. Yao, D. Zhang, C. Shan, J. Han, Rgb-t salient object detection via fusing multi-level cnn features, IEEE Trans. Image Process. 29 (2019) 3321–3335.

[21] D. Brockmann, T. Geisel, The ecology of gaze shifts, Neurocomputing 32 (2000) 643–650.

[22] G. Ciaparrone, F.L. Sánchez, S. Tabik, L. Troiano, R. Tagliaferri, F. Herrera, Deep learning in video multi-object tracking: a survey, Neurocomputing 381 (2020) 61–88.

[23] X. Wu, D. Sahoo, S.C. Hoi, Recent advances in deep learning for object detection, Neurocomputing 396 (2020) 39–64.

[24] P. Viola, M. Jones, Rapid object detection using a boosted cascade of simple features, in: Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001, vol. 1, IEEE, 2001, pp. I-I. .

[25] R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2014, pp. 580–587.

[26] R. Girshick, Fast r-cnn, in: Proceedings of the IEEE international conference on computer vision, 2015, pp. 1440–1448. .

[27] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in: 2009 IEEE conference on computer vision and pattern recognition, Ieee, 2009, pp. 248–255. .

[28] M. Tan, R. Pang, Q.V. Le, Efficientdet: Scalable and efficient object detection, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 10781–10790.

[29] C. Zhu, F. Chen, Z. Shen, M. Savvides, Soft anchor-point object detection, arXiv preprint arXiv:1911.12448 (2019)..

[30] Z. Yang, S. Liu, H. Hu, L. Wang, S. Lin, Reppoints: Point set representation for object detection, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 9657–9666.

[31] H. Qiu, Y. Ma, Z. Li, S. Liu, J. Sun, Borderdet: Border feature for dense object detection, in: European Conference on Computer Vision, Springer, 2020, pp. 549–564. .

[32] Y. Li, Y. Chen, N. Wang, Z. Zhang, Scale-aware trident networks for object detection, in: Proceedings of the IEEE international conference on computer vision, 2019, pp. 6054–6063.

[33] E.H. Adelson, C.H. Anderson, J.R. Bergen, P.J. Burt, J.M. Ogden, Pyramid methods in image processing, RCA Engineer 29 (6) (1984) 33–41.

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

[35] M. Kisantal, Z. Wojna, J. Murawski, J. Naruniec, K. Cho, Augmentation for small object detection, arXiv preprint arXiv:1902.07296 (2019). .

[36] R. LaLonde, D. Zhang, M. Shah, Clusternet: Detecting small objects in large scenes by exploiting spatio-temporal information, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 4003–4012. .

[37] F. Yang, H. Fan, P. Chu, E. Blasch, H. Ling, Clustered object detection in aerial images, in: Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 8311–8320.

[38] X. Yuan, L. Fang, Q. Dai, D.J. Brady, Y. Liu, Multiscale gigapixel video: a cross resolution image matching and warping approach, in: 2017 IEEE International Conference on Computational Photography (ICCP), IEEE, 2017, pp. 1–9.

[39] Y. Wang, L. Wang, H. Lu, Y. He, Segmentation based rotated bounding boxes prediction and image synthesizing for object detection of high resolution aerial images, Neurocomputing 388 (2020) 202–211.

[40] J. Zhou, C.-M. Vong, Q. Liu, Z. Wang, Scale adaptive image cropping for uav object detection, Neurocomputing 366 (2019) 305–313.

[41] J. Gao, Q. Wang, Y. Yuan, Scar: Spatial-/channel-wise attention regression networks for crowd counting, Neurocomputing 363 (2019) 1–8.

[42] Z. Li, M. Dong, S. Wen, X. Hu, P. Zhou, Z. Zeng, Clu-cnns: Object detection for medical images, Neurocomputing 350 (2019) 53–59.

[43] M. Najibi, B. Singh, L.S. Davis, Autofocus: Efficient multi-scale inference, in: Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 9745–9755. .

[44] B. Singh, M. Najibi, L.S. Davis, Sniper: Efficient multi-scale training, in: Advances in neural information processing systems, 2018, pp. 9310–9320. .

[45] M. Gao, R. Yu, A. Li, V.I. Morariu, L.S. Davis, Dynamic zoom-in network for fast object detection in large images, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 6926–6935.

[46] A. Neubeck, L. Van Gool, Efficient non-maximum suppression, in: 18th International Conference on Pattern Recognition (ICPR'06), vol. 3, IEEE, 2006, pp. 850–855. .

[47] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al., Pytorch: An imperative style, high-performance deep learning library, in: Advances in neural information processing systems, 2019, pp. 8026–8037. .

[48] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556 (2014). .

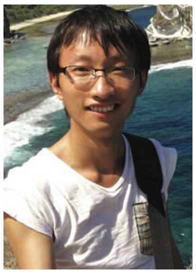
Kai Chen received his M. Sc. degree and B. Sc. degree from school of Software, Tsinghua University, Beijing, China in 2017 and 2014 respectively. He is a Ph. D. candidate in School of Software, Tsinghua University, Beijing, China. His research interests focuses on visual perception tasks via deep learning methods, including object detection, person re-identification, etc.

Zerun Wang received his B. Sc. degree from School of Software, Tsinghua University, Beijing, China in 2019. Now he is a master candidate in School of Software, Tsinghua University, Beijing, China. His research interests focuses on deep learning-based visual tasks including object detection, image restoration, etc.

Xueyang Wang is currently a master student in Tsinghua-Berkeley Shenzhen Institute (TBSI), Tsinghua University. He received B.E. from Xi'an Jiaotong University in 2019. His research interest is computer vision.

Dahan Gong received the B.S. and M.S. degree from School of Software, Tsinghua University, Beijing, China in2015 and 2018 respectively, and currently is a Ph.D. student in the same department. His research interests include computer vision and inference optimization.

Longlong Yu Graduated as an undergraduate from School of Software, Tsinghua University, Beijing, China in 2013. He was an Engineer in YQ Education & Technology Group Inc, between 2013 and 2017. Now he is in charge of artificial intelligence research in his own startup company.



Yuchen Guo received his Ph. D. degree and B. Sc. degree from School of Software, Tsinghua University, Beijing, China in 2018 and 2013 respectively. He was a Postdoc researcher in Department of Automation, Tsinghua Univerisity, between 2018 and 2020. Now he is an assistant researcher in Beijing National Research Center for Information Science and Technology, Tsinghua University. His research interests focuses on brain inspired artificial intelligence.

Guiguang Ding is currently an Associate Professor with the School of Software, Tsinghua University, China. His research interests include the areas of multimedia information retrieval, computer vision, and machine learning.