INTEGRATING NEAR AND LONG-RANGE EVIDENCE FOR VISUAL DETECTION

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

NERMİN SAMET

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
COMPUTER ENGINEERING

SEPTEMBER 2021

Approval of the thesis:

INTEGRATING NEAR AND LONG-RANGE EVIDENCE FOR VISUAL DETECTION

submitted by **NERMİN SAMET** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Halit Oğuztüzün
Head of Department, **Computer Engineering** _____

Assist. Prof. Dr. Emre Akbaş
Supervisor, **Computer Engineering, METU** _____

Examining Committee Members:

Assoc. Prof. Dr. Sinan Kalkan
Computer Engineering, METU _____

Assist. Prof. Dr. Emre Akbaş
Computer Engineering, METU _____

Prof. Dr. Pinar Duygulu Şahin
Computer Engineering, Hacettepe University _____

Assoc. Prof. Dr. Erkut Erdem
Computer Engineering, Hacettepe University _____

Assist. Prof. Dr. Gökberk Cinbiş
Computer Engineering, METU _____

Date:08.09.2021

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Nermin Samet

Signature :

ABSTRACT

INTEGRATING NEAR AND LONG-RANGE EVIDENCE FOR VISUAL DETECTION

Samet, Nermin
Ph.D., Department of Computer Engineering
Supervisor: Assist. Prof. Dr. Emre Akbaş

September 2021, 124 pages

This thesis presents HoughNet, a one-stage, anchor-free, voting-based, bottom-up object detection method. Inspired by the Generalized Hough Transform, HoughNet determines the presence of an object at a certain location by the sum of the votes cast on that location. Votes are collected from both near and long-distance locations based on a log-polar vote field. Thanks to this voting mechanism, HoughNet is able to integrate both near and long-range, class-conditional evidence for visual recognition, thereby generalizing and enhancing current object detection methodology, which typically relies on only local evidence. On the COCO dataset, HoughNet's best model achieves 46.4 AP (and 65.1 AP_{50}), performing on par with the state-of-the-art in bottom-up object detection and outperforming most major one-stage and two-stage methods. We further validate the effectiveness of our proposal in other visual detection tasks, namely, video object detection, instance segmentation, 3D object detection, keypoint detection for human pose estimation and whole-body human pose estimation, face detection and an additional "labels to photo" image generation task, where the integration of our voting module consistently improves performance in all cases.

In order to show the effectiveness of our proposal on whole-body human pose estima-

tion task, we developed a bottom-up, one-stage method called HPRNet. In HPRNet, we build a hierarchical regression mechanism, where we define each of the whole-body keypoints with a relative location (i.e. offset) to a specific point on the person box.

In the context of this thesis we also propose a one-stage, anchor-free object detector, PPDet, which integrates short-range interactions through voting. PPDet sum-pools predictions stemming from individual features into a single prediction which allows the model to reduce the contributions of non-discriminatory features during training.

Keywords: Object detection, voting, bottom-up recognition, Hough Transform, video object detection, instance segmentation, 3D object detection, human pose estimation, whole-body human pose estimation, face detection, image-to-image translation, label-to-image translation

ÖZ

GÖRSEL TANIMA PROBLEMLERİNE YAKIN VE UZUN MESAFELİ KANITLARIN ENTEGRE EDİLMESİ

Samet, Nermin
Doktora, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Dr. Öğr. Üyesi. Emre Akbaş

Eylül 2021 , 124 sayfa

Bu tez, tek-aşamalı, sınırlayıcı kutu içermeyen, oylamaya dayalı, aşağıdan-yukarıya nesne tanıma yöntemi olan HoughNet'i sunar. Genelleştirilmiş Hough Dönüşümü'nden esinlenen HoughNet, belirli bir konumdaki bir nesnenin varlığını, o konuma verilen oyların toplamına göre belirler. Oylar, log-polar oy alanına dayalı olarak hem yakın hem de uzak mesafelerden toplanır. Bu oylama mekanizması sayesinde, HoughNet görsel tanıma için hem yakın hem de uzun mesafelı, sınıf koşullu kanıtları entegre edebilir, böylece tipik olarak yalnızca yerel kanıtlara dayanan mevcut nesne algılama metodolojisini genelleştirir ve geliştirir. COCO veri kümesinde, HoughNet'in en iyi modeli 46.4 AP (ve 65.1 AP_{50}) elde ederek aşağıdan-yukarıya nesne tanıma yöntemleri ile benzer seviyede başarım göstermiş ve bir çok ana tek-aşamalı ve iki-aşamalı nesne tanıma yöntemlerini geride bırakmıştır. Önerdiğimiz yöntemin etkinliğini diğer görsel tanıma problemlerinde, yani videolarda nesnesi tanıma, nesne bölütleme, 3B nesne tanıma, insan pozisyon kestirimi, tüm-vücut insan pozisyon kestirimi, yüz tanıma ve ek olarak "etiketten fotoğrafa" görüntü oluşturma probleminde doğruladık. Buna göre, oylama modülümüz entegre edildiği her durumda performansı sürekli ola-

rak iyileştirmiştir.

Önerimizin tüm-vücut insan pozisyon kestirimi için etkinliğini göstermek için HPR-Net adını verdiğimiz aşağıdan-yukarıya tek-aşamalı bir yöntem geliştirdik. HPR-Net'te, tüm-vücut ana noktalarının her birini, insan sınırlayıcı kutu üzerindeki belirli noktalara göreli bir konumla tanımladığımız hiyerarşik bir regresyon mekanizması oluşturuyoruz.

Bu tez bağlamında ayrıca, oylama yoluyla kısa mesafeli etkileşimleri entegre eden, tek-aşamalı, sınırlayıcı kutu içermeyen bir nesne tanıma yöntemi olan PPDet'i öneriyoruz. PPDet, tekil özniteliklerden elde edilen tahminleri tek bir tahminde toplar, bu sayede eğitim sırasında ayırt edici olmayan özniteliklerin katkılarının azaltmasına olanak tanır.

Anahtar Kelimeler: Nesne tanıma, oylama, aşağıdan-yukarıya nesne tanıma, Hough Dönüşümü, videolarda nesne tanıma, nesne bölütleme, 3B nesne tanıma, insan pozisyon kestirimi, tüm-vücut insan pozisyon kestirimi, yüz tanıma, görüntüden görüntüye çeviri, etiketten görüntüye çeviri

This thesis is dedicated to Mustafa Kemal ATATÜRK.

ACKNOWLEDGMENTS

This thesis is being written during the tail end of the COVID19 pandemic with the support of the great people in my life.

I cannot thank my supervisor Assist. Prof. Dr. Emre Akbas enough for all of his guidance, knowledge, help and support over the years. I have learned so much from him. His acute insight brought my research to a higher level.

I would like to thank my doctoral thesis monitoring committee members, Assoc. Prof. Dr. Sinan Kalkan and Assoc. Prof. Dr. Erkut Erdem, for their valuable discussions and feedback throughout my studies.

I would also like to thank my thesis defense jury members, Prof. Dr. Pinar Duygulu and Assist. Prof. Dr. Gokberk Cinbis for accepting to review my thesis.

My warmest thanks also go to Prof. Dr. Fatos Yarman Vural and my labmates at METU ImageLab for the joyful moments we had in the lab.

Finally, my most sincere thanks go out to my family, my friends - Hunlar, and lastly but not least, to Samet, my best friend and love who shared every moment of struggle, fail, success and excitement with me on this journey.

This work was supported by the Scientific and Technological Research Council of Turkey (TÜBİTAK) through the project titled “Object Detection in Videos with Deep Neural Networks" (grant #117E054). The numerical calculations reported in this paper were partially performed at TÜBİTAK ULAKBİM, High Performance and Grid Computing Center (TRUBA resources). I also gratefully acknowledge the support of the AWS Cloud Credits for Research program.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

2D	2 Dimensional
3D	3 Dimensional
AOS	Average Orientation Score
AP	Average Precision
BEV	Bird-eye-view Bounding Box
BKH	Body Keypoint Heatmap
CNN	Convolutional Neural Network
DLA	Deep Layer Aggregation
FCN	Fully Convolutional Networks
FL	Focal Loss
FN	False Negative
FP	False Positive
FPN	Feature Pyramid Network
FPS	Frame per Second
GHT	Generalized Hough Transform
GT	Ground-truth
HG	Hourglass-104
HM-I	Hierarchical Model-I
HM-II	Hierarchical Model-II
HPRNet	Hierarchical Point Regression Network
IoU	Intersection over Union
ISM	Implicit Shape Model
LPIPS	Learned Perceptual Image Patch Similarity

LRP	Localization Recall Precision
mAP	Mean Average Precision
moLRP	Mean Optimal LRP
MS	Multi-scale
NLNN	Non-local Neural Networks
NMS	Non-maximum Supression
OAP	Objects as Points
PAF	Part Affinity Fields
PCH	Person Center Heatmap
PPDet	Prediction Pooling Detector
RL	Regression Loss
RN	Relation Networks
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent
SN	Single Network
SOTA	State-of-the-art
SS	Single-scale
VEP	Visual Evidence Prediction

CHAPTER 1

INTRODUCTION

This chapter is adopted from our ECCV'20 paper [2] and its extension [3].

Deep learning has brought on remarkable improvements in object detection. Performance on widely used benchmark datasets, as measured by mean average-precision (mAP), has at least doubled (from 0.33 mAP [4] [5] to 0.80 mAP on PASCAL VOC [6]; and from 0.2 mAP [7] to around 0.5 mAP on COCO [8]) in comparison to the pre-deep-learning, shallow methods. Current state-of-the-art, deep learning based object detectors [9, 10, 11, 8] predominantly follow a top-down approach where objects are detected holistically via rectangular region classification. This was not the case with the pre-deep-learning methods. The bottom-up approach was a major research focus as exemplified by the prominent voting-based (the Implicit Shape Model [12]) and part-based (the Deformable Parts Model [13]) methods. However, today, among deep learning based object detectors, the bottom-up approach has not been sufficiently explored with a few exceptions (e.g. CornerNet [14], ExtremeNet [15]).

1.1 Problem Definition and Scope of the Thesis

In this thesis, we propose HoughNet, a one-stage, anchor-free, voting-based, bottom-up object detection method. HoughNet is based on the idea of voting, inspired by the Generalized Hough Transform [16, 17]. In its most generic form, the goal of GHT is to detect a whole shape based on its parts. Each part produces a hypothesis, i.e. casts its vote, regarding the location of the whole shape. Then, the location with the most votes is selected as the result. Similarly, in HoughNet, the presence of an object belonging to a certain class at a particular location is determined by the sum of the

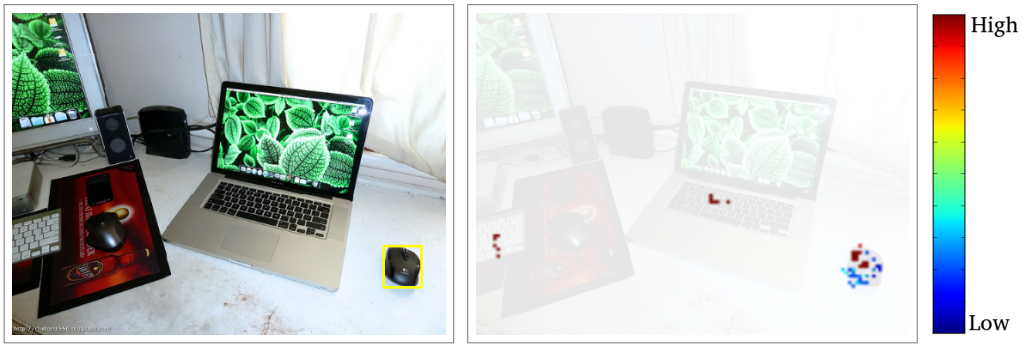


Figure 1.1: (Left) A sample "mouse" detection, shown with yellow bounding box, by HoughNet. (Right) The locations that vote for this detection. Colors indicate vote strength. In addition to the local votes originating from the mouse itself, there are strong votes from nearby "keyboard" objects, which shows that HoughNet is able to utilize both short and long-range evidence for detection. More examples can be seen in Figure 4.2. Image is taken from our ECCV'20 paper [2].

class-conditional votes cast on that location (Figure 1.1). HoughNet processes the input image using a convolutional neural network to produce an intermediate score map per class. Scores in these maps indicate the presence of visual structures that would support the detection of an object instance. These structures could be object parts, partial objects or patterns belonging to the same or other classes. We name these score maps as "visual evidence" maps. Each spatial location in a visual evidence map votes for target areas that are likely to contain objects. Target areas are determined by placing a log-polar grid, which we call the "vote field," centered at the voter location. The purpose of using a log-polar vote field is to reduce the spatial precision of the vote as the distance between voter location and target area increases. This is inspired by foveated vision systems found in nature, where the spatial resolution rapidly decreases from the fovea towards the periphery [18]. Once all visual evidence is processed through voting, the accumulated votes are recorded in object presence maps, where the peaks indicate the presence of object instances.

Current state-of-the-art object detectors rely on local or short-range visual evidence to decide whether there is an object at that location (as in top-down methods) or an important keypoint such as a corner (as in bottom-up methods). On the other hand, HoughNet is able to integrate both short and long-range visual evidence at

Figure 1.2: (Left) A sample "baseball bat" detection, shown with yellow bounding box, by HoughNet. (Right) The locations that vote for this detection. Colors indicate vote strength. A ball on the right-edge of the image is voting for the baseball bat on the left-edge. Image is taken from our ECCV'20 paper [2].

the same time through voting. An example is illustrated in Figure 1.1, where the detected mouse gets strong votes from two keyboards, one of which is literally at the other side of the image. In another example (Figure 1.2), a ball on the right-edge of the image is voting for the baseball bat on the left-edge. On the COCO dataset, HoughNet achieves comparable results with the state-of-the-art bottom-up detector CenterNet [19], while being the fastest among bottom-up detectors. It outperforms prominent one-stage (RetinaNet [8]) and two-stage detectors (Faster RCNN [9], Mask RCNN [20]). Within the scope of this study, we published the following paper:

- N. Samet, S. Hicsonmez, E. Akbas, "HoughNet: Integrating near and long-range evidence for bottom-up object detection", European Conference on Computer Vision (ECCV), 2020.

During model development of HoughNet to deal with the large number of training experiments and ablation studies, we curated "COCO `minitrain`", a mini training set for COCO. We validated COCO `minitrain` in two ways by (i) showing that the COCO `val2017` performance of a model trained on COCO `minitrain` is strongly positively correlated with the performance of the same model trained on COCO `train2017`, and (ii) showing that COCO `minitrain` set preserves the object instance statistics.

We showed that our Hough voting module is effective not only in object detection but

also in other visual detection tasks as well: video object detection (Sections 3.4 and 4.3.1), instance segmentation (Section 4.3.2), human pose estimation, i.e. keypoint detection (Section 4.3.4), whole-body human pose estimation (Section 4.3.5), 3D object detection (Section 4.3.3) and face detection (Section 4.3.6). For video object detection we extended voting in spatial domain to the temporal domain by developing a new method, which takes the difference of features from two frames, and applies spatial and temporal voting using our "temporal voting module" to detect objects. We showed the effectiveness of this method in video object detection (Sections 4.3.1). We also developed a "scalable" variant of HoughNet where the number of voting operations does not depend on the number of object classes (Section 4.2.5) and showed that it dramatically improves inference speed with a slight drop in accuracy. Furthermore, we examined the relations between vote-giver and vote-getter classes (Section 4.2.4), and evaluated HoughNet's performance using localisation, recall, precision (LRP) metrics [21, 22] to identify and compare the source of errors. Our extended work on HoughNet is currently under review at a reputable journal.

1.1.1 Prediction Pooling Detector

While working on HoughNet, we also developed another object detection model, PPDet, which integrates only short-range interactions through voting (Chapter 5). PPDet effectively reduces the label noise during training. Current anchor-free object detectors label all the features that spatially fall inside a predefined central region of a ground-truth box as positive. This approach causes label noise during training, since some of these positively labeled features may be on the background or an occluder object, or they are simply not discriminative features. In PPDet, we propose a new labeling strategy aimed to reduce the label noise in anchor-free detectors. We sum-pool predictions stemming from individual features into a single prediction. This allows the model to reduce the contributions of non-discriminatory features during training. We develop a new one-stage, anchor-free object detector, PPDet, to employ this labeling strategy during training and a similar prediction pooling method during inference. On the COCO dataset, PPDet achieves the best performance among anchor-free top-down detectors and performs on-par with the other state-of-the-art methods. It also outperforms all major one-stage and two-stage methods in small ob-

ject detection (APS 31.4). Within the scope of this study, we published the following paper:

- N. Samet, S. Hicsonmez, E. Akbas, "Reducing Label Noise in Anchor-Free Object Detection", British Machine Vision Conference (BMVC), 2020.

In both HoughNet (ECCV'20) and PPDet (BMVC'20) publications, there was a third author: Samet Hicsonmez, who is a PhD candidate at the Department of Computer Engineering, Hacettepe University. He was responsible for preparing visual results and managing experimental runs, in both publications.

1.1.2 Hierarchical Point Regression for Whole-Body Human Pose Estimation

One task we show the effectiveness of our voting module is whole-body pose estimation. Whole-body pose estimation task is a very recent task [23]. Even though existing two-stage methods obtains state-of-the-art results, their run time increases as the number of person instances increase, and existing one-stage methods performs poorly. Based on this fact, we first developed HPRNet (Chapter 6). In HPRNet, we present a new bottom-up one-stage method for whole-body pose estimation, which we call "hierarchical point regression," or HPRNet for short. In standard body pose estimation, the locations of ~ 17 major joints on the human body are estimated. Differently, in whole-body pose estimation, the locations of fine-grained keypoints (68 on face, 21 on each hand and 3 on each foot) are estimated as well, which creates a scale variance problem that needs to be addressed. To handle the scale variance among different body parts, we build a hierarchical point representation of body parts and jointly regress them. The relative locations of fine-grained keypoints in each part (e.g. face) are regressed in reference to the center of that part, whose location itself is estimated relative to the person center. In addition, unlike the existing two-stage methods, our method predicts whole-body pose in a constant time independent of the number of people in an image. On the COCO WholeBody dataset, HPRNet significantly outperforms all previous bottom-up methods on the keypoint detection of all whole-body parts (i.e. body, foot, face and hand); it also achieves state-of-the-art results on face (75.4 AP) and hand (50.4 AP) keypoint detection. Within the scope of

this study, we published the following paper:

- Nermin Samet, Emre Akbas, "HPRNet: Hierarchical point regression for whole-body human pose estimation", Image and Vision Computing, Volume 115, 2021, 104285, ISSN 0262-8856, https://doi.org/10.1016/j.imavis.2021.104285.

1.2 Contributions

Our contributions in this thesis are as follows:

- We developed HoughNet, a novel voting-based detection method that is able to integrate near and long-range evidence.
- We extended HoughNet and showed that the voting mechanism of HoughNet is effective for other visual detection tasks as well: instance segmentation, 3D object detection, keypoint detection for human pose estimation and whole-body human pose estimation, face detection and “labels to photo” image generation task.
- We developed a spatio-temporal voting module and showed its effectiveness on video object detection task.
- We developed a novel object detection model PPDet and showed its effectiveness on reducing label noise in top-down anchor-free object detection.
- We developed a bottom-up one stage whole-body human pose estimation model called HPRNet.

1.3 The Outline of the Thesis

Chapter 2 explains Generalized Hough Transform and presents the previous works related to non-deep voting-based object detection methods, deep voting-based object detection methods, bottom-up object detection methods, log-polar fields/representations and context modeling in object detection. In Chapter 3, we describe our method

and explain the details of the log-polar "vote field", voting module, our network architecture, spatio-temporal voting process. In Chapter 4, we show the effectiveness of our proposed method on object detection and other visual detection tasks, namely, video object detection, instance segmentation, 3D object detection, keypoint detection for 2D human pose estimation and 2D whole-body human pose estimation, and face detection. We mostly focus on object detection and include (i) ablation experiments through which we studied how different parameters of the vote field affect the performance, (ii) comparison with baseline, (iii) comparison with state-of-the-art, (iii) an analysis section and (iv) a scalable voting approach where the number of "visual evidence tensors" does not depend on the number of object classes. We also include an image generation task in the context of "label-to-photo" translation at the end. Chapter 5 describe PPDet and Chapter 6 presents HPRNet in detail. Finally, Chapter 7 concludes the thesis.

Chapter 1, Chapter 2, Chapter 3 and Chapter 4 are adopted from our ECCV'20 paper [2] and its extension [3]. Chapter 6 is based on our BMVC'20 paper [24] and Chapter 6 is based on our IMAVIS journal paper [25].

CHAPTER 2

RELATED WORK

This chapter is adopted from our ECCV'20 paper [2] and its extension [3].

2.1 Generalized Hough Transform

Hough transformation is a voting scheme that was first developed to detect analytical features such as lines, circles, ellipses [16]. For this purpose, parameter space is discretized into boxes. Each feature of the image, votes for the boxes in the parameter space that likely to create itself. A line in the image space corresponds to a point in the Hough space (see Figure 2.1). Since the values that the parameters can get are not limited in the Cartesian space and the vertical lines are faced with the infinite slope problem, the votes are collected in the polar coordinate system in the Hough space. Figure 2.2 shows the collected votes from the points on two lines in the Hough space. The regions that did not get any votes are black, while the points that collected the most votes are observed as bright.

The Hough Transformation was later expanded to the generalized Hough Transform (GHT) [17] to be used for the detection of arbitrary shapes. Generalized Hough Transform is essentially a method for object recognition and consists of the following steps; 1) a reference point is selected on the object, 2) the displacement vectors between the selected reference point and the points on the boundary of the object, and the angle θ (gradient direction) are calculated (see Figure 2.3), 3) in the last step, the calculated displacement vectors are indexed with the angle θ and stored in a table. With the help of this table, we switch to the Hough space.

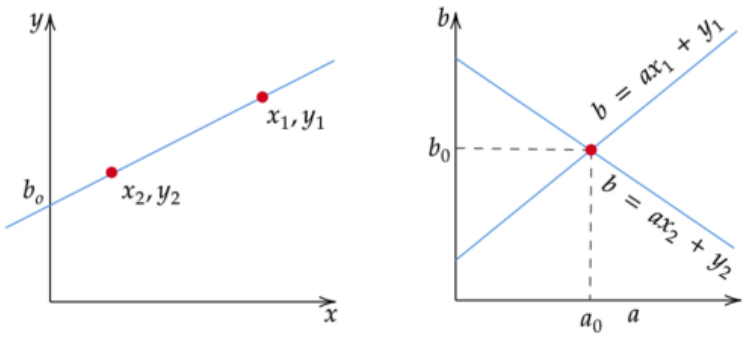


Figure 2.1: A line in the image space corresponds to a point in the Hough space. Image source: `https://towardsdatascience.com/lines-detection-with-hough-transform-84020b3b1549`

The GHT model is noise tolerant and robust to partial or slightly deformed shapes (i.e., robust to recognition under occlusion). The Hough transform is also robust to scale variations and rotations.

2.2 Non-deep, Voting-based Object Detection Methods

In the pre-deep learning era, Generalized Hough Transform (GHT) based voting methods have been used for object detection. The most influential work was the Implicit Shape Model (ISM) [12]. In ISM, Leibe et al. [12] applied GHT for object detection/recognition and segmentation. During the training of the ISM, first, interest points are extracted and then a visual codebook (i.e. dictionary) is created using an unsupervised clustering algorithm applied on the patches extracted around interest points. Next, the algorithm matches the patches around each interest point to the visual word with the smallest distance. In the last step, the positions of the patches relative to the center of the object are associated with the corresponding visual words and stored in a table. During inference, patches extracted around interest points are matched to closest visual words. Each matched visual word casts votes for the object center. In the last stage, the location that has the most votes is identified, and object detection is performed using the patches that vote for this location. Later, ISM was further extended with discriminative frameworks [26, 27, 28, 29, 30]. Okada [26] ensembled randomized trees using image patches as voting elements. Similarly, Gall

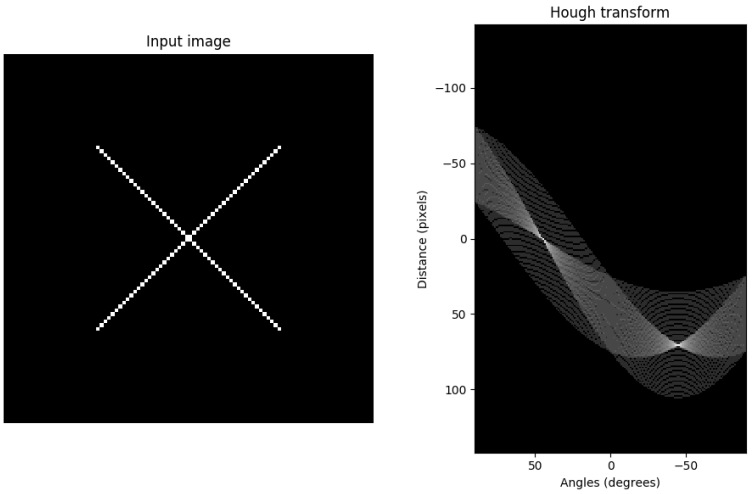


Figure 2.2: An example result of a Hough transform on a raster image containing two lines. Image source: `https://scikit-image.org/docs/0.11.x/auto_examples/plot_line_hough_transform.html`

and Lempitsky [27] proposed to learn a mapping between image patches and votes using random forests. In order to fix the accumulation of inconsistent votes of ISM, Razavi et al. [28] augmented the Hough space with latent variables to enforce consistency between votes. In Max-margin Hough Transform [29], Maji and Malik showed the importance of learning visual words in a discriminative max-margin framework. Barinova et al. [30] detected multiple objects using energy optimization instead of non-maxima suppression peak selection of ISM.

HoughNet is similar to ISM and its variants described above only at the idea level as all are voting based methods. There are two major differences: (i) HoughNet uses deep neural networks for part/feature (i.e. visual evidence) estimation, whereas ISM uses hand-crafted features; (ii) ISM uses a discrete set of visual words (obtained by unsupervised clustering) and each word's vote is exactly known (stored in a table) after training. In HoughNet, however, there is not a discrete set of words and vote is carried through a log-polar vote field which takes into account the location precision as a function of target area.

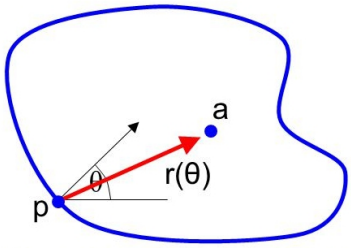


Figure 2.3: A shape defined by its boundary points and a reference point. For each point p on the boundary, we compute the displacement vector as a function of gradient orientation θ. Image source: `https://slideplayer.com/slide/5310314/`

2.3 Deep, Voting-based Object Detection Methods

Qi et al. [31] apply Hough voting for 3D object detection in point clouds. Sheshkus et al. [32] utilize Hough transform for vanishing points detection in the documents. For automatic pedestrian and car detection, Gabriel et al. [33] proposed using discriminative generalized Hough transform for proposal generation in edge images, later to further refine the boxes, they fed these proposals to deep networks. In the deep learning era, we are not the first to use a log-polar vote field in a voting-based model. Lifshitz et al. [34] used a log-polar map to estimate keypoints for single person human pose estimation. Apart from the fact that they are tackling a different problem (human pose estimation), there are several subtle differences. First, they prepare ground truth voting maps for each keypoint such that keypoints vote for every other one depending on its relative position in the log polar map. This requires manually creating static voting maps. Specifically, their model learns $H \times W \times R \times C$ voting map, where R is the number of bins and C is the augmented keypoints. In order to produce keypoint heatmaps, they perform vote agregation at test phase. Second, this design restricts the model to learn only the keypoint locations as voters. When we consider the object detection task and its complexity, it is not trivial to decide on vote-giving locations for objects or prepare ground-truth voting maps as in human pose estimation. Moreover, this design limits the voters to reside only inside of the object (e.g. person), whereas in our approach an object could get votes from far away regions. To overcome these issues, unlike their model we apply vote aggregation during training (they perform

vote agregation only at test phase). This allows us to expose the latent patterns between objects and voters for each class. In this way, our voting module is able to get votes from non-labeled objects (e.g. "candle", which is not a COCO class, voting for dining table; see the last row of Figure 4.2). To the best of our knowledge, we are the first to use a log-polar vote field in a voting-based deep learning model to integrate the **long range interactions** for object detection.

2.4 Bottom-up Object Detection Methods

Apart from the classical one-stage [10, 35, 36, 37, 8] vs. two-stage [9, 20] categorization of object detectors, we can also categorize the current approaches into two: top-down and bottom-up. In the top-down approach [10, 35, 8, 9], a near-exhaustive list of object hypotheses in the form of rectangular boxes are generated and objects are predicted in a holistic manner based on these boxes. Designing the hypotheses space (e.g. parameters of anchor boxes) is a problem by itself [38]. Typically, a single template is responsible for the detection of the whole object. In this sense, recent anchor-free methods [39, 40] are also top-down. On the other hand, in the bottom-up approach, objects *emerge* from the detection of parts or sub-object structures. For example, in CornerNet [14], top-left and bottom-right corners of objects are detected first, and then, they are paired to form whole objects. Following CornerNet, ExtremeNet [15] groups extreme points (e.g. left-most, etc.) and center points to form objects. Together with corner pairs of CornerNet [14], CenterNet [19] adds center point to model each object as a triplet. HoughNet follows the bottom-up approach based on a voting strategy: object presence score is voted (aggregated) from a wide area covering short and long-range evidence.

2.5 Methods using Log-polar Fields and Representations

Many biological systems have foveated vision where the spatial resolution decreases from the fovea (point of fixation) towards the periphery. Inspired by this phenomenon, computer vision researchers have used log-polar fields for many different purposes including shape description [41], feature extraction [42] and foveated sampling/imaging

[43].

2.6 Context Modeling in Object Detection

Context has long been part of object detection and explored to improve object detection performance. Context modeling has many aspects such as scene-level features, object-to-scene relations and object-to-object relations. Even though, HoughNet is not a proper context model, it is relevant to the object-to-object relations aspect.

In context modeling, there are two common approaches based on the source of context: detection with scene level context and detection with instance-level object to object/scene relations.

2.6.1 Detection with Scene Level Context

In literature, scene level context works are divided into two; using local context and using global context. Local context refers to the context around the object. It is known that local context improves the detection performance. Today the success of most of the object detectors relies on the implicit use of local context. Further, in order to use local context effectively, the deep learning-based object detectors enlarge the receptive field of network, use a larger size of object proposals, and/or directly use proposal regions as context [44, 45, 46, 47, 48, 49, 50]. For example, Kim et al. [51] propose a context model that uses manually picked regions as context.

Before deep learning era, global context expressed as a statistical summary of the image that represents the scene like Gist [52]. In deep learning methods, in order to integrate global context, large receptive fields [53] and global pooling operations [54, 55] are used. As a different approach, Bell et al. [56] use RNNs to extract contextual information from an image.

2.6.2 Detection with Instance Level Context

Some recent works aim to exploit context by using the relationship between individual objects [5, 57, 58, 59, 60]. Chen and Gupta [59] use a memory module on Fast R-CNN to represent object-to-object relationships. Hu et. a [60] propose elation networks module to be integrated into a two-stage detector. Relation networks define multiple relations between each proposal so that each proposal can send messages to others. Thus, relation networks models object-object relations explicitly for proposal-based two-stage detectors. Similarly, Arbel et al. [61] extend Fast R-CNN to refine the score of the final proposal by using other proposals. Chen et al. [62] selects important context regions based on IoU criteria between proposal and the other regions.

The other popular approach is to use the dependencies between objects and scenes [63, 64]. Liu et al. [64] integrate Structure Inference Network (SIN) into Faster-RCNN. Their method uses both scene-level and instance-level context so that it combines the features of both objects and scene. Chu and Cai [65] attempt to improve performance of Faster-RCNN by refining proposal scores based on object relations and global scene context. Li et al. [55] use both local and global context. They use discriminative parts as local context to infer object classes. They claim that not all global regions are useful for object detection. Based on this assumption they extract positive global context using an attention model.

There are also studies work on the analysis of the importance of context [66, 67, 68]. Dvornik et al. [66]. analyze the effect of context by using a context-driven data augmentation method. The improved results using context-driven data augmentation prove that CNN based detectors implicitly use contextual information. They also show that when objects are detached from their context and randomly placed performance of object detector decreases. Mottaghi et al. [67], analysis role of context both for object detection and segmentation on PASCAL VOC 2010 dataset. They also purpose a novel deformable part-based model that considers global context and local context around candidate object regions. The proposed model especially effective for tiny objects and improves overall detection performance as well. Qiao et al. [68] approach the context differently. Instead of using contextual information for object detection, they predict missing scene context using category, shape and position in-

formation of standalone objects. Ignoring the fine details, their model reconstructs the realistic scenes from given objects.

Compared to HoughNet, (i) current context studies mostly developed to be integrated into two-stage object detectors as an extension which lead to more complex object detection models, and (ii) one-stage context modeling object detectors relies on networks with large receptive fields to capture context information at larger scales. However, Luo et al. [69] introduces the concept of "Effective Receptive Field" and showed that not all pixels in the receptive field contribute equally to the output unit's response (i.e. effect of the receptive field on output looks like Gaussian distribution not uniform distribution).

In contrast to CNN features that ignores object pose, Capsule Networks [70] learn to pay attention to object's pose and its other aspects as well. In Capsule Networks, groups of neurons encode spatial information and also they predict the probability of an object being present. Capsule networks look at an image and predict what the instantiation parameters for objects are. Capsule Networks use dynamic routing algorithms to estimate features of object poses such as position, size, orientation, deformation, velocity, hue, texture etc. At that level of idea, it is similar to HoughNet. HoughNet also learns the evidences to verify the existence of an object. Capsule Networks focus on the orientation of parts and build hierarchical relationships to identify images. For example, let's consider that we have five low-level features and the predictions of these five low-level features indicate the same orientation and 'car' object, then the 'car' representation will be the "high-level" feature. However, in HoughNet we do not build hierarchical representations of object poses to detect objects. HoughNet integrates high-level short and long-range evidences through voting and the importance of the votes are determined based on log-polar vote-field. Morover, even though Capsule Networks obtained state of the art performance on simple datasets such as MNIST, it struggles on more complex data such as Imagenet.

Similar to HoughNet, Non-local neural networks (NLNN) [71] integrate long-range features. As a fundamental difference, in NLNN, the relative displacement between interacting features is not taken into account. However, HoughNet uses this information encoded through the regions of the log-polar vote field.

Concurrently with HoughNet [72], several object detectors have been introduced [73, 74, 75]. Among them, the transformer-based DETR [73], in particular, shares with HoughNet the idea of using both short and long range interactions. In DETR, features at a specific location are updated through interactions with features at other locations using encoder-decoder based transformers [76]. Although locations of features are taken into account using positional encoding, DETR models all possible pairs of features. Unlike DETR, HoughNet explicitly takes into account the spatial precision of the vote depending on the relative displacement between voter location and target (voted) area. Although attention-based neural networks and HoughNet share the idea of using long-range interactions, an in-depth comparison is beyond the scope of this work. Briefly; there are a variety of ways to encode location in attention-based neural networks: (i) simply no encoding – location, whether absolute or relative, is ignored [71], (ii) using sine and cosine functions with varying frequencies [73, 76], (iii) using a general function [77]. It is not trivial to compare these encodings to the explicit encoding of relative location using a log-polar field in HoughNet. Another fundamental difference is that transformers operate at the feature level by modifying features through interactions with other features, whereas HoughNet operates at the detection score level.

CHAPTER 3

HOUGHNET: MODELS AND METHOD

This chapter is taken from our ECCV'20 paper [2] and its extension [3].

Brief overview. In HoughNet, the input image first passes through a backbone convolutional neural network (CNN), the output of which is connected to three different branches carrying out the predictions of (i) visual evidence scores, (ii) objects' bounding box dimensions (width and height), and (iii) objects' center location offsets. The first branch is where the voting occurs. Before we describe our voting mechanism in detail, we first introduce the log-polar vote field below. The overall processing pipeline of HoughNet is illustrated in Figure 3.1.

3.1 The Log-polar "Vote Field"

We use the set of regions in a standard log-polar coordinate system to define the regions through which votes are collected. A log-polar coordinate system is defined by the number and radii of eccentricity bins (or rings) and the number of angle bins. We call the set of cells or regions formed in such a coordinate system as the "vote

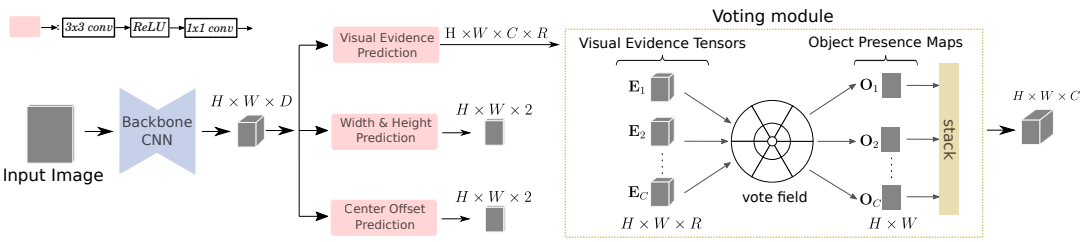


Figure 3.1: Overview of the processing pipeline of HoughNet. Image is taken from our ECCV'20 paper [2].

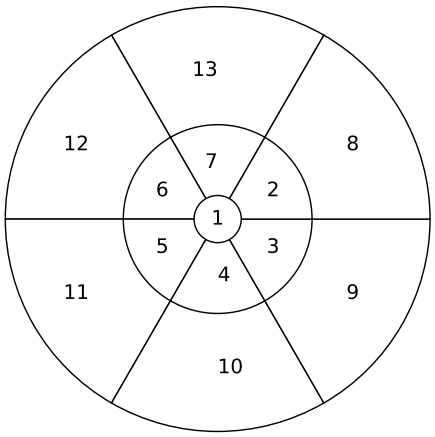


Figure 3.2: A log-polar "vote field" used in the voting module of HoughNet. Numbers indicate region ids. A vote field is parametrized by the number of angle bins, and the number and radii of eccentricity bins, or rings. In this particular vote field, there are a total of 13 regions, 6 angle bins and 3 rings. The radii of the rings are 2, 8 and 16, respectively. Image is taken from our ECCV'20 paper [2].

field" (Figure 3.2). In our experiments, we used different vote fields with different parameters (number of angle bins, etc.) as explained in Chapter 4. In the following, R denotes the number of regions in the vote field and K_r is the number of pixels in a particular region r. $\Delta_r(i)$ denotes the relative spatial coordinates of the i^{th} pixel in the r^{th} region, with respect to the center of the field. We implement the vote field as a fixed-weight (non-learnable) transposed-convolution filter as further explained below.

3.2 Voting Module

After the input image is passed through the backbone network and the "visual evidence" branch, the voting module of HoughNet receives C tensors $\mathbf{E}_1, \mathbf{E}_2, \ldots, \mathbf{E}_C$, each of size $H \times W \times R$, where C is the number of classes, H and W are spatial dimensions and R is the number of regions in the vote field. Each of these tensors contains class-conditional (i.e. for a specific class) "visual evidence" scores. The job of the voting module is to produce C "object presence" maps $\mathbf{O}_1, \mathbf{O}_2, \ldots, \mathbf{O}_C$, each of size $H \times W$. Then, peaks in these maps will indicate the presence of object instances. The voting process, which converts the visual evidence tensors (e.g. $\mathbf{E}_c$) to

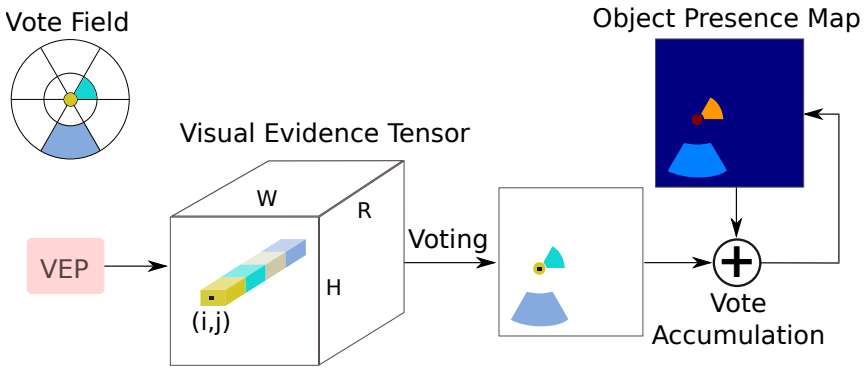


Figure 3.3: Voting process for a single class. Visual evidence prediction (VEP) branch outputs $H \times W \times R$ dimensional visual evidence tensor for the class. Here the voting process is illustrated for just a single location (i, j) and for 3 regions of the vote field shown with yellow, green and blue colors. The values at (i, j) corresponding to these 3 regions are added as votes to the appropriate locations in the Object Presence Map tensor. These locations are determined by the vote field shown at top left. Votes from all locations are similarly accumulated in the Object Presence Map. Image is taken from our ECCV'20 paper extension [3].

object presence maps (e.g. $\mathbf{O}_c$), works as described below.

The voting process is best explained using an example. Suppose we are to process the visual evidence at the i^{th} row, j^{th} column and the r^{th} channel of a visual evidence tensor $\mathbf{E}$. We first place our vote field (Figure 3.2) centered at (i, j) on the r^{th} channel, which is a 2D map. The region r of the vote field marks the target area to be voted on, whose coordinates can be calculated by adding the coordinate offsets $\Delta_r(\cdot)$ to (i, j). Then, we add the visual evidence score $\mathbf{E}(i, j, r)$ to the target area of the object presence map. Note that this operation can be efficiently implemented using the "transposed convolution" (or "deconvolution") operation. Visual evidence scores from locations other than (i, j) are processed in the same way and the scores are accumulated in the object presence map. We formally define this procedure in Algorithm 1, which takes in a visual evidence tensor as input and produces an object presence map[1]. Figure 3.3 also illustrates the defined voting process on a single class.

[1] We provide a step-by-step animation of the voting process in Appendix A.

Algorithm 1 Voting process.

Require: Visual evidence tensor $\mathbf{E}_c$, Vote field relative coordinates Δ

Ensure: Object presence map $\mathbf{O}_c$

```
Initialize O_c with all zeros
for each pixel (i, j, r) in E_c do
    /* K_r: number of pixels in the vote field region r */
    for k = 1 to K_r do
        (y, x) ← (i, j) + Δ_r(k)
        O_c(y, x) ← O_c(y, x) + (1/K_r) E_c(i, j, r)
    end for
end for
```

3.3 Network Architecture

Our network architecture design follows that of "Objects as Points" (OAP) [40]. HoughNet consists of a backbone and three subsequent branches which predict (i) visual evidence scores, (ii) bounding box widths and heights, and (iii) center offsets. Our voting module is attached to the visual evidence branch (Figure 3.1).

The output of the backbone network is a feature map of size $H \times W \times D$, which is the result of an input image of size $4H \times 4W \times 3$. The backbone's output is fed to all three branches. Each branch has one convolutional layer with 3×3 filters followed by a ReLU layer and another convolutional layer with 1×1 filters. The visual evidence branch outputs $H \times W \times C \times R$ sized output where C and R correspond to the number of classes and vote field regions, respectively. The width & height prediction branch outputs $H \times W \times 2$ sized output which predicts heights and widths for each object center. Finally, the center offset branch predicts relative displacement of center locations across the spatial axes. These offsets help recover the lost precision of the center points due to down-sampling operations through the network. Both the width & height branch and the center offset branch are class-agnostic.

Objective functions. For the optimization of the visual evidence branch, we use the modified focal loss [8] introduced in CornerNet [14] (also used in [15, 40]). We optimize the center offset branch using the L_1 loss as the other bottom-up detec-

tors [14, 15, 40] do. Finally, for the width & height prediction branch, we use L_1 loss by scaling the loss by 0.1 as proposed in OAP [40]. The overall loss is the sum of the losses from all branches.

3.4 Spatio-temporal Voting

Unlike static images, videos have rich temporal information. In order to benefit from the temporal clues in videos, researchers developed several methods to aggregate information locally and globally using two or more frames [78, 79, 80, 81]. Similarly, we extend HoughNet with a new temporal voting module to incorporate temporal information using an additional (auxiliary) frame.

Before describing the temporal voting process, we first explain the temporal log-polar vote field. The temporal vote field has 4 regions, 90° angle bin and a single ring with radii 8. Each region of the temporal vote field stands for a motion direction. For example, the temporal vote-field region with id 1 in Figure 3.4, corresponds to relative motion in $+x$ and $+y$ direction. When the temporal voting field is centered at a voter location, it votes for the locations in the target area depending on relative motion direction between frames.

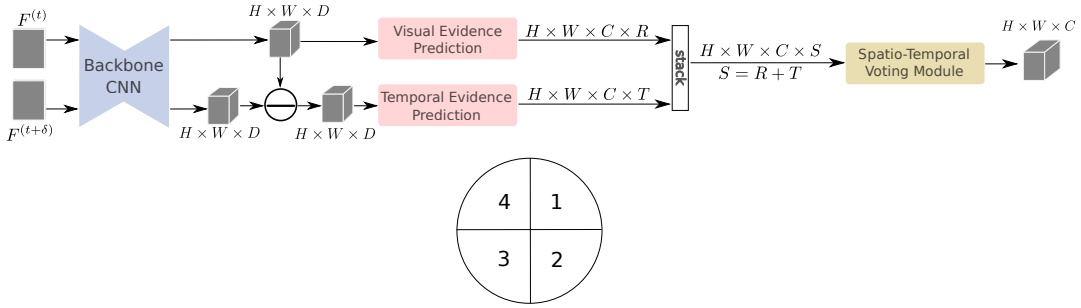


Figure 3.4: (Top) Overall processing pipeline of HoughNet for video object detection. (Bottom) Temporal vote field. Image is taken from our ECCV'20 paper extension [3].

We show the general processing pipeline for video object detection in Figure 3.4. In addition to the reference frame at time *t*, we choose a random auxiliary frame within δ time interval around the reference frame. First, both frames pass through the backbone and we obtain feature maps of size $H \times W \times D$ for each. Then, we

calculate motion features between these frames by subtracting the feature map of the auxiliary frame from the feature map of the reference frame. Bertasius et al.showed that using motion features is effective for pose detection in videos [82]. They interpret these features as discriminative motion features, where the network learns to ignore uninformative motion regions while focusing on discriminative motion cues.

Later, relative motion features are fed to the temporal evidence branch. Finally, visual and temporal evidence tensors are stacked and forwarded to the spatio-temporal voting module to output $H \times W \times C$ dimensional object presence maps. Spatio-temporal voting aggregates votes in the same way as described in Section 3.2. Temporal evidence branch has the same architecture as the visual evidence branch.

CHAPTER 4

EXPERIMENTAL ANALYSIS OF HOUGH VOTING ON DIFFERENT VISION PROBLEMS

This chapter is adopted from our ECCV'20 paper [2] and its extension [3].

In this chapter, we show the effectiveness of our proposed method on object detection and other visual detection tasks, namely, video object detection, instance segmentation, 3D object detection, keypoint detection for 2D human pose estimation and 2D whole-body human pose estimation, and face detection. Since we mostly focus on object detection, the object detection section is more detailed compared to others. It includes (i) ablation experiments through which we studied how different parameters of the vote field affect the performance, (ii) comparison with baseline, (iii) comparison with state-of-the-art, (iii) an analysis section and (iv) a scalable voting approach where the number of "visual evidence tensors" does not depend on the number of object classes. The sections for other tasks mostly contain comparisons with baseline. We also include an image generation task in the context of "label-to-photo" translation at the end. In order to handle a large volume of ablation experiments, we created a small training set for COCO, called "COCO `minitrain`", which is described first in the following.

We ran our experiments on 4 V100 GPUs. For training, we used 512×512 images unless stated otherwise. The training setup is not uniform across different experiments, mainly due to different backbones. However, the inference pipeline is common for all HoughNet models. We extract center locations by applying a 3×3 max pooling operation on object presence heatmaps and pick the highest scoring 100 points as detections. Then, we adjust these points using the predicted center offset values. Final bounding boxes are generated using the predicted width & height values on these

detections.

4.1 COCO minitrain

To facilitate model development and faster analysis in ablation experiments, we carefully curated a mini training set for COCO, dubbed "COCO `minitrain`". It is a subset of the COCO `train2017` dataset, containing $25K$ images (about 20% of `train2017`) and around $184K$ objects across 80 object categories. We randomly sampled these images from the full set while preserving the following three quantities as much as possible: (i) proportion of object instances from each class, (ii) overall ratios of small, medium and large objects, (iii) per class ratios of small, medium and large objects.

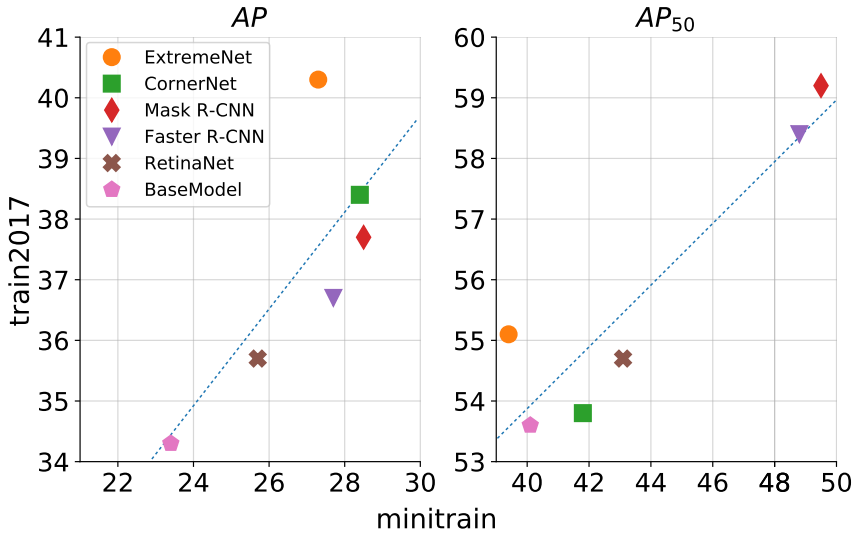


Figure 4.1: Performance validation for COCO `minitrain`. The x-axis is the `val2017` performance (AP on the left, AP50 on the right) of a model when it is trained on `minitrain`. The y-axis is the performance of the model when it is trained on the full COCO training set, `train2017`. Fitted lines with Pearson correlation coefficients 0.74 and 0.92, respectively for AP and AP_{50}, show strong positive correlation. This figure is based on Table 4.1. Image is taken from our ECCV'20 paper extension [3].

To validate COCO `minitrain`, we computed the correlation between the `val2017`

performance of a model when it is trained on `minitrain` with the same as when it is trained on `train2017`. Over six different object detectors (Faster R-CNN, Mask R-CNN, RetinaNet, CornerNet, ExtremeNet and HoughNet), the Pearson correlation coefficients turned out to be 0.74 and 0.92 for AP and AP_{50}, respectively (Figure 4.1), which indicate strong positive correlation. Figure 4.1 is based on the results in Table 4.1. Table 4.2 presents the full results obtained on COCO `val2017` performances when models are trained on COCO `minitrain`. Further details on `minitrain` and the dataset itself can be found in Appendix B.

Table 4.1: Object Detector performances trained on `minitrain` vs `train2017`. Models are evaluated on `val2017`.

Method	Backbone	Scale	`minitrain`			`train2017`		
			AP	AP_{50}	AP_{75}	AP	AP_{50}	AP_{75}
Two-stage detectors:								
Faster R-CNN	ResNet-50 *w* FPN	800	27.7	48.8	28.4	36.7	58.4	39.6
Mask R-CNN	ResNet-50 *w* FPN	800	28.5	49.5	29.4	37.7	59.2	40.9
One-stage detectors:								
RetinaNet	ResNet-50 *w* FPN	800	25.7	43.1	26.8	35.7	54.7	38.5
CornerNet	Hourglass-104	511	28.4	41.8	29.5	38.4	53.8	40.9
ExtremeNet	Hourglass-104	511	27.3	39.4	28.9	40.3	55.1	43.7
HoughNet (ours)	ResNet-101	512	23.4	40.1	23.6	34.3	53.6	36.6

Table 4.2: Object Detector performances. Models are trained on `minitrain` and evaluated on `val2017`.

Method	BackBone	Scale	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Two-stage detectors:								
Faster-R-50-FPN	ResNet-50 *w* FPN	800	27.7	48.8	28.4	14.7	29.8	36.4
Mask-R-50-FPN	ResNet-50 *w* FPN	800	28.5	49.5	29.4	14.7	30.7	37.6
One-stage detectors:								
RetinaNet-R-50-FPN	ResNet-50 *w* FPN	800	25.7	43.1	26.8	12.1	28.6	34.2
CornerNet	Hourglass-104	511	28.4	41.8	29.5	11.3	29.6	39.2
ExtremeNet	Hourglass-104	511	35.8	49.9	38.8	17.2	38.6	49.0

4.2 Hough Voting for Object Detection

In the evaluation of object detection models, we follow the other bottom-up methods [15, 40, 14] and use two modes: (i) single-scale, horizontal-flip testing (SS testing mode), and (ii) multi-scale, horizontal-flip testing (MS testing mode). In MS, we use the following scale values, $0.6, 1.0, 1.2, 1.5, 1.8$. To merge augmented test results, we use Soft-NMS [83], and keep the top 100 detections. All tests are performed on a single V100 GPU.

4.2.1 Ablation Experiments

Here we analyze the effects of the number of angle and ring bins of the vote field on performance. Models are trained on COCO `minitrain` and evaluated on `val2017` set with SS testing mode. The backbone is Resnet-101 [6]. In order to get higher resolution feature maps, we add three deconvolution layers on top of the default Resnet-101 network, similar to [84]. We add 3×3 convolution filters before each 4×4 deconvolution layer, and put batchnorm and ReLU layers after convolution and deconvolution filters. We trained the network with a batch size of 44 for 140 epochs with Adam optimizer [85]. Initial learning rate 1.75×10^{-4} was divided by 10 at epochs 90 and 120.

4.2.1.1 Angle Bins

We started with a large, 65 by 65, vote field with 5 rings. We set the radius of these rings from the most inner one to the most outer one as 2, 8, 16, 32 and 64 pixels, respectively. We experimented with $60°$, $90°$, $180°$ and $360°$ bins. We do not split the center ring (i.e. region with id 1 in Figure 3.2) into further regions. Results are presented in Table 4.3a. For the $180°$ experiment, we divide the vote field horizontally. $90°$ yields the best performance considering both *AP* and AP_{50}. We used this setting in the rest of the experiments.

Table 4.3: Ablation experiments for the vote field. (a) Effect of angle bins on performance. Vote field with 90° has the best performance (considering *AP* and AP_{50}). (b) Effect of central and peripheral regions. Here, the angle bin is 90° and the ring count is four. Disabling any of center or periphery hurts performance, cf. (a). (c) Effect of number of rings. Angle is 90° and vote field size is updated according to the radius of the last ring. Using 3 rings yields the best result. It is also the fastest model.

Model	*AP*	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	FPS
60°	**24.6**	41.3	25.0	**8.2**	27.7	36.2	3.4
90°	**24.6**	**41.5**	25.0	**8.2**	27.7	36.2	**3.5**
180°	24.5	41.1	24.8	8.1	27.7	**36.3**	**3.5**
360°	**24.6**	41.1	**25.1**	8.0	**27.8**	**36.3**	**3.5**
(a) Varying the Number of Angle Bins							
Only Center	23.8	39.5	24.5	**7.9**	26.8	34.7	**3.5**
No Center	**24.4**	**40.9**	**24.9**	7.4	**27.6**	**37.1**	3.3
Only Context	23.6	39.7	24.2	7.4	26.4	35.9	3.4
(b) Effectiveness of Votes from Center or Periphery							
5 Rings	24.6	**41.5**	25.0	8.2	27.7	36.2	3.5
4 Rings	24.5	41.1	25.3	8.2	**27.8**	36.1	7.8
3 Rings	**24.8**	41.3	**25.6**	**8.4**	27.6	**37.5**	**15.6**
(c) Varying Ring Counts							

4.2.1.2 Effects of Center and Periphery

We conducted experiments to analyze the importance of votes coming from different rings of the vote field. Results are presented in Table 4.3b. In the *Only Center* case, we only keep the center ring and disable the rest. In this way, we only aggregate votes from features of the object center directly, which corresponds to a traditional object detector where only local (short-range) evidence is used. This experiment shows that votes from outer rings help improve performance. For the *No Center* case, we only disable the center ring. We observe that there is only 0.2 decrease in *AP*. This suggests that the evidence for successful detection is embedded mostly around the object center not directly inside the object center. In order to observe the power

of long-range votes, we conducted another experiment called "Only Context," where we disabled the two most inner rings and used only the three outer rings for vote aggregation. This model reduced *AP* by 1.0 point compared to the full model.

4.2.1.3 Ring Count

To find out how far an object should get votes from, we discard outer ring layers one by one as presented in Table 4.3c. The models with 5 rings, 4 rings and 3 rings have 17, 13 and 9 voting regions and 65, 33 and 17 vote field sizes, respectively. The model with 3 rings yields the best performance on *AP* metric and is the fastest one at the same time. On the other hand, the model with 5 rings yields 0.2 AP_{50} improvement over the model with 3 rings.

From all these ablation experiments, we decided to use the model with 5 rings and 90° as our *Base Model*. Considering both speed and accuracy, we decided to use the model with 3 rings and 90° as our *Light Model*.

4.2.1.4 Voting Module vs. Dilated Convolution

Dilated convolution [86], which can include long-range features, could be considered as an alternative to our voting module. To compare performance, we trained models on `train2017` and evaluated them on `val2017` using the SS testing mode.

Baseline: We consider OAP with ResNet-101-DCN backbone as baseline. The last 1×1 convolution layer of center prediction branch in OAP, receives $H \times W \times D$ tensor and outputs object center heatmaps with a tensor of size $H \times W \times C$.

Baseline + Voting Module: We first adapt the last layer of center prediction branch in baseline to output $H \times W \times C \times R$ tensor, then attach our voting module on top of the center prediction branch. Adding the voting module increases parameters of the layer by R times. The log-polar vote field is 65×65, and has 5 rings (90°). With 5 rings and 90° we end up with $R = 17$ regions.

Baseline + Dilated Convolution: We use dilated convolution with kernel size 4×4

Table 4.4: Comparing our voting module to an equivalent dilated convolution filter, in terms of number of parameters and the spatial filter size, on COCO `val2017` set. Models are trained on COCO `train2017` and results are presented on SS testing mode.

Method	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Baseline	36.2	54.8	38.7	16.3	41.6	52.3
+ Dilated Conv.	36.6	56.1	39.2	16.7	42.0	53.6
+ Voting Module	**37.3**	**56.6**	**39.9**	**16.8**	**42.6**	**55.2**

Table 4.5: HoughNet results on COCO `val2017` set for different training setups. † indicates initialization with CornerNet weights, * indicates initialization with ExtremeNet weights. Results are given for SS and MS testing modes, respectively.

Models	Backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	FPS
Base	R-101	36.0 / 40.7	55.2 / 60.6	38.4 / 43.9	16.2 / 22.5	41.7 / 44.2	52.0 / 55.7	3.5 / 0.5
Base	R-101-DCN	37.3 / 41.6	56.6 / 61.2	39.9 / 44.9	16.8 / 22.6	42.6 / 44.8	55.2 / 58.8	3.3 / 0.4
Light	R-101-DCN	37.2 / 41.5	56.5 / 61.5	39.6 / 44.5	16.8 / 22.5	42.5 / 44.8	54.9 / 58.4	14.3 / 2.1
Light	HG-104	40.9 / 43.7	59.2 / 61.9	44.1 / 47.3	23.8 / 27.5	45.3 / 45.9	52.6 / 56.2	6.1 / 0.8
Light	HG-104†	41.7 / 44.7	60.5 / 63.2	45.6 / 48.9	23.9 / 28.0	45.7 / 47.0	54.6 / 58.1	5.9 / 0.8
Light	HG-104*	43.0 / **46.1**	62.2 / **64.6**	46.9 / **50.3**	25.5 / **30.0**	47.6 / **48.8**	55.8 / **59.7**	5.7 / 0.8

and dilation rate 22 for the last layer of the center prediction branch in baseline. Using 4×4 kernel increases parameters 16 times which is approximately equal to R in the *Baseline + Voting Module*. Using dilation rate 22, the filter size becomes 67×67 which is close to 65×65 log-polar vote field.

For a fair comparison with *Baseline*, both *Baseline + Voting Module* and the *Baseline + Dilated Convolution* use ResNet-101-DCN backbone. Our voting module outperforms dilated convolution in all cases (Table 4.4).

4.2.2 Comparison with Baseline

In Table 4.5, we present the performance of HoughNet for different backbone networks, initializations and our base-vs-light model, on the `val2017` set. There is a

Table 4.6: Comparison of object detection with baseline (OAP) on `val2017`. Results are given for SS and MS test modes, respectively.

Method	*AP*	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	*moLRP* ↓
Baseline *w* R-101-DCN	36.2 / 39.2	54.8 / 58.6	38.7 / 41.9	16.3 / 20.5	41.6 / 42.6	52.3 / 56.2	71.1 / 68.3
+ Voting Module	**37.2 / 41.5**	**56.5 / 61.5**	**39.6 / 44.5**	**16.8 / 22.5**	**42.5 / 44.8**	**54.9 / 58.4**	**69.9 / 66.6**
Baseline *w* HG-104	42.2 / 45.1	61.1 / 63.5	46.0 / 49.3	25.2 / 27.8	46.4 / 47.7	55.2 / **60.3**	66.1 / 63.9
+ Voting Module	**43.0 / 46.1**	**62.2 / 64.6**	**46.9 / 50.3**	**25.5 / 30.0**	**47.6 / 48.8**	**55.8** / 59.7	**65.6 / 63.1**

significant speed difference between Base and Light models. Our light model with R-101-DCN backbone is the second fastest one (14.3 FPS) achieving 37.2 *AP* and 56.5 AP_{50}. We observe that initializing the backbone with a pretrained model improves the detection performance. In Table 4.6, we compare HoughNet's performance with its baseline OAP [40] for two different backbones. HoughNet is especially effective for small objects, it improves the baseline by 2.1 and 2.2 *AP* points for R-101-DCN and HG-104 backbones, respectively. We also provide results for the recently introduced *moLRP* [21] metric, which combines localization, precision and recall in a single metric. Lower values are better.

4.2.3 Comparison with the State-of-the-art

For comparison with the state-of-the-art, we use Hourglass-104 [14] backbone. We train Hourglass model with a batch size of 36 for 100 epochs using the Adam optimizer [85]. We set the initial learning rate to 2.5×10^{-4} and divided it by 10 at epoch 90. Table 4.7 presents performances of HoughNet and several established state-of-the-art detectors. First, we compare HoughNet with OAP [40] since it is the model on which we built HoughNet. In OAP, they did not present any results for "from scratch" training. Instead they fine-tuned their model from ExtremeNet weights. When we do the same (i.e. initialize HoughNet with ExtremeNet weights), we obtain better results than OAP. However as expected, HoughNet is slower than OAP. Among the one-stage bottom-up object detectors, HoughNet performs on-par with the best bottom-up object detector by achieving 46.4 *AP* against 47.0 *AP* of CenterNet [19]. HoughNet outperforms CenterNet on AP_{50} (65.1 AP_{50} vs. 64.5 AP_{50}). Note that, since our

Table 4.7: Comparison with the state-of-the-art on COCO `test-dev`. The methods are divided into three groups: two-stage, one-stage top-down and one-st age bottom-up. The best results are boldfaced separately for each group. Backbone names are shortened: R is ResNet, X is ResNeXt, F is FPN and HG is HourGlass.* indicates that the FPS values were obtained on the same AWS machine with a V100 GPU using the official repos in SS setup. The rest of the FPS are from their corresponding papers. F. R-CNN is Faster R-CNN.

Method	Backbone	Initialize	Train size	Test size	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	FPS
Two-stage detectors:											
R-FCN [87]	R-101	ImageNet	800×800	600×600	29.9	51.9	-	10.8	32.8	45.0	5.9
CoupleNet [49]	R-101	ImageNet	ori.	ori.	34.4	54.8	37.2	13.4	38.1	50.8	-
F. R-CNN+++ [6]	R-101	ImageNet	1000×600	1000×600	34.9	55.7	37.4	15.6	38.7	50.9	-
F. R-CNN [88]	R-101-F	ImageNet	1000×600	1000×600	36.2	59.1	39.0	18.2	39.0	48.2	5.0
Mask R-CNN [20]	X-101-F	ImageNet	1300×800	1300×800	39.8	62.3	43.4	22.1	43.2	51.2	11.0
Cascade R-CNN [89]	R-101	ImageNet	-	-	42.8	62.1	46.3	23.7	45.5	55.2	**12.0**
PANet [90]	X-101	ImageNet	1400×840	1400×840	**47.4**	**67.2**	**51.8**	**30.1**	**51.7**	**60.0**	-
One-stage detectors:											
Top Down:											
SSD [10]	VGG-16	ImageNet	512×512	512×512	28.8	48.5	30.3	10.9	31.8	43.5	-
YOLOv3 [35]	Darknet	ImageNet	608×608	608×608	33.0	57.9	34.4	18.3	35.4	41.9	**20.0**
DSSD513 [36]	R-101	ImageNet	513×513	513×513	33.2	53.3	35.2	13.0	35.4	51.1	-
RefineDet (SS) [91]	R-101	ImageNet	512×512	512×512	36.4	57.5	39.5	16.6	39.9	51.4	-
RetinaNet [8]	X-101-F	ImageNet	1300×800	1300×800	40.8	61.1	44.1	24.1	44.2	51.2	5.4
RefineDet (MS) [91]	R-101	ImageNet	512×512	≤2.25×	41.8	62.9	45.7	25.6	45.1	54.1	-
OAP (SS) [40]	HG-104	ExtremeNet	512×512	ori.	42.1	61.1	45.9	24.1	45.5	52.8	9.6*
FSAF (SS) [92]	X-101	ImageNet	1300×800	1300×800	42.9	63.8	46.3	26.6	46.2	52.7	2.7
FSAF (MS) [92]	X-101	ImageNet	1300×800	∼≤2.0×	44.6	65.2	48.6	29.7	47.1	54.6	-
FCOS [39]	X-101-F	ImageNet	1300×800	1300×800	44.7	64.1	48.4	27.6	47.5	55.6	7.0*
FreeAnchor (SS) [93]	X-101-F	ImageNet	1300×960	1300×960	44.9	64.3	48.5	26.8	48.3	55.9	-
OAP (MS) [40]	HG-104	ExtremeNet	512×512	≤1.5×	45.1	63.9	49.3	26.6	47.1	57.7	-
FreeAnchor (MS) [93]	X-101-F	ImageNet	1300×960	∼≤2.0×	**47.3**	**66.3**	**51.5**	**30.6**	**50.4**	**59.0**	-
Bottom Up:											
ExtremeNet (SS) [15]	HG-104	-	511×511	ori.	40.2	55.5	43.2	20.4	43.2	53.1	3.0*
CornerNet (SS) [14]	HG-104	-	511×511	ori.	40.5	56.5	43.1	19.4	42.7	53.9	5.2*
CornerNet (MS) [14]	HG-104	-	511×511	≤1.5×	42.1	57.8	45.3	20.8	44.8	56.7	-
ExtremeNet (MS) [15]	HG-104	-	511×511	≤1.5×	43.7	60.5	47.0	24.1	46.9	57.6	-
CenterNet (SS) [19]	HG-104	-	511×511	ori.	44.9	62.4	48.1	25.6	47.4	57.4	4.8*
CenterNet (MS) [19]	HG-104	-	511×511	≤1.8×	**47.0**	64.5	**50.7**	28.9	**49.9**	**58.9**	-
HoughNet (SS)	HG-104	-	512×512	ori.	40.8	59.1	44.2	22.9	44.4	51.1	**6.4***
HoughNet (MS)	HG-104	-	512×512	≤1.8×	44.0	62.4	47.7	26.4	45.4	55.2	-
HoughNet (SS)	HG-104	ExtremeNet	512×512	ori.	43.1	62.2	46.8	24.6	47.0	54.4	**6.4***
HoughNet (MS)	HG-104	ExtremeNet	512×512	≤1.8×	46.4	**65.1**	**50.7**	**29.1**	48.5	58.1	-

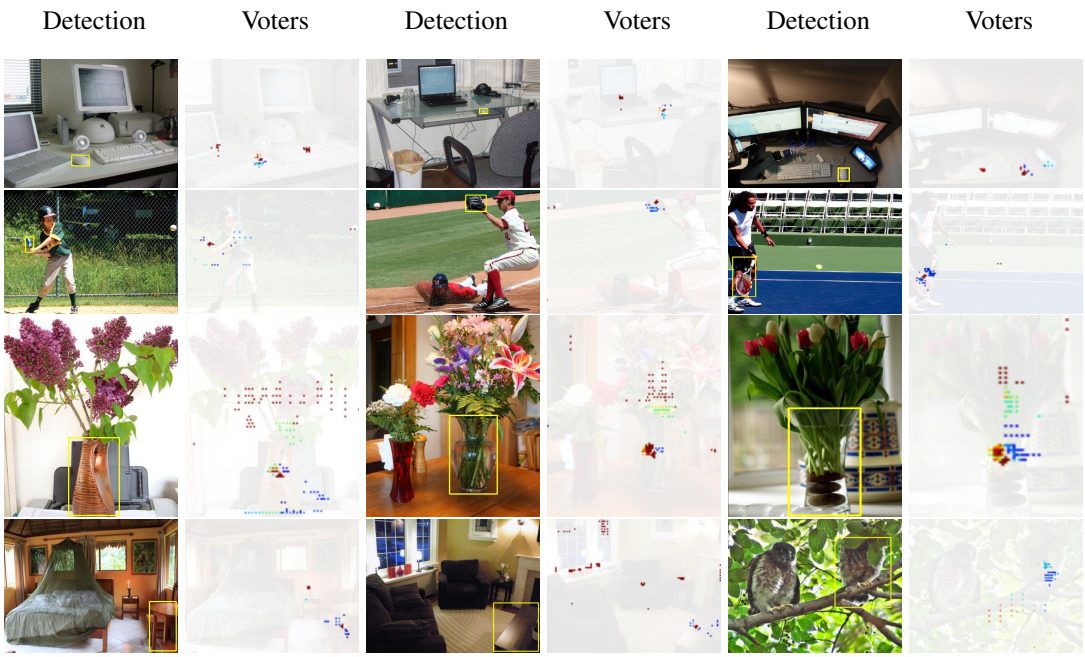


Figure 4.2: Sample detections of HoughNet and their vote maps. In the "detection" columns, we show a correctly detected object, marked with a yellow bounding box. In the "voters" columns, the locations that vote for the detection are shown. Colors indicate vote strength based on the standard "jet" colormap (red is high, blue is low; Figure 1.1). In the **top row**, there are three "mouse" detections. In all cases, in addition to the local votes (that are on the mouse itself), there are strong votes coming from nearby "keyboard" objects. This voting pattern is justified given that mouse and keyboard objects frequently co-appear. A similar behavior is observed in the detections of "baseball bat", "baseball glove" and "tennis racket" in the **second row**, where they get strong votes from "ball" objects that are far-away. Similarly, in the **third row**, "vase" detections get strong votes from the flowers. In the first example of the **bottom row**, "dining table" detection gets strong votes from the candle object, probably because they co-occur frequently. Candle is not among the 80 classes of COCO dataset. Similarly, in the second example in the **bottom row**, "dining table" has strong votes from objects and parts of a standard living room. In the last example, partially occluded bird gets strong votes (stronger than the local votes on the bird itself) from the tree branch. More visual results could be found in Appendix C. Image is taken from our ECCV'20 paper [3].

model is initialized with ExtremeNet weights, which makes use of the segmentation masks in its own training, our model effectively uses more data compared to CenterNet. HoughNet is the fastest among one-stage bottom-up detectors. It is faster than CenterNet, CornerNet and more than twice as fast as ExtremeNet.

We provide visualization of votes for sample detections of HoughNet for qualitative visual inspection (Figure 4.2). These detections clearly show that HoughNet is able to make use of long-range visual evidence.

4.2.4 Analysis

Here we analyse the effect of Hough voting from two aspects. First, we inspect the error sources of both HoughNet and its baseline, and compare the two. This lets us understand how the voting module has improved the baseline. Secondly, we collect votes cast on a large number of images and analyse them to deduce interactions among object classes

4.2.4.1 Error Sources

Here we use the recently introduced "localisation recall precision" (LRP) metric [21, 22], which provides us with componentwise errors. These errors and the relative improvements yielded by the voting module are shown in Table 4.8. The largest relative improvement ($+4.4\%$) is obtained in the "false negative" (FN) component, which shows that the voting module increases the recall rate, that is, object instances normally missed by the baseline are successfully recovered with the integration of the voting module. This shows the importance of voting. "False positive" (FP) component is also improved ($+3.3\%$), which indicates that object instances marked as background by the baseline are corrected by the voting module. These improvements come with no degradation in localisation performance.

Table 4.8: Effect of voting module on three major error types, namely, localisation (Loc), false positive (FP) and false negative (FN), as measured by the LRP metric [21, 22]. Relative improvement is calculated as $(B - V)/B$ where B and V are the LRP errors of baseline and "baseline+voting module", respectively. Lower is better. Largest relative improvement is achieved in the FN component, which indicates that voting module increases the recall over the baseline.

Model	*Loc*	*FP*	*FN*
Baseline *w* R-101-DCN	17.1	27.1	50.2
+ Voting Module	17.1	**26.2**	**48.0**
Relative Improvement	0%	+3.3%	+4.4%

4.2.4.2 Interaction among Object Classes

We build a $C \times C$ (C is the number of classes) matrix to visualize voting relations among classes on the COCO dataset using the R-101-DCN backbone of HoughNet. In this matrix, rows represent vote-getters and columns represent vote-givers. For each vote-getter class, we first identify center points of detections and find all locations that vote for these centers. Then, we sum class probabilities of all the voter-giver locations and obtain a final C dimensional vector that summarizes the votes received by the vote-getter class. Figure 4.3 shows a 10×15 matrix, which we selected from the full 80×80 matrix based on the following criterion: we selected the top 10 vote-getter classes with the maximum total votes. Next, for each vote-getter class, we selected the top 15 vote-giver classes that have the maximum voting activity. We provide the full 80×80 matrix in Appendix D.

This matrix reflects many object co-occurrence relations and it captures object-to-object context well. For example, *mouse* class gets votes mostly from *mouse*, *keyboard* and *laptop* classes. *toaster* objects get votes from kitchen items. As a part of tableware items, both *spoon* and *knife* get votes from each other and other tableware items like *fork* and *bowl*. *Potted plant* objects get from indoor objects such as *chair* and *couch*.

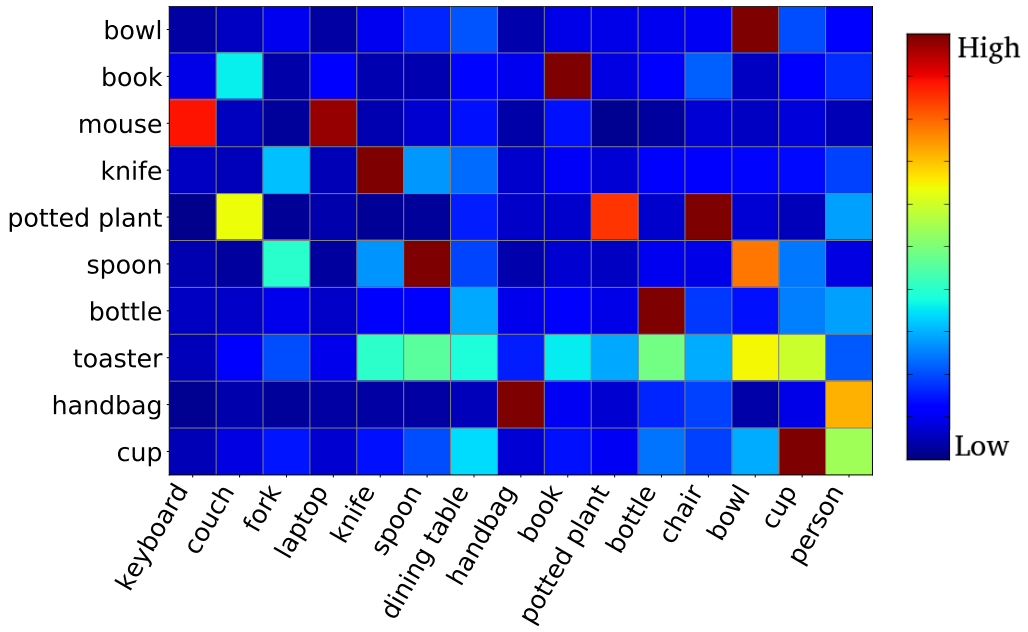


Figure 4.3: Object detection voting activity among 10 vote-getter classes (rows) and 15 vote-giver classes (columns) on the COCO dataset. Color decodes voting activity (red: high, blue: low). Each detection in a vote-getter class might get votes from multiple object classes. For example, the *cup* objects (i.e. the "cup" row) get relatively high votes from *cup*, *person*, *dining table* and *bowl* classes. The full 80×80 matrix is provided in Appendix D. Image is taken from our ECCV'20 paper extension [3].

4.2.5 A Scalable Approach to Voting (independent of number of classes)

In HoughNet, there is a separate visual evidence tensor per object class (Figure 3.1) and a voting process is run for each of these (Section 3.2). This linear dependence on the number of object classes might be problematic when the number of classes increases dramatically, which is the case for newer datasets such as the 1000-class LVIS dataset [94]. With this in mind, we designed a scalable variant of our voting module where, instead of having a separate visual evidence tensor per class, we create N tensors with $N << C$ (Figure 4.4). In this design, these N tensors contain "visual evidence" scores that are shared among and common to all classes. As a result, the voting process is carried out N times instead of C times. We convert the resulting $H \times W \times N$ dimensional voting maps to $H \times W \times C$ dimensional object presence maps using a ReLU layer followed by a convolutional layer with 3×3 filters. We

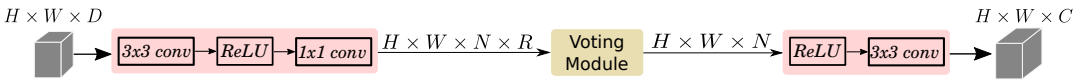


Figure 4.4: A scalable variant of our voting process. Instead of having a separate visual evidence tensor for each of the C classes, we create N ($N << C$) visual evidence tensors that are shared and common to all classes. Image is taken from our ECCV'20 paper extension [3].

illustrate the scalable voting module in Figure 4.4.

Table 4.9: Ablation experiments for the scalable voting module. Models are trained on COCO `minitrain` and results are presented on SS testing mode. Using 8 visual evidence tensors yields the best result for all *AP* metrics.

N	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
4	24.9	40.7	25.9	7.7	27.5	37.5
8	**25.9**	**42.3**	**26.7**	**8.5**	**28.4**	**39.1**
12	24.6	40.6	25.6	7.8	27.0	36.6
16	25.0	41.1	25.8	8.2	27.4	37.9

To analyze the scalable approach, we conducted ablation experiments for different values of N. We used the vote field from the *Light Model* setup with 3 rings and $90°$ bins, using the ResNet-101-DCN backbone. Models are trained on COCO `minitrain` and evaluated on `val2017` set in SS testing mode. Results are presented in Table 4.9. The model with $N = 8$ performs best among others.

To compare the scalable model with the light model, we trained it on COCO `train2017` and obtained results on `val2017`. The scalable model is almost 2 times faster than the light model (26.3 FPS vs. 14.3 FPS) in SS testing mode. However, it performs 1 *AP* point worse than the base and light models with the same backbone, obtaining 36.3 *AP*.

4.3 Hough Voting for Other Visual Detection Tasks

4.3.1 Video Object Detection

We conducted our experiments on the ImageNet VID dataset [95]. ImageNet VID dataset has 30 object categories. For training and evaluation, we followed widely adopted protocols [78, 81] and trained our models on 3,862 videos in the training set and evaluated their performances on 555 videos in the validation set using mean average precision (mAP) as the evaluation metric. In addition to the overall mAP, we also present mAP results for slow, medium, and fast groups of videos as done in previous work [78]. During the training of temporal models, we use a pair of video frames; a reference frame at time *t* and a nearby auxiliary frame at time $t + \delta$ where δ is randomly picked from the set $\{-4, ..., +4\}$. At inference, we use fixed δ values and present our results for $\delta = \{-4\}$ and $\delta = \{-4, +4\}$ settings.

First, we obtained single-frame baseline results for our baseline (OAP [40]) and HoughNet. We trained OAP with 140 epochs using their official repository. For faster analysis, we trained our models with 80 epochs and divided the initial learning rate 1.25×10^{-4} by 10 at epoch 50. All models are trained with a batch size of 32. Results in Table 4.10 show that HoughNet outperforms its baseline (68.8 vs 65.0) in this setting. Later, we conducted an ablation experiment to compare the performance of proposed temporal voting with feature aggregation over reference and auxiliary frames. To this end, both reference and auxiliary frames are passed through the backbone, then the output feature maps are concatenated and fed to the visual evidence prediction branch. This model improved the single-frame baseline result of HoughNet by 2.7 mAP points. Next, we experimented with temporal voting using motion features. Temporal voting improved the performance further by 2.4 points. Using two auxiliary frames (δ is $\{-4, +4\}$), we obtained an even better result achieving 74.9 mAP.

Table 4.10: Results of video object detection on ImageNet VID validation set. Results are obtained without any test time augmentation. δ corresponds to the time offset of the auxiliary frame during inference.

Method	δ	*mAP*	mAP_F	mAP_M	mAP_S
Single-frame models:					
Baseline (OAP)	-	65.0	41.4	61.9	76.4
+ Voting (HoughNet)	-	68.8	45.8	66.1	79.1
Temporal voting models					
+ Feat. Agg.	-4	71.5	46.6	67.7	82.1
+ Motion Features	-4	73.9	50.4	71.5	82.8
+ Motion Features	-4,+4	**74.9**	**51.5**	**72.5**	**83.6**

4.3.2 Instance Segmentation

In order to show the effectiveness of voting for instance segmentation, we first extended our baseline OAP to perform instance segmentation. Inspired by the recent method BlendMask [96], we added to OAP a new *prototype mask prediction* branch to predict category independent, single prototype mask. This branch outputs a $H \times W \times 1$-dimensional feature map. In order to predict instance specific masks, we added another *attention map prediction* branch which outputs $H \times W \times 196$ dimensional attention features. These new branches have the same layer structure as other branches (see Figure 3.1). We provide the network architecture of instance segmentation for baseline model in Figure 4.5.

During training, we first get the *instance specific local prototypes* for each instance by cropping them from the prototype mask according to their location and box size. Next, we extract 1×196 *instance specific attention maps* using the center points of instances, from the predicted attention map. Later we obtain 14×14 instance specific attention maps by reshaping the attention features. We apply sigmoid normalization for both instance specific local prototypes and attention maps, and finally blend them by applying element-wise product. We use binary cross entropy as our loss function for segmentation. To extend HoughNet for the instance segmentation task, we add the

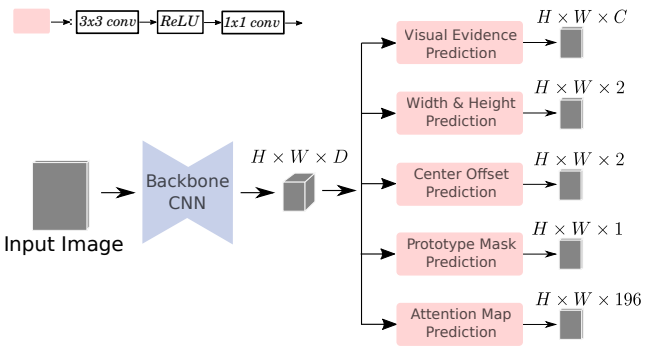


Figure 4.5: Overall processing pipeline of baseline model for instance segmentation. Image is taken from the supplementary document of our ECCV'20 paper extension work [3].

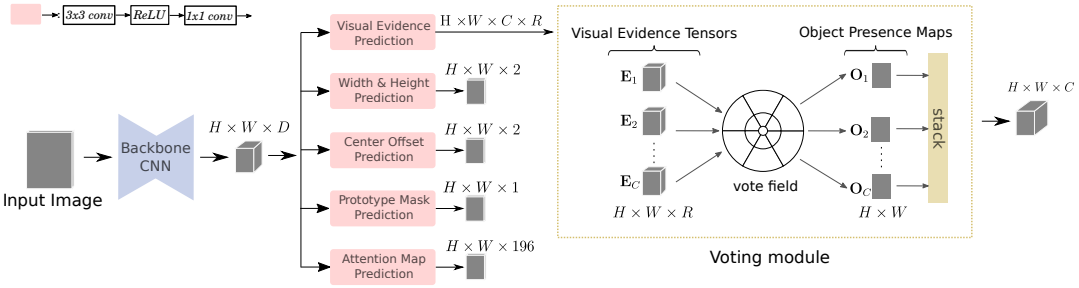


Figure 4.6: Overall processing pipeline of HoughNet for instance segmentation. Image is taken from the supplementary document of our ECCV'20 paper extension work [3].

new branches as described above and follow the same training process. We provide the network architecture of instance segmentation for HoughNet in Figure 4.6.

We present the instance segmentation results on the COCO dataset in Table 4.11. Both baseline and HoughNet models are trained for 80 epochs with a batch size of 32. Initial learning rate 1.25×10^{-4} is divided by 10 at epoch 50. Voting module of HoughNet improves instance segmentation performance by 1.2 AP points and outperforms the baseline for all instance segmentation and box AP metrics.

Table 4.11: Effect of voting module for the instance segmentation task on COCO `val2017` set. Results are shown for both COCO segmentation and box AP. Models are trained on COCO `train2017`. Results are obtained without any test time augmentation.

Method	AP^{seg}	AP^{seg}_{50}	AP^{seg}_{75}	AP^{seg}_{S}	AP^{seg}_{M}	AP^{seg}_{L}	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	AP^{box}_{S}	AP^{box}_{M}	AP^{box}_{L}
Baseline	27.2	46.4	28.0	8.6	31.3	43.9	33.9	51.3	36.5	14.7	39.3	50.0
+ Voting	**28.4**	**48.0**	**28.8**	**9.1**	**32.1**	**46.1**	**35.0**	**52.9**	**37.6**	**15.0**	**40.0**	**52.2**

4.3.3 3D Object Detection

Here we conduct experiments in 3D object detection to see whether our voting module is useful. In addition to the class and location prediction as done in 2D object detection, a 3D object detector has to predict additional *depth*, *3D dimension* and *orientation* attributes. For this task also, we consider OAP [40] as our baseline. In addition to the branches from 2D object detection, OAP adds separate branches for these additional three attributes.

To show the effectiveness of the voting module, similar to 2D object detection we attach our voting module with 3 rings and 90° to the class prediction branch. We experimented with *car* classes of KITTI dataset [97] using the training and validation splits from SubCNN [98]. Following OAP, we use the original image resolution 1280×384 both during training and inference. We trained the network with a batch size of 16 for 70 epochs with Adam optimizer [85]. Initial learning rate 1.25×10^{-4} was divided by 10 at epochs 45 and 60. For fair comparison with the baseline, we use Deep Layer Aggregation (DLA) [99] backbone as in OAP. More details related to both training and inference could be found in OAP [40].

Table 4.12 shows results for bounding box AP, average orientation score (AOS) and bird-eye-view bounding box AP (BEV AP) at 3 levels of difficulty, namely, *easy*, *medium* and *hard*. AP values in the table correspond to the average AP at IoU threshold 0.5 at 11 recalls from 0.0 to 1.0 with a step size of 0.1. As also identified by OAP, since the recall threshold is small, the evaluation measures fluctuate up to 10% *AP*. To smooth the effect of fluctuations we trained 5 models and reported the averages

Table 4.12: Effect of voting module for the 3D object detection task on KITTI. Mean AP values of 5 models are presented with their standard deviations. Results are obtained without any test time augmentation.

Method	AP_e	AP_m	AP_h	AOS_e	AOS_m	AOS_h	$BEV\,AP_e$	$BEV\,AP_m$	$BEV\,AP_h$
Baseline	**90.2**±1.2	80.4±1.4	**71.1**±1.6	85.3±1.7	75.0±1.6	66.2±1.8	31.4±3.7	26.5±1.6	23.8±2.9
+ Voting	89.2±0.4	**80.6**±3.0	70.0±0.2	**86.0**±0.8	**77.0**±3.1	**66.5**±0.6	**35.0**±0.8	**28.0**±0.8	**26.1**±0.7

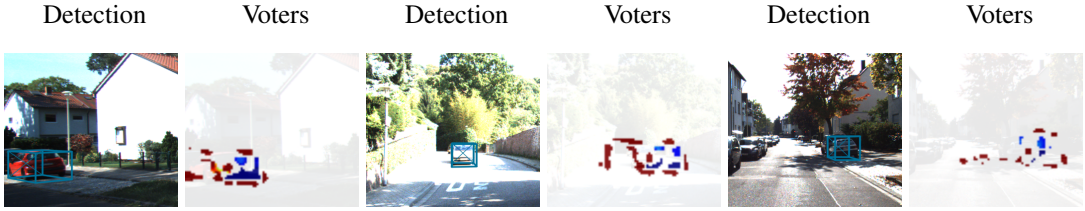


Figure 4.7: Sample *car* detections of HoughNet from KITTI dataset and their vote maps. In the "detection" columns, we show a correctly detected object, marked with a blue 3D bounding box. In the "voters" columns, the locations that vote for the detection are shown. Colors indicate vote strength based on the standard "jet" colormap (red is high, blue is low; Figure 1.1). In all detections, in addition to the local votes, there are strong votes come from surroundings such as road, buildings and wall. Image is taken from our ECCV'20 paper extension [3].

together with standard deviation for each metric as in OAP.

The model with our voting module performs on par with the baseline for bounding box AP, and it outperforms baseline on AOS and BEV AP. Especially BEV AP is significantly improved and is more stable with much less variance compared to baseline. In Figure 4.7, we show visualization of votes for sample *car* detections.

4.3.4 2D Human Pose Estimation

In this section, we show the effectiveness of our voting module for 2D Human Pose Estimation task. Human Pose Estimation is the problem of detecting human joints (i.e. keypoints) in images. It is another detection task that our voting module could

Table 4.13: Comparing our voting module with baseline for 2D human pose estimation on COCO `val2017` set. The voting module is attached to the person classification and keypoint estimation branches separately and concurrently. Results are shown for both COCO keypoint and box AP. Models are trained on COCO `train2017`. Results are presented on SS testing mode.

Method	AP^{kp}	AP^{kp}_{50}	AP^{kp}_{75}	AP^{kp}_{M}	AP^{kp}_{L}	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	AP^{box}_{S}	AP^{box}_{M}	AP^{box}_{L}
Baseline	54.7	**81.7**	59.4	49.0	64.6	47.5	63.9	52.8	15.9	65.8	79.3
+ Voting for Person Class.	**56.9**	81.6	61.9	**51.3**	67.9	50.1	71.4	54.1	16.9	64.7	79.3
+ Voting for Keypoint Est.	56.8	81.5	61.2	50.7	68.0	50.2	70.9	54.3	**17.2**	64.4	79.4
+ Voting for Both	**56.9**	81.6	**62.1**	50.8	**68.4**	**50.4**	**71.7**	**54.4**	17.0	64.9	**79.8**

help by leveraging the interactions both among keypoints and other visual parts.

Our baseline (OAP [40]) considers pose estimation as a regression task and defines each keypoint with an offset to the center of the instance and directly regresses them in a separate branch. In order to refine keypoints, OAP also employs the common bottom-up multi-human pose estimation approach as in [100, 101, 102] and predicts heatmaps for each of k human joints in another branch. More detail on training and inference processes can be found in OAP [40].

We analyze the effect of voting by attaching our voting module to the class prediction and the keypoint heatmap prediction branches both separately and at the same time. We conducted our experiments on COCO dataset which has 17 keypoints for person object instances. For fair comparison with the baseline, we use DLA backbone and initialize it with the center detection model of OAP. We followed exactly the same training setup as in Section 4.2.1. The models are trained on `train2017` and evaluated on `val2017`. In Table 4.13, we show the results for both keypoint AP (AP^{kp}) and box AP (AP^{box}). Baseline results are obtained using the publicly available equivalent model trained with 140 epochs from the official repository of OAP. In all cases, attaching our voting module improved the baseline. Using the voting module in classification and keypoint heatmap prediction branches at the same time gives the best performance for both keypoint AP^{kp} and box AP^{box}. Attaching the voting module improves the baseline by 2.2 and 2.9 points for AP^{kp} and AP^{box}, respectively. Especially in AP^{box}_{50}, our voting module dramatically improved baseline by 7.8 points. We

provide visualization of votes for sample detections in Figure 4.8.

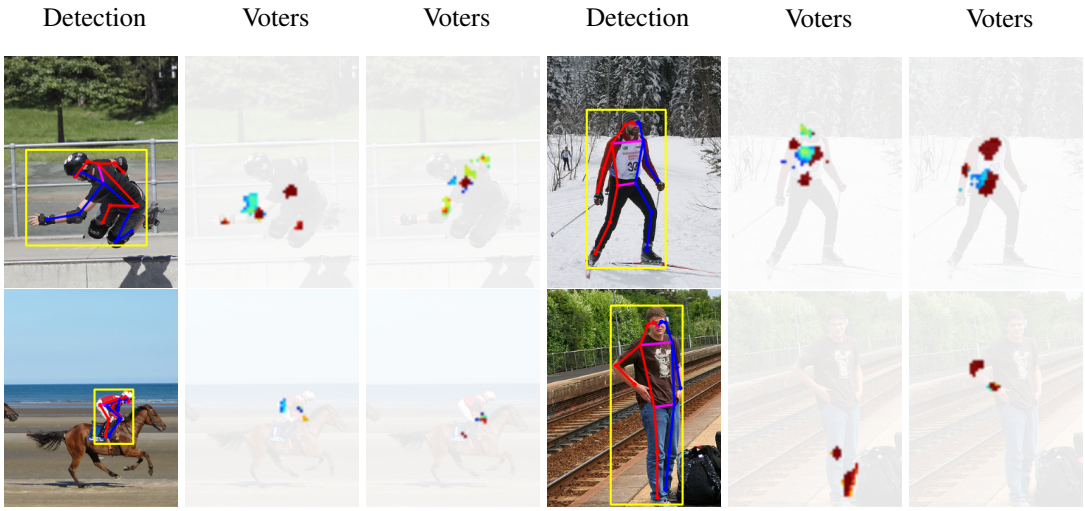


Figure 4.8: Sample detections of HoughNet and their vote maps. In the "detection" columns, we show correctly detected objects and their poses. Detection box is marked with a yellow bounding box, and pose estimation is shown with blue color for the left parts, red color for the right parts. In the "voters" columns, the locations that vote for the detection are shown. Colors indicate vote strength based on the standard "jet" colormap (red is high, blue is low; Figure 1.1). For **the first image of the top row**, votes for the *left-elbow* and *left-shoulder* are shown respectively. The *left-elbow* detection gets strong votes from the *left-shoulder*, *left-knee* and *left-wrist*. The *left-shoulder* detection gets votes from the area spanning from *right-elbow* to *left-elbow*. In **the second image of top row**, the *right-shoulder* and *right-elbow* keypoint detections get strong votes from the upper-body parts. In **the first example of the bottom row**, votes for the *left-elbow* and *left-knee* are shown respectively. Despite heavy occlusion, *left-elbow* gets votes especially from the region of wrinkles. Moreover the invisible *left-knee* gets strong votes from the *right-ankle* and arm region. In **the second image of bottom row**, *right-knee* and *right-wrist* detections and their vote maps are given. Image is taken from our ECCV'20 paper extension [3].

Table 4.14: Comparing our voting module with baseline model (HPRNet) for 2D whole-body human pose estimation on COCO WholeBody validation set. Models are trained on COCO WholeBody training set from scratch. Results are presented on SS testing mode.

Method	body		foot		face		hand		whole-body		all-mean	
	AP^{kp}	AR^{kp}	AP^{kp}	AR^{kp}	AP^{kp}	AR^{kp}	AP^{kp}	AR^{kp}	AP^{kp}	AR^{kp}	AP^{kp}	AR^{kp}
HPRNet	55.2	63.1	49.1	60.9	74.6	83.7	47.0	60.8	31.5	44.6	51.5	62.6
+ Voting	**55.9**	**63.4**	**49.6**	**61.5**	**75.0**	**83.9**	**47.5**	**61.2**	**31.7**	**44.8**	**51.9**	**63.0**

4.3.5 2D Whole-body Human Pose Estimation

In order to show the effectiveness of our voting module for whole-body human pose estimation, we first developed HPRNet. Details on whole-body pose estimation task and HPRNet could be found in Chapter 6. Later on top of HPRNet we attached our voting module to the class prediction branch. We experimented with DLA backbone and followed exactly the same training setup as in Section 4.2.1. As the Table 4.14 shows, attaching our voting module consistently improves the performance for all body parts.

4.3.6 Face Detection

Here we show the effectiveness of our voting module for face detection on Wider Face benchmark [103]. Again, we consider OAP [40] as our baseline. As in 2D object detection we attach our voting module with 3 rings and 90° to the class (i.e face) prediction branch. We experimented with Resnet-18 [6] backbone and followed exactly the same training setup as in Section 4.2.1.

Table 4.15 presents mAP results at 3 levels of difficulty, namely, *easy*, *medium* and *hard*. Our voting module improved the performance by 2.2 point for easy group, 2.0 point for medium group and 1.8 point for hard group.

We further improved the face detection performance adding landmark regression as a supervision task. For this purpose, we define each face landmark with an offset to

Table 4.15: Comparing our voting module with baseline for face detection on Wider Face `val` set. The voting module is attached to the face classification branch. *mAP* results are shown for easy, medium and hard groups. Models are trained on Wider Face `train`. Results are presented on SS testing mode.

Method	mAP_E	mAP_M	mAP_H
Baseline	91.2	89.1	70.1
+ Voting	93.4	91.1	71.9
+ Voting *w* Landmark Supervision	**93.6**	**91.3**	**72.7**

the face center and directly regress them in a separate branch as in 2D Human Pose Estimation in Section 4.3.4. Landmark supervision especially effective for hard group of faces.

4.4 Hough Voting for an Image Generation Task

Another task where long-range interactions could be useful is the task of image generation from a given label map. There are two main approaches to solve this task; using unpaired and paired data for training. We take CycleGAN [104] and Pix2Pix [105] as our baselines for unpaired and paired approaches, respectively. We attach our voting module at the end of CycleGAN [104] and Pix2Pix [105] models.

Table 4.16: Comparison of FCN and LPIPS Scores for the "Labels to Photo" Task on the Cityscapes Dataset [1].

Method	*Per-pixel acc.* ↑	*Per-class acc.* ↑	*Class IOU* ↑	*LPIPS* ↓
CycleGAN	0.43	0.14	0.09	**0.52**
+ Voting	**0.52**	**0.17**	**0.13**	0.53
pix2pix	0.71	**0.25**	0.18	**0.43**
+ Voting	**0.76**	**0.25**	**0.20**	0.44

For quantitative comparison, we use the Cityscapes [1] dataset. In Table 4.16, we present FCN scores [106] (which is used as the measure of success in this task) of

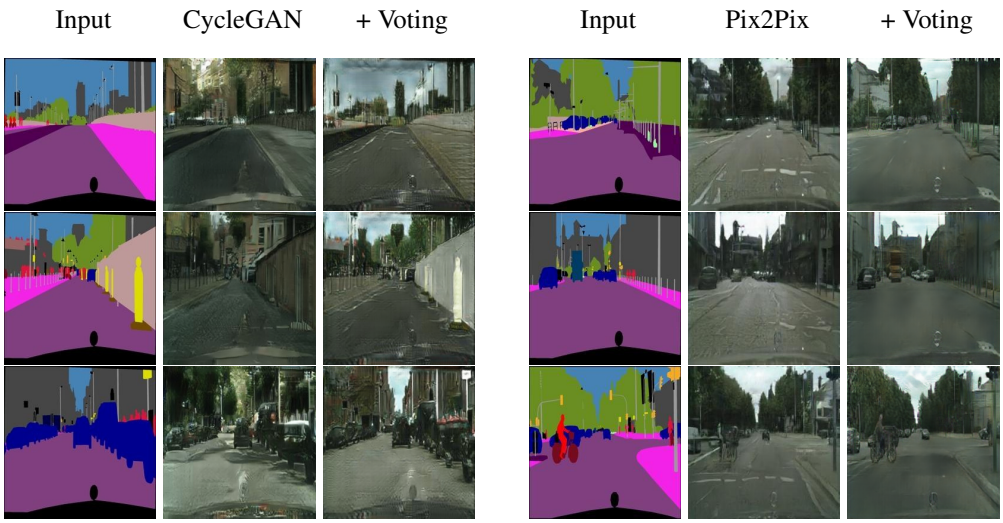


Figure 4.9: Sample qualitative results for the "labels to photo" task. When integrated with CycleGAN, our voting module helps generate better images in the sense that the image conforms to the input label map better. In all three images, CycleGAN fails to generate sky, buildings and falsely generates vegetation in the last image. When used with Pix2Pix, it helps generate more detailed images. In the first row, cars and buildings can be barely seen for Pix2Pix. Similarly, a bus is generated as a car and a bicycle is silhouetted in the second and third images, respectively. Our voting module fixes these errors. Image is taken from our ECCV'20 paper [3].

CycleGAN and Pix2Pix with and without our voting module. To obtain the "without" result, we used the already trained model shared by the authors. We obtained the "with" result using the official training code from their repositories. In both cases evaluation was done using the official test and evaluation scripts from their repos. Results show that using the voting module improves FCN scores by large margins.

For the realism analysis of generated images, we also present Learned Perceptual Image Patch Similarity (LPIPS) [107] scores in Table [106]. Lower LPIPS values are better. Attaching our voting module has a slightly negative effect on realism of generated images. For this part, further research is needed.

Qualitative inspection also shows that when our voting module is attached, the generated images conform to the given input segmentation maps better (Figure 4.9). This is the main reason for the quantitative improvement. Since Pix2Pix is trained with

Table 4.17: Comparison of HoughNet with NLNN and RN context models.

Method	*Baseline*	*Backbone*	*Improvement (AP)*
RN	FasterRCNN [9]	ResNet-101 *w* DCN	1.0
NLNN	MaskRCNN [20]	ResNet-101	1.3
HoughNet	OAP [40]	ResNet-101 *w* DCN	**2.2**

paired data, generated images follow input segmentation maps, however, Pix2Pix fails to generate small details.

4.5 Comparing HoughNet with Context Models

We compare HoughNet with the established context models Non-local neural networks (NLNN) [71] and Relation networks (RN) [60] for object detection. NLNN integrates long-range features. There is one fundamental difference between NLNN and HoughNet. In NLNN, the relative displacement between interacting features is not taken into account. However, HoughNet uses this information encoded through the regions of the log-polar vote field. NLNN and HoughNet do not have a common experimental configuration. In the closest experiment, NLNN reports 1.3 *AP* improvement over MaskRCNN [20] with ResNet-101 backbone, and HoughNet reports 2.2 *AP* improvement over OAP [40] with the similar ResNet-101 *w* DCN backbone (see Table 4.17).

RN is a method developed for proposal-based two-stage detectors. Object-object relations are explicitly modeled. In this sense, long range evidence is used. RN and HoughNet do not have a common experimental configuration. In the closest experiment, RN reports 1.0 *AP* improvement over FasterRCNN [9] with ResNet-101 *w* DCN backbone, and HoughNet reports 2.2 *AP* improvement over OAP [40] with the similar ResNet-101 *w* DCN backbone (see Table 4.17).

CHAPTER 5

REDUCING LABEL NOISE IN ANCHOR-FREE OBJECT DETECTION

This chapter is taken from our BMVC'20 paper [24].

5.1 Introduction

Early deep learning based object detectors were two-stage, proposal driven methods [9, 108]. In the first stage, a sparse set of object proposals are generated and a convolutional neural network (CNN) categorizes them in the second stage. Later, the idea of unified detection in a single stage has gained increasing attention [10, 35, 8, 36], where proposals were replaced with predefined anchors. On the one hand, anchors have to cover the image densely (in terms of location, shape and scale) so as to maximize recall; on the other hand, their number should be kept at a minimum to reduce both the inference time and the imbalance problems [109] they create during training.

A considerable amount of effort has been spent on addressing the drawbacks of anchors: several methods have been proposed to improve the quality of anchors [38, 110], to address the extreme foreground-background imbalance [111, 8, 109], and recently, one-stage anchor-free methods have been developed. There are two main groups of prominent approaches in anchor-free object detection. The first group is keypoint based, bottom-up methods, popularized after the pioneering work CornerNet [14]. These detectors [14, 15, 40, 112] first detect keypoints (e.g. corners, center and extreme points) of objects, and then group them to yield whole-object detections. The second group of anchor-free object detectors [39, 113, 92] follow a top-down approach, and directly predict class and bounding box coordinates at each location in the final feature map(s).

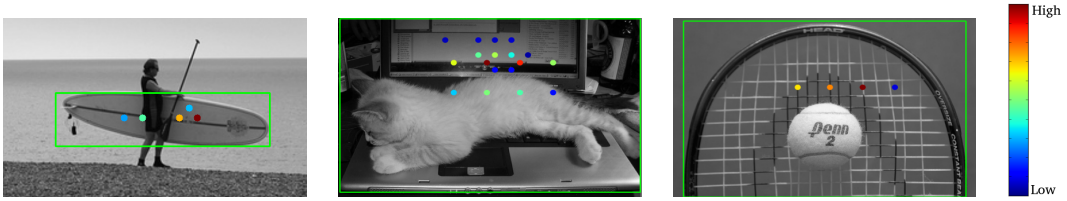


Figure 5.1: Three sample detections by PPDet, from left to right: surfboard, laptop and racket. The colored dots show the locations whose predictions are pooled to generate the final detection shown in the green bounding box. The color denotes the contribution weight. Highest contributions are coming from the objects and not occluders or background areas. Images are from COCO `val2017` set. Image is taken from our BMVC'20 paper [24].

One important aspect of object detector training is the strategy used to label object candidates, which could be proposals, anchors or locations (i.e. features) in the final feature map. In order to label a candidate 'positive' (foreground) or 'negative' (background) during training, a variety of strategies have been proposed, based on *Intersection over Union (IoU)* [9, 87, 10, 8], *keypoints* [14, 15, 40, 112] and *relative location to a ground-truth box* [39, 114, 113]. Specifically in top-down anchor-free object detectors, after the input image is passed through the backbone feature extractor and the FPN [88], features that spatially fall inside a ground-truth box are labeled as positive and others as negative – there is also an "ignore" region in between. Each of these positively-labeled features contributes to the loss function as a separate prediction. The problem with this approach is that some of these positive labels might be plain-wrong or of poor quality, hence, they inject label noise during training. Noisy labels come from (i) non-discriminatory features that are on the object, (ii) background features within the ground-truth box, and (iii) occluders (Figure 5.1). In this work, we propose an anchor-free object detection method, which relaxes the positive labeling strategy so that the model is able to reduce the contributions of non-discriminatory features during training. In accordance with this training strategy, our object detector employs an inference method where highly-overlapping predictions enforce each other.

In our method, during training, we define a "positive area" within a ground-truth (GT)

box, which is co-centric and has the same shape with the GT box. We experimentally adjust the size of the positive area relative to the GT box. As this is an anchor-free method, each feature (i.e. location in the final feature maps) predicts a class probability vector and bounding box coordinates. The class predictions from the positive area of a GT box get pooled together and contribute to the loss as a single prediction. This sum-pooling alleviates the noisy-labels problem mentioned above since the contributions of features from non-object (background or occluded) areas, and non-discriminatory features are automatically down weighted during training. At inference, class probabilities of highly overlapping boxes are again pooled together to obtain the final class probabilities. We name our method as "PPDet", which is short for "prediction pooling detector."

Our contributions with this work are two fold: (i) a relaxed labelling strategy, which allows the model to reduce the contribution of non-discriminatory features during training, and (ii) a new object detection method, PPDet, which uses this strategy for training and a new inference procedure based on prediction pooling. We show the effectiveness of our proposal on the COCO dataset. PPDet outperforms all anchor-free top-down detectors and performs on-par with the other state-of-the-art methods. PPDet is especially effective for detecting small objects ($31.4\ AP_S$, better than state-of-the-art).

5.2 Related Work

Apart from the classical one-stage [10, 35, 8, 36] vs. two-stage [9, 108, 87] categorization of object detectors, we can also categorize the current approaches into two: anchor-based and anchor-free. Top-down anchor-free object detectors simplify the training process by eliminating complex IoU operations and focus on identifying the regions that may contain objects. In that sense, FCOS [39], FSAF [92] and FoveaBox [113] first map GT boxes onto the FPN levels, then label the locations, i.e. features, as positive or negative based on whether they are inside a GT box. Bounding box prediction is only for positively-labeled locations. FoveaBox [113] and FSAF [92] define three areas for each object instance; *positive* area, *ignore* area and *negative* area. FoveaBox defines the *positive* (*fovea*) area as the region which

is co-centric with the GT box, and whose dimensions are scaled by a (shrink) factor 0.3. All locations within this positive area are labeled as positive. Similarly, another area is obtained using a shrink factor of 0.4. Any location that is outside this area is labeled as negative. If a location is neither positive nor negative, it is ignored during training. FSAF follows the same approach and uses shrink factors 0.2 and 0.5, respectively. Instead of having pre-defined discrete areas as in [92, 113, 114], FCOS down-weights the features based on their distance to the center using a centerness branch. FCOS and FoveaBox implement static feature-pyramid level selection where they assign objects to levels based on GT box scale and GT box regression distance, respectively. Unlike them, FSAF relaxes the feature selection step and dynamically assigns each object to the most suitable feature-pyramid level.

Bottom-up anchor-free object detection methods [14, 15, 40, 112] aim to detect certain keypoints of objects, such as corners and the center. Their labeling strategy uses heatmaps, and in this sense, it is considerably different from that of top-down anchor-free methods. Also, HoughNet, a novel, bottom-up voting-based method that can utilize both near and long-range evidence to detect object centers, has shown comparable performance with major one-stage and two-stage top-down methods (see Chapter 4).

In the anchor-based approaches [9, 87, 10, 35, 8, 93, 114], objects are predicted from regressed anchor boxes. During training, the label of an anchor box is determined based on its intersection over union (IoU) with a GT box. Different detectors use different criteria, e.g. Faster RCNN [9] labels an anchor as positive if $IoU > 0.7$, and negative if $IoU < 0.3$; R-FCN [87], SSD [10] and Retinanet [8] use $IoU > 0.5$ for positive labeling but slightly different criterias for negative labeling. There are two prominent anchor-based methods which directly address the labeling problem. Guided Anchoring [114] introduces a new adaptive anchoring scheme that learns arbitrary shaped boxes instead of dense and predefined ones. Similar to FSAF [92], FoveaBox [113] and our method PPDet, Guided Anchoring follows region based labelling and defines three types of regions for each ground-truth object; *center* region, *ignore* region and *outside* region, and labels the generated anchors positive if it resides inside the *center* region, negative if in *outside* region and ignores the rest. On the other hand, FreeAnchor [93] applies the idea of relaxing positive labels for anchor-based

detectors. This is the most similar method to ours. It replaces hand-crafted anchor assignment with a maximum likelihood estimation procedure, where anchors are set free to choose their GT box. Since FreeAnchor is optimizing object-anchor matching using a customized loss function, it can not be directly applied to anchor-free object detectors.

5.3 Methods

5.3.1 Labeling Strategy and Training

Anchor-free detectors limit prediction of GT boxes by assigning them to appropriate FPN levels based on their scales [113] or target regression distances [39]. Here, we follow the scale-based assignment strategy [113] since it is a way of naturally associating GT boxes with feature pyramid levels. Then, we construct two different regions for each GT box. We define the "positive area" as the region that is co-centric with the GT box and having the same shape as the GT box. We experimentally set the size of the "positive area". Then, we identify all the locations (i.e. features) that spatially fall inside the "positive area" of a GT box as "positive (foreground)" features and the rest as "negative (background)" features. Each positive feature is assigned to the ground-truth box that contains it. In Figure 5.2, blue and red cells represent foreground cells and the rest (empty or white) are background cells. The blue cells are assigned to the *frisbee* object and the red cells to the *person* object. To obtain the final detection score for an object instance, we pool the classification scores of all the features that are assigned to that object, by adding them together to obtain a final C-dimensional vector where C is the number of the classes. All features except the positively labelled ones are negatives. Each negative feature contributes individually to the loss (i.e. no pooling). This final prediction vector is fed to the focal loss (FL). For example, suppose $\{\mathbf{p}_i | i = 1, 2, \ldots, N\}$ represent the red, foreground features that are assigned to the person object in Figure 5.2. Let $\mathbf{y}$ be the ground-truth, one-hot vector for the person class. Then, this particular object instance contributes "FL($\sum_i \mathbf{p}_i, \mathbf{y}$)" to the loss function in training. Each object instance is represented with a single prediction.

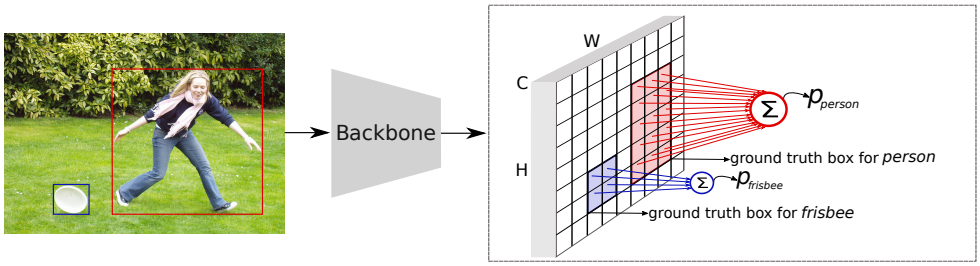


Figure 5.2: Prediction pooling during training of *PPDet*. For simplicity, it is illustrated on a single FPN level and the bounding box regression branch is not shown. Blue and red cells are foreground cells. Same color foreground cells, each of which is a C-dimensional vector, are pooled, i.e. summed together, to form the final prediction score for the corresponding object. These pooled scores (i.e., $p_{\text{person}}, p_{\text{frisbee}}$), are fed to the loss function (i.e., focal loss). Image is taken from our BMVC'20 paper [24].

By default, we assign positive features to the object instance of the box they are in. At this point, assignment of features in the intersection areas of different GT boxes is an issue to be handled. In such cases, we assign those features to the GT box with the smallest distance to their centers. Similar to other anchor-free methods [39, 92, 113, 40], in our model each foreground feature assigned to an object is trained to predict the coordinates of its object's GT box.

We use the focal loss [8] ($\alpha = 0.4$ and $\gamma = 1.5$) for the classification branch and smooth L_1 loss [108] for the regression branch.

5.3.2 Inference

Inference pipeline of PPDet is given in Figure 5.3. First, the input image is fed to a backbone neural network model (described in the next section) which produces the initial set of detections. Each detection is associated with (i) a bounding box, (ii) an object class (chosen as the class with maximum probability) and (iii) a confidence score. Within these detections, those labeled with the background class are eliminated. We consider each remaining detection at this stage as a vote for the object that it belongs to, where the box is an hypothesis for the location of the object and the confidence score is the strength of the vote. Next, these detections are pooled together

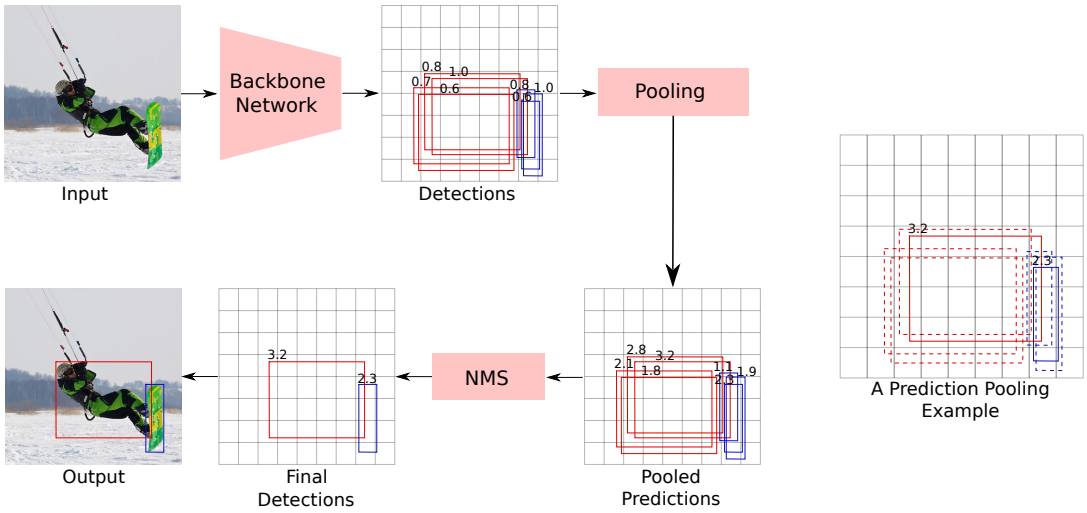


Figure 5.3: (Left) Illustration of PPDet's inference pipeline. Predicted boxes for *person* and *snowboard* are shown in red and blue, respectively. Red and blue boxes vote for each other among themselves. See text for details. (Right) A pooling example. The dashed-boundary red boxes vote for the solid red box and the dashed-boundary blue boxes vote for the solid blue box. Final scores (after aggregation) of solid boxes are shown. Image is taken from our BMVC'20 paper [24].

as follows. If two detections belonging to the same object class overlap more than a certain amount (i.e. intersection over union (IoU) > 0.6), then we consider them as voting for the same object and the score of each detection is increased by $k^{(IoU-1.0)}$ times the score of the other detection, where k is a constant. The more the IoU, the higher the increase. After applying this process to every pair of detections, we obtain the scores for final detections. This step is followed by the class aware non-maxima suppression (NMS) operation which yields the final detections.

Note that although the prediction pooling used in inference might seem to be different from the pooling employed in training, in fact, they are the same process. The pooling used in training makes the assumption that the bounding boxes predicted by the features in the positive area overlap among each other perfectly (i.e. IoU=1).

5.3.3 Network Architecture

PPDet uses the network model of RetinaNet [8] which consists of a backbone convolutional neural network (CNN) followed by a feature pyramid network (FPN) [88]. The FPN computes a multi-scale feature representation and produces feature maps at five different scales. There are two separate, parallel networks on the top of each FPN layer, namely classification network and regression network. The classification network outputs a $W \times H \times C$ tensor where W and H are spatial dimensions (width and height, respectively) and C is the number of the classes. Similarly, the regression network outputs a $W \times H \times 4$ tensor where 4 is the number of bounding box coordinates. We refer to each pixel in these tensors as a *feature*.

5.4 Experiments

This section describes the experiments we conducted to show the effectiveness of our proposed method. First, we present ablation experiments to find the optimal relative area of the positive region within GT boxes and the regression loss weight. Next, we present several performance comparisons on the COCO dataset. Finally, we provide sample heatmaps which show the GT box relative locations of features responsible for correct detections.

5.4.1 Implementation Details

We use Feature Pyramid Network (FPN) [88] on top of ResNet [6] and ResNeXt [115] as our backbone networks for ablations and state of the art comparison, respectively. For all experiments, we resize the images such that their shorter side is 800 pixels and longer side is maximum 1300 pixels. The constant k used in vote aggregation (i.e., $k^{\mathrm{IoU}-1}$) was set to 40 experimentally. We trained all of the experiments on 4 Tesla V100 GPUs, and tested using a single Tesla V100 GPU. We used MMDetection [116] framework with Pytorch [117] to implement our models.

5.4.2 Ablation Experiments

Unless stated otherwise, in ablation experiments we used ResNet-50 with FPN backbone. They are trained with a batch size of 16 for 12 epochs using stochastic gradient descent (SGD) with weight decay of 0.0001 and momentum of 0.9. Initial learning rate 0.01 was dropped $10\times$ at epochs 8 and 11. All ablation models are trained on COCO [7] `train2017` dataset and tested on `val2017` set.

5.4.2.1 Size of the "Positive Area"

As explained before, we define the "positive area" as the region that is co-centric with the GT box and that has the same shape as the GT box. We adjust the size of this "positive area" by multiplying its width and height with a shrink factor. We experimented with shrink factors between 1.0 and 0.2. Performance results are presented in Table 5.1. From shrink factor 1.0 to 0.4, *AP* increases, however, after that point performance degrades dramatically. Based on this ablation, we set the shrink factor to 0.4 for the rest of our experiments.

Table 5.1: Experiments to determine the best shrink factor which defines the relative size of the "positive area" with respect to the GT box. Models were trained on `train2017` and results were obtained on `val2017`.

Shrink Factor	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
1.0	30.0	44.4	32.5	17.2	33.9	38.4
0.8	32.4	47.9	35.1	18.0	36.4	41.9
0.6	34.5	51.3	37.5	19.6	38.5	44.5
0.4	**36.0**	**53.6**	**39.0**	**20.4**	**39.6**	**46.6**
0.2	32.6	50.3	34.7	17.9	36.3	42.5

5.4.2.2 Regression Loss Weight

To find the optimal balance between the classification and regression loss, we conducted ablation experiments on the regression loss weight. As shown in Table 5.2,

0.75 yields the best results. We set the weight of the regression loss to 0.75 for the rest of our experiments.

Table 5.2: Experiments on regression loss (RL) weight. Models were trained on `train2017` and results were obtained on `val2017`.

RL weight	*AP*	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
1.00	36.0	53.6	39.0	20.4	39.6	46.6
0.90	36.0	53.9	39.3	20.1	39.6	47.2
0.75	**36.3**	54.3	**39.5**	**21.1**	39.5	**47.5**
0.60	36.2	**54.6**	**39.5**	21.0	**40.1**	47.1

5.4.2.3 Improvements

We also employed improvements used in other state-of-the-art object detectors [39, 113, 40]. First, we trained our baseline model using ResNet-101 with FPN backbone. Later, we replaced the last convolution layer before class prediction in the classification branch with deformable convolutional layers. This modification improved the performance around 0.3 for all *APs* (see Table 5.3). Later, on top of this modification, we add another one where we adopt group normalization after each convolution layer in the regression and classification branches. As seen in Table 5.3, this modification increased *AP* by 0.6 and AP_{50} by 1.1. In this table, we also provide results for the recently introduced *moLRP* [21] metric, which combines localization, precision and recall in a single metric. Lower values are better. Models are trained with a batch size of 16 for 24 epochs using stochastic gradient descent (SGD) with weight decay of 0.0001 and momentum of 0.9. Initial learning rate 0.01 was dropped $10\times$ at epochs 16 and 22. We include these two modifications in our final model.

5.4.2.4 Class Imbalance

PPDet sum-pools predictions into a single prediction per object instance which reduces the number of positives during training. One may think that it exacerbates the

Table 5.3: Experiments on improvements. Using deformable convolution in the classification branch and group normalization layers further improve detection performances. Models are trained on `train2017` and tested on `val2017` set.

Method	*AP*	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	*moLRP* $\downarrow$
Baseline	39.6	58.0	43.4	23.9	44.1	51.0	68.9
+ Deform. Conv.	39.9	58.4	43.7	24.2	44.4	51.3	68.7
+ Group Norm.	**40.5**	**59.5**	**44.2**	**25.4**	**44.7**	**52.3**	**67.8**

class imbalance [109] even more. To analyse the issue, we calculated the average number of positives per image, which is 7 for PPDet, 41 for FoveBox and 165 for RetinaNet. PPDet considerably decreases the number of positives. However, this is still small compared to the number of negatives (tens of thousands), hence, it does not exacerbate the existing class imbalance problem. We use focal loss to tackle the imbalance.

5.4.3 State-of-the-art Comparison

To compare our model with the state-of-the-art methods, we used ResNet-101 with FPN and ResNeXt-101-64x4d with FPN backbones. They are trained with batch sizes of 16 and 8 for 24 and 16 epochs, respectively, using SGD with weight decay of 0.0001 and momentum of 0.9. For the ResNet backbone, initial learning rate 0.01 was dropped $10\times$ at epochs 16 and 22. For the ResNeXt backbone, initial learning rate 0.005 was dropped $10\times$ at epochs 11 and 14. The models are trained on COCO [7] `train2017` dataset and tested on `test-dev` set. We used $(800, 480)$, $(1067, 640)$, $(1333, 800)$, $(1600, 960)$, $(1867, 1120)$, $(2133, 1280)$ scales for multi-scale testing. Table 5.4 presents performances of PPDet and several established state-of-the-art detectors.

FSAF [92] and FoveaBox [113] use a similar approach to ours to build the "positive area". While single scale testing performance of PPDet is comparable with that of FSAF on the same ResNeXt-101-64x4d with FPN backbone, PPDet's multi-scale

Table 5.4: Detection performances on COCO `test-dev` set. The methods are divided into three groups: two-stage, one-stage anchor-based and one-stage anchor-free. The best results are boldfaced separately for each group. PPDet achieves state-of-the-art results on the AP_S metric among all the detectors. * results are taken from MMDetection. † MS test for FoveaBox is implemented by us on top of the original code.

Method	Backbone	Train size	Test size	*AP*	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	FPS
Two-stage detectors:										
R-FCN [87]	ResNet-101	800×800	600×600	29.9	51.9	-	10.8	32.8	45.0	5.9
CoupleNet [49]	ResNet-101	ori.	ori.	34.4	54.8	37.2	13.4	38.1	50.8	-
Faster R-CNN+++ [6]	ResNet-101	1000×600	1000×600	34.9	55.7	37.4	15.6	38.7	50.9	-
Faster R-CNN [88]	ResNet-101-FPN	1000×600	1000×600	36.2	59.1	39.0	18.2	39.0	48.2	5.0
Mask R-CNN [20]	ResNeXt-101-FPN	1300×800	1300×800	39.8	62.3	43.4	22.1	43.2	51.2	11.0
Cascade R-CNN [89]	ResNet-101	-	-	42.8	62.1	46.3	23.7	45.5	55.2	**12.0**
PANet [90]	ResNeXt-101	1400×840	1400×840	**47.4**	**67.2**	**51.8**	**30.1**	**51.7**	**60.0**	-
One-stage, anchor-based:										
SSD [10]	VGG-16	512×512	512×512	28.8	48.5	30.3	10.9	31.8	43.5	-
YOLOv3 [35]	Darknet-53	608×608	608×608	33.0	57.9	34.4	18.3	35.4	41.9	**20.0**
DSSD513 [36]	ResNet-101	513×513	513×513	33.2	53.3	35.2	13.0	35.4	51.1	-
RefineDet (SS) [91]	ResNet-101	512×512	512×512	36.4	57.5	39.5	16.6	39.9	51.4	-
RetinaNet [8]	ResNet-101-FPN	1300×800	1300×800	39.1	59.1	42.3	21.8	42.7	50.2	10.9*
RetinaNet [8]	ResNeXt-101-FPN	1300×800	1300×800	40.8	61.1	44.1	24.1	44.2	51.2	7.0*
RefineDet (MS) [91]	ResNet-101	512×512	≤2.25×	41.8	62.9	45.7	25.6	45.1	54.1	-
GA-RetinaNet [114]*	ResNet-101	1300×960	1300×800	41.9	62.2	45.3	24.0	45.3	53.8	-
FreeAnchor (SS) [93]	ResNeXt-101-FPN	1300×960	1300×960	44.9	64.3	48.5	26.8	48.3	55.9	8.4*
FreeAnchor (MS) [93]	ResNeXt-101-FPN	1300×960	∼≤2.0×	**47.3**	**66.3**	**51.5**	**30.6**	**50.4**	**59.0**	-
Anchor-free, bottom-up:										
ExtremeNet (SS) [15]	Hourglass-104	511×511	ori.	40.2	55.5	43.2	20.4	43.2	53.1	3.1
CornerNet (SS) [14]	Hourglass-104	511×511	ori.	40.5	56.5	43.1	19.4	42.7	53.9	4.1
CornerNet (MS) [14]	Hourglass-104	511×511	≤1.5×	42.1	57.8	45.3	20.8	44.8	56.7	-
CenterNet (SS) [40]	Hourglass-104	512×512	ori.	42.1	61.1	45.9	24.1	45.5	52.8	**7.8**
HoughNet (SS) [2]	Hourglass-104	512×512	≤ ori.×	43.1	62.2	46.8	24.6	47.0	54.4	6.4
ExtremeNet (MS) [15]	Hourglass-104	511×511	≤1.5×	43.7	60.5	47.0	24.1	46.9	57.6	-
CenterNet (SS) [112]	Hourglass-104	511×511	ori.	44.9	62.4	48.1	25.6	47.4	57.4	3.0
CenterNet (MS) [40]	Hourglass-104	512×512	≤1.5×	45.1	63.9	49.3	26.6	47.1	57.7	-
HoughNet (MS) [2]	Hourglass-104	512×512	≤1.8×	46.4	**65.1**	**50.7**	**29.1**	48.5	58.1	-
CenterNet (MS) [112]	Hourglass-104	511×511	≤1.8×	**47.0**	64.5	**50.7**	28.9	**49.9**	**58.9**	-
Anchor-free, top-down:										
FoveaBox [113] (SS)	ResNet-101-FPN	1300×800	1300×800	40.6	60.1	43.5	23.3	45.2	54.5	-
FoveaBox [113] (SS)	ResNeXt-101-FPN	1300×800	1300×800	42.1	61.9	45.2	24.9	46.8	55.6	-
FSAF (SS) [92]	ResNeXt-101-FPN	1300×800	1300×800	42.9	63.8	46.3	26.6	46.2	52.7	2.7
FoveaBox [113] (MS)†	ResNet-101-FPN	1300×800	1300×800	44.2	**65.4**	47.8	28.8	46.7	53.7	-
FSAF (MS) [92]	ResNeXt-101-FPN	1300×800	∼≤2.0×	44.6	65.2	48.6	29.7	47.1	54.6	-
FCOS [39]	ResNeXt-101-FPN	1300×800	1300×800	44.7	64.1	48.4	27.6	47.5	55.6	7.0*
PPDet (SS)	ResNet-101-FPN	1300×800	1300×800	40.7	60.2	44.5	24.5	44.4	49.7	**7.5**
PPDet (SS)	ResNeXt-101-FPN	1300×800	1300×800	42.3	62.0	46.3	26.2	46.0	51.9	4.1
PPDet (MS)	ResNet-101-FPN	1300×800	∼≤2.0×	45.2	63.5	50.3	30.0	48.6	54.7	-
PPDet (MS)	ResNeXt-101-FPN	1300×800	∼≤2.0×	**46.3**	64.8	**51.6**	**31.4**	**49.9**	**56.4**	-

testing performance is 1.7 *AP* points better than that of FSAF's. Our both models with single-scale testing get slightly better results than FoveaBox while outperforming it on small objects by more than 1.0. The results of our multi-scale testing outperforms FoveaBox by 1 *AP* on the same ResNet-101 with FPN backbone.

Our multi-scale performance is the best among all the anchor-free top-down methods. Moreover, our multi-scale performance on small objects (i.e. AP_S) sets the new state-of-the-art among all detectors in Table 5.4.

We conducted experiments to analyse the effect of the prediction pooling for training and inference. When we removed the prediction pooling from the inference pipeline of our ResNet-101-FPN backbone model, we observed that *AP* goes down by 2.5 points on `val2017` set. To analyse the effect of prediction pooling for training, we added prediction pooling to RetinaNet [8] and FoveaBox [113] only during inference (so, no PP in training). This resulted in 0.5 and 2.8 points drop in *AP* for RetinaNet and FoveaBox, respectively.

We also conducted another experiment to test the effectiveness of sum-pooling over max-pooling. For max-pooling, we identified the feature within the positive area, whose predicted box overlaps the most with the GT box. Then, only this feature is included in focal loss to represent its GT box during training. This strategy dropped *AP* by more than 2 points, yielding 38.4 with ResNet101 with FPN backbone.

As an additional result, we present the performance of PPDet on the PASCAL VOC dataset [118]. For training, we used the union set of PASCAL VOC 2007 `trainval` and VOC 2012 `trainval` images ("07+12"). For testing, we used the `test` set of PASCAL VOC 2007. Our PPDet model achieves 77.8 mean average precision (mAP) outperforming FoveaBox [113] at 76.6 mAP, which we consider as a baseline here, when both use the ResNet-50 backbone.

Figure 5.4 shows the heatmap of cell centers relative to the ground-truth box, which are responsible for detection. The heatmaps of RetinaNet are concentrated at the center of the ground-truth object boxes. In contrast, PPDet's final detections are formed from a relatively wider area verifying its dynamic and automatic characteristics on assigning weights to the features in the positive area. In addition to the detections

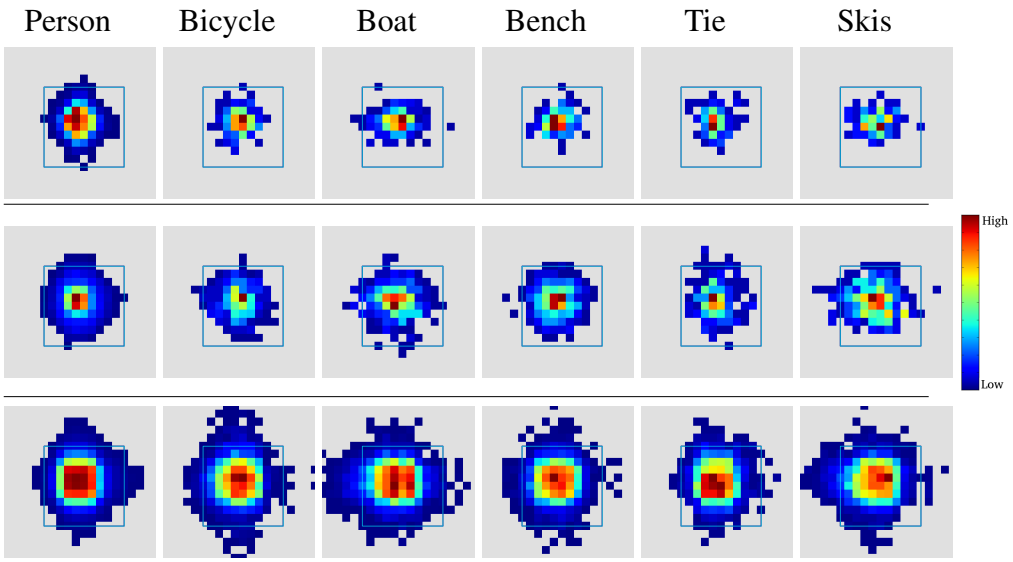


Figure 5.4: Feature locations that are responsible for detection during inference, relative to the ground-truth box (blue rectangle). To bring different ground-truth boxes into the same plot, we normalized each ground-truth box to a canonical size. Relative locations of responsible features were normalized accordingly. **Top row** shows the heatmap of responsible feature locations for anchor-based RetinaNet. **Second row** shows the same for anchor-free object detector FoveaBox. **Bottom row** shows the same for PPDet. Heatmaps were obtained on COCO `val2017` images with ResNet-101 with FPN backbone. RetinaNet detects objects mostly with center cells. In terms of peakyness, FoveaBox's heatmaps are similar to RetinaNet's. PPDet detects objects from a wider area, also from outside of the object box. Image is taken from our BMVC'20 paper [24].

coming from the center of the ground-truth box, they may heavily come from the different parts of the ground-truth box.

5.5 Conclusion

In this work, we introduced a novel labeling strategy for the training of anchor-free object detectors. While current anchor-free methods force positive labels on all the features that are spatially inside a predefined central region of a ground-truth box, our labeling strategy relaxes this constraint by sum-pooling predictions stemming

from individual features into a single prediction. This allows the model to reduce the contributions of non-discriminatory features during training. We developed PPDet, a one-stage, anchor-free object detector which employs the new labeling strategy during training and a new inference method based on pooling predictions. We analyzed our idea by conducting several ablation experiments. We reported results on COCO `test-dev` and show that PPDet performs on par with the state-of-the-art and achieves state-of-the-art results on small objects (AP_S 31.4). We further validated the effectiveness of our method through visual inspections.

CHAPTER 6

HIERARCHICAL POINT REGRESSION FOR WHOLE-BODY HUMAN POSE ESTIMATION

This chapter is taken from our IMAVIS journal paper [25].

6.1 Introduction

As a challenging computer vision task, human pose estimation aims to localize human body keypoints in images and videos. Human pose estimation has an important role in several vision tasks and applications such as action recognition [119, 120, 121, 122, 123], human mesh recovery [124, 125, 126, 127], augmented/virtual reality [128, 129, 130], animation and gaming [131, 132, 133, 134]. Unlike the standard human pose estimation task, whole-body pose estimation aims to detect face, hand and foot keypoints in addition to the standard human body keypoints. The challenge in this problem is the extreme scale variance or imbalance among different whole-body parts. For example, the relatively small scale of face and hand keypoints make accurate localization of face and hand keypoints more difficult compared to the standard body keypoints such as elbow, knee and hip. Direct application of existing human pose estimation methods do not yield satisfactory results due to this scale variance problem.

Even though human pose estimation has been well studied for the past few decades, the whole-body pose estimation task has not been sufficiently explored, mainly due to the lack of large-scale fully annotated whole-body keypoint datasets. The previous few methods [135, 136], trained several deep networks separately on different face, hand and body datasets, and ensembled them during inference. These methods suffer

from issues arising from datasets' biases, variations of illumination, pose and scales, and complex training and inference pipelines.

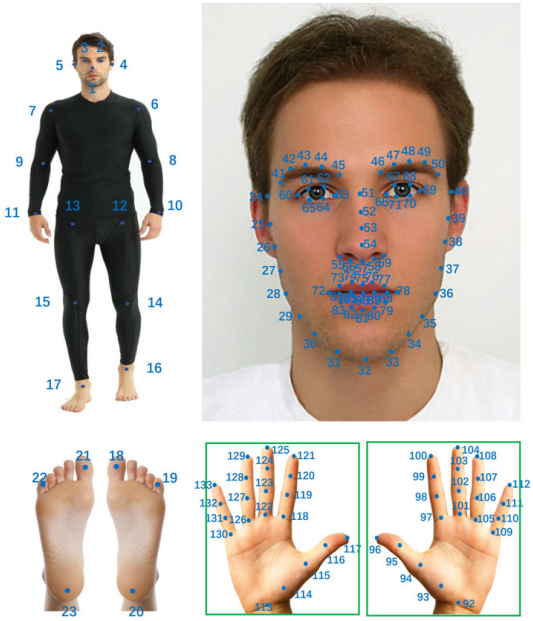


Figure 6.1: Whole-body keypoints as defined in the COCO WholeBody dataset. There is a total of 133 keypoints. In addition to standard 17 human body keypoints (top-left) from the COCO keypoints dataset, there are 68 face (top-right), 42 hand (21 keypoints for each) (bottom-right) and 6 foot (3 for each) (bottom-left) keypoints are annotated. Image source: `https://github.com/jin-s13/COCO-WholeBody`

Recently, in order to address the missing benchmark issue, Jin et al. [23] introduced a novel dataset for whole-body pose estimation, called COCO WholeBody. COCO WholeBody extends COCO keypoints dataset [137] by further annotating face, hand and foot keypoints. In addition to the standard, 17 human body keypoints from the COCO keypoints dataset; 68 facial landmarks, 42 hand keypoints and 6 foot keypoints are annotated (Figure 6.1). Along with these 133 whole-body keypoint annotations, the dataset also has face and hand bounding box annotations that were automatically computed from the extreme keypoints of the corresponding part. They also proposed a strong baseline, called ZoomNet, which has set the state of the art. ZoomNet is a top-down, two-stage method based on the human pose estimation model HRNet [138]. Given an image, ZoomNet first detects person instances using the FasterRCNN [9]

person detector, then it predicts 17 body and 6 foot keypoints using a CNN model. Later, to overcome the scale variance between whole-body parts, ZoomNet crops the hand and face areas that it detected and transforms them to higher resolutions using seperate CNNs to further perform face and hand keypoint estimation.

There are two main approaches for human pose and whole-body pose estimation; *bottom-up* [139, 140, 141, 142, 143, 144, 145, 146, 136, 147, 148, 149, 150, 151] and *top-down* [152, 153, 20, 154, 138, 155]. Bottom-up methods directly detect human body keypoints and later group them to obtain final poses for each person in a given image. On the other hand, top-down methods (e.g. ZoomNet) first detect and extract person instances, then apply pose estimation on each instance separately. The grouping stage of bottom-up methods is more efficient than repeating pose estimation for each person instance. As a result, top-down methods slow down with the increasing number of people (Figure 6.6). However, compared to bottom-up methods, better accuracies are obtained by top-down approaches.

In this work, we propose a new bottom-up method, HPRNet, that explicitly handles the hierarchical nature of whole-body pose estimation by regressing keypoints hierarchically. To this end, in addition to the estimation of standard body keypoints, we define the bounding box centers of relatively small body parts such as face and hands with offsets to the person instance center (Figure 6.3). Concurrently, we build another level of regression where we define each hand and face keypoints with an offset to their corresponding hand and face bounding box centers. We jointly train each level of regression hierarchy and regress all whole-body keypoints with respect to their defined center points. This hierarchical bottom-up approach brings two benefits. First, the scale variance among different body parts are handled naturally as the relative distances within each part are in a similar range and each part-type is processed by a separate sub-network. Second, being a bottom-up method, HPRNet's inference speed is minimally affected by the number of persons in the input image. This is in contrast to the top-down methods such as ZoomNet, which significantly slows down with more person instances (65.7 ms for an image containing 1 person versus 668.2 ms for an image with 10 persons). Our method is based on the center-point based bottom-up object detection methods [40, 2, 19, 14]. These methods can easily be extended to the keypoint estimation task [40, 3].

We validated the effectiveness of our method through ablation experiments and comparisons with the state of the art (SOTA) on the COCO WholeBody dataset. Our method significantly outperforms all bottom-up methods. It also outperforms the SOTA top-down method ZoomNet in the detection of face and hand keypoints, while being significantly faster than ZoomNet.

Our major contribution in this work is the proposal of a one-stage, bottom-up method to close the performance gap between the bottom-up and top-down methods. In contrast to top-down methods, our method runs almost in constant time, independent from the number of persons in the input image.

6.2 Related Work

6.2.1 Human Body Pose Estimation

We can categorize the current approaches for multi-person pose estimation into two: bottom-up and top-down. In the bottom-up methods [139, 140, 141, 142, 143, 144, 145, 146, 136, 147, 148, 149, 150, 151], given an image, body keypoints detected first, without knowing the number or location of person instances or to which person instances these keypoints belong. Later, detected keypoints are grouped and assigned to person instances. Recently, center-based object detection methods [40] have been extended to perform human pose estimation [40, 3]. These methods represent keypoints with an offset value to the center of the person box and directly regresses them during training. In order to improve localization of keypoints, they also estimate the heatmap of each keypoint as in other bottom-up methods [135, 141, 144, 143]. At inference, using center offsets, they group and assign keypoints to person instances. Since bottom-up methods detect all people keypoints at once, they are fast.

Top-down methods [152, 153, 20, 154, 138, 155] first detect person instances in the input image. Commonly, they use an off-the-shelf object detector (e.g. Faster-RCNN [9]) to obtain person boxes. Next, top-down methods estimate a single person pose for each cropped person box. By cropping and resizing each person box, top-down methods have the advantage to zoom into the details of each person. Therefore,

top-down approaches are more capable of handling scale variance issues. As a result, state-of-the-art results are obtained by top-down methods and there is an accuracy gap between top-down and bottom-up approaches. However, since a pose estimation model is run for each person instance, top-down methods tend to be slow on average, that is, they get significantly slower with increasing number of persons in an image (Figure 6.6).

One may think that using human body pose estimation methods on a whole-body pose estimation dataset (i.e. COCO WholeBody) could be a solution for whole-body pose estimation. However, as it is stated in the COCO WholeBody dataset paper [23], due to the large scale variance between whole body parts, applying these methods directly results in suboptimal accuracies.

6.2.2 Whole-body Pose Estimation

Whole-body pose estimation requires accurate localization of keypoints on body, face, hand and feet. Detection of keypoints is well studied for each of these body parts independently, under face alignment [156, 157, 158, 159], facial landmark detection [160, 161], hand pose estimation [162, 163], hand tracking [164, 165] and feet keypoint detection [135] topics. However, there are not many works on the whole-body pose estimation mostly due to lack of a large-scale annotated dataset. Prior to the release of the COCO WholeBody dataset [23], OpenPose [135] attempted to detect the whole-body keypoints. For this purpose, OpenPose ensembles 5 separately trained models namely human body pose estimation, hand detection, face detection, hand pose estimation and face pose estimation. Due to these multiple models, training and inference of OpenPose are complex and costly. Our end-to-end trainable single network eliminates these drawbacks.

Hidalgo et al. presented a bottom-up method called SN [136]. Their model extends PAF [139] for whole-body pose estimation. Similar to PAF [139], they predict heatmaps for each keypoint and use part affinity maps for grouping. SN model is trained on a dataset that is sampled from different datasets. Both SN and our proposed model HPRNet are bottom-up methods. However, SN falls short of handling scale variations between whole-body parts whereas hierarchical point representation

of HPRNet overcomes this issue.

The first step towards having a whole-body pose estimation benchmark is the release of the COCO WholeBody dataset [23]. Jin et al. extended the existing COCO keypoints [137] dataset by further annotating face, hands and feet keypoints (Figure 6.1). They also proposed a strong, two-stage, top-down model to perform whole-body pose estimation on the COCO WholeBody dataset. Similar to top-down human pose estimation methods, Jin et al. [23] first obtain candidate person boxes in an image using FasterRCNN [9]. Next, using a single network called ZoomNet, detection of whole-body keypoints is performed on the person boxes. ZoomNet is composed of 4 sub CNN networks. First, FeatureNet processes input person boxes and extracts shared features at two scales. Next, using features from FeatureNet, BodyNet detects body and foot keypoints. BodyNet is also responsible for the prediction of the face and hand bounding box corner points to roughly extract face and hand areas. Later, cropped face and hand bounding boxes are fed to the FaceHead and HandHead networks to detect the keypoints on face and hands. They use HRNet-W32 [138] network for the BodyNet and HRNetV2p-W18 [166] network for the FaceHead and HandHead networks.

Even though bottom-up approaches are fast, they are not robust enough to handle the scale variance across the whole-body parts. However, we hypothesize that representing each keypoint with an offset value to a carefully selected location can handle the scale variance. Based on this, we extend the center-based human pose estimation method [40] to perform whole-body pose estimation by introducing hierarchical regression of keypoints. We also show that hierarchical regression of keypoints for small scale whole-body parts (i.e. face and hand) is more effective than cropping and zooming into them.

6.3 Model

HPRNet is a one-stage end-to-end trainable network that learns regressing the whole-body keypoints. In HPRNet, the input image first passes through a backbone network and output of the backbone is fed to 8 separate branches, namely; *Person Center*

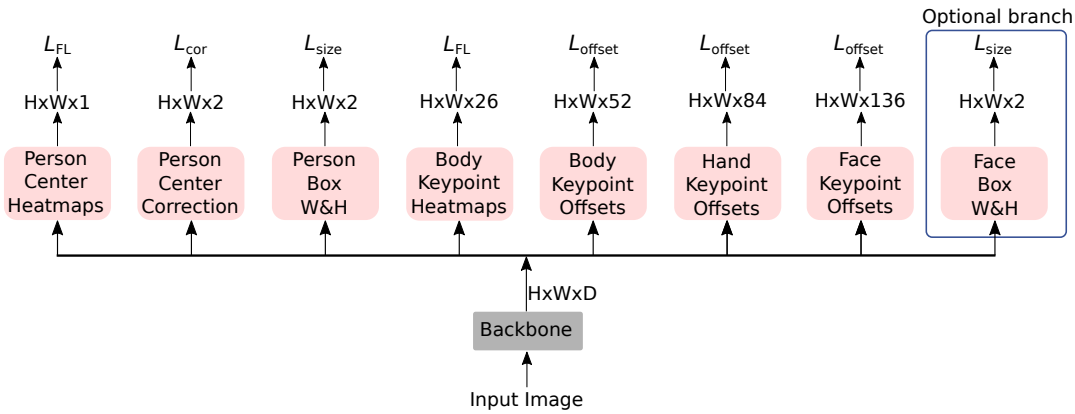


Figure 6.2: Network architecture of the proposed HPRNet for whole-body keypoint detection. Image is taken from our IMAVIS journal paper [25].

Heatmap, *Person Center Correction*, *Person W & H*, *Body Keypoint Offsets*, *Body Keypoint Heatmaps*, *Hand Keypoint Offsets*, *Face Keypoint Offsets* and *Face Box W & H*. We show the network architecture of HPRNet in Figure 6.2.

6.3.1 Hierarchical Regression of Whole-Body Keypoints

In HPRNet, we build a hierarchical regression mechanism, where we define each of the whole-body keypoints with a relative location (i.e. offset) to a specific point on the person box.

We represent each of the (standard) 17 keypoints on the body with an offset to the center of the person bounding box. Unlike the body; face, hand and foot are small parts. Based on this, we define each of this parts with a relative location to their part center as follows; (i) each of 68 face keypoints is defined with an offset to the center of face bounding box, (ii) each of 21 left hand keypoints is defined with an offset to left hand bounding box center, (iii) each of 21 right hand keypoints is defined with an offset to right hand bounding box center, (iv) each of 3 left foot keypoints is defined with an offset to left foot bounding box center, (v) each of 3 right foot keypoints is defined with an offset to right foot bounding box center. Face, hand and foot bounding boxes are automatically extracted from the groundtruth keypoint annotations.

We treat the bounding box center of the face, left hand, right hand, left foot and

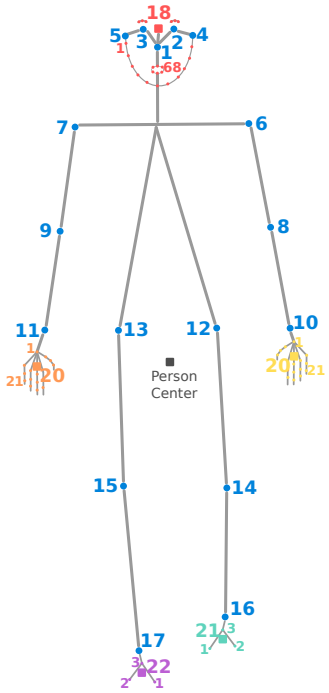


Figure 6.3: All regressed keypoints in HPRNet. Blue keypoints are body keypoints as defined in COCO keypoints and COCO WholeBody datasets. Colored square points correspond to the face (18), left hand (19), right hand (20), left foot (21) and right foot (22) box centers. Blue keypoints (1-17) and colored square points are defined with an offset to the center of the person instance. For simplicity, face and hand keypoints are sparsely illustrated. Image is taken from our IMAVIS journal paper [25].

right foot as a body part keypoint and define each of them with an offset value to the person box center (Figure 6.3). We illustrate the hierarchical regression of whole-body keypoints in Figure 6.4b.

At inference, after detecting all the keypoints in the input image, we group and assign them to person instances. To achieve this, we get predicted person centers from the output of *Person Center Heatmap* branch as in CenterNet [40]. Next, we obtain the offset values on the predicted person center locations of *Body Keypoint Offsets* branch output. After that, we add these offsets to person centers to obtain the *regressed* body keypoint locations. At the same time, we extract the *detected* body keypoints from the outputted heatmap of the *Body Keypoint Heatmaps* branch. At the last step, we match the *detected* and *regressed* keypoints based on L2 distance and only take the keypoints inside the predicted person bounding box.

Next, we group face and hand keypoints (and foot keypoints as well, if we are using the Hierarchical Model-I (Figure 6.4b). We obtain predicted part centers from the output of *Body Keypoint Heatmaps* branch. Then, we collect the offset values on the corresponding predicted part center locations of *Hand Keypoint Offsets* and *Face Keypoint Offsets* branch output. Finally, we add these offsets to the part centers to obtain the face and hand keypoints.

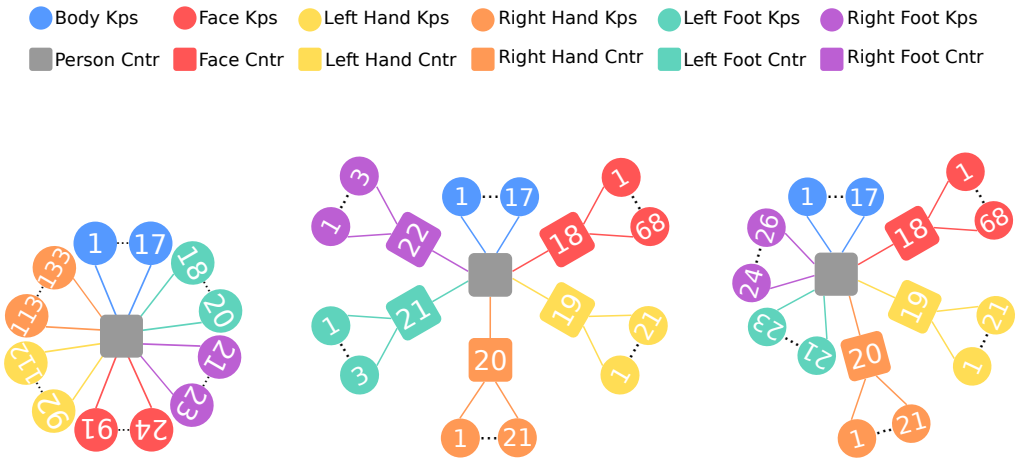
6.3.2 Regression of Foot Keypoints

Ideally, each labeled foot part in the COCO WholeBody dataset should have 3 keypoint annotations. However, more than 20% of annotated feet have missing annotations (i.e. they have one or two keypoints annotations, instead of three). These missing annotations present a challenge to HPRNet, since we automatically extract foot centers from the annotated extreme points. In the case of the missing foot keypoints, the obtained foot center point is not reliable. To deal with this issue, we treat the foot keypoints as body keypoints as shown in Figure 6.4c, and represent them by their offsets to the center of the person bounding box.

6.3.3 Network Architecture

Given an input image of size $4H \times 4W \times 3$, the backbone network outputs a feature map of size $H \times W \times D$. The backbone's output is fed to the following subsequent branches. Each branch has one convolutional layer with 3×3 filters followed by a ReLU layer and another convolutional layer with 1×1 filters.

- *Person Center Heatmap* branch outputs $H \times W$ sized tensor for person center point predictions.
- *Person Center Correction* branch predicts $H \times W \times 2$ sized tensor for the local offsets of center locations across the spatial axes. These offsets help to recover the lost precision of the center points due to down-sampling operations through the network.
- *Person Box W & H* branch outputs $H \times W \times 2$ sized tensor of widths and



(a) Baseline Model (b) Hierarchical Model-I (c) Hierarchical Model-II

Figure 6.4: Hierarchical representations of whole-body keypoints. (a) Each of 133 whole-body keypoints is defined with an offset to the person box center. (b) Body keypoints and other part centers (i.e. foot, face and hand) are defined with offsets to the person box center. Foot, face and hand keypoints are defined with offsets according to their corresponding part centers. (c) Considering the sparsity of foot keypoint annotations, we define them with their offset values to the person box center. In both Hierarchical Model-I and Hierarchical Model-II, body keypoints are defined with offset values to the person box center. Each face and hand keypoint is defined with an offset to face and hand bounding box centers, respectively. Image is taken from our IMAVIS journal paper [25].

heights for each person instance center.

- *Body Keypoint Offsets* branch predicts offset values of 26 keypoints (17 body keypoints + 6 foot keypoints + Center of Face Box + Center of Left Hand Box + Center of Right Hand Box) to the person box center across the x and y axes.
- *Body Keypoint Heatmaps* branch outputs $H \times W \times 26$ sized heatmap tensor for the 26 keypoints.
- *Hand Keypoint Offsets* branch outputs $H \times W \times 84$ sized tensor of offset values between 21 left hand keypoints and left hand box center; and the offset values between the 21 right hand keypoints and right hand box center across the spatial axes.

- *Face Keypoint Offsets* branch outputs $H \times W \times 136$ sized tensor of offset values between 68 face keypoints and face box center across the spatial axes.

- *Face Box W & H* branch outputs $H \times W \times 2$ sized tensor of widths and heights for each face. It is an optional branch.

6.3.4 Objective Functions

For the optimization of the *Person Center Heatmap (PCH)* and *Body Keypoint Heatmap (BKH)* branches, we use the modified focal loss [8] as done in previous work [14, 15, 40, 2]. Modified focal loss (FL) is presented in Equation 6.1. $I \in R^{4W \times 4H \times 3}$ is our input image. In HPRNet,due to downsampling operations, the spatial output size of each branch is 4 times smaller resulting in $W \times H$. Therefore, $Y \in [0,1]^{W \times H \times C}$ is the ground truth heatmap for person centers and keypoints. C corresponds to class number and keypoint types. For instance, in the *Person Center Heatmap* branch, we have only *person* class, thus $C = 1$. $\hat{Y} \in [0,1]^{W \times H \times C}$ is the predicted heatmap output by the branches where $\hat{Y}_{x,y,c} = 1$ indicates presence of a person center or keypoint at location (x, y) for class c. In the following all equations, N is the total number of ground truth person centers or keypoints in image I. α and β are focal loss parameters and set as $\alpha = 2$ and $\beta = 4$ as in CornerNet [14].

$$L_{\mathrm{FL}} = \frac{-1}{N} \sum_{xyc} \begin{cases} \left(1 - \hat{Y}_{xyc}\right)^{\alpha} \log\left(\hat{Y}_{xyc}\right) & \text{if } Y_{xyc} = 1 \\ \left(1 - Y_{xyc}\right)^{\beta} \left(\hat{Y}_{xyc}\right)^{\alpha} & \\ \log\left(1 - \hat{Y}_{xyc}\right) & \text{otherwise} \end{cases} \tag{6.1}$$

To compensate for the discretization error of the person center points due to downsampling operations through the network, we optimize the *Person Center Correction* according to the following L1 loss similar to the bottom-up object detectors [14, 15, 40, 2]. $\hat{T} \in \mathcal{R}^{W \times H \times 2}$ is the predicted local offset by the network to recover the lost precision of person center points. $p \in \mathcal{R}^2$ is a ground truth keypoint and $\tilde{p} = \left\lfloor \frac{p}{4} \right\rfloor$ is

the corresponding ground keypoint location at low-resolution.

$$L_{\text{cor}} = \frac{1}{N} \sum_{p} \left| \hat{T}_{\tilde{p}} - \left(\frac{p}{4} - \tilde{p} \right) \right| \tag{6.2}$$

We optimize the *Body Keypoint Offset*, *Hand Keypoint Offset* and *Face Keypoint Offset* branches using the L1 loss. The generic formulation of keypoint regression is presented in Equation 6.3. In the equation, $\hat{O} \in \mathbb{R}^{H \times W \times k \times 2}$ is the regression output of keypoints k for a specific whole-body part (i.e. body, face, hand), and B_{part} is the ground truth center of that part's bounding box.

$$L_{\text{offset}} = \sum_{k} \left| \hat{O}_{[k]} - B_{\text{part}} \right| \tag{6.3}$$

Finally, for the *Person Box H & W* and *Face Box H & W* branches, we use L1 loss and scale it by 0.1 as in CenterNet [40]. In the Equation 6.4, $s_n = (w, h)$ is the width and height values of the each object (or face) n and $\hat{S} \in \mathcal{R}^{W \times H \times 2}$ is the predicted width and height values.

$$L_{\text{size}} = \frac{1}{N} \sum_{n=1}^{N} \left| \hat{S}_{p_n} - s_n \right| \tag{6.4}$$

We obtain the overall loss by summing the losses from all branches as follows:

$$L_{\text{overall}} = L_{\text{FL}}^{PCH} + L_{\text{FL}}^{BKH} + L_{\text{cor}} + L_{\text{offset}}^{body} + L_{\text{offset}}^{face} + L_{\text{offset}}^{hand} + 0.1 L_{\text{size}}^{person} + 0.1 L_{\text{size}}^{face}$$

6.4 Experiments

This section describes the experiments we conducted to show the effectiveness of our proposed method. First, we present ablation experiments to compare hierarchical models I and II shown in Figure 6.4. Next, we compare our method with our baseline CenterNet [40] (Figure 6.4a). Finally, we provide performance comparison with the state of the art and a run-time analysis.

6.4.1 Implementation Details

We use Deep Layer Aggregation (DLA) [99] backbone for ablation and baseline comparison experiments, and Hourglass-104 [14] as our backbone network for state of the art comparison. For all experiments, during training we resize the images to 512×512 pixels. At inference we use images with their original sizes without applying any scaling. We train all the models with a batch size of 32 for 140 epochs using the Adam optimizer [85]. We set the initial learning rate to 1.25×10^{-4} and divided it by 10 at epochs 90 and 120. We trained all of the models on 4 Tesla V100 GPUs, and tested using a single GTX 1080 TI GPU. We used PyTorch [117] to implement our models. All of our experiments are conducted on the COCO WholeBody Dataset [23] and results are presented in keypoint AP (AP^{kp}) and keypoint recall AR (AR^{kp}) metrics without any test time augmentation. All results are obtained on the COCO WholeBody validation set.

6.4.2 Hierarchical Model-I vs Hierarchical Model-II

In Table 6.1, we compare Hierarchical Model-I and Hierarchical Model-II (see Figure 6.4). As it can be seen from the table, regressing foot keypoints as a part of the body keypoints, improves the foot AP^{kp} significantly by 15.6 points (33.5 vs. 49.1). Moreover, hand and whole-body AP^{kp}s also improved about 3 points in this setup. Based on these results, for the rest of the experiments we use the Hierarchical Model-II.

Table 6.1: Comparison of Hierarchical Model-I (HM-I) and Hierarchical Model-II (HM-II) as in Figure 6.4. Training foot keypoints with offset values to the person box center outperforms the model when trained with offset values to the foot part centers. Both models are trained with DLA backbone.

Method	body		foot		face		hand		whole-body		all-mean	
	AP^{kp}	AR^{kp}	AP^{kp}	AR^{kp}	AP^{kp}	AR^{kp}	AP^{kp}	AR^{kp}	AP^{kp}	AR^{kp}	AP^{kp}	AR^{kp}
HM-I	**55.5**	**63.4**	33.5	55.3	**74.6**	83.5	44.1	57.8	28.0	40.5	47.1	60.1
HM-II	55.2	63.1	**49.1**	**60.9**	**74.6**	**83.7**	**47.0**	**60.8**	**31.5**	**44.6**	**51.5**	**62.6**

6.4.3 Comparison with Baseline

To obtain the baseline results, we regress all the 133 keypoints to the person instance box center during training as in CenterNet [40] (see Figure 6.4b). In Table 6.2, we compare HPRNet with the baseline model in terms of accuracy and recall. Our proposed HPRNet significantly outperforms the baseline results for all AP^{kp} and AR^{kp} metrics except the whole-body AP^{kp}.

Table 6.2: Comparing HPRNet with the baseline model. To obtain the baseline results, we regress all the 133 keypoints to the person instance box center during training as in Figure 6.4b. Both models are trained with DLA backbone.

Method	body		foot		face		hand		whole-body		all-mean	
	AP^{kp}	AR^{kp}	AP^{kp}	AR^{kp}	AP^{kp}	AR^{kp}	AP^{kp}	AR^{kp}	AP^{kp}	AR^{kp}	AP^{kp}	AR^{kp}
Baseline	46.7	55.5	33.6	48.9	52.0	60.2	26.4	39.1	**33.3**	43.4	38.4	49.4
HPRNet	**55.2**	**63.1**	**49.1**	**60.9**	**74.6**	**83.7**	**47.0**	**60.8**	31.5	**44.6**	**51.5**	**62.6**

6.4.4 Comparison with the State-of-the-art

Table 6.3 presents the performance of our models and several established keypoint estimation models on the COCO WholeBody validation set. We also present average run times if available. **HPRNet performs best among the bottom-up methods.** Other bottom-up methods especially fail to accurately localize foot keypoints. The performance gap between the second best performing bottom-up method and our method is 40.9 AP^{kp} points on the foot keypoint detection. Similarly, our method outperforms other bottom-up methods for the body, face, hand and whole-body keypoint detection by a large margin. Among the top-down methods, ZoomNet outperforms the well known OpenPose [135] and HRNet [138]. Here, ZoomNet is a two-stage framework where at the first stage person candidates are extracted with FasterRCNN and at the second stage ZoomNet is run on these candidate boxes. HRNet can be seen as a one-stage counterpart of ZoomNet and finally OpenPose is a multi-model which requires separate training for each whole-body part. HPRNet obtains state-of-the-art results on face and hand keypoint detection. Our model with Hourglass-104 back-

Table 6.3: Comparison with the state-of-the-art on COCO WholeBody validation set. The methods are divided into two groups: top-down and bottom-up. The best results and run times are boldfaced separately for each group. HPRNet performs best among the bottom-up methods. HPRNet also obtains state-of-the-art results on face and hand keypoint detection outperforming ZoomNet. Among all methods, HPRNet with DLA backbone is the fastest one. * indicates that run time linearly increases as the number of people in an image increases. HG is Hourglass-104. R. time is Running time.

Method	body		foot		face		hand		whole-body		all-mean		R. time (ms)
	AP^{kp}	AR^{kp}	AP^{kp}	AR^{kp}	AP^{kp}	AR^{kp}	AP^{kp}	AR^{kp}	AP^{kp}	AR^{kp}	AP^{kp}	AR^{kp}	
Top-down methods:													
OpenPose [135]	56.3	61.2	53.2	64.5	48.2	62.6	19.8	34.2	33.8	44.9	42.3	53.5	**45**
HRNet* [138]	65.9	70.9	31.4	42.4	52.3	58.2	30.0	36.3	43.2	52.0	44.6	52.0	-
ZoomNet* [23]	**74.3**	**80.2**	**79.8**	**86.9**	**62.3**	**70.1**	**40.1**	**49.8**	**54.1**	**65.8**	**62.1**	**70.6**	175
Bottom-up methods:													
PAF [139]	26.6	32.8	10.0	25.7	30.9	36.2	13.3	32.1	14.1	18.5	19.0	29.1	100
SN [136]	28.0	33.6	12.1	27.7	38.2	44.0	13.8	33.6	16.1	20.9	21.6	32.0	216
AE [141]	40.5	46.4	7.7	16.0	47.7	58.0	34.1	43.5	27.4	35.0	31.5	39.8	-
Ours (HPRNet-DLA)	55.2	63.1	49.1	60.9	74.6	83.7	47.0	60.8	31.5	44.6	51.5	62.6	**37**
Ours (HPRNet-HG)	**59.4**	**68.3**	**53.0**	**65.4**	**75.4**	**86.8**	**50.4**	**64.2**	**34.8**	**49.2**	**54.6**	**66.8**	101

bone outperforms ZoomNet on the detection of face keypoints by 13.1 AP^{kp} points and hand keypoints by 10.3 AP^{kp} points. These successful results on the face and hand keypoint detection, further shows the effectiveness of our proposed bottom-up hierarchical approach over the ZoomNet's zoom-in mechanism. However, for the detection of the body and whole-body keypoints ZoomNet performs best among all methods. **Among all methods, our HPRNet with the DLA backbone is the fastest one (37 ms) with constant run time**. In Figure 6.5, we show sample qualitative results for our approach.

6.4.5 Runtime Analysis

Average run time of ZoomNet (including Faster RCNN for person detector) on a single image is 174.7 ms. Similarly, the average run time of HPRNet with DLA and Hourglass-104 backbones is 37 ms (26 ms for feedforward and 11 ms for keypoint grouping and assignment) and 101 ms (90 ms for feedforward and 11 ms for keypoint

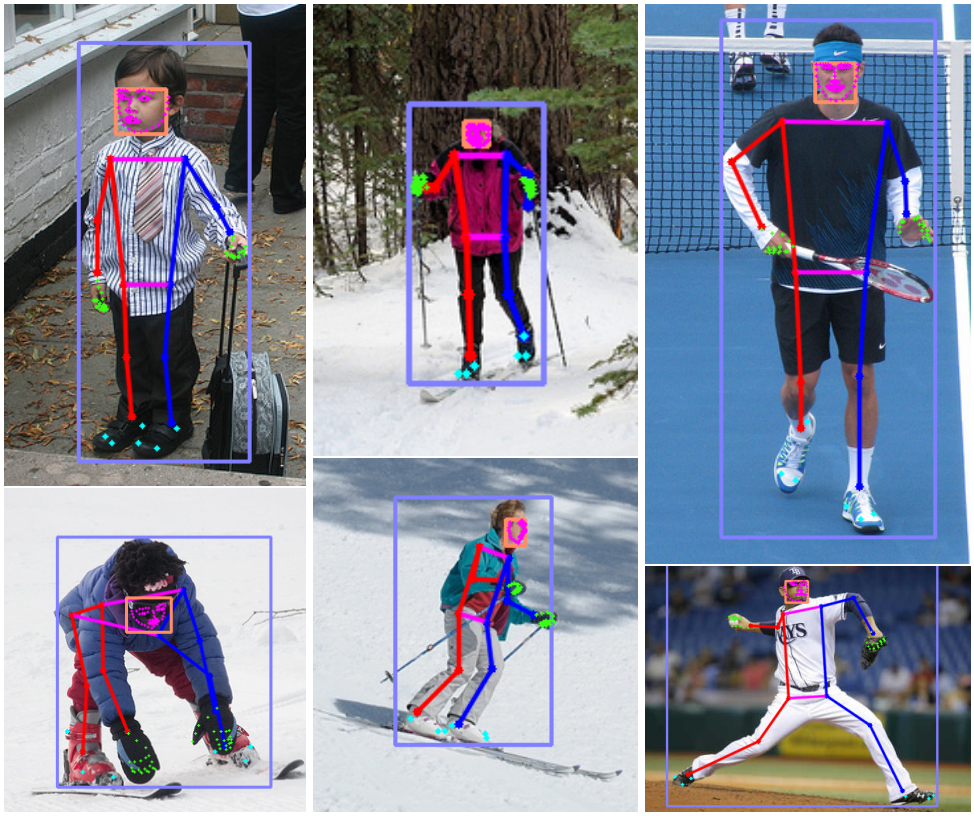


Figure 6.5: Sample whole-body keypoint detection results of HPRNet. We show correctly detected people, and their whole-body poses. Detection box is marked with a purple bounding box, and body pose estimation is shown with blue color for the left parts, red color for the right parts. For clarity, we mark the detected keypoints on face, hand and foot with magenta, green and cyan colors, respectively. Detected faces are marked with an orange bounding box. Image is taken from our IMAVIS journal paper [25].

grouping and assignment). HPRNet is significantly faster than ZoomNet. Moreover, as a top-down method, run time of ZoomNet increases as the number of people on an image increases. We compare the run time of our models and ZoomNet in Figure 6.6.

6.4.6 Face Detection from Keypoints

In this section, we studied the face detection task and compared HPRNet and ZoomNet. We first extracted face boxes using extreme face keypoints and calculated AP scores as in object detection. Our model outperformed ZoomNet in face detection

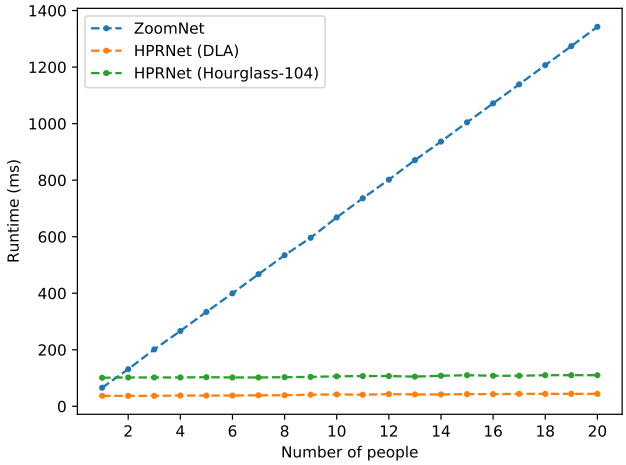


Figure 6.6: Runtime analysis of ZoomNet and our models with respect to number of people in an image. As the number of people in an image increases, the runtime of ZoomNet linearly increases. Whereas, our models almost have constant run time. Image is taken from our IMAVIS journal paper [25].

(46.2 AP vs 37.7 AP). Later, using an additional branch for face detection we train another model (see Figure 6.2). Our model with an extra face detection branch further improved the performance of HPRNet for face detection achieving 55.8 AP and 56.4 AP with DLA and Hourglass-104 backbones respectively. Results are presented in Table 6.4.

Table 6.4: Face detection results. The first group of results are obtained from extreme face keypoints for both ZoomNet and HPRNet. The HPRNet results in the second group are obtained with an extra face detection branch. HG is Hourglass-104.

Method	AP	AP_{50}	AP_{75}	AP_M	AP_L
When face boxes are extracted from extreme face keypoints:					
ZoomNet	37.7	64.5	41.1	25.8	44.9
HPRNet (DLA)	46.2	70.6	54.8	32.2	53.8
HPRNet (HG)	46.1	70.9	53.6	33.4	53.1
Our model with an extra face detection branch					
HPRNet (DLA)	55.8	82.3	66.2	40.0	**63.6**
HPRNet (HG)	**56.4**	**82.4**	**67.1**	**43.4**	63.3

6.5 Conclusion

In this work, we introduced HPRNet as a bottom-up, one-stage method for whole-body keypoint detection. HPRNet handles scale variance among whole-body parts by hierarchically regressing whole-body keypoints. We evaluated the effectiveness of our method through baseline comparison and ablation experiments on hierarchical structure of whole-body keypoints. Our method achieves state-of-the-art results in the detection of face and hand keypoints on the COCO WholeBody dataset; it also outperforms all other bottom-up methods in the detection of all whole-body parts. We conducted a run time analysis between HPRNet and ZoomNet and showed that in contrast to ZoomNet, HPRNet runs in constant time, independent of the number of persons in an image.

CHAPTER 7

CONCLUSION

This chapter is adopted from our ECCV'20 paper [2] and its extension [3].

In this thesis, we presented HoughNet, a new, one-stage, anchor-free, voting-based, bottom-up object detection method. HoughNet determines the presence of an object at a specific location by the sum of the votes cast on that location. Voting module of HoughNet is able to use both short and long-range evidence through its log-polar vote field. Thanks to this ability, HoughNet generalizes and enhances current object detection methodology, which typically relies on only local (short-range) evidence. We show that HoughNet performs on-par with the state-of-the-art bottom-up object detectors, and obtains comparable results with one-stage and two-stage methods. To further validate our proposal, we used the voting module of HoughNet in video object detection, instance segmentation, 3D object detection, 2D human pose estimation, 2D whole-body human pose estimation, face detection and "labels to photo" image generation tasks, and showed that our voting module consistently improves the baseline performances.

For video object detection we extended voting in spatial domain to the temporal domain and developed a new method, which takes the difference of features from two frames, and applies spatial and temporal voting using our "temporal voting module" to detect objects.

We also developed a "scalable" variant of HoughNet where the number of voting operations does not depend on the number of object classes. We showed that scalable Hough voting dramatically improves inference speed with a slight drop in accuracy.

We also conducted analysis studies on HoughNet. First, we examined the relations

between vote-giver and vote-getter classes of HoughNet. Secondly, we evaluated HoughNet's performance using localisation, recall, precision metrics and compared the source of errors with the baseline model.

In order to show the effectiveness of our proposal on whole-body human pose estimation task, we developed a bottom-up, one-stage method called HPRNet. HPRNet handles scale variance among whole-body parts by hierarchically regressing whole-body keypoints. We showed that HPRNet is both fast and accurate. HPRNet runs in constant time, independent of the number of persons in an image, and also performs best among bottom-up methods and achieves state-of-the-art results on face and hand keypoint detection.

We also developed a one-stage, anchor-free object detector, PPDet, which integrates short-range interactions through voting. PPDet employs the new labeling strategy during training and a new inference method based on pooling predictions. This allows PPDet to reduce the contributions of non-discriminatory features during training. We showed that PPDet performs on par with the state-of-the-art and especially very effective for small object detection.

7.1 Limitations and Future Work

The current design of Hough voting module has limitations. First, in the current design, one could only apply voting at the last layer to detect object classes. Applying voting at the early layers on intermediate feature tensors are computationally very costly. With a more efficient design voting could be applied at early layers and also performance could be improved further.

Second, the current design is not compatible with object detector networks with FPN [88]. One has to search for the best log-polar vote-field for each FPN layer manually. Moreover, applying voting at each output of FPN layer is computationally costly. In that sense, a different voting mechanism could be developed particularly for network architectures with FPN.

Third, in the current design we apply voting at each spatial location. However, not all

the locations contain useful short or long-range evidence for the detection of objects. Applying voting only on the locations with strong short and long-range evidence will improve the run time of HoughNet.

HoughNet exposes the latent patterns between objects and its voters. In this way, our voting module is able to get votes from non-labeled objects in a dataset. Based on this observation, in future we aim to extend our work for the discovery of unseen object classes.

Both HoughNet and transformer based object detectors integrates short and long-range interactions. We aim to compare HoughNet with transformer based object detectors, and possibly extend the log-polar voting module to be used within transformers.

In the current version of HPRNet, we rely on manually assigned hierarchy labels of whole-body keypoints. Manually assigned hierarchy labels force the network to regress each whole-body keypoint according to a hand crafted pre-defined value. Relaxing this regressing strategy will let each whole-body keypoint to leanr its own hierarchy level during training. Adopting the network to relax regressing of whole-body keypoints could improve the performance further.

In the current design of PPDet, we manually define the positive area of a ground-truth box. Dynamic selection of this positive area could improve the performance of PPDet further. To this end, a new branch could be integrated to the network to predict the positive area of the corresponding object's ground-truth area.

REFERENCES

[1] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3213–3223, 2016.

[2] N. Samet, S. Hicsonmez, and E. Akbas, "HoughNet: Integrating near and long-range evidence for bottom-up object detection," in *European Conference on Computer Vision*, 2020, in press.

[3] N. Samet, S. Hicsonmez, and E. Akbas, "Houghnet: Integrating near and long-range evidence for visual detection," 2021.

[4] R. B. Girshick, P. F. Felzenszwalb, and D. McAllester, "Discriminatively trained deformable part models, release 5." http://people.cs.uchicago.edu/ rbg/latent-release5/.

[5] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1627–1645, 2009.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

[7] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *European Conference on Computer Vision*, pp. 740–755, Springer, 2014.

[8] T. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *IEEE International Conference on Computer Vision*, 2017.

[9] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems*, pp. 91–99, 2015.

[10] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European Conference on Computer Vision*, 2016.

[11] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[12] B. Leibe, A. Leonardis, and B. Schiele, "Robust object detection with interleaved categorization and segmentation," *International Journal of Computer Vision*, vol. 77, pp. 259–289, May 2008.

[13] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part based models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2010.

[14] H. Law and J. Deng, "Cornernet: Detecting objects as paired keypoints," in *European Conference on Computer Vision*, pp. 734–750, 2018.

[15] X. Zhou, J. Zhuo, and P. Krähenbühl, "Bottom-up object detection by grouping extreme and center points," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[16] P. V. C. Hough, "Machine Analysis of Bubble Chamber Pictures," vol. C590914, pp. 554–558, 1959.

[17] D. H. Ballard *et al.*, "Generalizing the hough transform to detect arbitrary shapes," *Pattern Recognition*, 1981.

[18] M. Land and B. Tatler, *Looking and acting: vision and eye movements in natural behaviour*. Oxford University Press, 2009.

[19] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian, "Centernet: Keypoint triplets for object detection," in *IEEE International Conference on Computer Vision*, 2019.

[20] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask r-cnn," *IEEE International Conference on Computer Vision*, pp. 2980–2988, 2017.

[21] K. Oksuz, B. Cam, E. Akbas, and S. Kalkan, "Localization recall precision (LRP): A new performance metric for object detection," in *European Conference on Computer Vision*, 2018.

[22] K. Oksuz, B. C. Cam, S. Kalkan, and E. Akbas, "One metric to measure them all: Localisation recall precision (LRP) for evaluating visual detection tasks," *under review at TPAMI*, 2020.

[23] S. Jin, L. Xu, J. Xu, C. Wang, W. Liu, C. Qian, W. Ouyang, and P. Luo, "Whole-body human pose estimation in the wild," in *European Conference on Computer Vision*, 2020.

[24] N. Samet, S. Hicsonmez, and E. Akbas, "Reducing Label Noise in Anchor-Free Object Detection," in *British Machine Vision Conference (BMVC)*, 2020.

[25] N. Samet and E. Akbas, "Hprnet: Hierarchical point regression for whole-body human pose estimation," *Image and Vision Computing*, vol. 115, p. 104285, 2021.

[26] R. Okada, "Discriminative generalized hough transform for object detection," in *IEEE International Conference on Computer Vision*, 2009.

[27] J. Gall and V. Lempitsky, "Class-specific hough forests for object detection," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[28] N. Razavi, J. Gall, P. Kohli, and L. Van Gool, "Latent hough transform for object detection," in *European Conference on Computer Vision*, 2012.

[29] S. Maji and J. Malik, "Object detection using a max-margin hough transform," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[30] O. Barinova, V. Lempitsky, and P. Kholi, "On detection of multiple object instances using hough transforms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 9, pp. 1773–1784, 2012.

[31] C. R. Qi, O. Litany, K. He, and L. J. Guibas, "Deep hough voting for 3d object detection in point clouds," in *IEEE International Conference on Computer Vision*, 2019.

[32] A. Sheshkus, A. Ingacheva, V. Arlazarov, and D. Nikolaev, "Houghnet: Neural network architecture for vanishing points detection," in *IEEE International Conference on Document Analysis and Recognition (ICDAR)*, pp. 844–849, 2019.

[33] E. Gabriel, M. Schleiss, H. Schramm, and C. Meyer, "Analysis of the discriminative generalized hough transform as a proposal generator for a deep network in automatic pedestrian and car detection," *Journal of Electronic Imaging*, vol. 27, no. 5, p. 051228, 2018.

[34] I. Lifshitz, E. Fetaya, and S. Ullman, "Human pose estimation using deep consensus voting," in *European Conference on Computer Vision*, 2016.

[35] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[36] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg, "Dssd: Deconvolutional single shot detector," *arXiv preprint arXiv:1701.06659*, 2017.

[37] J. Ren, X. Chen, J. Liu, W. Sun, J. Pang, Q. Yan, Y.-W. Tai, and L. Xu, "Accurate single stage detector using recurrent rolling convolution," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[38] J. Wang, K. Chen, S. Yang, C. C. Loy, and D. Lin, "Region proposal by guided anchoring," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[39] Z. Tian, C. Shen, H. Chen, and T. He, "Fcos: Fully convolutional one-stage object detection," in *IEEE International Conference on Computer Vision*, 2019.

[40] X. Zhou, D. Wang, and P. Krähenbühl, "Objects as points," in *arXiv preprint arXiv:1904.07850*, 2019.

[41] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition

using shape contexts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, pp. 509–522, April 2002.

[42] E. Akbas and M. P. Eckstein, "Object detection through search with a foveated visual system," *PLoS computational biology*, vol. 13, no. 10, p. e1005743, 2017.

[43] V. J. Traver and A. Bernardino, "A review of log-polar imaging for visual perception in robotics," *Robotics and Autonomous Systems*, vol. 58, no. 4, pp. 378–398, 2010.

[44] S. Zagoruyko, A. Lerer, T.-Y. Lin, P. O. Pinheiro, S. Gross, S. Chintala, and P. Dollár, "A multipath network for object detection," *arXiv preprint arXiv:1604.02135*, 2016.

[45] X. Zeng, W. Ouyang, B. Yang, J. Yan, and X. Wang, "Gated bi-directional cnn for object detection," in *European Conference on Computer Vision*, pp. 354–369, Springer, 2016.

[46] X. Zeng, W. Ouyang, J. Yan, H. Li, T. Xiao, K. Wang, Y. Liu, Y. Zhou, B. Yang, Z. Wang, *et al.*, "Crafting gbd-net for object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 9, pp. 2109–2123, 2017.

[47] W. Ouyang, K. Wang, X. Zhu, and X. Wang, "Learning chained deep features and classifiers for cascade in object detection," *arXiv preprint arXiv:1702.07054*, 2017.

[48] S. Gidaris and N. Komodakis, "Object detection via a multi-region and semantic segmentation-aware cnn model," in *IEEE International Conference on Computer Vision*, pp. 1134–1142, 2015.

[49] Y. Zhu, C. Zhao, J. Wang, X. Zhao, Y. Wu, and H. Lu, "Couplenet: Coupling global structure with local parts for object detection," in *IEEE International Conference on Computer Vision*, pp. 4126–4134, 2017.

[50] C. Chen, M.-Y. Liu, O. Tuzel, and J. Xiao, "R-cnn for small object detection," in *Asian Conference on Computer Vision*, pp. 214–230, Springer, 2016.

[51] Y. Kim, T. Kim, B.-N. Kang, J. Kim, and D. Kim, "Ban: Focusing on boundary context for object detection," in *Asian Conference on Computer Vision*, pp. 555–570, Springer, 2018.

[52] S. K. Divvala, D. Hoiem, J. H. Hays, A. A. Efros, and M. Hebert, "An empirical study of context in object detection," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1271–1278, IEEE, 2009.

[53] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[54] Z. Li, Y. Chen, G. Yu, and Y. Deng, "R-FCN++: Towards accurate region-based fully convolutional networks for object detection," in *AAAI Conference on Artificial Intelligence*, 2018.

[55] J. Li, Y. Wei, X. Liang, J. Dong, T. Xu, J. Feng, and S. Yan, "Attentive contexts for object detection," *CoRR*, vol. abs/1603.07415, 2016.

[56] S. Bell, C. Lawrence Zitnick, K. Bala, and R. Girshick, "Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2874–2883, 2016.

[57] C. Desai, D. Ramanan, and C. C. Fowlkes, "Discriminative models for multi-class object layout," *International Journal of Computer Vision*, vol. 95, no. 1, pp. 1–12, 2011.

[58] Q. Chen, Z. Song, J. Dong, Z. Huang, Y. Hua, and S. Yan, "Contextualizing object detection and classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 1, pp. 13–27, 2014.

[59] X. Chen and A. Gupta, "Spatial memory for context reasoning in object detection," in *IEEE International Conference on Computer Vision*, pp. 4086–4096, 2017.

[60] H. Hu, J. Gu, Z. Zhang, J. Dai, and Y. Wei, "Relation networks for object detection," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3588–3597, 2018.

[61] N. Arbel, T. Avraham, and M. Lindenbaum, "Inner-scene similarities as a contextual cue for object detection," *arXiv preprint arXiv:1707.04406*, 2017.

[62] Z. Chen, S. Huang, and D. Tao, "Context refinement for object detection," in *European Conference on Computer Vision*, pp. 71–86, 2018.

[63] S. Gupta, B. Hariharan, and J. Malik, "Exploring person context and local scene context for object detection," *arXiv preprint arXiv:1511.08177*, 2015.

[64] Y. Liu, R. Wang, S. Shan, and X. Chen, "Structure inference net: Object detection using scene-level context and instance-level relationships," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6985–6994, 2018.

[65] W. Chu and D. Cai, "Deep feature based contextual model for object detection," *Neurocomputing*, vol. 275, pp. 1035–1042, 2018.

[66] N. Dvornik, J. Mairal, and C. Schmid, "On the importance of visual context for data augmentation in scene understanding," *arXiv preprint arXiv:1809.02492*, 2018.

[67] R. Mottaghi, X. Chen, X. Liu, N.-G. Cho, S.-W. Lee, S. Fidler, R. Urtasun, and A. Yuille, "The role of context for object detection and semantic segmentation in the wild," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[68] X. Qiao, Q. Zheng, Y. Cao, and R. W. Lau, "Tell me where i am: Object-level scene context prediction," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2633–2641, 2019.

[69] W. Luo, Y. Li, R. Urtasun, and R. Zemel, "Understanding the effective receptive field in deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, pp. 4898–4906, 2016.

[70] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," *arXiv preprint arXiv:1710.09829*, 2017.

[71] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7794–7803, 2018.

[72] N. Samet, S. Hicsonmez, and E. Akbas, "Houghnet: Integrating near and long-range evidence for bottom-up object detection," in *ECCV*, pp. 406–423, Springer, 2020.

[73] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," *European Conference on Computer Vision*, 2020.

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

[75] C. Chi, F. Wei, and H. Hu, "Relationnet++: Bridging visual representations for object detection via transformer decoder," *Advances in Neural Information Processing Systems*, 2020.

[76] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, pp. 5998–6008, 2017.

[77] H. Zhao, J. Jia, and V. Koltun, "Exploring self-attention for image recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10076–10085, 2020.

[78] X. Zhu, Y. Wang, J. Dai, L. Yuan, and Y. Wei, "Flow-guided feature aggregation for video object detection," in *ICCV*, 2017.

[79] H. Deng, Y. Hua, T. Song, Z. Zhang, Z. Xue, R. Ma, N. Robertson, and H. Guan, "Object guided external memory network for video object detection," in *ICCV*, 2019.

[80] C. Guo, B. Fan, J. Gu, Q. Zhang, S. Xiang, V. Prinet, and C. Pan, "Progressive sparse local attention for video object detection," in *ICCV*, 2019.

[81] Y. Chen, Y. Cao, H. Hu, and L. Wang, "Memory enhanced global-local aggregation for video object detection," in *CVPR*, 2020.

[82] G. Bertasius, C. Feichtenhofer, D. Tran, J. Shi, and L. Torresani, "Learning temporal pose estimation from sparsely labeled videos," in *NIPS*, 2019.

[83] N. Bodla, B. Singh, R. Chellappa, and L. S. Davis, "Soft-nms–improving object detection with one line of code," in *IEEE International Conference on Computer Vision*, pp. 5561–5569, 2017.

[84] B. Xiao, H. Wu, and Y. Wei, "Simple baselines for human pose estimation and tracking," in *European Conference on Computer Vision*, pp. 466–481, 2018.

[85] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[86] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *CoRR*, 2015.

[87] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: Object detection via region-based fully convolutional networks," in *Advances in Neural Information Processing Systems*, pp. 379–387, 2016.

[88] T. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie, "Feature pyramid networks for object detection," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 936–944, 2017.

[89] Z. Cai and N. Vasconcelos, "Cascade R-CNN: Delving into high quality object detection," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6154–6162, 2018.

[90] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8759–8768, 2018.

[91] S. Zhang, L. Wen, X. Bian, Z. Lei, and S. Z. Li, "Single-shot refinement neural network for object detection," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4203–4212, 2018.

[92] C. Zhu, Y. He, and M. Savvides, "Feature selective anchor-free module for single-shot object detection," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[93] X. Zhang, F. Wan, C. Liu, R. Ji, and Q. Ye, "Freeanchor: Learning to match

anchors for visual object detection," in *Advances in Neural Information Processing Systems*, 2019.

[94] A. Gupta, P. Dollar, and R. Girshick, "LVIS: A dataset for large vocabulary instance segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[95] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "Imagenet large scale visual recognition challenge," *IJCV*, 2015.

[96] H. Chen, K. Sun, Z. Tian, C. Shen, Y. Huang, and Y. Yan, "Blendmask: Top-down meets bottom-up for instance segmentation," in *CVPR*, 2020.

[97] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3354–3361, 2012.

[98] Y. Xiang, W. Choi, Y. Lin, and S. Savarese, "Subcategory-aware convolutional neural networks for object proposals and detection," in *IEEE Winter Conference on Applications of Computer Vision*, pp. 924–933, 2017.

[99] F. Yu, D. Wang, E. Shelhamer, and T. Darrell, "Deep layer aggregation," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2403–2412, 2018.

[100] A. Newell, Z. Huang, and J. Deng, "Associative embedding: End-to-end learning for joint detection and grouping," in *Advances in Neural Information Processing Systems*, pp. 2277–2287, 2017.

[101] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7291–7299, 2017.

[102] G. Papandreou, T. Zhu, L.-C. Chen, S. Gidaris, J. Tompson, and K. Murphy, "Personlab: Person pose estimation and instance segmentation with a bottom-up, part-based, geometric embedding model," in *European Conference on Computer Vision*, pp. 269–286, 2018.

[103] S. Yang, P. Luo, C. C. Loy, and X. Tang, "Wider face: A face detection benchmark," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[104] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *IEEE International Conference on Computer Vision*, pp. 2223–2232, 2017.

[105] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1125–1134, 2017.

[106] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3431–3440, 2015.

[107] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *IEEE Conference on Computer Vision and Pattern Recognition*, June 2018.

[108] R. Girshick, "Fast R-CNN," in *IEEE International Conference on Computer Vision*, 2015.

[109] K. Oksuz, B. C. Cam, S. Kalkan, and E. Akbas, "Imbalance Problems in Object Detection: A Review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2020.

[110] T. Yang, X. Zhang, W. Zhang, and J. Sun, "Metaanchor: Learning to detect objects with customized anchors," in *NIPS*, 2018.

[111] A. Shrivastava, A. Gupta, and R. Girshick, "Training region-based object detectors with online hard example mining," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 761–769, 2016.

[112] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian, "Centernet: Keypoint triplets for object detection," in *IEEE International Conference on Computer Vision*, 2019.

[113] T. Kong, F. Sun, H. Liu, Y. Jiang, L. Li, and J. Shi, "Foveabox: Beyond anchor-based object detector," *IEEE Transactions on Image Processing*, pp. 7389–7398, 2020.

[114] J. Wang, K. Chen, S. Yang, C. C. Loy, and D. Lin, "Region proposal by guided anchoring," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2965–2974, 2019.

[115] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1492–1500, 2017.

[116] K. Chen, J. Wang, J. Pang, Y. Cao, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Xu, Z. Zhang, D. Cheng, C. Zhu, T. Cheng, Q. Zhao, B. Li, X. Lu, R. Zhu, Y. Wu, J. Dai, J. Wang, J. Shi, W. Ouyang, C. C. Loy, and D. Lin, "MMDetection: Open mmlab detection toolbox and benchmark," *arXiv preprint arXiv:1906.07155*, 2019.

[117] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, pp. 8024–8035, 2019.

[118] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes (VOC) Challenge," *International Journal of Computer Vision*, pp. 303–338, 2010.

[119] Y. Du, W. Wang, and L. Wang, "Hierarchical recurrent neural network for skeleton based action recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[120] M. Li, S. Chen, X. Chen, Y. Zhang, Y. Wang, and Q. Tian, "Actional-structural graph convolutional networks for skeleton-based action recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[121] A. Yan, Y. Wang, Z. Li, and Y. Qiao, "Pa3d: Pose-action 3d machine for video

recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[122] L. Huang, Y. Huang, W. Ouyang, and L. Wang, "Part-aligned pose-guided recurrent network for action recognition," *Pattern Recognition*, 2019.

[123] D. C. Luvizon, D. Picard, and H. Tabia, "2d/3d pose estimation and action recognition using multitask deep learning," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[124] H. Choi, G. Moon, and K. M. Lee, "Pose2mesh: Graph convolutional network for 3d human pose and mesh recovery from a 2d human pose," in *European Conference on Computer Vision*, 2020.

[125] J. N. Kundu, M. Rakesh, V. Jampani, R. M. Venkatesh, and R. V. Babu, "Appearance consensus driven self-supervised human mesh recovery," in *European Conference on Computer Vision*, 2020.

[126] U. Iqbal, K. Xie, Y. Guo, J. Kautz, and P. Molchanov, "Kama: 3d keypoint aware body mesh articulation," *arXiv preprint arXiv:2104.13502*, 2021.

[127] A. Kanazawa, M. J. Black, D. W. Jacobs, and J. Malik, "End-to-end recovery of human shape and pose," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[128] G. Cimen, C. Maurhofer, B. Sumner, and M. Guay, "Ar poser: Automatically augmenting mobile pictures with digital avatars imitating poses," in *12th International Conference on Computer Graphics, Visualization, Computer Vision and Image Processing*, 2018.

[129] A. Elhayek, O. Kovalenko, P. Murthy, J. Malik, and D. Stricker, "Fully automatic multi-person human motion capture for vr applications," in *International Conference on Virtual Reality and Augmented Reality*, 2018.

[130] W. Xu, A. Chatterjee, M. Zollhoefer, H. Rhodin, P. Fua, H.-P. Seidel, and C. Theobalt, "Mo2cap2: Real-time mobile 3d motion capture with a cap-mounted fisheye camera," *IEEE transactions on visualization and computer graphics*, 2019.

[131] "Azure kinect body tracking joints.".

[132] "3d skeletal tracking on azure kinect.".

[133] "How huawei ml kit's face detection and hand keypoint detection capabilities helped with creating the game crazy rockets." https://medium.com/huawei-developers/how-huawei-ml-kits-face-detection-and-hand-keypoint-detection-capabilities-helped-with-creating-6a22fdb7f967.

[134] L. Kumarapu and P. Mukherjee, "Animepose: Multi-person 3d pose estimation and animation," *Pattern Recognition Letters*, 2021.

[135] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh, "Openpose: realtime multi-person 2d pose estimation using part affinity fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[136] G. Hidalgo, Y. Raaj, H. Idrees, D. Xiang, H. Joo, T. Simon, and Y. Sheikh, "Single-network whole-body pose estimation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[137] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *European Conference on Computer Vision*, 2014.

[138] K. Sun, B. Xiao, D. Liu, and J. Wang, "Deep high-resolution representation learning for human pose estimation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[139] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[140] G. Ning, Z. Zhang, and Z. He, "Knowledge-guided deep fractal neural networks for human pose estimation," *IEEE Transactions on Multimedia*, 2017.

[141] A. Newell, Z. Huang, and J. Deng, "Associative embedding: End-to-end learning for joint detection and grouping," *arXiv preprint arXiv:1611.05424*, 2016.

[142] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *European Conference on Computer Vision*, 2016.

[143] M. Kocabas, S. Karagoz, and E. Akbas, "Multiposenet: Fast multi-person pose estimation using pose residual network," in *European Conference on Computer Vision*, 2018.

[144] G. Papandreou, T. Zhu, L.-C. Chen, S. Gidaris, J. Tompson, and K. Murphy, "Personlab: Person pose estimation and instance segmentation with a bottom-up, part-based, geometric embedding model," in *European Conference on Computer Vision*, 2018.

[145] A. Bulat and G. Tzimiropoulos, "Human pose estimation via convolutional part heatmap regression," in *European Conference on Computer Vision*, 2016.

[146] L. Pishchulin, E. Insafutdinov, S. Tang, B. Andres, M. Andriluka, P. V. Gehler, and B. Schiele, "Deepcut: Joint subset partition and labeling for multi person pose estimation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[147] E. Insafutdinov, M. Andriluka, L. Pishchulin, S. Tang, E. Levinkov, B. Andres, and B. Schiele, "Arttrack: Articulated multi-person tracking in the wild," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[148] E. Insafutdinov, L. Pishchulin, B. Andres, M. Andriluka, and B. Schiele, "Deepercut: A deeper, stronger, and faster multi-person pose estimation model," in *European Conference on Computer Vision*, 2016.

[149] U. Iqbal, A. Milan, and J. Gall, "Posetrack: Joint multi-person pose estimation and tracking," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[150] S. Jin, W. Liu, W. Ouyang, and C. Qian, "Multi-person articulated tracking with spatial and temporal embeddings," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[151] S. Jin, X. Ma, Z. Han, Y. Wu, W. Yang, W. Liu, C. Qian, and W. Ouyang, "Towards multi-person pose tracking: Bottom-up and top-down methods," in *ICCV PoseTrack Workshop*, 2017.

[152] Y. Chen, Z. Wang, Y. Peng, Z. Zhang, G. Yu, and J. Sun, "Cascaded pyramid network for multi-person pose estimation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[153] G. Papandreou, T. Zhu, N. Kanazawa, A. Toshev, J. Tompson, C. Bregler, and K. Murphy, "Towards accurate multi-person pose estimation in the wild," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[154] H.-S. Fang, S. Xie, Y.-W. Tai, and C. Lu, "Rmpe: Regional multi-person pose estimation," in *IEEE International Conference on Computer Vision*, 2017.

[155] B. Xiao, H. Wu, and Y. Wei, "Simple baselines for human pose estimation and tracking," in *European Conference on Computer Vision*, 2018.

[156] X. Cao, Y. Wei, F. Wen, and J. Sun, "Face alignment by explicit shape regression," *International Journal of Computer Vision*, 2014.

[157] G. Tzimiropoulos, "Project-out cascaded regression with an application to face alignment," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[158] G. Trigeorgis, P. Snape, M. A. Nicolaou, E. Antonakos, and S. Zafeiriou, "Mnemonic descent method: A recurrent process applied for end-to-end face alignment," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[159] Z. Zhang, P. Luo, C. C. Loy, and X. Tang, "Learning deep representation for face alignment with auxiliary attributes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2015.

[160] J. Deng, J. Guo, E. Ververas, I. Kotsia, and S. Zafeiriou, "Retinaface: Single-shot multi-level face localisation in the wild," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[161] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint face detection and alignment using multitask cascaded convolutional networks," *IEEE Signal Processing Letters*, 2016.

[162] M. Oberweger, P. Wohlhart, and V. Lepetit, "Hands deep in deep learning for hand pose estimation," *20th Computer Vision Winter Workshop*, 2015.

[163] M. Oberweger and V. Lepetit, "Deepprior++: Improving fast and accurate 3d hand pose estimation," in *IEEE International Conference on Computer Vision Workshops*, 2017.

[164] T. Sharp, C. Keskin, D. Robertson, J. Taylor, J. Shotton, D. Kim, C. Rhemann, I. Leichter, A. Vinnikov, Y. Wei, *et al.*, "Accurate, robust, and flexible real-time hand tracking," in *Proceedings of the 33rd annual ACM conference on human factors in computing systems*, 2015.

[165] S. Sridhar, F. Mueller, A. Oulasvirta, and C. Theobalt, "Fast and robust hand tracking using detection-guided optimization," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[166] K. Sun, Y. Zhao, B. Jiang, T. Cheng, B. Xiao, D. Liu, Y. Mu, X. Wang, W. Liu, and J. Wang, "High-resolution representations for labeling pixels and regions," *arXiv preprint arXiv:1904.04514*, 2019.

Appendix A

A STEP-BY-STEP ANIMATION OF THE VOTING PROCESS.

Figure A.1 illustrates the vote aggregation process for two steps, for a specific class. Each row corresponds to a single step. $\mathbf{E}$ is the visual evidence tensor for a specific class. Its size is $H \times W \times R$. $\mathbf{O}$ is the corresponding object presence map with size $H \times W$.

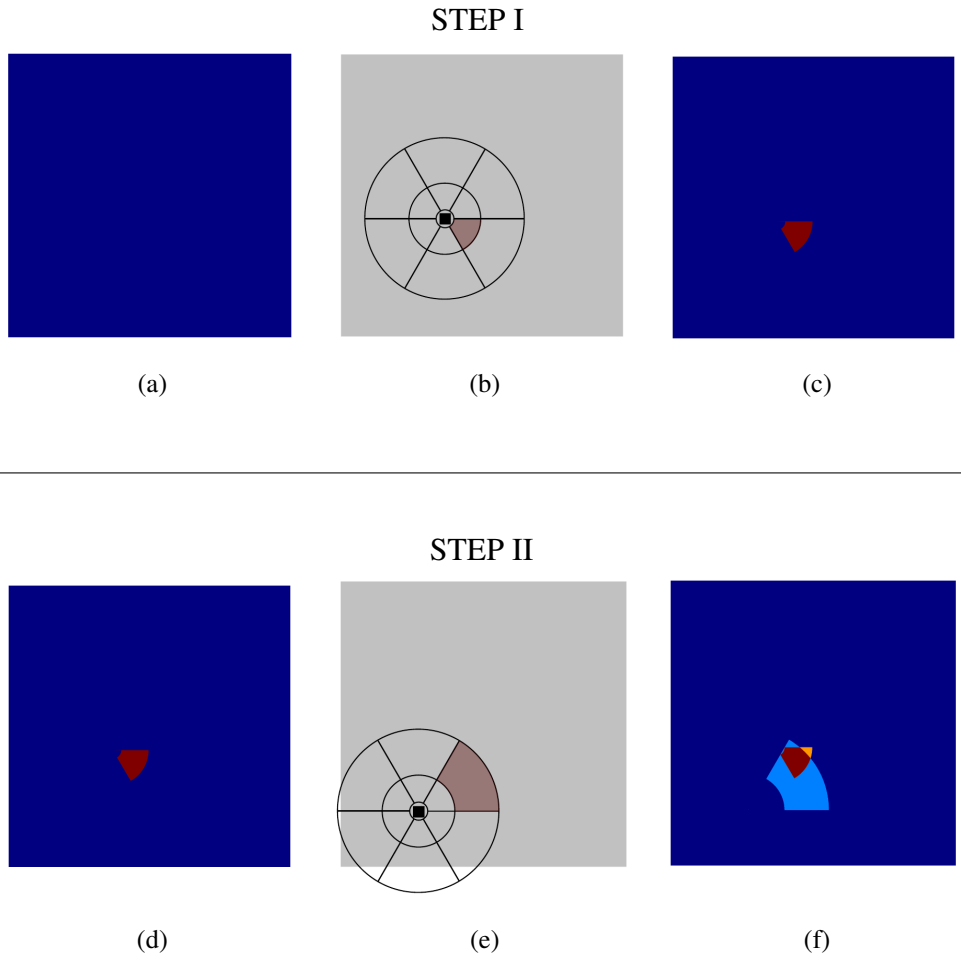


Figure A.1: (a) **O** is initialized with zeros. (b) We pick an arbitrary location (i, j) and the 3rd channel of **E**. We place the vote field centered at the location (i, j). Since, we are using the 3rd channel, the relevant region of the vote field is 3 (shown with pink color on **E**). Finally we apply the voting. (c) **O** after voting. (d) **O** before the voting. (e) To further illustrate the voting process, we pick another location and the 8th channel in **E**. We place the vote field centered at the location. The region 8 marks the target area to be voted on. We apply the voting. (f) **O** after voting. The colors in **O** indicate accumulated vote strength.

Appendix B

COCO `MINITRAIN` STATISTICS

We present several statistics about COCO `minitrain` and show that `minitrain`'s object instance statistics match those of `train2017`. We show Person class separately, because it is the most dominant class in the dataset. When we add person class to the figures, it becomes harder to see the details of other classes.

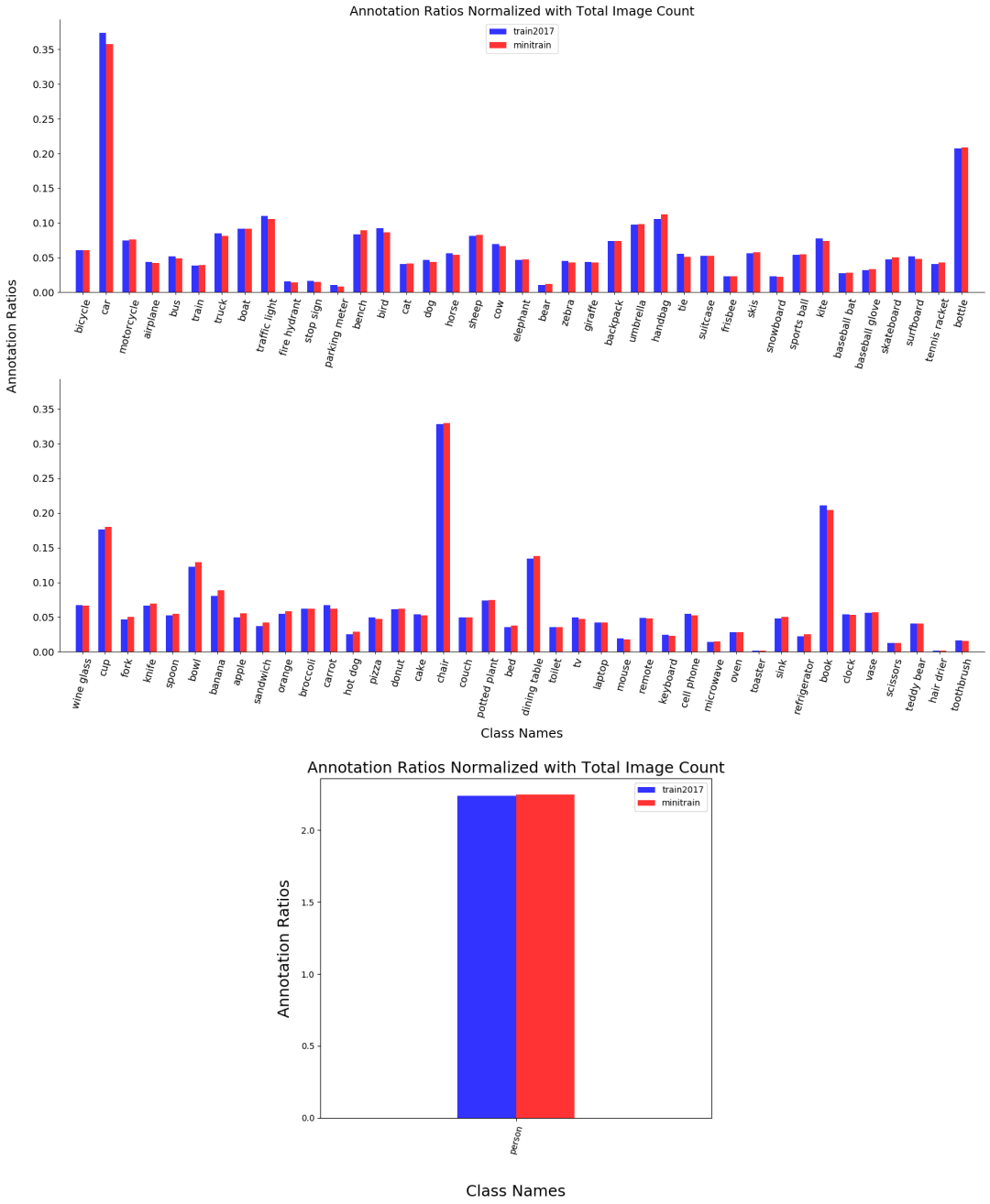


Figure B.1: (Top) Total annotations (i.e. object instances) normalized with total image counts in the dataset. (Bottom) *Person* annotations normalized with total image counts in the dataset.

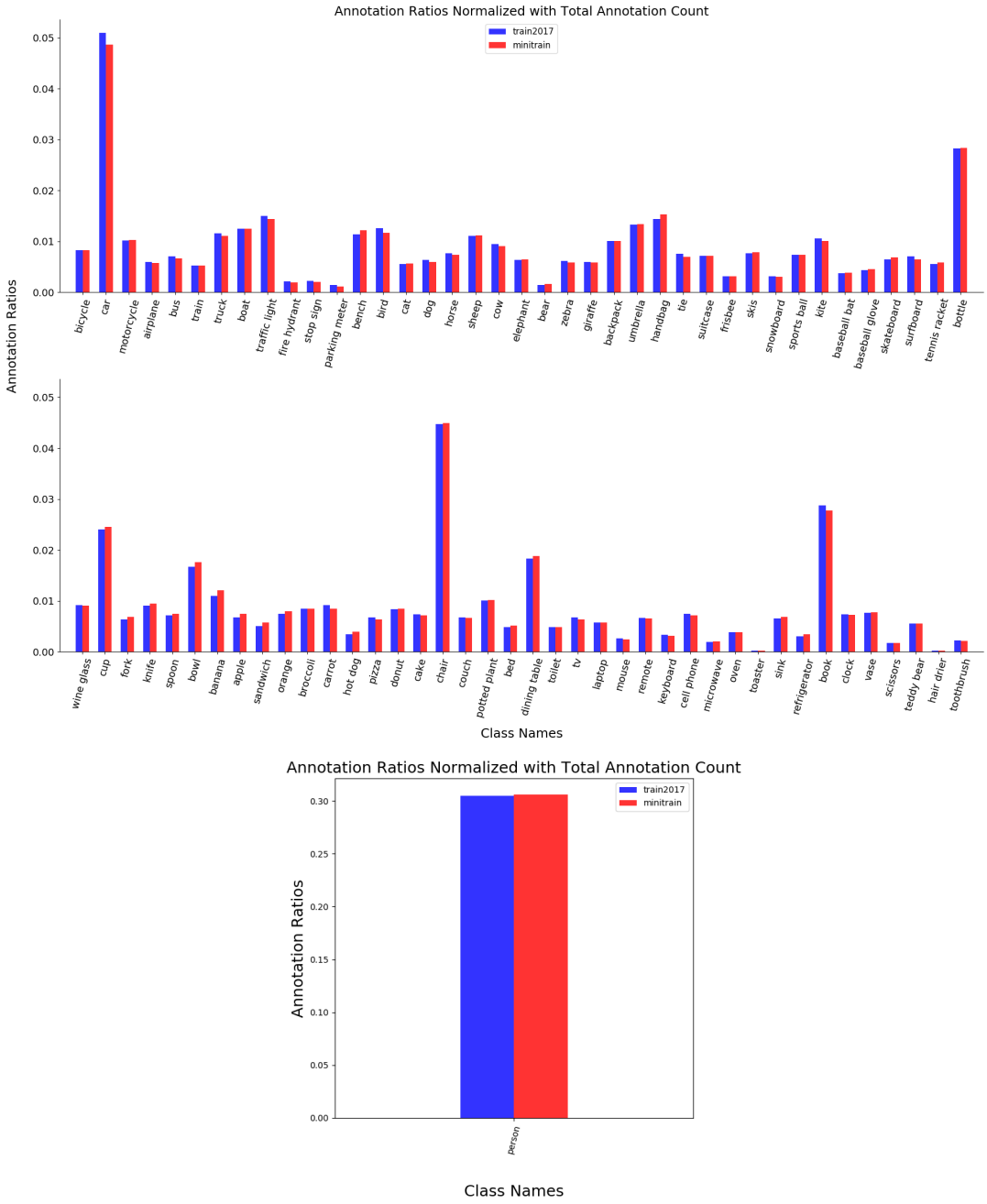


Figure B.2: (Top) Total annotations normalized with total annotation counts in the dataset. (Bottom) *Person* annotations normalized with total annotation counts in the dataset.

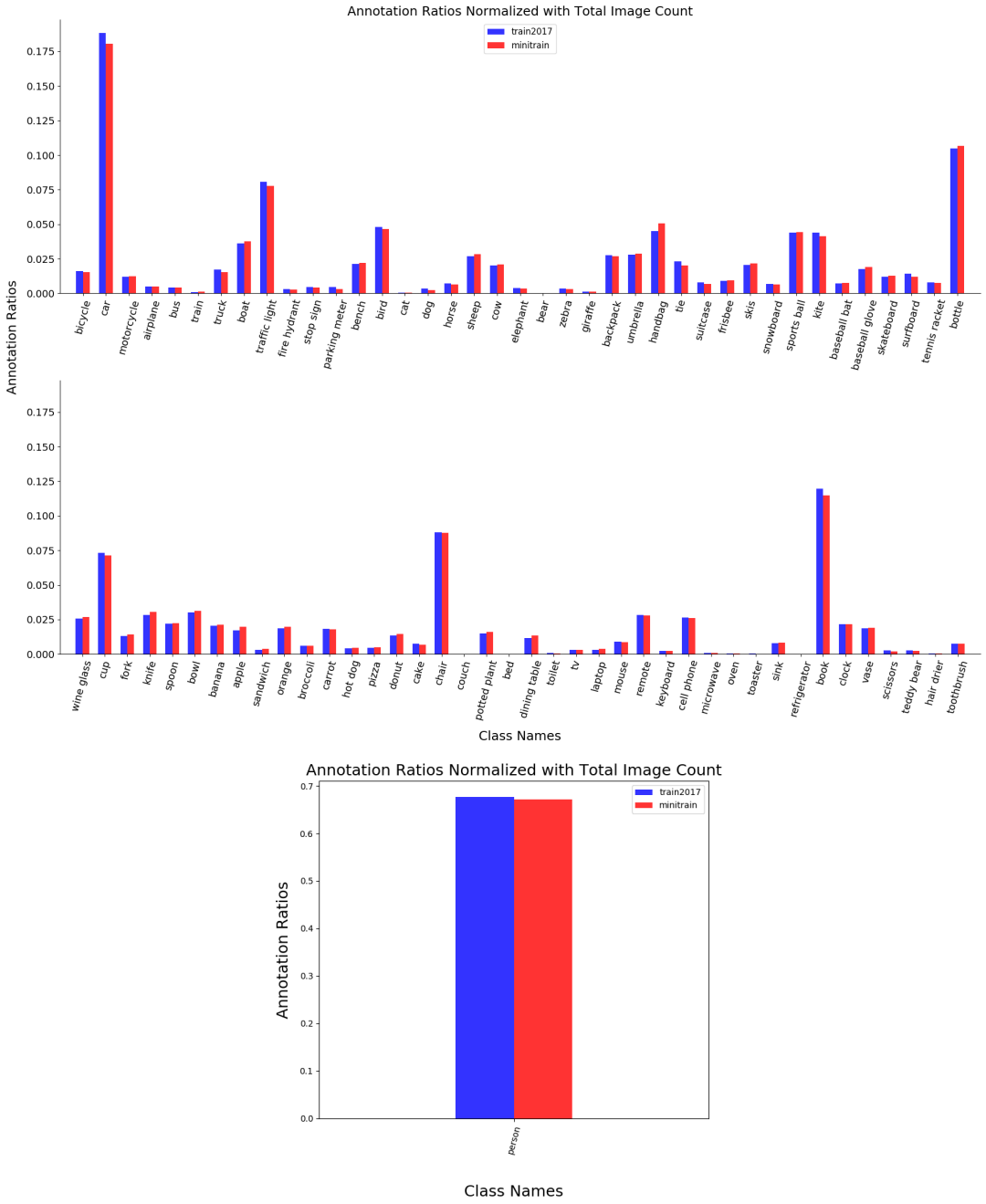


Figure B.3: (Top) *Small* annotations normalized with total image counts in the dataset. (Bottom) *Small Person* annotations normalized with total image counts in the dataset.

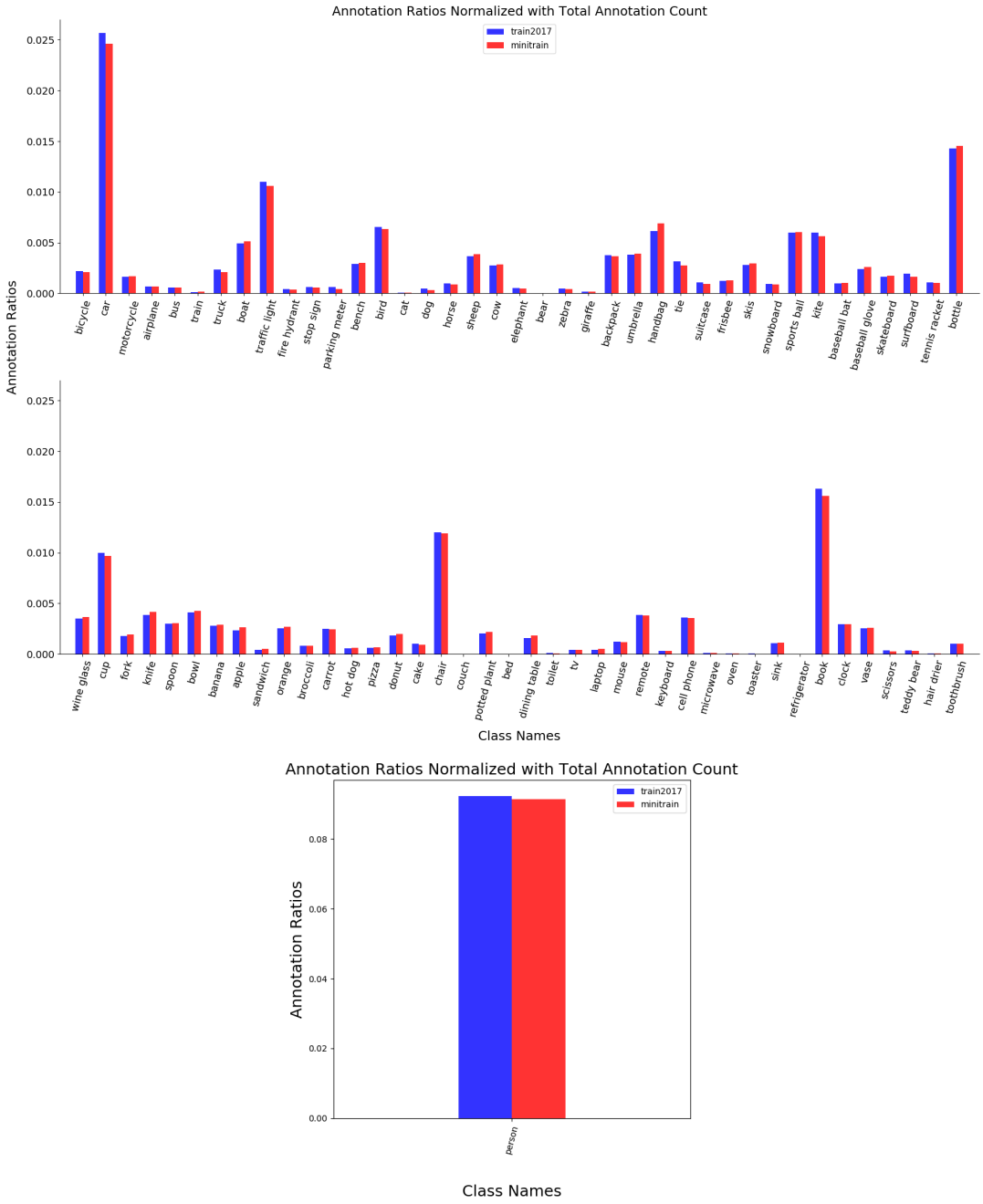


Figure B.4: (Top) *Small* annotations normalized with total annotation counts in the dataset. (Bottom) *Small Person* annotations normalized with total annotation counts in the dataset.

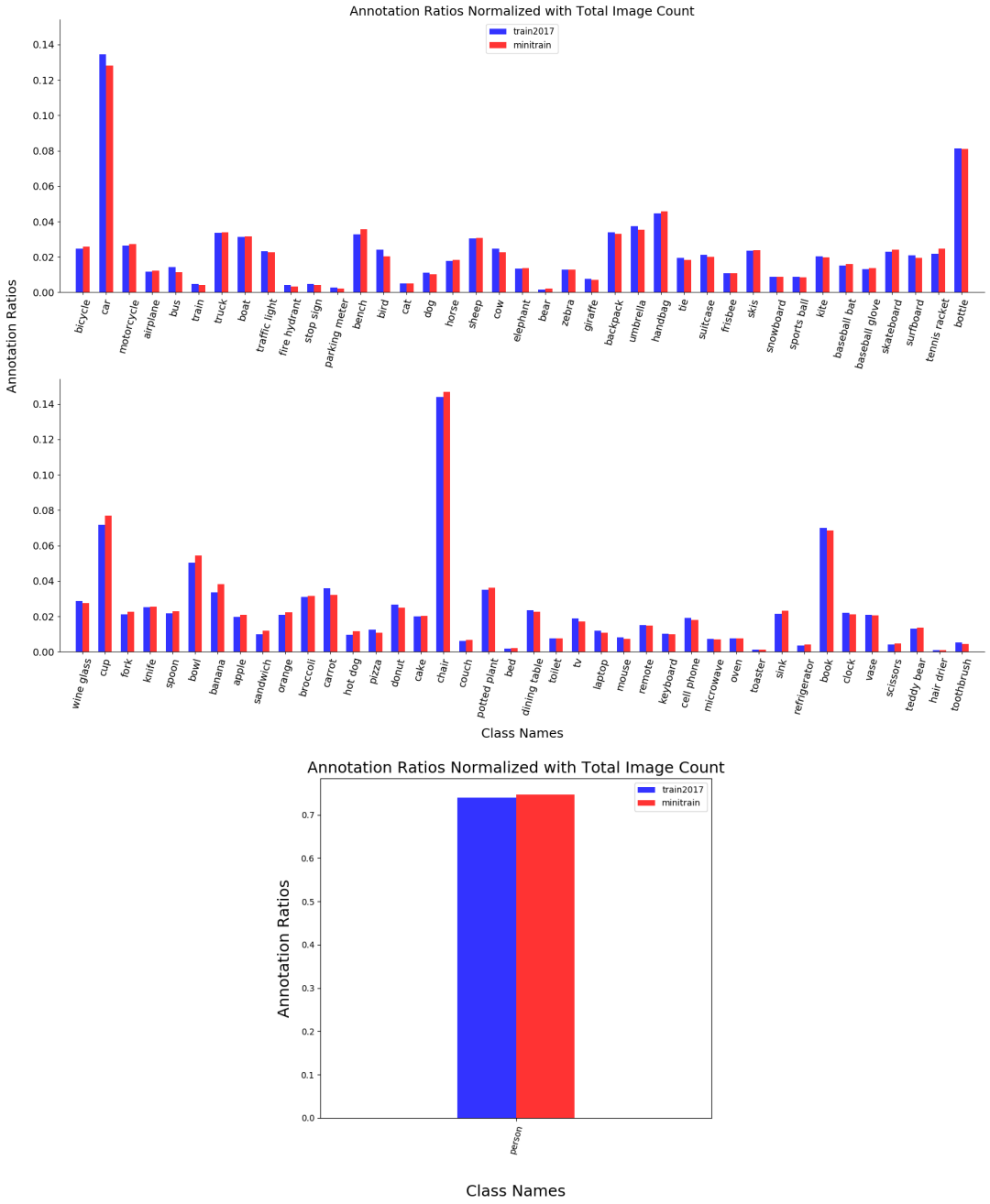


Figure B.5: (Top) *Medium* annotations normalized with total image counts in the dataset. (Bottom) *Medium Person* annotations normalized with total image counts in the dataset.

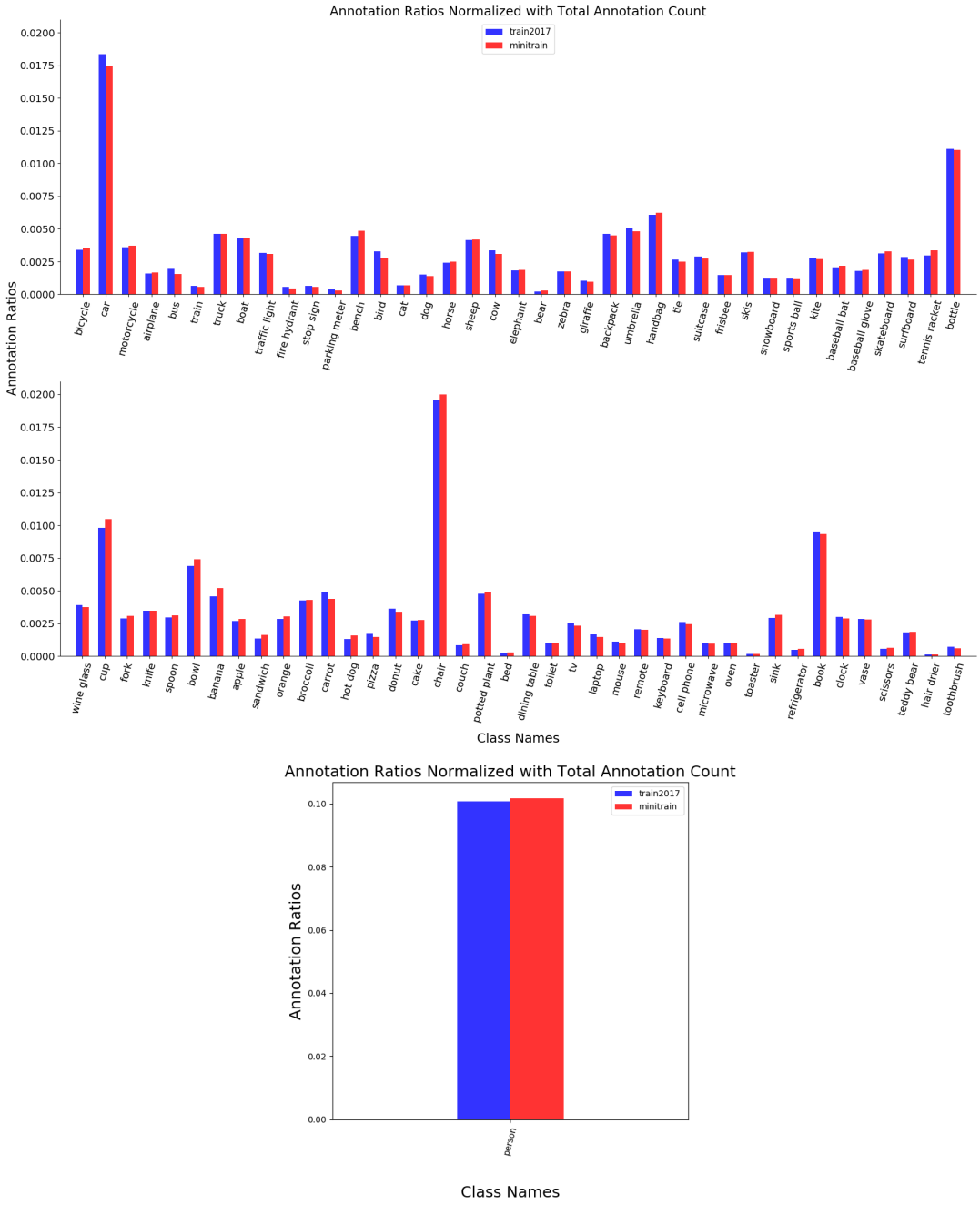


Figure B.6: (Top) *Medium* annotations normalized with total annotation counts in the dataset. (Bottom) *Medium Person* annotations normalized with total annotation counts in the dataset.

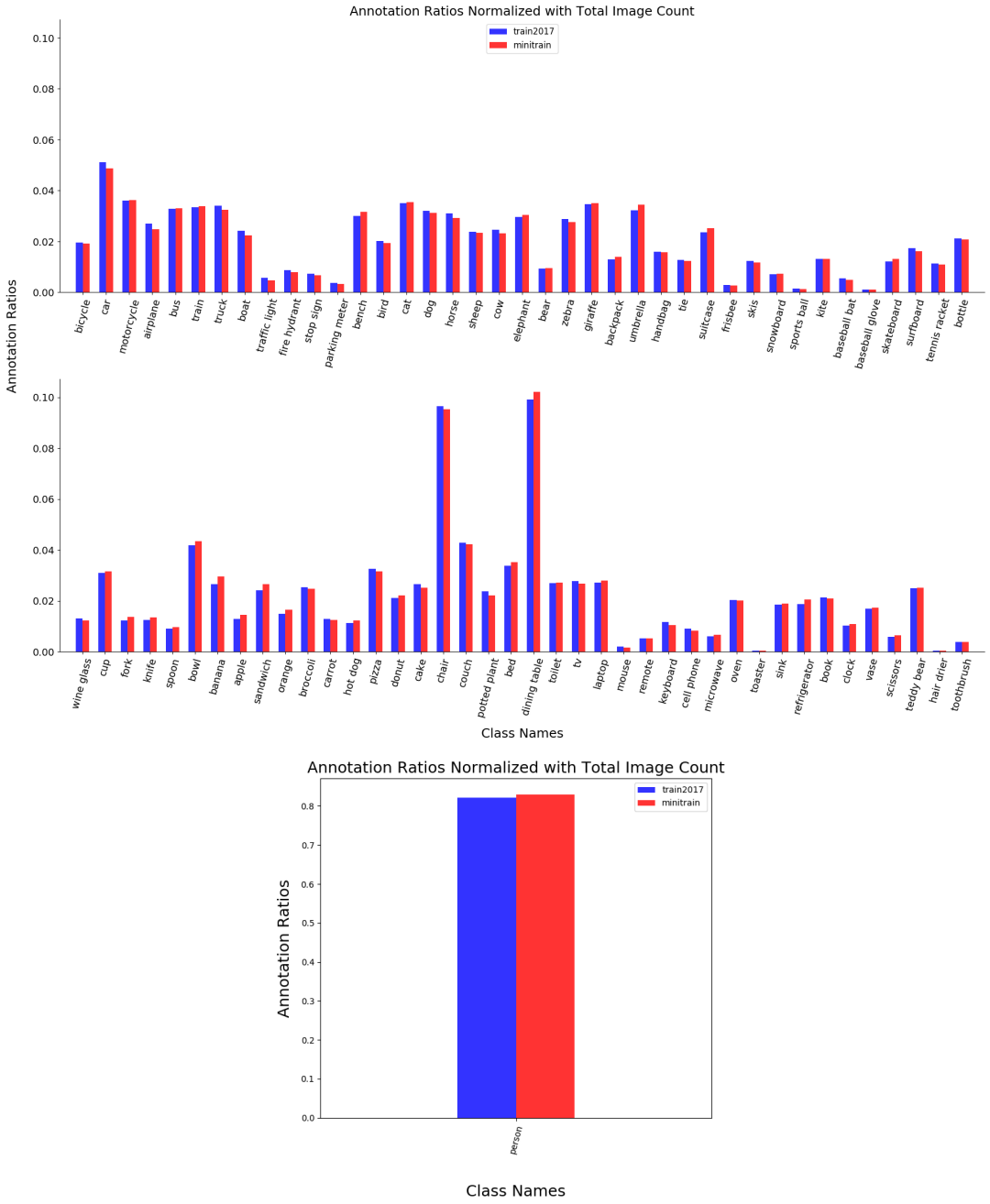


Figure B.7: (Top) *Large* annotations normalized with total image counts in the dataset. (Bottom) *Large Person* annotations normalized with total image counts in the dataset.

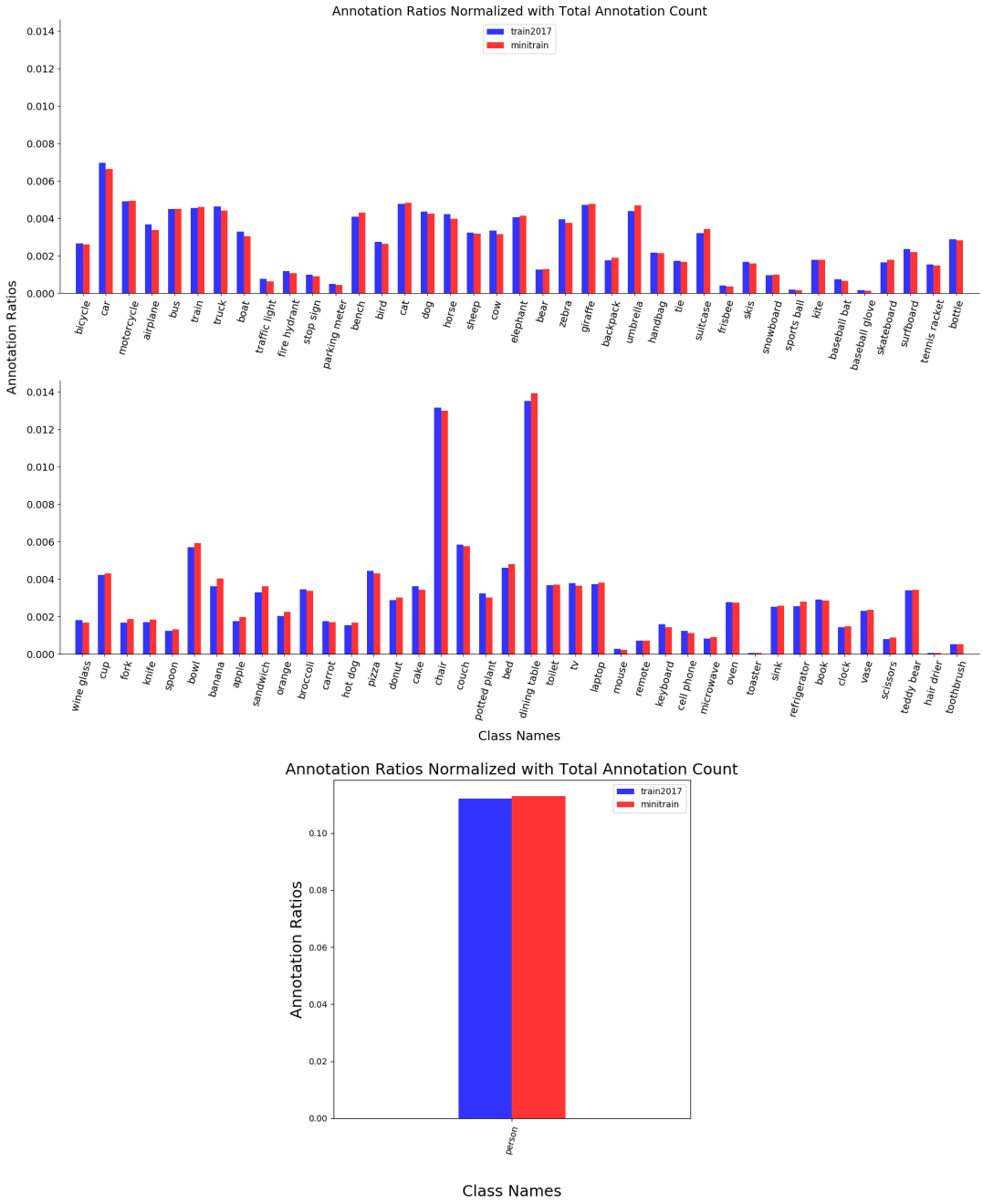


Figure B.8: (Top) *Large* annotations normalized with total annotation counts in the dataset. (Bottom) *Large Person* annotations normalized with total annotation counts in the dataset.

Appendix C

MORE VISUAL RESULTS ON OBJECT DETECTION

Following, we present more visual results on object detection.

Figure C.1: *Fire hydrant* detection gets strong votes from *cars*, *person*, *buildings* and *road*.

Figure C.2: *Tennis racket* detection gets strong votes from *person*.

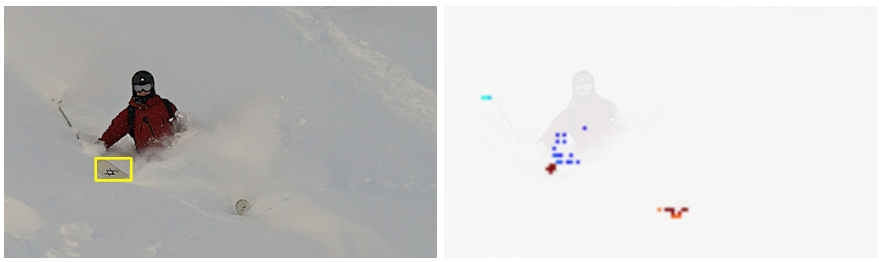


Figure C.3: *Ski* detection gets strong votes from other *ski*, *ski baton* and *person*.

Figure C.4: *Kite* detection gets strong votes from *person* and *sky*.

Figure C.5: *Sports ball* detection gets strong votes from *person*.

Figure C.6: *Television* detection gets strong votes from common things in a living room such as *paintings* at the wall and *books* in the shelf.

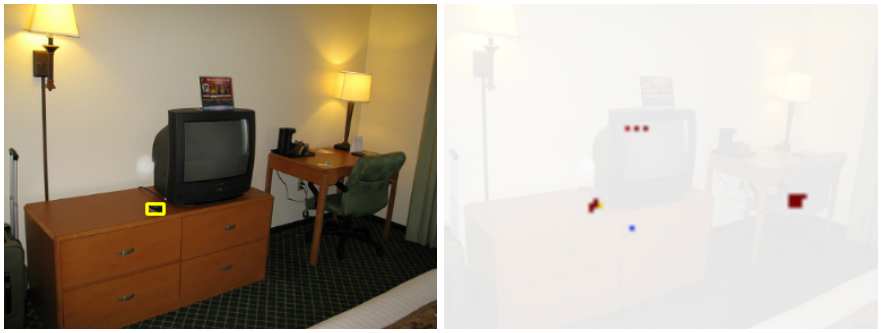


Figure C.7: *Remote* detection gets strong votes from *television* and *chair* objects.

Figure C.8: *Television* detection gets strong votes from things in a living room such as *lamp* (is not among 80 classes of COCO dataset), *chair* and *couch*.

Figure C.9: *Television* detection gets strong votes from things in a kitchen such as *lamp* (is not among 80 classes of COCO dataset), and *couch*.

Appendix D

INTERACTION AMONG OBJECT CLASSES

We present the full 80×80 matrix to visualize voting relations between classes on the COCO dataset in Figure D.1.

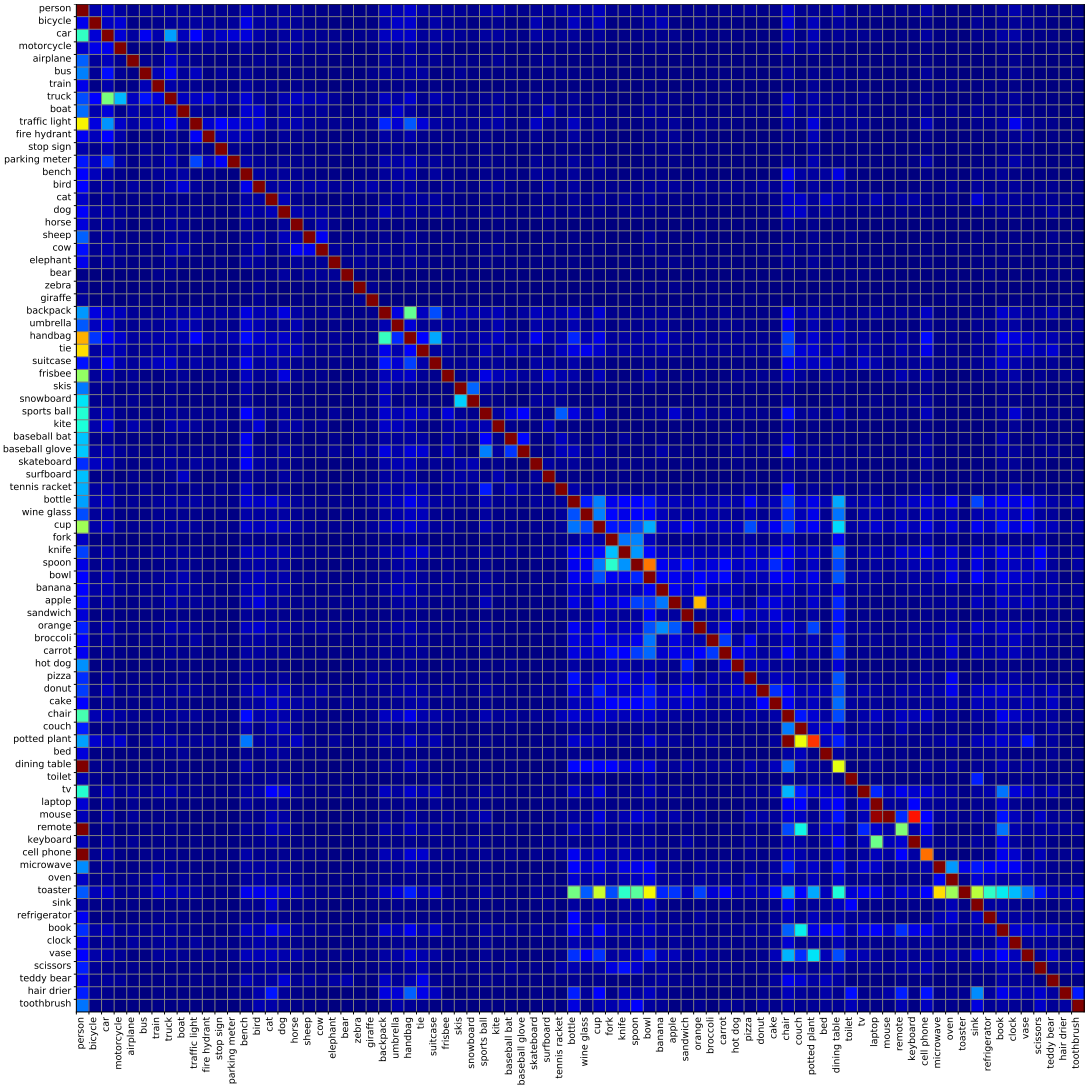


Figure D.1: We present the 80×80 matrix to visualize voting relations between classes on the COCO dataset. Matrix rows are vote-getters classes and columns are voters.