Addressing Occlusion in Panoptic Segmentation

Ajit Sarkaar

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Engineering

A. Lynn Abbott, Chair
Bert Huang
Creed F. Jones

December 17, 2020
Blacksburg, Virginia

Keywords: Deep Learning, Image Segmentation, Object Detection, Image Classification,
Autonomous Systems

Addressing Occlusion in Panoptic Segmentation

Ajit Sarkaar

(ABSTRACT)

Visual recognition tasks have witnessed vast improvements in performance since the advent of deep learning. Despite the gains in performance, image understanding algorithms are still not completely robust to partial occlusion. In this work, we propose a novel object classification method based on compositional modelling and explore its effect in the context of the newly introduced panoptic segmentation task. The panoptic segmentation task combines both semantic and instance segmentation to perform labelling of the entire image. The novel classification method replaces the object detection pipeline in UPSNet, a Mask R-CNN based design for panoptic segmentation. We also discuss an issue with the segmentation mask prediction of Mask R-CNN that affects overlapping instances. We perform extensive experiments and showcase results on the complex COCO and Cityscapes datasets. The novel classification method shows promising results for object classification on occluded instances in complex scenes.

Addressing Occlusion in Panoptic Segmentation

Ajit Sarkaar

(GENERAL AUDIENCE ABSTRACT)

Visual recognition tasks have witnessed vast improvements in performance since the advent of deep learning. Despite making significant improvements, algorithms for these tasks still do not perform well at recognizing partially visible objects in the scene. In this work, we propose a novel object classification method that uses compositional models to perform part based detection. The method first looks at individual parts of an object in the scene and then makes a decision about its identity. We test the proposed method in the context of the recently introduced panoptic segmentation task. The panoptic segmentation task combines both semantic and instance segmentation to perform labelling of the entire image. The novel classification method replaces the object detection module in UPSNet, a Mask R-CNN based algorithm for panoptic segmentation. We also discuss an issue with the segmentation mask prediction of Mask R-CNN that affects overlapping instances. After performing extensive experiments and evaluation, it can be seen that the novel classification method shows promising results for object classification on occluded instances in complex scenes.

Acknowledgments

First and foremost, I would like to thank Dr. A. Lynn Abbott for his constant guidance and support throughout this work and graduate studies as well. I am grateful for all the teachings and valuable feedback provided through engrossed discussions that helped me improve as a researcher.

I would also like to thank Dr. Huang and Dr. Jones for serving on my committee and providing valuable insights on this work.

I am grateful to the Graduate School for their continued assistance and Advanced Research Computing at Virginia Tech for concise tutorials and help during office hour sessions.

I would like to thank my family for all their support and always pushing me to do my best.

Finally, I am also incredibly grateful to all my teachers at Sahyadri and Riverdale for providing a great learning experience and motivating me to pursue a career in engineering science.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Visual recognition tasks have witnessed vast improvements in performance since the advent of deep learning methods. Some important visual recognition tasks include image classification, object detection and semantic segmentation. Visual recognition tasks have applicability across a wide range of domains such as autonomous systems, robotics, medical imaging and diagnosis and space science, to name a few. The current state of deep learning research on these tasks offers a number of options available for use in systems, with trade-offs in accuracy, speed and size. This takeover of deep learning and its prevalence seen today can, in part, be attributed to the image classification task of the ImageNet challenge [64], as AlexNet [40] successfully demonstrated the potential that deep neural networks possessed and commenced this revolution. Since then, research on image classification has progressed to a point that image classification is now considered as a solved task with the ResNet [29] architecture achieving a phenomenal 3.57% Top-5 error rate (ensemble version). Strides have also been taken in the research of object detection, which combines both classification and localization by predicting a precise bounding box around each object and identifying its class. For object detection, region proposal based methods such as the R-CNN family of algorithms [26] [25] [62] take the lead in terms of performance numbers and qualitative performance, but are slower than near real-time object detectors like YOLO [60] and SSD

[53], as both YOLO and SSD make predictions for a fixed number of proposals and sacrifice some performance for faster runtime. Another important visual recognition task is image segmentation. Unlike classification or detection, the progress for the image segmentation task has been somewhat disjointed.

Image segmentation is the task of dividing an image into regions or groups of pixels. The task of segmentation has been studied since the early days of computer vision. Historically, countable things such as people, man-made objects, animals received much of the attention until Adelson [1] emphasized the importance of studying amorphous semantic regions as well. Semantic and instance segmentation have since been studied separately and this divide between the two sub tasks has persisted over the years, and exists, even today.

Kirillov et al. [36] cite lack of performance metrics on a joint task (task combining semantic and instance segmentation) as one of the reasons that the sub-tasks have been studied in isolation. As complete scene understanding has become increasingly important due to its critical role in robotics, autonomous systems and numerous other fields, it is not enough to study either one of these tasks in isolation. In an effort to combine study of semantic and instance classes, Kirillov et al. [36] introduced a unified task termed Panoptic Segmentation. Panoptic Segmentation involves performing segmentation of the complete image, whilst also counting the number of instances present for each *foreground* category of classes. The unified task is explained in Figure 1.1. It includes the study of both *thing* (countable objects such as person, car, etc.) and *stuff* (amorphous regions such as sky, river, etc.) classes. Kirillov et al. also outline the novel segmentation task format, provide a thorough review of available datasets, human consistency studies and introduce a new performance metric for this unified task called Panoptic Quality. Panoptic Quality is a unified metric that gives *thing* and *stuff* classes equal importance and provides a single metric for both object detection and segmentation while maintaining simplicity.

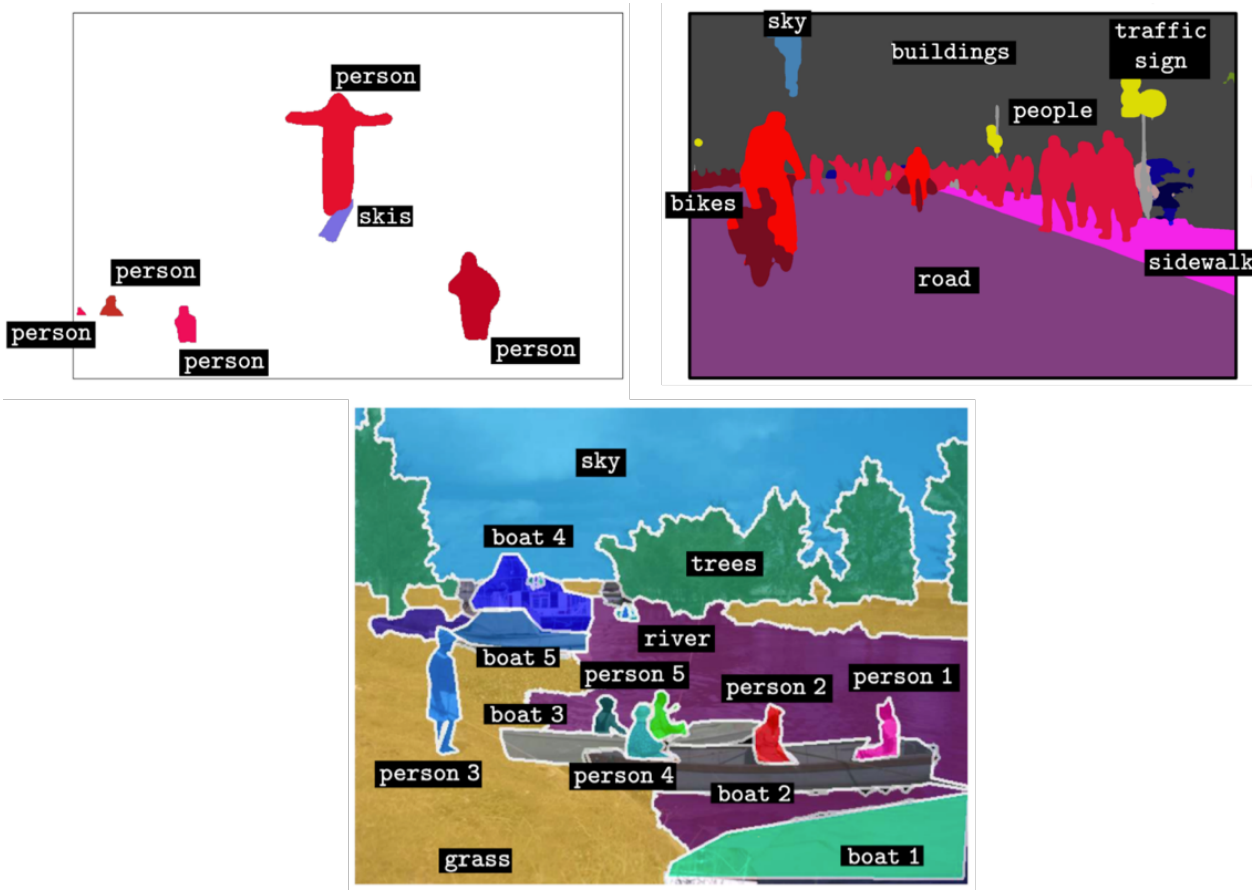

Figure 1.1: The unified task of panoptic segmentation. The example in the top left shows what instance segmentation attends to, while the example in the top right shows the working of semantic segmentation. As seen, instance segmentation has no understanding about general scene layout and context. Semantic segmentation on the other hand, cannot reason about separate instances of objects. Panoptic segmentation *(bottom row)* combines the two into a unified task. Note that the instances are numbered since multiple instances of the same class are identified separately. *(Image credit: [36])*

Occlusion plays a crucial role in scene understanding. Occlusion occurs when 3D objects that are separated along the axis of the viewpoint, are projected onto the same points in the 2D image plane. Objects closer to the image plane partially or completely block the objects located at a farther distance. The problem of occlusion has been extensively studied in the field of computer vision [78] [18] [69] [32]. Despite the impressive performance of deep learning based methods on image recognition tasks, deep learning based image classifiers are

still not robust to the occlusion problem, as shown in [91], [39].

One might argue that appending additional data or increasing dataset size by using data augmentation techniques might mitigate degradation of performance and overcome these problems. However, experiments carried out in [38] show that these techniques by themselves do not help in improving performance of Deep Convolutional Neural Network (DCNN) based classifiers in the presence of occlusion. Figure 1.2 shows a few examples of real world scenes where some objects are occluded and cause DCNN based classifiers to make incorrect predictions.

In addition to the occlusion problem, an image classifier should also be robust to variations in object pose, scene lighting and noise. As learnings of deep learning based classifiers are derived entirely from large sets of training data, if the data does not contain enough examples that help the classifier model these variations, performance of deep learning models may drop even further. Several works [70] [68] [19] have simulated attacks on deep learning models to investigate their stability and performance under adverse conditions as the problem of AI safety becomes increasingly important. Fawzi and Frossard [19] carried out mask attacks where a part of the object was masked out and caused the DCNN based classifiers to make implausible predictions. Figure 1.3 shows how the VGG [66] classifier breaks and makes incorrect predictions even though a large portion of the input still remains unchanged. Szegedy et al. [70] subject deep neural networks to stress to determine just how many pixels in the input need to be altered, to fool the network. They show that applying a certain hardly perceptible perturbation to the input image can cause the network to completely misclassify an image. The perturbation signal is found by maximizing the network's prediction error. Su et al. [68] use the differential evolution technique to find that a change as minute as modifying just a single pixel in the input can cause these classifiers to make wildly different predictions and brings the brittle understanding of these image classifiers to light. Figure

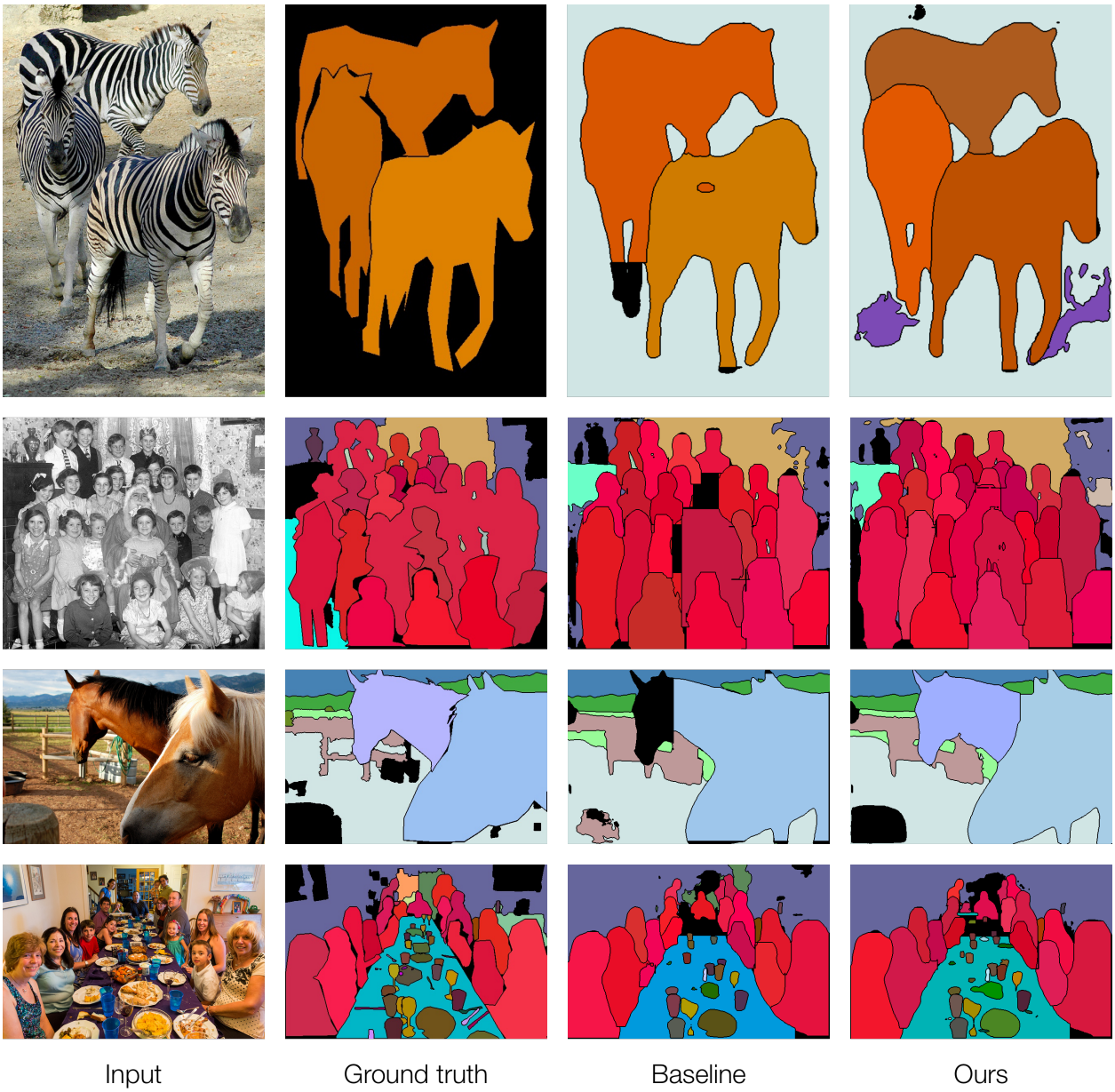

Figure 1.2: Panoptic segmentation results on images from the COCO dataset. First two columns are input and ground truth data. The third and fourth columns show results of the baseline and our approach respectively. Overlapping animal instances in the first and third rows are detected successfully by our approach. Some people missed by the baseline in the second and fourth rows are also detected well.

1.4 shows the findings of [68]. As systems that rely on 2D perception for reasoning become more and more ubiquitous, the importance of designing robust classifiers cannot be stressed

enough.

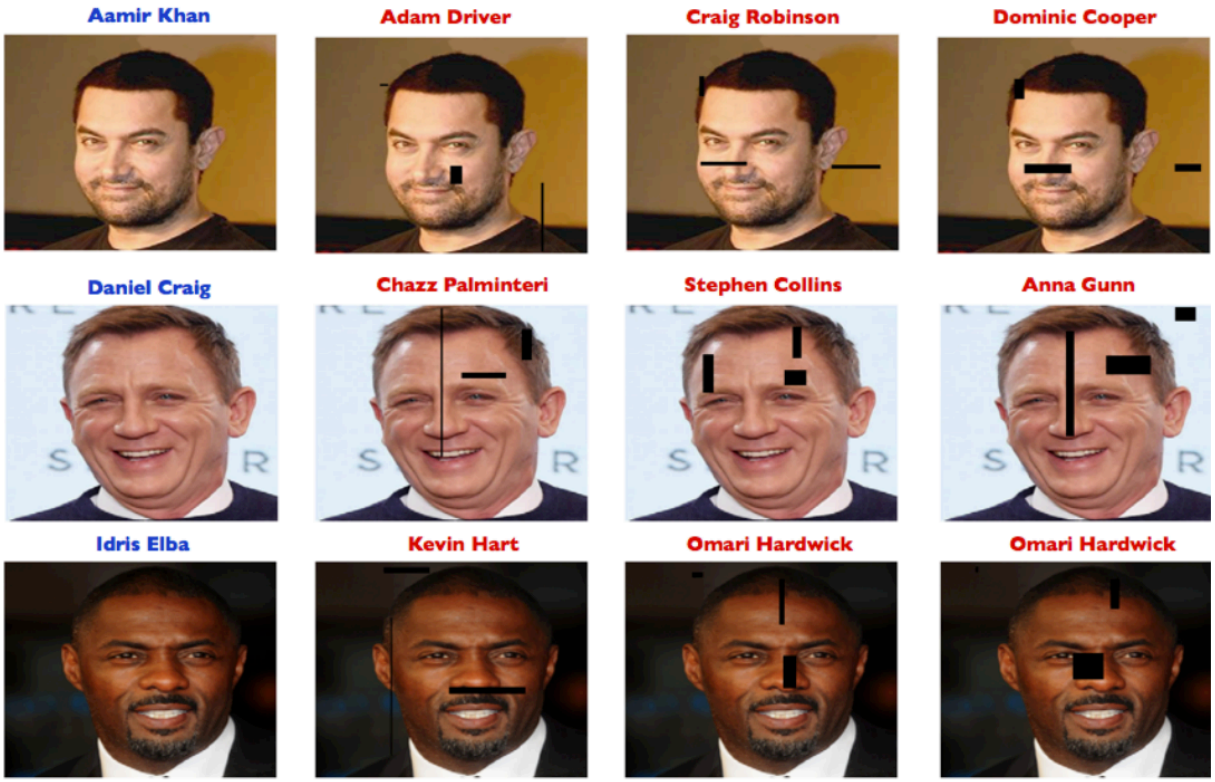

Figure 1.3: A simulation of the mask attack on the VGG [66] image classification network shown in [19]. The first column shows the input with its true class in blue. The other columns show inputs with added masks and predictions in red. As seen from the figure, setting values inside small patches to zero causes the classifier to make incorrect predictions. Note that an incorrect prediction is made even though a large portion of the input remains visible.

Compositionality [23] [4] is the ability of human beings to represent and perceive entities as hierarchies of parts, with the parts themselves being meaningful entities that are reusable in the form of valid combinations. The compositionality property has been shown to be beneficial in tackling several high level tasks. Neuroscientific research shows that human cognition uses compositionality for object classification and allows it to detect severely occluded objects with high precision. In the context of computer vision, compositional models recursively represent object parts and make decisions about object parts (in a hierarchical

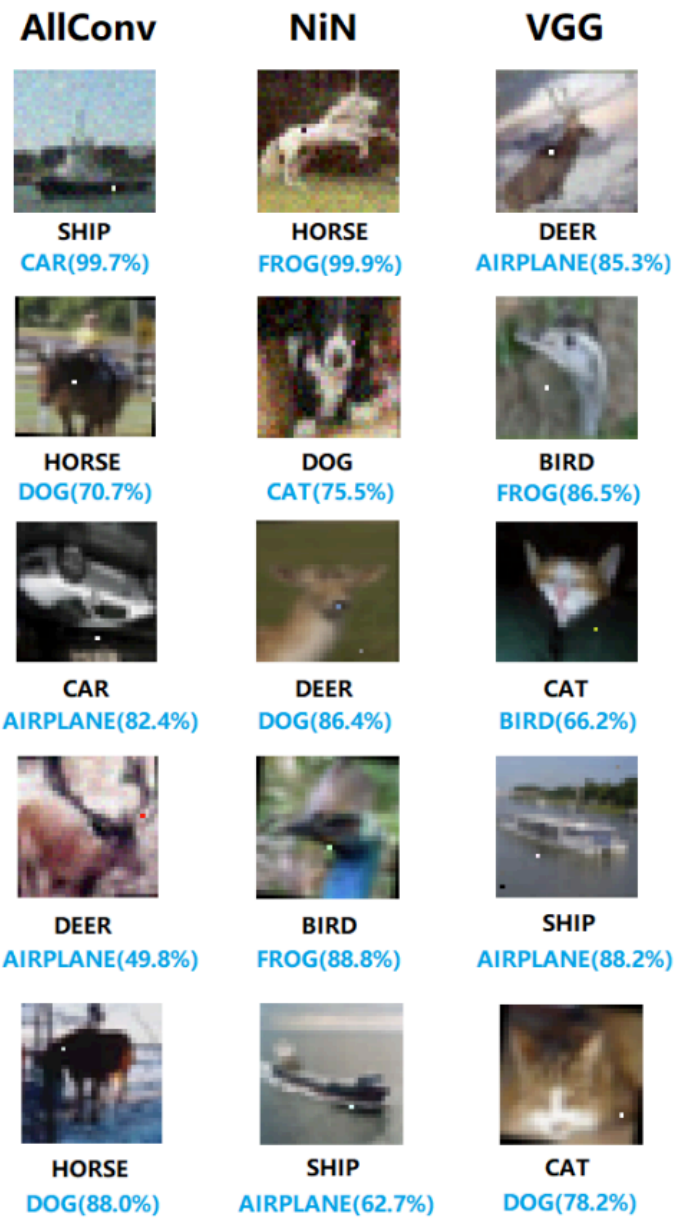

Figure 1.4: One pixel attacks created by [68] to fool DNN classifiers. Results of three networks AllConv, NiN and VGG are shown on images from the CIFAR-10 dataset. *Black* denotes the true class while *blue* denotes the predictions after modifying one pixel of the input.

manner) which contribute to the final result. Several works [24] [37] [77] [88] in computer vision that use compositional models have also shown that they can perform image classification well in the presence of occlusion. By distributing responsibility of classifying individual

parts across the spatial dimension, compositional models can detect occluded entities by recognizing learned patterns that belong to the corresponding category. This strategy can also be used to defend against adversarial attacks as objects parts are now being modelled as opposed to learning just one model for the whole category.

In this work, we introduce a novel procedure to perform *foreground* object classification that is relatively robust to occlusion and other variations in the input. Unlike the brittle understanding that vanilla CNNs hold, our approach to classification results in a stronger classifier. We perform classification using compositional models by first encoding parts of the input into high dimensional feature vectors. These spatially localized features are then fed to the compositional model which consists of several mixture models that correspond to patterns or object parts. We then obtain the likelihood of feature vectors and then combine these scores to make final class predictions for an input image. As parts of the image are fed separately into the compositional model, the responsibility of making final class predictions is split in the spatial dimension and reduces incorrect predictions caused by perturbations in some parts of the image.

We test our new classification strategy on the novel panoptic segmentation task, which is focused on complete scene understanding. We integrate our classification mechanism into UPSNet [81], an adaptation of the Mask R-CNN [30] framework for the panoptic segmentation task. We choose to adopt [81] to test our approach as the design is *top down*, highly modular and delivers competent performance on panoptic segmentation. The design starts with a feature extraction backbone which is made up of the popular ResNet [29] architecture, followed by a FPN module to generate multi-scale features. The multi-scale feature maps are shared by two task specific heads which perform instance and semantic segmentation. The semantic head is a series of deformable convolution layers that produces class predictions for each pixel in the input. The instance head consists of three parallel branches for

classification, bounding box regression and segmentation mask prediction. We introduce a novel classification pipeline in the instance head that is based on compositional modelling. The outputs of the semantic and instance heads are then combined in the final parameter free panoptic head, to produce a single pixel-wise segmentation output. To improve instance mask segmentation predictions, we also introduce an attention mechanism which helps in differentiating instances that are closely packed together and share a boundary. We test the performance of our approach on two popular datasets for panoptic segmentation: COCO [48], and Cityscapes [11] and show qualitative and quantitative improvements on the panoptic segmentation task.

1.2 Main Contributions

In this work, we propose a novel object classification method that uses compositional models to perform part based detection. The method first looks at individual parts of an object in the scene and then makes a decision about the object's identity. We test the proposed method in the context of the newly introduced panoptic segmentation task. The panoptic segmentation task combines both semantic and instance segmentation to perform labelling of the entire image. The main contributions of this work are:

- Introduce a novel compositional model based object classification approach that is more robust to occlusion and shows promising results on complex scenes.
- Develop a new training strategy for object classification that uses ground truth mask information to perform part based detection.
- Discover an issue with fully convolutional segmentation mask prediction of Mask R-CNN and propose the Subject Attention Module that learns to correctly identify pixels

of *foreground* objects.

- Discuss an issue with instance segmentation datasets that highlights a discrepancy between qualitative results and quantitative metrics.

1.3 Thesis Organization

The rest of the thesis is organized in the following manner:

- Chapter 2 sheds more light on segmentation tasks and their evolution. This chapter also talks about related work done on segmentation and addressing the occlusion problem.
- Chapter 3 discusses the proposed design and its modules. Some novel training ideas and issues with current algorithms are also briefly covered.
- Chapter 4 presents implementation details and results of extensive experiments carried out to evaluate the proposed algorithm.
- Finally, Chapter 5 presents a summary and introduces future research possibilities.

Chapter 2

Review of Literature

2.1 Image Segmentation

Image segmentation is the task of dividing an image into regions or groups of pixels. It has become a popular task in the field of computer vision because of a wide range of applications, from autonomous vehicles and robotics to medical imaging and diagnosis. The task of segmentation has been studied for a long time in the field of computer vision and image processing. But almost all works on segmentation have focused on either semantic classes, which are uncountable such as the sky, water bodies or vegetation, or instance categories, such as people, motor vehicles or furniture. Due to this trend, advances made in recent years have focused on only one of the categories, but not both. Using separate algorithms to segment semantic and instance classes and unifying them for achieving complete segmentation would not be prudent as these algorithms are often computationally heavy.

2.1.1 Semantic Segmentation

Semantic segmentation involves the study of *stuff* classes such as the sky, water bodies or vegetation. These classes are usually uncountable and amorphous. This task is defined as assigning a class label to each pixel in the image. Before the advent of deep learning based algorithms, semantic segmentation was performed using thresholding, edge based par-

titioning, region based methods (region growing, region splitting or merging) and watershed algorithms. Clustering based techniques were also a popular choice.

The first standout work on semantic segmentation using deep learning was Fully Convolutional Networks (FCN) [54]. FCNs use a series of convolutional layers in an encoder-decoder style architecture. The decoder section uses transposed convolutions, which are essentially learnable layers to upsample feature maps. The PSPNet [90] model showed the importance of using multi-scale features for semantic segmentation and proposed the pyramid pooling module to learn feature representations at different scales. Yu and Koltun [86] use atrous convolutions that allow for larger receptive fields without increasing model size, also use multi scale features to improve segmentation performance. Valada et al. [74] also use atrous convolutions combined with multi-scale residual units to learn multi-scale features. DeepLab [5] propose Atrous Spatial Pyramid Pooling (ASPP) module that concatenates feature maps using parallel atrous convolutions with different dilation rates and a global pooling layer. Even though ASPP substantially gives a performance boost by capturing long-range context information, it is accompanied by increased computational complexity. This can be overcome by using Dense Prediction Cells (DPC) [6] and Efficient Atrous Spatial Pyramid Pooling (eASPP) [75] that show improved performance than ASPP while being approximately ten times more efficient. Recently, Li et al. [45] suggest that global feature aggregation, though important for scene parsing tasks, often leads to over-smooth regions, and local information containing boundaries or small objects is often lost, leading to sub-optimal performance. To overcome this problem, the authors propose an end-to-end global aggregation coupled with local distribution module which preserves local information and yields a balanced output.

2.1.2 Instance Segmentation

Some early approaches made use of conditional random fields (CRF) [31], Markov random fields with CNNs [89], and recurrent neural networks [63] [61]. Fast forward to today, a majority of the CNN based approaches can be categorized into two broad architectural patterns: *top down*, which predict instance masks for region proposals, produced by state-of-the-art detectors and *bottom up*, which aggregate pixel-level predictions produced by the segmentation module to form the instance level predictions.

Bottom up approaches often employ FCNs to obtain a rich feature representation and then apply transformations to generate instance predictions. Dai et al. [12] use local coherence for predicting instance affinity while Uhrig et al. [73] encode the direction which points to the instance center for each pixel. SSAP [21] uses pixel pair affinity pyramids for computing the probability that two pixels belong to the same instance. Bai and Urtasun [3] use convolutional layers to produce feature maps which represent energy levels of the image and then make cuts at a single energy level to obtain instance masks. Despite introducing innovative solutions to predict instance affinity, *bottom up* approaches have lost popularity compared to *top down* methods due to lower performance scores. In proposal based methods, Hariharan et al. [27] propose a method that uses Multiscale Combinatorial Grouping [2] proposals as input to CNNs for feature extraction and then employ a Support Vector Machine (SVM) for region classification. Subsequently, Hariharan et al. [28] propose hypercolumn pixel descriptors for simultaneous detection and segmentation.

DeepMask [58] feeds patches sampled from an input image to a CNN to produce class-agnostic segmentation masks and likelihood of objects present in the patch. Dai et al. [13] use separate networks for generating box proposals, mask prediction and object classification. FCIS [47] performs joint detection and segmentation on position sensitive score maps

generated by classification of pixels on the basis of relative locations.

Mask R-CNN [30] is one of the most popular and well performing *top down* instance segmentation approaches. It extends the Faster R-CNN architecture to instance segmentation by adding a segmentation mask prediction branch parallel to the classification and bounding box regression branch. Some works that improve upon Mask R-CNN are Mask Scoring R-CNN [34] which relates mask quality with mask score, and Path Aggregation Network for Instance Segmentation [52] that adds bottom-up path augmentation to enhance the object localization ability in earlier layers of the network.

2.2 Panoptic Segmentation

Image segmentation is a computer vision task which has myriad applications, and therefore, has been studied since the early days of the field itself. In the beginning, study of countable objects in images received a majority of attention with little or almost no focus on studying unstructured regions in the image, such as the sky, vegetation or terrain. After the work of Adelson [1], a new segmentation subproblem called semantic segmentation, which focused on identifying these amorphous regions, began to garner attention. Semantic segmentation is defined simply as assigning a class label to each pixel in the input image. In contrast, instance segmentation involves identifying each countable object in the input image and predicting a segmentation mask for it, to label the pixels that belong to a particular instance. The advent of deep learning has catapulted progress towards solving these segmentation problems. However, even though these tasks are inherently related, significant differences in performance measurement metrics, datasets, and viewpoints have created a divide and caused a rift in the methods developed for these tasks. Approaches to semantic segmentation use cascaded fully convolutional layers in the form of encoder - decoder architectures, while

instance segmentation methods typically use object proposals for localization, followed by task specific heads for classification, regression and mask prediction. But many segmentation applications require reliable performance on inputs sampled from the real world, which, almost always contains a mix of both *thing* (instance) and *stuff* (semantic) regions in the scene. The need to segment both sets of categories jointly calls for the development of algorithms which reliably solve this unified task and yield coherent scene segmentations. Works [71] [72] [85] done before the siege of deep learning called scene parsing, image parsing and holistic scene understanding were essentially identical to this unified task.

The term Panoptic Segmentation was introduced by Kirillov et al. in [36]. Panoptic Segmentation combines instance and semantic segmentation to achieve a single output, which contains class labels and an instance ID for each pixel. The instance IDs for *stuff* classes are ignored, while the IDs for *thing* classes are used to reason about which instance of a *thing* class the pixel belongs to. Citing the use of disjoint metrics as a reason for separate study of instance and semantic segmentation, the authors introduce a new task metric called panoptic quality which measures the performance for both semantic and instance classes in an uniform manner. Since the inception of the novel panoptic segmentation task, almost all of the algorithms that have been proposed can be categorized into two broad groups based on their topology. These groups are *top down* and *bottom up*. Figure 2.1 shows an illustration that compares the two topologies.

2.2.1 Top Down

The *top down* category of algorithms first predict bounding boxes around possible objects and then generate a segmentation mask to associate each pixel that lies within the mask with its corresponding class score. As inferred from quantitative performance, the *top down* ap-

proaches achieve better performance than *bottom up* strategies for the panoptic segmentation task. Many approaches [46] [59] [41] [83] of this type use [30] as a foundation and propose novel additions to improve segmentation performance. Together with the introduction of the new task, Kirillov et al. also propose a strong baseline in the form of Panoptic FPN [35] that is formed by adding a semantic segmentation branch to Mask R-CNN [30]. UPSNet [81] uses a parameter-free panoptic head to resolve conflicts between *thing* and *stuff* predictions by also predicting an extra "unknown" category. Li et al. propose TASCNet [43] which uses the Panoptic FPN design and adds a *things* vs. *stuff* mask which joins the instance and semantic heads to maintain consistency between *foreground* and *background* predictions. AUNet [46] also join the two heads by adding attention mechanisms to produce coherent outputs and minimize conflict between instance and *stuff* predictions. Petrovai and Nedevschi [57] propose a panoptic segmentation network for automated driving that combines the benefits of PSPNet for semantic segmentation and Mask R-CNN for instance segmentation. Liu et al. [51] propose the Spatial Ranking module to resolve conflicts between overlapping instance masks.

2.2.2 Bottom up

The *bottom up* methods make class predictions for each pixel before using grouping strategies to detect and localize instances as shown in Figure 2.1. These approaches can be seen as extensions of popular fully-convolutional methods for semantic segmentation and sometimes offer faster runtime than *top down* algorithms.

Panoptic DeepLab [10] is a bottom-up approach which borrows heavily from the DeepLab segmentation architecture. An encoder backbone consisting of atrous convolutions in the last layers is used for feature extraction. This is followed by two instance and semantic de-

coders with ASPP modules. The semantic segmentation branch is similar to the DeepLabV3 [7] design. The instance segmentation masks are predicted using center of mass of detections. The final pixel assignment uses a voting mechanism similar to the one presented in DeeperLab [82]. AdaptIS [67] first performs semantic segmentation by generating pixel-wise predictions and then uses point proposals to generate instance masks. Li and Arnab [44] use weak supervision in the form of bounding-boxes and image-level tags to produce non-overlapping instance masks while performing semantic segmentation. DeeperLab [82] also uses an encoder-decoder architecture followed by an instance head that adopts a keypoint based representation for instances. The instance predictions are made using four parallel heads with one head producing keypoint heatmaps and small, medium and long range offsets predicted by the others.

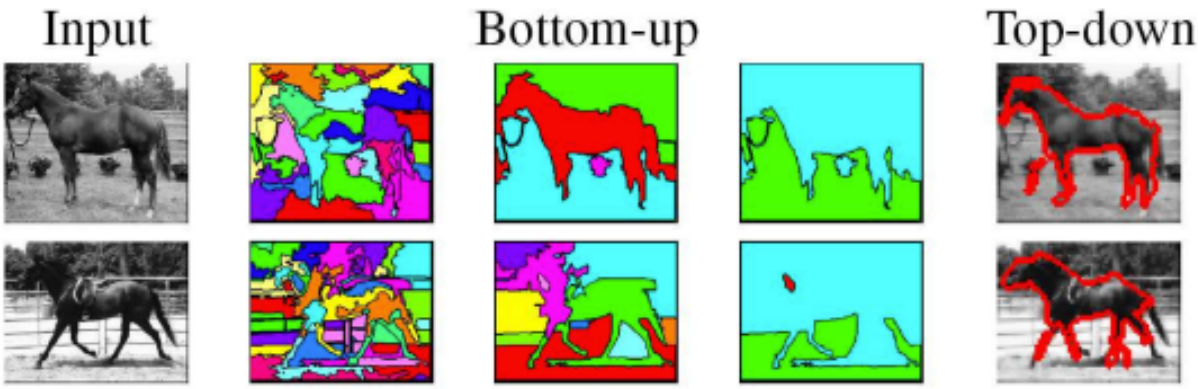

Figure 2.1: An example from [42] showing the difference between *top down* and *bottom up* segmentation approaches. *Bottom up* proceeds by using a hierarchical grouping strategy as seen in columns two and three. *Top down* first identifies an object and then determines which pixels belong to the object.

FPSNet [16] prioritizes runtime and attempts to perform both semantic and instance segmentation as a fully convolutional model by predicting a class score and an instance ID for each pixel. For training instance mask predictions, ground truth masks are converted into soft attention masks. The design also uses RetinaNet as the backbone as it is much

faster than other heavy backbones. Hou and Li [33] is another single shot fully convolutional panoptic segmentation strategy. Hou and Li also focus on runtime and argue that the "detect" phase of proposal based methods is slow and discards a lot of useful information. They make use of proposals discarded by NMS to recover instance masks directly.

2.3 Occlusion Handling

Occlusion occurs when 3D objects that are separated along the axis of the viewpoint, are projected onto the same points in the 2D image plane. Objects closer to the image plane partially or completely block the objects located at a farther distance. The problem of occlusion has been extensively studied in the field of computer vision [78] [18] [69] [32]. Although the task of scene understanding has witnessed significant progress in recent years, the problem of occlusion still persists in scene parsing systems and continues to hurt performance of image analysis tasks.

For dealing with occlusion in segmentation problems, Scene Parsing with Object Instances and Occlusion Ordering [71] first obtain pixel wise labels and a set of candidate object instances for hundreds of object classes. Overlapping regions are then used to obtain an occlusion ordering using graph theory to achieve an output which explains the image well. Finally, the method alternates between using the instance predictions to refine the pixel labels and vice versa, till neither of them changes. Multi-instance Object Segmentation with Occlusion Handling [9] collects segmentation masks and class scores for objects that are possibly occluded and formulates them into an energy minimization framework. Wang et al. [79] introduce repulsion loss which forces bounding box regressors to move towards the correct target while also being repelled by other nearby target proposals, which makes the detector more robust to densely populated scenes and often suffer from occlusion. Self-

Supervised Scene De-occlusion [87] takes a self supervised approach to recover hidden scene structures, specifically, object masks and RGB content by completing invisible parts of an occluded object. In this way, the method recovers a progressive ordering of objects present in the image. SOGNet: Scene Overlap Graph Network for Panoptic Segmentation [83] uses category, geometry and appearance features to form an embedding which is used to create a 2D matrix that explicitly encodes overlap relations between instance masks. Any overlap resolution is done through a differentiable weakly supervised overlap resolving module, with the help of supervision from per pixel instance ID classification in the panoptic head. The panoptic segmentation model proposed by Lazarow el al. [41] has an identical design to [81] with a ResNet FPN backbone followed by two task specific heads, Mask R-CNN for instance and FCNs for semantic tasks. An occlusion head is supplemented to explicitly deal with overlapping instance detections to resolve assignment of pixels claimed by multiple detections. For two overlapping mask detections m_i and m_j, the occlusion head determines which mask detection will claim the intersecting region $I(m_i, m_j)$ of pixels, essentially placing one mask on top of the other to settle the conflict. Some other works on dealing with the occlusion problem for segmentation include Semantic Amodal Segmentation [92], An End-to-End Network for Panoptic Segmentation [50] and UPSNet: A Unified Panoptic Segmentation Network [81].

2.4 Review

In this work, we introduce an approach to object classification that is based on compositional models. We substitute this classification pipeline into UPSNet [81], to test its performance on the panoptic segmentation task. The architecture is based on the Mask R-CNN [30] instance segmentation model with an added fully convolutional semantic segmentation branch. The

proposed architecture belongs to the *top down* category of algorithms. Through experiments on the COCO and Cityscapes datasets, we show how the compositional model performs on scenes with a high degree of complexity.

Chapter 3

Approach

In this chapter, we talk about our proposed approach to solve the panoptic segmentation task. We propose a modular deep learning architecture which consists of a shared backbone, followed by multiple task specific heads to compute results of subtasks. These results are then combined by the final merging stage to give a single output required for the panoptic segmentation task. Our architecture is based on the Mask R-CNN [30] architecture which gives strong results for the instance segmentation task and borrows some modules from the UPSNet [81] model, which performs well on the panoptic segmentation task. The proposed architecture belongs to the *top down* category of algorithms.

3.1 Model Architecture

The proposed architecture is shown in Figure 3.1. The design consists of a shared backbone, which acts as a feature extractor. This is followed by the instance and semantic segmentation heads, which take the multi-scale feature map generated by the backbone and perform segmentation for *thing* and *stuff* classes respectively. Two parallel branches for *foreground* object detection and mask segmentation make up the instance head. In this work, we replace the detect branch with a new design that is based on a compositional model. The final stage is the parameter free fusion head, identical to the one introduced in [81].

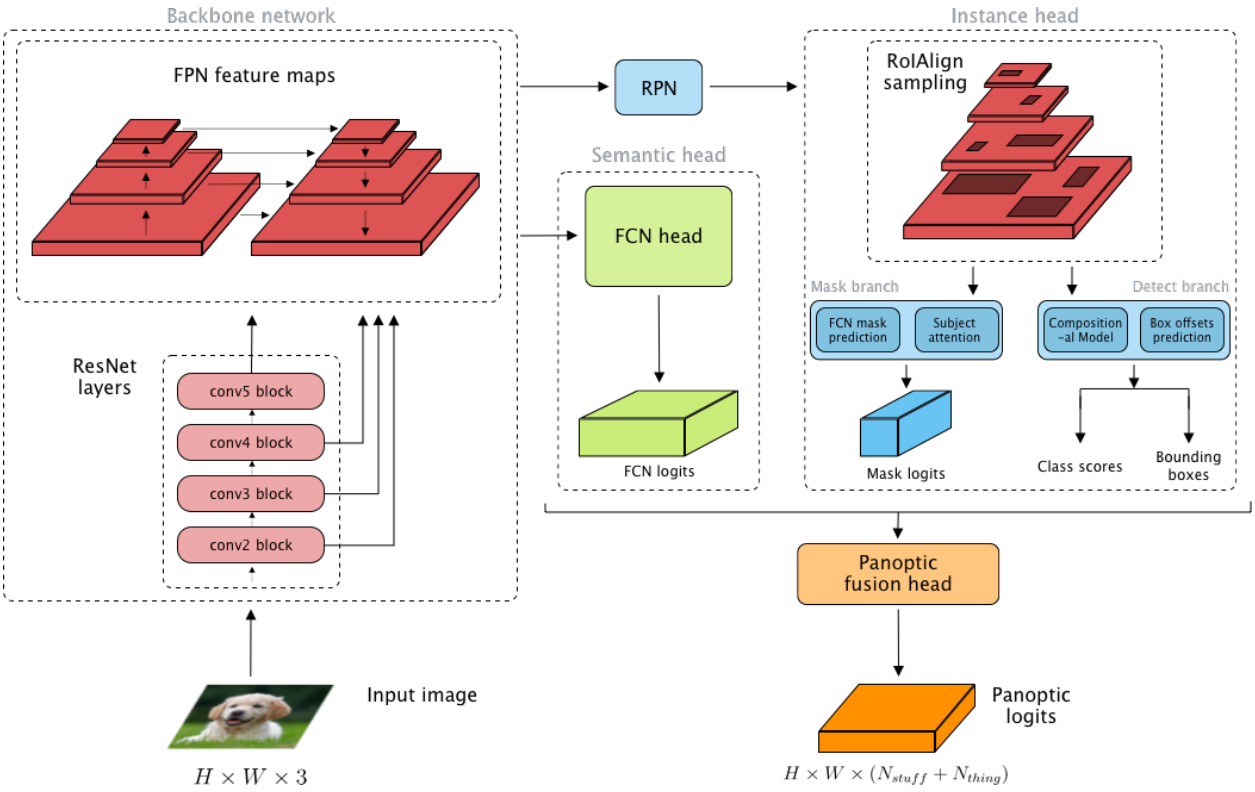

Figure 3.1: The high level architecture of our model. The backbone network first performs feature extraction using ResNet [29] layers and generates a pyramid of feature maps with FPN [49]. The feature maps are then used by two task specific heads: semantic head for *stuff* classes and instance head for *thing* classes. The results of the task heads are then combined by the fusion head to generate final panoptic logits.

3.1.1 Shared Backbone

The backbone network of our architecture is identical to [30], a combination of ResNet [29] and Feature Pyramid Network [49]. The ResNet architecture is made up of residual blocks which are groups of 1×1 and 3×3 convolutional layers. There are shortcut connections that bypass each block to merge input of a particular block with its output, which is then passed to the following block. These shortcut connections help gradient flow during training. The ResNet model has been widely adopted as a deep neural feature extractor due to its state-of-the-art performance on the image classification task of the ImageNet challenge. The final dense layers of the ResNet model are dropped and the output of the final residual block

is used as a feature map when ResNet is used as a feature extractor.

Image pyramids are often used in recognition tasks as they allow feature representation at multiple scales. The Feature Pyramid Network (FPN) [49] takes a standard feature extractor (such as ResNet [29]) and generates a pyramid of feature maps which yield superior performance than standard backbone designs as demonstrated by the authors in [49]. The ResNet architecture has four main stages which generate fine to coarse feature maps, with each stage being made up of multiple residual blocks. The feature maps generated by each stage have a stride of (4, 8, 16, 32) and are termed as $conv_2$, $conv_3$, $conv_4$ and $conv_5$, respectively. The lower level feature maps (e.g., $conv_2$), are rich in local feature information but contain very little semantic knowledge as they have been subsampled a smaller number of times. In contrast, higher level feature maps, which are also spatially smaller in size, have more semantic awareness. FPNs combine these feature maps using top-down and bottom-up pathways to yield multi-scale feature maps P_2, P_3, P_4 and P_5. Starting from the coarsest map $conv_5$, each feature map C_i is upsampled and then merged with a lower level feature map C_{i-1}, which undergoes 1×1 convolution to match the number of channels (bottom) before element-wise addition to get P_{i-1}. The merged map is then subjected to a 3×3 convolution to reduce the aliasing effect of upsampling.

3.1.2 Semantic Head

The semantic head is a fully convolutional subnetwork formed by a series of deformable convolution layers [14] that takes the pyramid of feature maps (P_2, P_3, P_4, P_5) generated by ResNet FPN backbone and predicts pixel-wise class scores. The architecture of the semantic head is shown in Figure 3.2. Since the primary focus of this work was to improve instance segmentation under occlusion, the semantic head is kept identical to [81]. The semantic

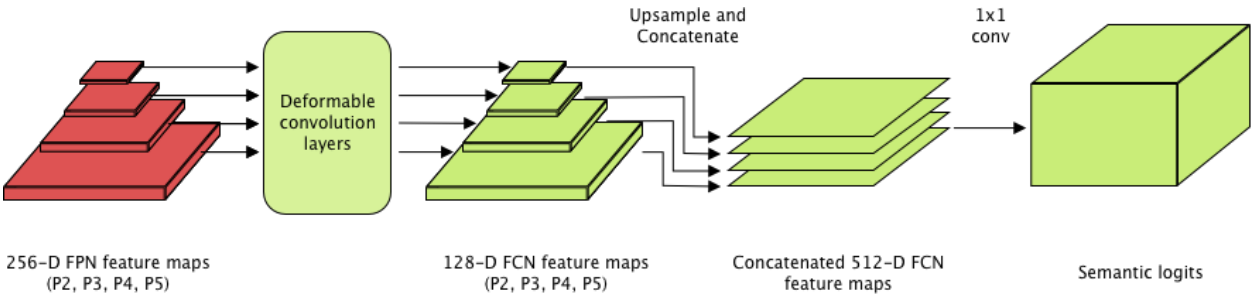

Figure 3.2: The architecture of the semantic head. The semantic head takes the pyramid of feature maps as input and applies deformable convolutions [14] on each feature map separately. The feature maps are then brought to the same dimensions and combined to get semantic logits.

head performs segmentation for all *thing* and *stuff* classes. The predictions for *thing* classes may be used to improve instance segmentation results as well. The semantic head makes use of deformable convolution layers which have the ability to learn transformations for sampling locations of convolutional layers as shown in Figure 3.3. As a result, the convolution operation is not restricted to standard shape and can act upon a local region of the feature map.

The deformable subnet consists of one $3 \times 3 \times 256$ followed by two $3 \times 3 \times 128$ convolution layers. Each of the of the feature maps (P_2, P_3, P_4, P_5) which are (1/4, 1/8, 1/16, 1/32) the scale of the input image and have 256 channels are passed through this subnet individually. The feature maps are then upsampled to bring all maps to the same scale and concatenated. The final concatenated set of features is then subjected to a 1×1 conv, followed by softmax to get the final class-wise logits for each pixel. The output of the semantic head has a 1/4 scale and is therefore, upsampled one last time to match the spatial dimensions of the input.

The training strategy of the semantic head is identical to [81]. In addition to using pixel-wise cross-entropy loss for training the semantic head, an RoI loss is also added to help improve segmentation performance of instances by adding an extra penalization term to pixels that belong instances. The RoI loss is calculated by using the ground truth instance masks resized

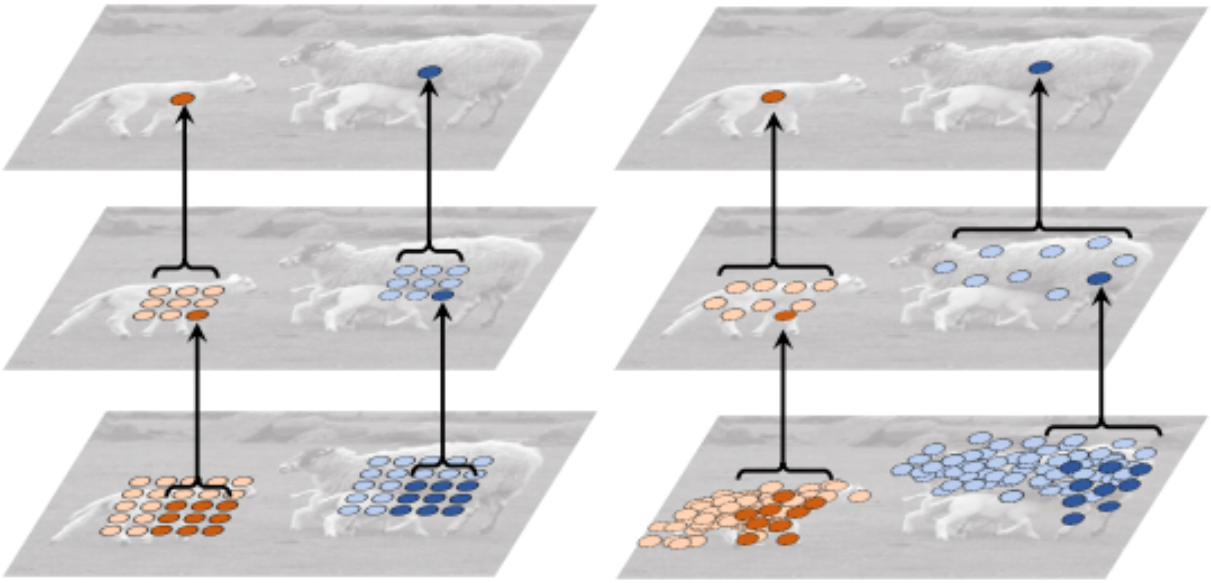

Figure 3.3: The working of Deformable Convolutional Networks (DCN) [14]. Unlike standard convolution layers *(left column)*, DCN layers *(right column)* are capable of learning sampling offsets to improve context of convolution operations. Figure taken from original paper.

to 28×28 and the corresponding patch of the final logit map. The loss is then the cross entropy between each of these pixels.

3.1.3 Instance Head

The instance head focuses on parts of the input image which correspond to objects that belong to *thing* classes. Bounding boxes that are likely to contain objects are termed as Regions of Interest (RoI). RoIs are first identified by the Region Proposal Network module introduced in [62]. The instance head processes each RoI and predicts a class score, box offsets and a segmentation mask. In contrast to the semantic head which performs segmentation for both *stuff* and *thing* classes, the instance head only deals only with *thing* classes.

The instance head consists of three parallel branches which are responsible for perform-

ing classification, bounding box regression and mask segmentation. The classification and bounding box regression pipeline can be paired together to form the detection branch, while another branch (fully convolutional) is responsible for generating segmentation masks for each RoI. This branch can be termed as the mask branch. In this work, we replace the detection branch of R-CNN [30] with a novel design that is based on a compositional model to improve classification performance of occluded objects. Our novel detection branch is shown in Figure 3.4.

To begin with, each RoI is first used to extract feature map crops from the pyramid of feature maps generated by the shared backbone using RoIAlign sampling introduced in [30]. The RoIAlign function maps a RoI of arbitrary size and aspect ratio to feature map of constant dimensions. We denote this feature tensor extracted by RoIAlign as $F \in \mathbb{R}^{k \times k \times D}$, where D is the number of channels of the feature map generated by the *backbone*. Each position (i, j) on the 2D lattice of size $k \times k$ (shown in *orange* in Figure 3.4) represents a D dimensional feature vector $F^D_{i,j}$. We refer to this 2D lattice as *RoI lattice* in the discussions that follow. The generated feature map for each RoI is then used by the detect and mask branches.

Our detect branch is shown in Figure 3.4. As shown in the figure, we use the cropped feature map $F^D_{i,j}$ to make class and bounding box offsets predictions. To perform classification that is robust to occlusion and possesses spatial awareness, we introduce a compositional module in the classification pipeline. We start by looking at features generated at each position (i, j) on the *RoI lattice* and find a matching amongst a reference set of patterns. These references correspond to frequently occurring patterns on instances of *thing* classes in the dataset and are often characteristic to each individual class. For example, an instance of the person category can be identified by first looking for its associated parts, such as a head, a pair of hands and finally, components of the lower body. Rather than simply associating low level image patterns to classes, the compositional model encodes high dimensional abstract

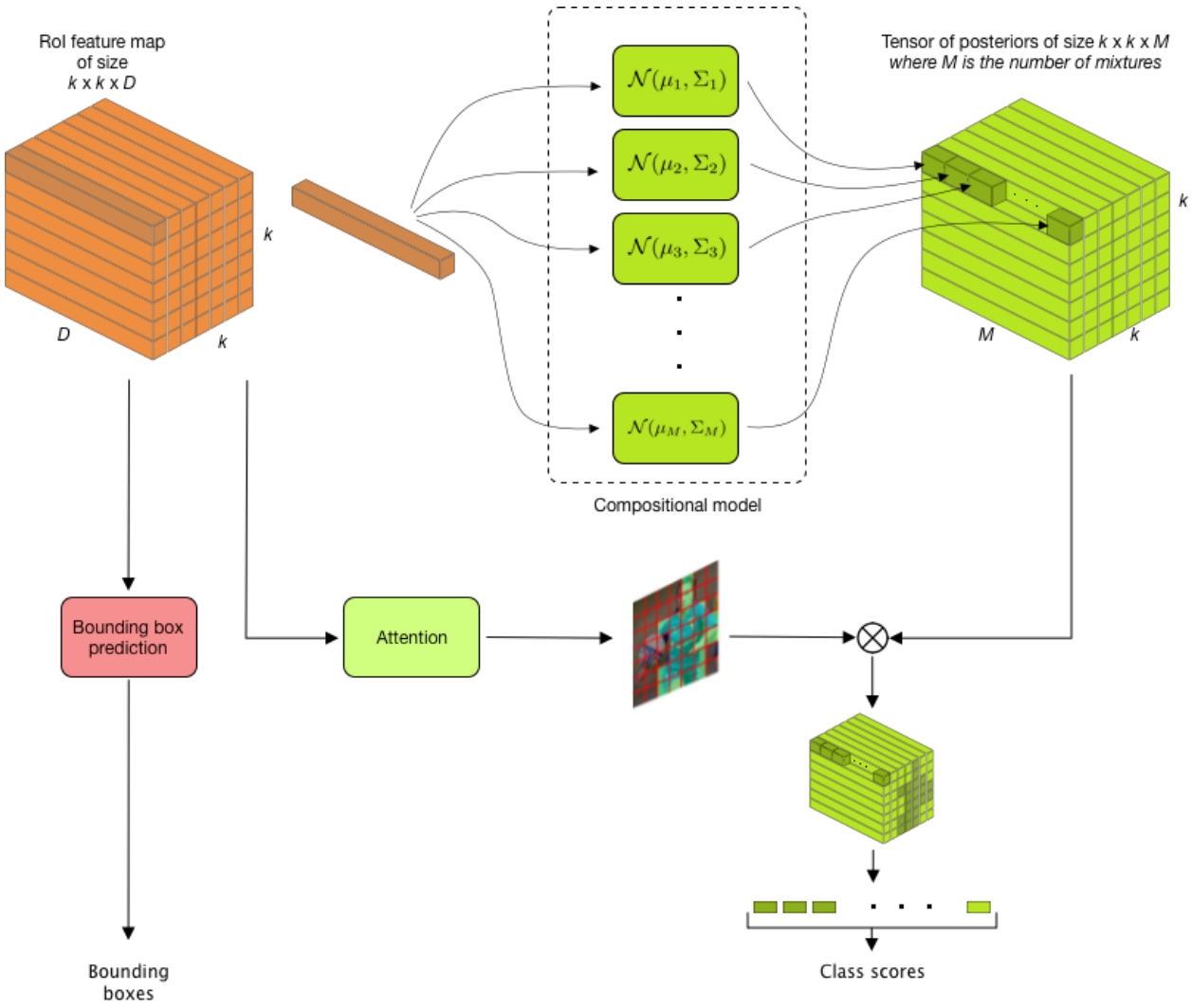

Figure 3.4: The novel compositional model used in the detect branch for instance segmentation. Features at each position on the $k \times k$ sized *RoI lattice* are processed by the compositional model to compute posterior values. The posterior values are then used to make final class score predictions.

feature vectors that correspond to parts present on instances of *thing* classes generated by the feature extractor backbone. Our compositional module consists of a mixture of multivariate Gaussian distributions. Using a mixture of Gaussians allows computation of soft assignments for each position on the *RoI lattice* and divides responsibility of classification amongst parts of the RoI. These assignments can then be combined to arrive at the final class prediction.

A mixture of multivariate Gaussians can be denoted as:

$$p(f_p^{(D)}) = \sum_{k=1}^{M} \pi_k \, \mathcal{N}(f_p^{(D)} | \mu_k, \Sigma_k) \tag{3.1}$$

where M is the number of components, each being D dimensional (superscript (D) represents dimensionality and not exponentiation), $f_p^{(D)}$ is the encoded feature vector, π_k represents the mixture co-efficient of the kth component and $\mathcal{N}(\mu_k, \Sigma_k)$ is the kth multivariate Gaussian distribution having mean μ_k and covariance Σ_k. As each component of the mixture is a multivariate Gaussian, the mean μ_k is a D dimensional vector and covariance Σ_k is a $D \times D$ matrix.

We can obtain another representation of $p(f_p^{(D)})$ (having M components) by introducing an M dimensional binary random variable z having a 1-of-M representation in which a particular element z_k (where $k = 1, 2, 3, ..., M$) is equal to 1 and all other elements are equal to 0. The values of z_k therefore satisfy $z_k \in \{0, 1\}$ and $\sum_k z_k = 1$. This means that there are M possible states for the variable z based on which element is non-zero in the vector. A mixture can then be defined as a combination of a marginal and a conditional distribution as follows:

$$\begin{aligned} p(f_p^{(D)}) &= \sum_z p(f_p^{(D)}, z) \\ &= \sum_z p(z) p(f_p^{(D)} | z) \end{aligned} \tag{3.2}$$

The marginal distribution $p(z)$ can be specified in terms of the mixture co-efficients as $p(z_k = 1) = \pi_k$. The mixture co-efficients must satisfy $0 \leq \pi_k \leq 1$ and $\sum_{k=1}^{M} \pi_k = 1$. The conditional distribution term $p(f_p^{(D)} | z)$ represents a multivariate Gaussian. The probability

density function of a multivariate Gaussian (with dimensionality D) is given as:

$$\mathcal{N}(f_p|\mu,\Sigma) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} exp\left(-\frac{1}{2}(f_p-\mu)^T\Sigma^{-1}(f_p-\mu)\right) \tag{3.3}$$

where f_p and μ are D dimensional vectors and Σ is the covariance matrix of size $D \times D$.

A mixture of multivariate Gaussians with M components has parameters π_M (mixture coefficients), μ_M (means) and Σ_M (covariances). To learn these parameters (π_M, μ_M and Σ_M), we collect features at locations where an instance of *thing* class is present in the image and subsequently use clustering. Upon convergence, the centers μ represent frequent occurring patterns that are characteristic to particular *thing* classes in the dataset. For simplicity, we treat all variables in the Gaussians to be independent, which reduces the covariance matrix to a diagonal matrix of variances $\Sigma_d = diag(\sigma_1^2, \sigma_2^2, \sigma_3^2, ..., \sigma_D^2)$ and simplifies the inverse Σ_d^{-1} to just inversion $\Sigma_d^{-1} = diag(\sigma_1^{-2}, \sigma_2^{-2}, \sigma_3^{-2}, ..., \sigma_D^{-2})$. Due to the independence assumption among variables, the computation also reduces to calculating a Gaussian PDF for each dimension independently and then taking their product to get the final result. So, the simplified calculation now takes the form:

$$\mathcal{N}(f_p|\mu,\Sigma) = \frac{1}{(2\pi)^{D/2}|\Sigma_d|^{1/2}} exp\left(-\frac{1}{2}\ ((f_p-\mu)\times(f_p-\mu))^T\ .\ \Sigma_d^{-1}\right) \tag{3.4}$$

where f_p, μ and Σ_d^{-1} are all D dimensional vectors and $\times$ represents element-wise multiplication.

Each position (i, j) (where $[i,\ j = 1,\ 2,\ 3,\ ...,\ k]$) on the $k \times k$ sized 2D lattice of the RoI defines a high dimensional feature vector that encodes the pattern contained in its receptive field. We start by first estimating $p(z|f_p)$ at each position by feeding these localized features to our compositional module to obtain likelihood scores at each position. This gives us

a M dimensional vector representing how active each of the M components is at every spatial position. We use hardware accelerated code to maintain fast inference times. The component that is most active at a particular position can be obtained by taking an *argmax* over all dimensions. The final mixture likelihoods are computed at each position by taking a dot product of the likelihoods and mixture co-efficients followed by normalization for each component k as follows:

$$p(z_k = 1|f_p) = \frac{\pi_k \, \mathcal{N}(f_p|\mu_k, \Sigma_k)}{\sum_{j=1}^{M} \pi_j \, \mathcal{N}(f_p|\mu_j, \Sigma_j)} \tag{3.5}$$

The object detection branch of the instance head is responsible for performing two tasks: classification and box regression. In the final step of the detection pipeline, we take the $k \times k \times M$ tensor of posterior values and predict the final class scores for each RoI. We apply a series of dense layers to the posterior values given by the compositional module. The compositional module allows us to process each spatial position (i, j) (where [i, j = 1, 2, 3, ..., k]) on the $k \times k$ sized 2D lattice separately and then use posterior values to make final class predictions for each RoI. As a result, the predictions are dependent on correctly identifying sub parts of the object, rather than using the entire feature map for performing classification. The RoIs generated by the Region Proposal Network are rough estimates and contain a high number of *background* boxes that have very low overlap with ground truth objects present in the image (as explained in [26] and [62]). Since the detection branch only deals with detecting *foreground* objects, assigning more importance to positions that contain parts of a *foreground* object helps improve classification. To predict the presence of a *foreground* object for each position, we add an attention mechanism parallel to the compositional model that assigns a score between 0 and 1 at each position on the 2D lattice of the feature map of size $k \times k$. The attention mechanism is fully connected layer followed by a sigmoid activation function to squash values between 0 and 1. It takes the feature map

$F_{i,j}^{D}$ as input and predicts object presence scores. To obtain the final prediction, the posterior values are combined with the attention scores and passed to the final dense layers for class predictions. Unlike Mask R-CNN [30], the bounding box offset prediction is separated from classification and gets its own parallel branch, formed using a couple of convolutional layers for channel adjustment followed by dense layers for prediction of final offsets of each RoI. The training strategy for the detection branch is discussed in 4.3.

The segmentation mask generation branch of the instance head is similar to the one introduced in [30]. The mask branch is fully convolutional and generates class agnostic binary masks for each RoI. Each RoI is used to predict a mask tensor of shape $k \times k \times C$ where k is the mask height and width, and C is the number of classes. The mask branch is made up of four 3×3 convolutional layers of size 256 with a stride and padding of 1 to maintain spatial layout, followed by a 2×2 transpose convolution with stride 2 to increase spatial dimensions. A final 1×1 convolution is applied to bring the channels to C to produce $k \times k$ sized binary masks for each class. The mask which corresponds to the predicted class for a RoI then becomes the final mask for that RoI. Since the mask picking strategy is based on the predicted class, conflicts may arise if more than one instance of the same class are present in a single RoI as explained in 3.1.5. To resolve such conflicts, we test an additional mechanism to resolve intra-RoI conflicts as discussed in 3.1.5.

3.1.4 Panoptic Fusion Head

The outputs of the semantic and instance heads are combined to form a single output in the panoptic head. The goal of panoptic segmentation is to generate an output image of the same spatial size as the input, with each pixel assigned a class label and an instance ID. The panoptic head is the final stage of the design that aggregates the outputs of the preceding

heads and produces a vector of logits per pixel. The architecture of the panoptic fusion head is shown in Figure 3.5.

The logits generated by the semantic head are denoted by X. These logits have the same spatial dimensions as that of the input image. We denote the height, width and channels of the semantic head output as H, W and C respectively. Because the semantic head performs segmentation for both *stuff* and *thing* classes, C can be broken down into $(C_{stuff} + C_{thing})$. Thus the semantic head logits are made up of two tensors X_{stuff} and X_{thing} having the same spatial dimensions H and W, but different channel sizes of C_{stuff} and C_{thing} respectively.

The panoptic head produces a logit tensor L with dimension $H \times W \times (N_{stuff} + N_{inst})$. The first N_{stuff} channels are taken from X_{stuff} produced by the semantic head. The calculation of instance logits needs additional computation. For every instance prediction i ($i = 1, 2, 3, \ldots, N_{inst}$), the instance head produces a mask M_i of size 28×28, bounding box B_i (as coordinates of the top left and bottom right vertices), and a class ID C_i. Another representation of the same instance can be generated from the semantic head logits for *thing* classes X_{thing}. This representation can be obtained by cropping the channel corresponding to class C_i of X_{thing} using bounding box coordinates B_i. The mask crop can then be pasted on an empty image, so that the mask representation S_i^M (for instance i from the semantic head; S denotes semantic head) will be of the input image size $H \times W \times 1$ with non-zero values only present only inside the corresponding bounding box B_i.

The 28×28 mask logits M_i generated by the instance head is also scaled to the same spatial dimensions B_i using bi-linear interpolation and placed on another empty image, to match the position of the mask S_i^M to obtain a representation I_i^M (for instance i from the instance head; I denotes instance head). The final representation for instance i in L is calculated by summing logits of the instance and semantic heads. $L_{N_{stuff}+i} = I_i^M + S_i^M$. This procedure is followed for all instance predictions to fill L.

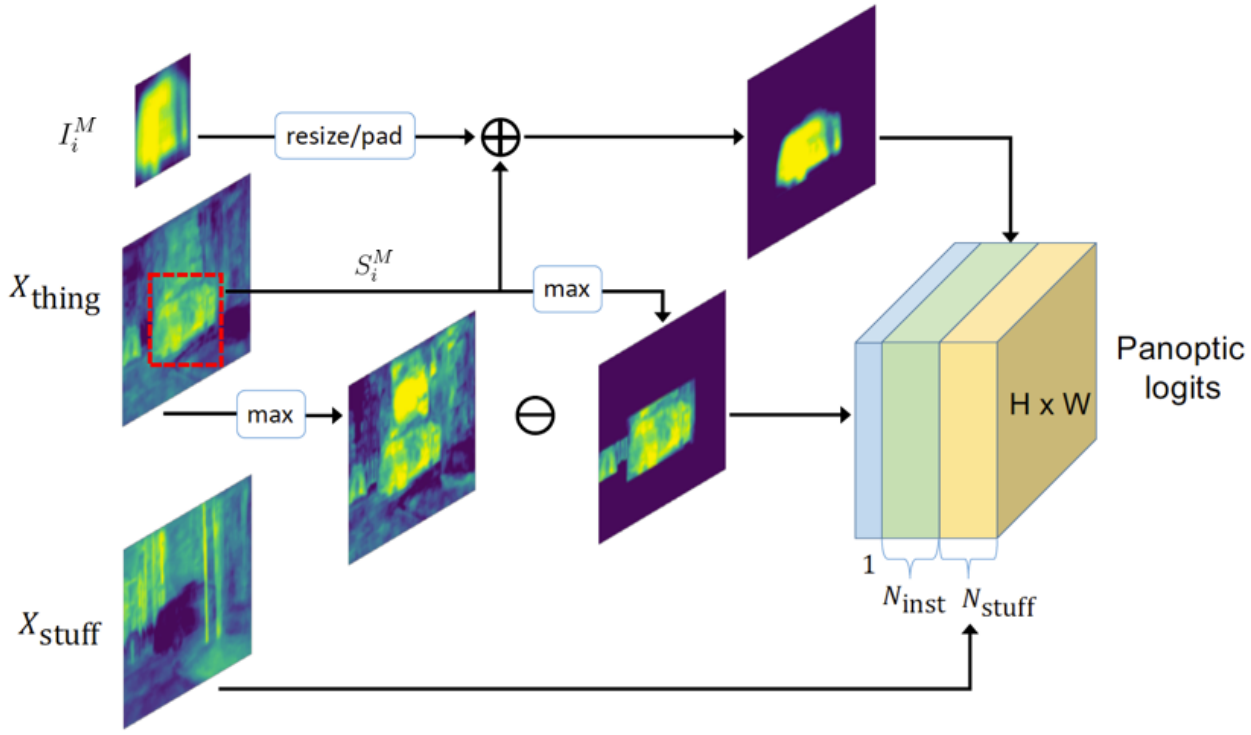

Figure 3.5: The panoptic head as shown in [81]. X_{stuff} and X_{thing} are the logits for *stuff* and *thing* classes generated by the FCN head in Figure 3.1 respectively. I_i^M are the logits for the ith instance mask generated by the instance head (mask logits in Figure 3.1) and S_i^M are the logits for the same mask cropped from the semantic logits for its class. H and W are the height and width of the input image and channels C of the output are calculated as $C = N_{stuff} + N_{inst} + 1$. The extra $(+1)$ channel in the final output is added for the *unknown* category. The symbols $\oplus$ and $\ominus$ in the figure represent element-wise addition and subtraction of logits, respectively.

The panoptic output is obtained by applying a softmax function along the channel dimension. If the *argmax* across channels lies in the N_{stuff} channels, the pixel belongs to a *stuff* class, otherwise to one of the instances. The class assignment for each instance is calculated by the strategy explained in [81].

To reduce the risk of making incorrect predictions, [81] also predict an additional category called *unknown*. The justification for adding this *unknown* class is that any pixel that receives an incorrect prediction will only contribute to a *false negative* of one class and

not also contribute to a *false positive* of some other class in the calculation of the panoptic segmentation quality metric. The logits for this *unknown* category are calculated as: $L_{unknown} = max_{channels}(X_{thing}) - max_{channels}(L_{inst})$, where X_{thing} is taken from semantic head logits while L_{inst} are all the instance logits (with channel size N_{inst}) in the final panoptic logit tensor L. We can understand the implications of introducing an unknown class by looking at its definition. In the case that some instance is missed, the term $max_{channels}(L_{inst})$ will be 0, or very close to 0. But if the semantic head predicts an instance with high confidence present in the same position, it is likely that the instance has been missed by the algorithm. In such cases, the *unknown* logits will take on high values, and, as a result, such pixels will be excluded when panoptic segmentation quality is measured. The method for generating ground truth data for training the *unknown* class is explained in [81].

3.1.5 Subject Attention Module

Achieving accurate segmentation of natural scenes that are often dynamic and suffer from occlusion is a non-trivial task. In natural scenes, it is likely that the region proposal may have more than one *foreground* object in the bounding box. To understand and overcome issues related to instance segmentation, we propose a scheme to classify region proposals (inputs to the instance head) to elicit understanding of issues, and, as a result, leading to improved segmentation head design. The task of panoptic segmentation deals with two broad categories of segmentation classes, *thing* corresponding to countable objects in a scene, and *stuff*, which include abstract classes such as the sky, water, road, wall, etc. We first categorize proposals into two broad groups: 1. *PURE*, region proposals that just have one instance of some *thing* class present with no interference such as occlusion or noise. We observe that such proposals are favorable for the segmentation mask prediction branch in [30] and yield good segmentation masks. 2. *IMPURE*, proposals of this type contain more than one

instance of *thing* classes. Presence of multiple instances also introduces ambiguity as multiple objects may be included in the segmentation mask. These proposals can be classified into two subtypes: A. *Type I*, proposals having multiple instances of the same *thing* category, B. *Type II*: proposals with multiple instances of two different categories of the *thing* classes. We note that proposals can be of both Type I and Type II simultaneously. Figure 3.6 shows a few examples of *IMPURE* proposals.

We introduce a Subject Attention Module (SAM), which aims to improve instance segmentation performance by reasoning about which locations contain the *subject* within a region proposal (also called RoI). In photography, the *subject* is the focus of the image, both as the sharpest point in the photograph and in a more figurative sense. For segmentation, a *subject* can be thought of as the object which the proposal was intended to detect. SAM is shown as a part of the Mask branch in Figure 3.1. The goal of SAM then becomes to identify the subject or the primary target of the region proposal. The architecture of SAM is similar to the mask branch of Mask R-CNN. But instead of generating segmentation masks for each class, SAM generates logits for each location of the binary mask, predicting how likely a location is to be a part of the subject. It consists of two fully convolutional layers followed by a dense layer. The last layer of SAM gives logits with the same spatial dimensions of the mask. Each location gives a weighting factor which is used to adjust the importance of locations where the subject is present. The final instance segmentation mask is obtained by combining the predicted mask logits with the weights generated by SAM.

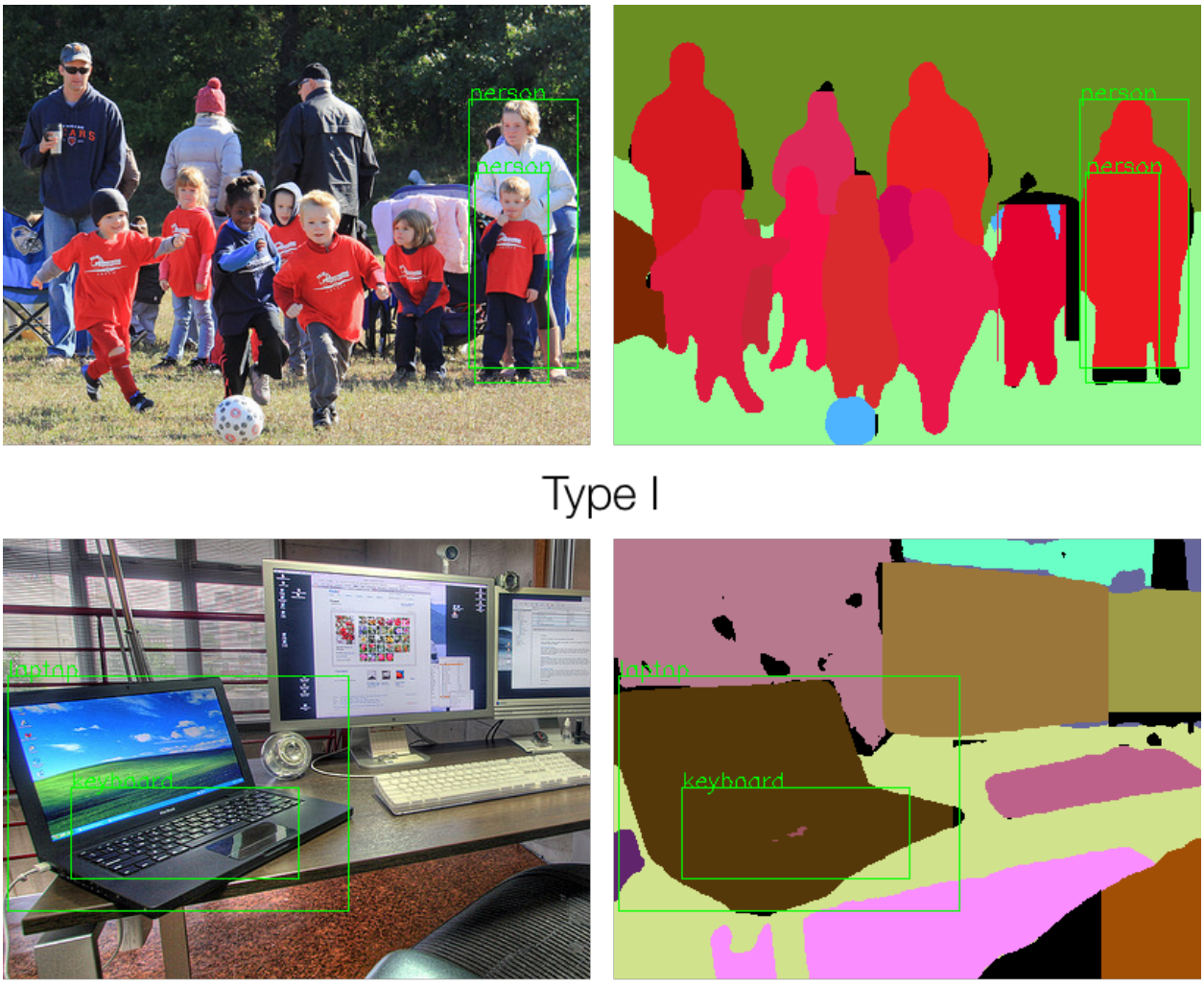

Figure 3.6: Examples of Type I and Type II *Impure* proposals. The first column shows the input with a few detected instances and their predicted classes annotated using *green boxes* (only instances being discussed are shown). The second column shows the predicted masks for the same detections. In the first example, the child instance is missed as the mask of another instance of the same class claims pixels that belong to the child instance. The second row shows an example of Type II where pixels of an instance of the *keyboard* class are claimed by the mask of another instance of the *laptop* class. The *keyboard* object is missed as panoptic segmentation requires a predicted segment to have an IoU of greater than 0.5 with a ground truth segment.

3.2 Implementation Details

3.2.1 Model Training

We implement our approach to panoptic segmentation in PyTorch [55]. The model is trained with up to 16 GPUs using the distributed training framework Horovod [65]. Each GPU is assigned a single image during training, therefore the minibatch size depends on the number of GPUs in use for training. Since the architecture is based on [81] and [30], we follow many of the settings used in those works.

The panoptic segmentation architecture is trained with 8 losses in total: panoptic head loss (pixel wise cross entropy loss for the unified panoptic output), semantic head loss (pixel wise cross entropy loss and RoI loss) and finally, instance head loss (5 losses for box classification, *foreground* attention, box regression, mask segmentation and subject attention). Each loss has a weighting factor associated with it to maintain balance and facilitate optimal learning of parameters.

In the inference phase, the first step of the instance head is to get the class scores and box offsets from the detection branch. The RoIs with class scores and box offset predictions are then filtered to only retain RoIs that have confidence scores greater than 0.6, followed by non-maximum suppression to reject duplicate predictions. The RoIs that remain are then fed to the mask segmentation branch that generates binary masks for each RoI and subject attention predictions. To combine mask predictions, a pruning process is used. First, RoIs (now with masks added) are sorted in the decreasing order of confidence. Each mask is then interpolated to the image scale and place onto an empty canvas. In the final output, there will be one canvas for each *foreground* class that has the same spatial size of the input image. If any mask happens to have an overlap greater than 0.3 with another that was placed earlier,

the mask being processed is discarded. Otherwise, the non-overlapping portion of the mask is copied to the canvas. In this way, logits for each *foreground* category are calculated and passed to the final fusion head for final panoptic predictions.

3.2.2 Learning Compositional Model Parameters

To perform classification in the instance head, we use a compositional model that matches features at every spatial position (i, j) on the 2D lattice of the RoI feature map with a reference set. This reference set of features represent frequently observed sub-parts of instances of some *thing* class. Our compositional model consists of M multivariate Gaussians that correspond to these commonly observed patterns. To learn the parameters of these Gaussians, we use the pre-trained backbone network to collect all features using the ground truth information of all *thing* classes. We leverage the available mask information to identify which locations on the 2D lattice of the RoI feature map correspond to instances of *thing* classes. The RoIAlign function maps an RoI having an arbitrary scale and aspect ratio to a feature map of constant size $k \times k$. We use the feature maps extracted by RoIAlign to learn parameters of the compositional model.

First, we bring the ground mask of each RoI to $k \times k$ using bi-linear interpolation. Then, for an RoI having an $k \times k$ sized mask m, we collect all features where $m_{i,j} = 1$, where [i, j = 1, 2, 3, ..., k]. This is shown in Figure 3.7. Figure 3.7 shown an example from the COCO dataset. In the left half of the figure, we show the input image with each RoI annotated in green. On the right, we show the extracted RoIs and place corresponding masks on top of them. The highlighted parts in the rightmost column correspond to $m_{i,j} = 1$. Features at these positions are used for learning parameters of the compositional model. Similar to [62], we also add an extra *background* class to improve learning of the classification pipeline.

Figure 3.7: *(left)*: An input image from the COCO dataset with grouth truth RoIs annotated in green. *(right)*: 3 RoIs extracted from the image with their corresponding ground truth masks placed on top. Highlighted boxes indicate the parts that belong to the instance in the RoI bounding box.

To sample features for the *background* category, we define 4 constant RoIs along the boundaries of the input image where no *foreground* RoIs are present. The coordinates of these RoIs in the format (x_0, y_0, x_1, y_1) (where (x_0, y_0) is the top left coordinate and x_1, y_1 is the bottom right coordinate of the bounding box) are: (1, 1, $argmin(R_x)$, H), ($argmin(R_x)$, 1, $argmax(R_x)$, $argmin(R_y)$), ($argmax(R_x)$, 1, W, H), ($argmin(R_x)$, $argmax(R_x)$, $argmax(R_x)$, H) where W, H are the width and height of the image, and R_x and R_y are all x and y coordinate values in the RoIs of the image respectively. These *background* RoIs are shown in the right half of Figure 3.8, with ground truth RoIs for instances of *foreground* classes shown in the left half. Note that since the RoIs are brought to a constant size and aspect ratio using the RoIAlign function, which adds some scale invariance to the classification pipeline. The parameters of the compositional model are learned before training the task heads using the standard EM algorithm [17]. We use the implementation of the machine learning frame-

work Scikit-learn [56]. The initial means are derived using K-means and set a convergence threshold of 0.001. The covariance type is set to *diag*, following the independence assumption explained in 3.1.3.

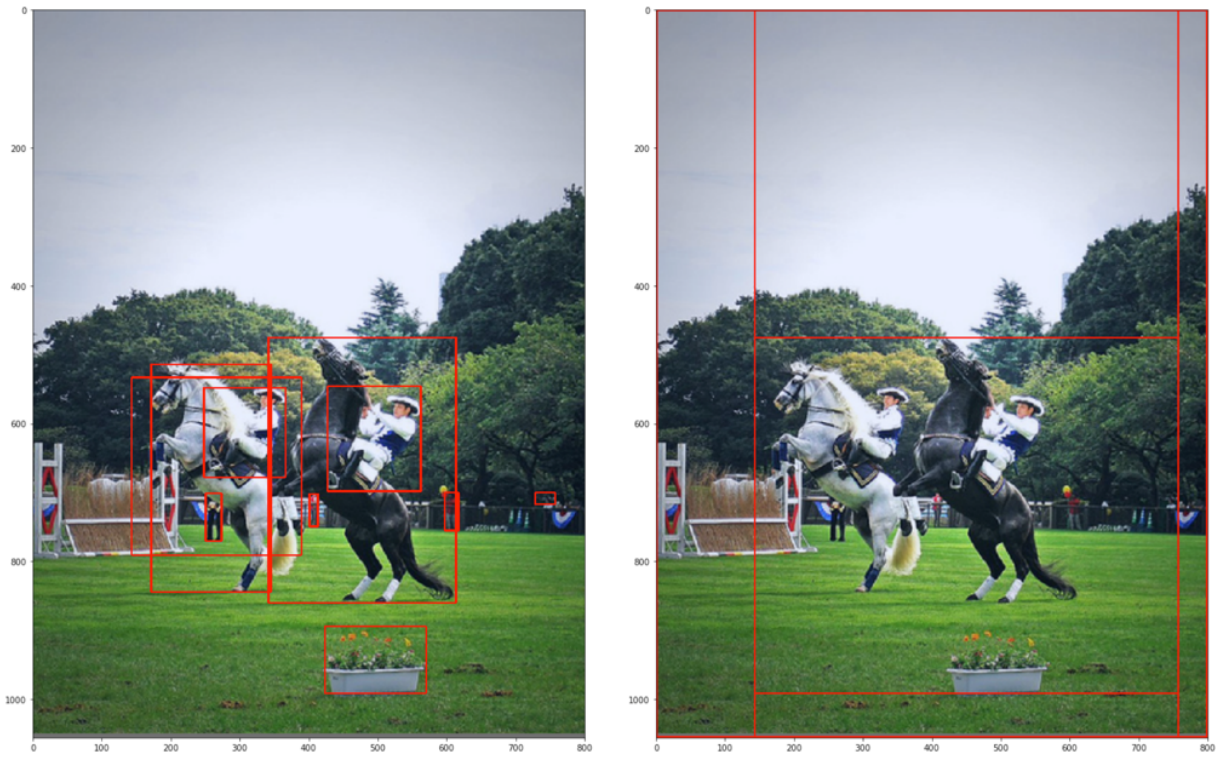

Figure 3.8: *(left)*: Image shows the ground truth RoIs annotated in red. *(right)*: An example showing the RoIs sampled for extracting features of the *background* class. Regions extending from the edges to the red lines correspond to *background* regions.

Chapter 4

Results

4.1 Datasets

We test our approach on the two most popular datasets used for the panoptic segmentation task. The Microsoft COCO [48] dataset is a large scale dataset with pixel-wise annotations for both semantic and instance segmentation tasks. The collection of images is representative of the complexity of natural scenes and includes scenarios encountered by vision systems in the wild. The dataset consists of 118k training images, 5k validation samples and 20k images for testing. For the panoptic segmentation task in COCO, there are a total of 133 categories, consisting of 80 *thing* or object categories and 53 *stuff* categories. We use the (*train*, *val*) phases of Microsoft COCO [48] dataset for corresponding tasks.

The second dataset we use is the Cityscapes dataset [11] which is a collection of urban street scenes from over 50 European cities and covers many common driving scenarios. The scenes have also been recorded over several seasons to capture diverse scenarios. The dataset contains pixel level annotations for a total of 19 classes of which 11 belong to *stuff* and 8 to the *thing* categories. There are 5000 finely annotated 2D images that have a resolution of 2048×1024 pixels captured using a 22cm baseline stereo camera. The images are divided into groups of 2975, 500 and 1525 for *train*, *val* and *test* respectively.

Figures 4.3 and 4.4 show some examples from both datasets coupled with their ground truth

annotations. As seen, both datasets are extremely challenging as they represent scenes from the wild and often contain occluded subjects.

4.2 Evaluation Metrics

The absence of a well defined metric for a unified task that studies both semantic and instance segmentation was one of the reasons that the two groups of classes were studied in isolation. To address this issue, Kirillov et al. also introduced a new unified metric with the introduction of the novel panoptic segmentation task. The metric, Panoptic Quality (PQ) was introduced in [36], and measures performance on the joint task of panoptic segmentation. PQ is shown in equation 4.1 here.

$$PQ = \frac{\sum_{p,g \in TP} IoU(p,g)}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|} \tag{4.1}$$

$IoU(p,\, g)$ is the Intersection over Union of a predicted segment p and a ground truth segment g, TP are the True Positive segments (p matched with a g, i.e., $IoU(p,\, g) > 0.5$, FP are False Positives (unmatched p) and FN being the False Negative segments (unmatched g). PQ can be calculated as a product of Segmentation Quality (SQ) and Recognition Quality (RQ) as shown in the equation below:

$$PQ = \underbrace{\frac{\sum_{p,g \in TP} IoU(p,g)}{|TP|}}_{\text{Segmentation Quality (SQ)}} \times \underbrace{\frac{|TP|}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|}}_{\text{Recognition Quality (RQ)}} \tag{4.2}$$

As seen in the figure, SQ is the average of Intersection over Unions ($IoU(p,\, g)$) of all TP segments. As explained in [36], RQ corresponds to the popular $F1$ measure. In addition to

PQ, PQ^{th} (PQ over all *thing* classes) and PQ^{st} (PQ over all *stuff* classes) are also calculated to understand the quality of instance and semantic segmentation respectively. For *stuff* classes, predictions for each *stuff* class in an image is combined and treated as one instance, irrespective of shape. The PQ metric is computed for each class independently and averaged over all classes to make the metric insensitive to imbalanced class distributions. The unknown category (explained in 3.1.4) is not included while calculating Panoptic Quality.

4.3 Experimental Setup

In this section, we present the experimental setup and design choices that were made while training our deep learning architecture. Unless otherwise specified, we use the ResNet-50 FPN backbone for most experiments. We also present results of the larger ResNet-101 FPN backbone in the results section. Results for all datasets are reported on the validation split.

We use standard stochastic gradient descent with momentum for optimization with a learning rate of 0.02 for experiments that use 16 GPUs. The learning rate is scaled as a function of the number of accelerators in use. The weight decay factor is set to 0.0001. A learning rate decay schedule of three stages is implemented. For the COCO dataset, the model is trained for 90k batches (with a batch size of 16) with the learning rate decayed by a factor of 10 at 60k and 80k iterations. For Cityscapes, the model is trained for 24k iterations (with a batch size of 8) with the learning rate decayed by a factor of 10 after 18k iterations. All the input images are resized to dimensions where the shorter side is 800 and the maximum possible largest side is 133. All images undergo horizontal flipping and per-channel normalization.

We use the pre-trained backbone and RPN weights from the baseline and retrain all other task specific heads. Since our detection head performs spatially aware classification, mask information is also necessary for training. Therefore, for the detection branch of the instance

head, we move on from the training strategy of the baseline and follow a new design. The RoIs used for training the detection branch still consist of a mixture of *foreground* and *background* samples. The *background* samples are needed to reject many region proposals that have low overlap values with the instances present in the image. For each image, we use the ground truth RoIs and their corresponding masks as the *foreground* samples. The ratio of *foreground* to *background* RoIs is set to 1:4. The *background* RoIs are chosen from a pool of high confidence proposals that have an IoU between 0 and 0.5 with any ground truth RoI. After combining the two groups of samples, the set then undergoes trimming to limit the count to 512. The training of other branches follows the baseline methodology. We learn the parameters of the compositional model with features extracted using the pre-trained weights of the baseline. All results are reported on setups with 320 mixtures (4 x number of classes) and having a feature vector size of 256. Most results are shown with a comparison to the baseline.

4.4 Panoptic Quality Computation

The panoptic segmentation task is a unified task for complete scene understanding of 2D visual data and encompasses major image understanding tasks such as classification, object detection and segmentation. The introduction of this novel task was accompanied by a new quantitative evaluation metric, aimed to capture the performance of the unified task fairly. This new metric, Panoptic Quality (PQ), is discussed in section 4.2. The task of panoptic segmentation, however, and the novel metric are still fairly recent and continue to evolve. In this section, we discuss the shortcomings of panoptic quality computation and discuss experiments that showcase these issues. Similar discussions and suggestions on making improvements to the metric can also be found in [82] [59] [20].

Computing panoptic quality can be defined as a two step procedure: First, segment matching of instance detections is performed. Each ground truth instance is matched with a prediction if the Intersection over Union of the two is greater than 0.5. Since the panoptic segmentation task requires non overlapping outputs, only one prediction can be matched with any ground truth annotation. After matching, each prediction falls into one of three groups: *TP* (matched pairs), *FP* (unmatched predictions), and *FN* (unmatched ground truth annotations). In the second step, these groups are then used to compute the final PQ using the equation in 4.2.

The COCO dataset is a comprehensive dataset of 118k training, 5k validation and 20k test images with a total of 133 categories. Despite possessing a large collection of annotated samples, many instances in the dataset are not individually annotated. For each image in the COCO dataset, if the image contains more than ten instances of the same category, they are labelled collectively as an "iscrowd" category. Each annotation has a parameter which indicates if the annotation is an *iscrowd*. During the calculation of quantitative metrics, the predictions that are predicted over regions that belong to the *iscrowd* category are ignored. However, many of the annotations in the dataset lack the parameter. The absence of this parameters causes the successful detection of instances present in complex scenes to be evaluated as false positives, even though they are, in fact, true positives, and contribute negatively to the quantitative performance. As a result, the quantitative metrics such as panoptic quality, average precision and recall for detection, that are used for evaluation of image recognition algorithms, are not fully representative of the true performance of algorithms. This is shown with the help of an example in Figure 4.1. Figure 4.1 shows an example from the COCO dataset. The first row shows the input image (*top left*) and the ground truth segmentation and bounding box annotations (*top right*). The ground truth bounding boxes are drawn in *green* and the *iscrowd* region is colored in painted in a single

large blob of red. Note that individual instances of the same class all have same color but different shades to separate instances that belong to the same class. The bottom row shows the results of the baseline (*bottom left*) and proposed (*bottom right*) approaches. Many of the instances present in the input image are unannotated, and therefore, predictions for missing instances contribute as false positives. As seen in the example, many of the predictions made on unannotated instances are correct, but contribute negatively during the calculation of quantitative metrics such as PQ and average precision.

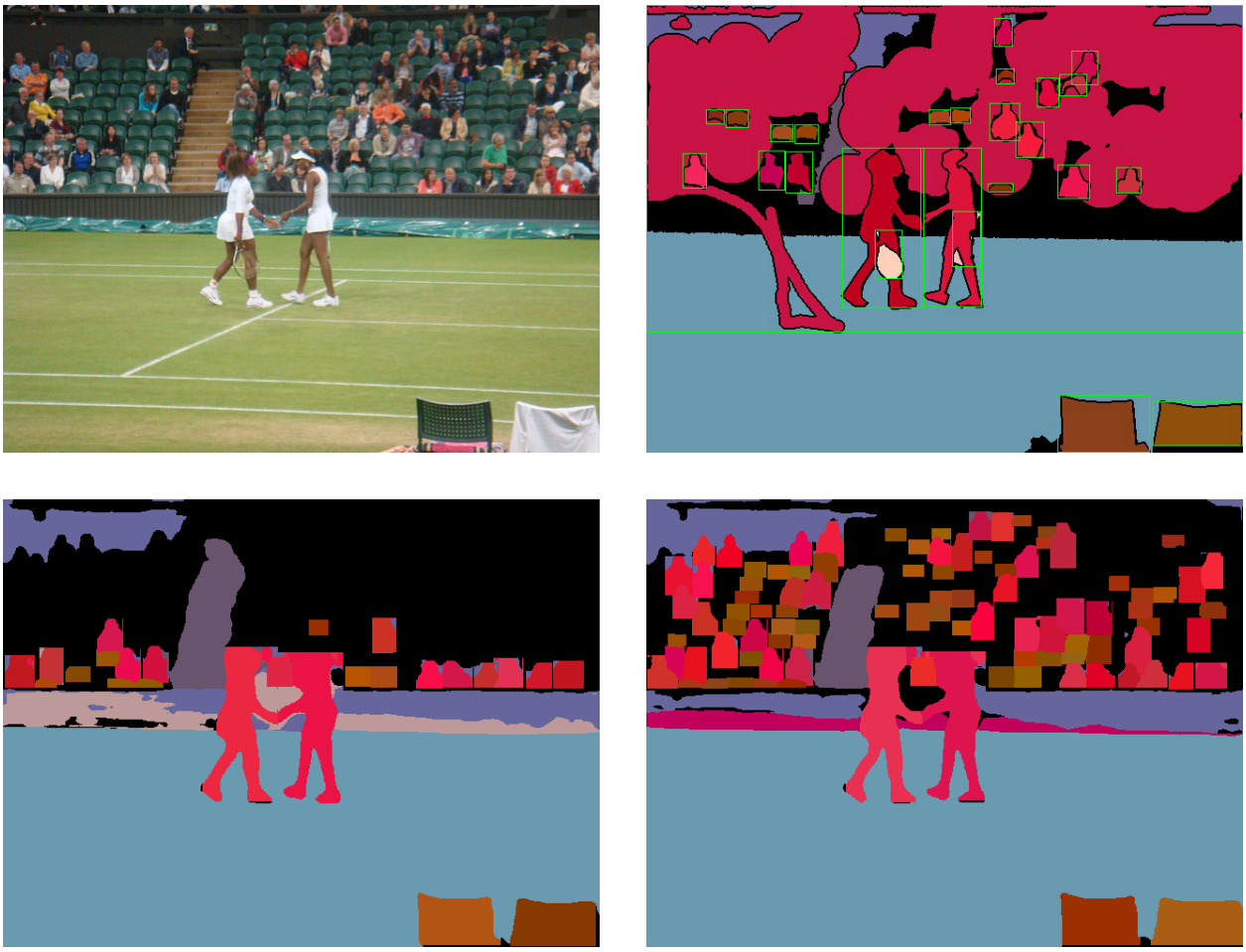

Figure 4.1: An example that shows ground truth annotations in the COCO dataset. The first row shows the input image (*left*) and the ground truth segmentation and bounding box annotations (*right*). The ground truth bounding boxes are drawn in *green*. The bottom row shows the results of the baseline (*left*) and proposed (*right*) approaches. Many of the instances present in the input image are unannotated, and therefore, predictions for missing instances contribute as false positives.

To explore quantitative metric calculation of detection metrics, we performed panoptic segmentation on a subset of images that give good detection results for instances that are unannotated in the dataset. These samples are shown in Figure 4.2. As seen in the figure, many of the results contain examples missed by the baseline but are detected by the proposed approach. The quantitative results for this subset of images are shown in Table 4.1.

Even though the proposed approach gives better qualitative results (if not matched) than the baseline, the quantitative metrics are not fully representative of the true performance. As seen in the table, RQ^{th}, which is similar to the F1 measure for *thing* classes, is comparatively lower, even though the proposed approach detects more instances in the inputs and generates better segmentation masks for the few instances that are present in the ground truth.

Due to this issue, we ask the readers to give equal importance to qualitative results and refer to quantitative metrics by keeping qualitative results into consideration. We think that observing relative counts of true detections and missed instances may also help in gauging the performance of algorithms. The counts of instances of classes that are detected well with our approach are shown in Table 4.3. As seen in the table, the approach detects a higher number of true positives while reducing the false negatives, which indicates that instances missed by the detector are successfully caught by the proposed approach.

Figure 4.2: Results of the experiment showing issues discussed in section 4.4. The 1st and 2nd columns show the input and ground truth respectively. The last two columns show the results of the proposed approach (3rd column) and the baseline (last column). The proposed approach yields better results than the baseline in all examples. Note that some examples are missing dense ground truth instance annotations which adversely affect the computation of quantitative metrics.

Table 4.1: Results of the experiment on some samples (shown in Figure 4.2) from the COCO *Val* dataset. Even though the proposed approach gives better qualitative results than the baseline, the quantitative metrics are not fully representative of the true performance. As seen in the table, RQ^{th}, which is similar to the F1 measure for *thing* classes, is comparatively lower, even though the proposed approach detects more instances in the inputs and generates better segmentation masks for the few instances that are present in the ground truth.

Method	Backbone	PQ	SQ	RQ	PQ^{th}	SQ^{th}	RQ^{th}	PQ^{st}	SQ^{st}	RQ^{st}
Baseline	ResNet-50 FPN	31.5	37.1	40.8	39.0	41.7	46.5	22.2	31.4	33.8
Proposed	ResNet-50 FPN	27.7	33.9	36.8	34.2	38.4	44.4	19.7	28.3	27.4

4.5 Qualitative Results

Examples from qualitative results of the proposed approach are shown in Figure 4.3. The first and second columns show the input data and the ground truth respectively. For comparison, the last two columns show the results of the baseline (third column) and our approach (last column).

Many of the examples contain instances that occur in close proximity and suffer from varying levels of occlusion. In the first row, a baby elephant is blocking a significant portion of another instance of the elephant class. This causes the baseline to assign a high confidence score to a RoI that includes both instances and fails to distinguish between overlapping instances of the same category. This effect can also be observed in the input on the third row, where the occluded instance is missed and included in another detected bounding box. This confusion is overcome by the proposed approach since both RoIs get assigned high confidence values and are detected as separate instances during final detection. The second and the last rows show examples where instances are missed by the baseline due to unusual pose and appearance. As our approach performs classification by calculating likelihoods of sub-parts, parts of the RoI that have a strong correspondence with some *thing* class ensure that ambiguity in some parts does not affect the final class score. The person sitting at the

back on the motorcycle in the second row and the person on the right in the last image are detected successfully by our approach.

The qualitative results on the Cityscapes dataset are shown in Figure 4.4. In contrast to COCO, the Cityscapes dataset is much smaller in terms of both, sample size and category count. Following the layout of previous figures, the first and second columns show the input data and the ground truth respectively. For comparison, the last two columns show the results of the baseline (third column) and our approach (last column). As seen in the figure, occluded *person* instances are successfully separated by our approach. As a large portion of the input is covered by semantic instances, the results are quite similar to the baseline.

Finally, the results of our approach on the COCO *Val* dataset using the ResNet-101 FPN backbone are shown in appendix A. Using a larger feature extractor backbone improves the overall performance as the feature maps contain richer feature representations of the input. This results in an improvement in both the semantic and instance segmentation performance. When shifted from the ResNet-50 to the ResNet-101 shared backbone, our approach makes a larger improvement than the baseline, which suggests that better feature representation may help improve the performance of our approach even further.

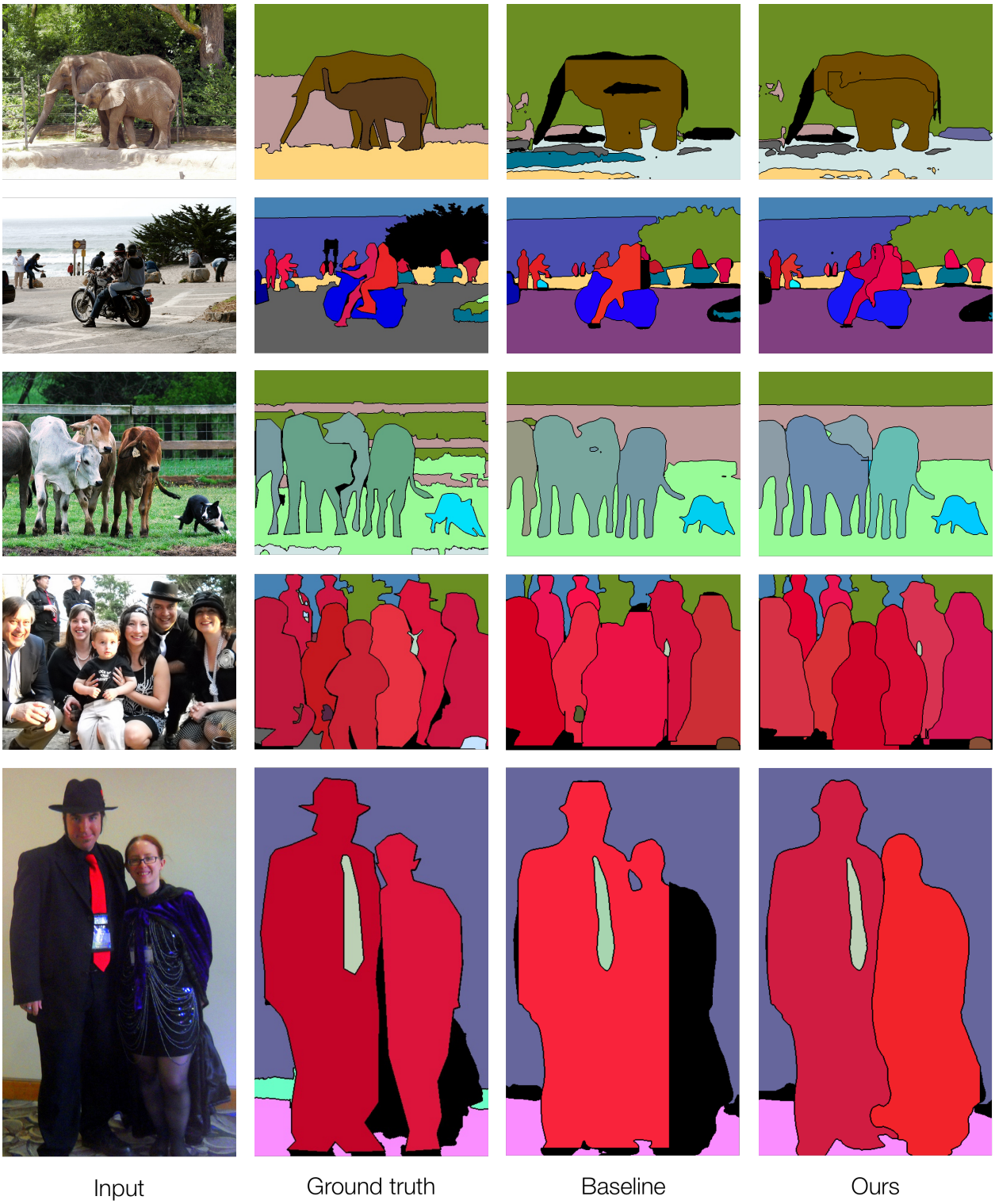

Figure 4.3: Qualitative results on examples from the COCO *Val* dataset. The overlapping elephant instances in the first row and people on the motorcycle in the second row are separated well by our approach. An occluded cow instance is also caught, even though the segmentation result is not ideal. Overlapping people in the final two examples are also detected successfully.

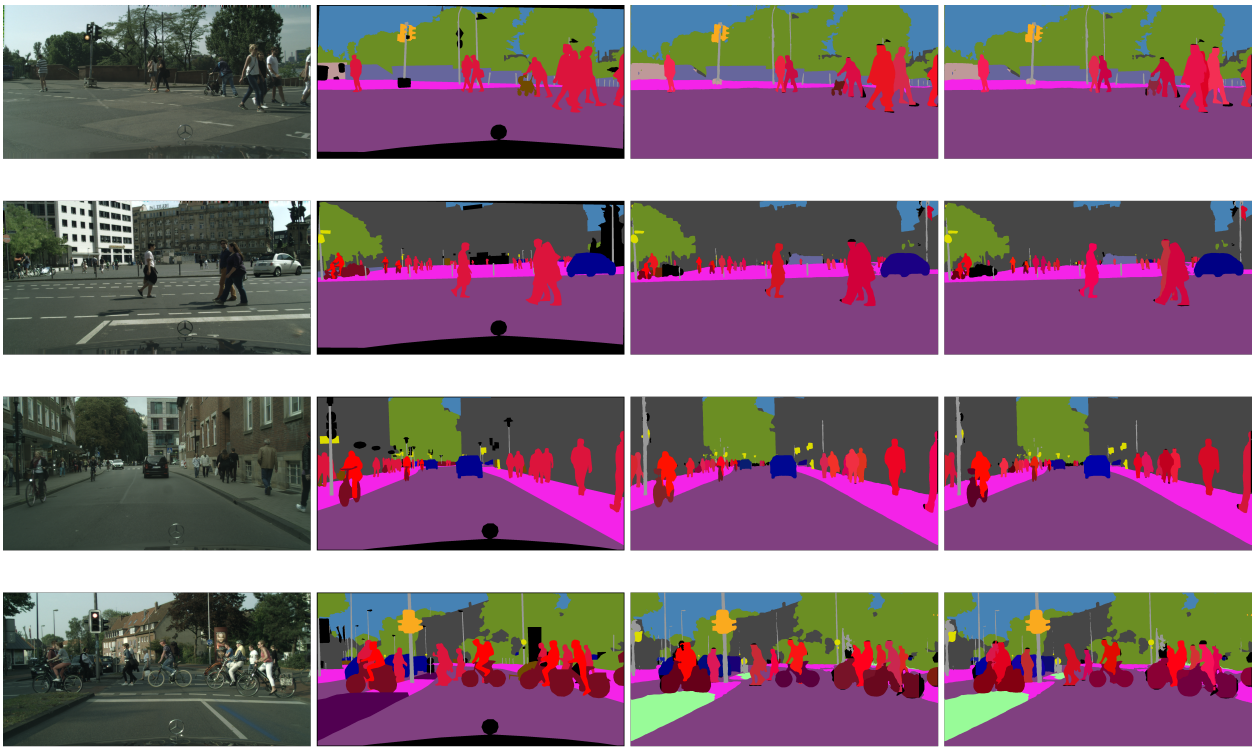

Figure 4.4: Qualitative results on examples from the Cityscapes *Val* dataset.

4.6 Crowded Scenes

In this section, we discuss the performance of our approach on complex scenes which contain a large number of instances. Due to the high density of instances, many are only partially visible and often have varying scales. As shown in the results in Figure 4.5, our approach performs well in detecting these instances. Since each RoI is brought to the same scale using the RoIAlign function and then goes through a compositional model, instances smaller in size as they are far away from the image capture device, or occluded, still get assigned high confidence scores.

Figure 4.5 shows some results on crowded scenes. For comparison, results of the baseline

are also included. In the first two rows, we can see that our approach performs well in detecting and separating out instances of the person category even though many of them are severely occluded and have an unusual pose. In the first example, a faintly visible baseball player behind the batter (player holding the baseball bat) is successfully detected. The third row shows a sample in which our approach detects both more oranges and apples. Finally, the last row shows a dense crowd of elephants with many of them being occluded by other elephant instances. Despite not achieving a perfect segmentation result, our approach is able to successfully detect more occluded instances than the baseline.

Figure 4.2 also shows more examples of successful detections in crowded scenes by our approach. As discussed in 4.4, many regions in such crowded scenes have been left unannotated which categorizes these dense detections as false positives that contribute negatively to quantitative performance metrics, even though the qualitative results tell a different story. We expect the performance of our approach on quantitative metrics to improve once the ground truth is augmented with the missing annotations.

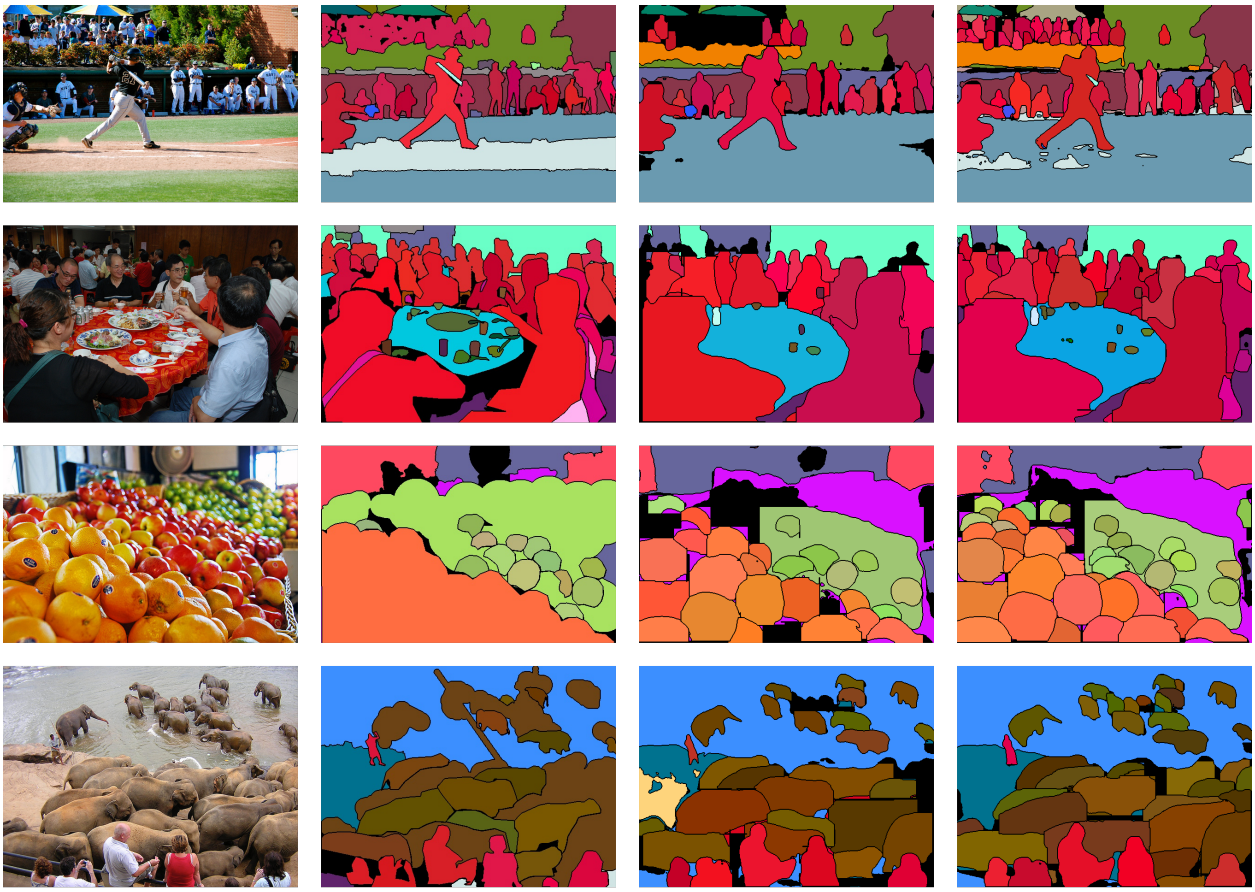

Figure 4.5: Results on crowded scenes from the COCO *Val* dataset. In the first example, the crowd in the top left and a faintly visible baseball player behind the batter (player holding the baseball bat) are successfully detected. The third row shows a sample in which our approach correctly detects more oranges as well as more apples. Finally, the last row shows a dense crowd of elephants with many of them being occluded by other elephant instances. As seen, our approach detects a significantly higher number of instances.

4.7 Quantitative Results

We show quantitative results on the validation split of both COCO and Cityscapes datasets. Table 4.2 shows overall results on the COCO *Val* dataset. The table reports the overall PQ score and PQ averaged over *thing* classes and *stuff* classes as well. The methods are grouped

into two broad categories called Single stage and Two stage. Two stage strategies perform inference in two steps. First, they identify regions with the possible presence of instances of *thing* classes called proposals and then process these proposals to get the final result. In contrast, single stage methods make predictions for *thing* and *stuff* classes in a single forward pass step. As seen in the table, our approach is similar to the Mask R-CNN [30] design and can therefore be categorized as a two stage method. It is interesting to note that six of the eleven two stage methods in the table use the same base design of Mask R-CNN [30]. For most methods, we include architecture versions that share the same backbone as our method for uniform comparison. Our baseline ranks second in terms of overall metrics. The performance using the ResNet-101 backbone are also shown in the table. Making use of the larger ResNet-101 backbone improves overall performance 4.8 points and improves RQ by 6.3 points, which are both better than improvements made by the baseline. Even though we report performance numbers on both datasets, we want to stress again that these metrics alone may not be enough to determine the actual performance of the algorithms, as these metrics have been computed with missing annotations in the dataset as discussed in section 4.4.

To exclude the incorrect false positive predictions, we report detection performance using the true positive and false negative counts and recall. In terms of successful detections of ground truth instances, our approach shows an improvement over the baseline. We note that the recall and precision metrics are related, and an increase in recall could possibly result in a lower precision score, which would explain lower precision values for our approach. However, since the proposed method performs exceedingly well on crowded scenes, a large portion of false positives are contributed by such samples, which, given perfect annotations would turn into true positives and would improve both precision and recall scores even further. Table 4.3 shows a comparison of our approach with the baseline. We see an improvement of 1.45

Table 4.2: Panoptic segmentation results on the MS-COCO 2018 *Val* dataset. Superscripts 'Th' and 'St' denote numbers for *thing* and *stuff* classes respectively. (All values are shown as percentages. Higher is better.) (Λ: Computed using missing annotations.)

Method	Backbone	PQ	SQ	RQ	PQ^{Th}	SQ^{Th}	RQ^{Th}	PQ^{St}	SQ^{St}	RQ^{St}
Single Stage										
DeeperLab [82]	Xception-71	33.8	-	-	-	-	-	-	-	-
Hou et al.[33]	ResNet-50 FPN	37.1	-	-	41.0	-	-	31.3	-	-
PCV [76]	ResNet-50 FPN	37.5	77.7	47.2	40.0	78.4	50.0	33.7	76.5	42.9
Panoptic DeepLab [10]	Xception-71	40.2	-	-	44.4	-	-	33.8	-	-
Two Stage										
JSIS-Net [15]		26.9	72.4	35.7	29.3	72.1	39.2	23.3	73.0	30.4
AUNet [46]	ResNet-50 FPN	38.6	76.4	47.5	46.2	80.2	56.2	27.1	70.8	34.5
AdaptIS [67]	ResNet-50 FPN	41.8	78.4	51.3	47.8	81.3	58.0	32.8	74.1	41.1
Panoptic FPN [35]	ResNet-50 FPN	39.0	-	-	45.9	-	-	28.7	-	-
OANet [50]	ResNet-50 FPN	39.0	77.1	47.8	48.3	81.4	58.0	24.9	70.6	32.5
SOGNet [84]	ResNet-50 FPN	43.7	-	-	50.6	-	-	33.6	-	-
SpatialFlow [8]	ResNet-50 FPN	39.3	-	-	45.1	-	-	30.5	-	-
Single-Shot [80]	ResNet-50 FPN	32.4	-	-	34.8	-	-	28.6	-	-
Naiyu Gao et al. [22]	ResNet-50 FPN	40.2	-	-	45.3	-	-	32.3	-	-
OCFusion [41]	ResNet-50 FPN	42.5	-	-	49.1	-	-	32.5	-	-
UPSNet [81]	ResNet-50 FPN	42.5	78.5	52.5	48.1	79.2	59.2	33.9	77.4	42.3
Ours$^{\Lambda}$	ResNet-50 FPN	40.1	78.1	49.7	44.2	78.6	54.6	33.9	77.4	42.3
Method	Backbone	PQ	SQ	RQ	PQ^{Th}	SQ^{Th}	RQ^{Th}	PQ^{St}	SQ^{St}	RQ^{St}
UPSNet [81]	ResNet-101 FPN	46.7	80.5	56.9	53.2	81.2	64.6	36.9	79.5	45.4
Ours$^{\Lambda}$	ResNet-101 FPN	44.9	80.5	54.7	50.3	81.6	60.9	36.8	78.4	45.3

points on the overall recall score, accompanied by a 2.53% increase in *TP* counts and a 3.36% decrease in *FP* numbers. Using the larger ResNet-101 backbone increases the overall recall improvement even further to 2.14 points. A class-wise breakdown of the improvements is discussed in the following section.

The results on the Cityscapes dataset are shown in Table 4.4. In contrast to COCO, the Cityscapes dataset is much smaller in terms of both, dataset size and category count. The dataset also contains a fairly large ratio of crowds, with very little visibility of occluded instances. This makes it tougher to detect instances well. Similar to the COCO dataset, closely packed instances are labelled as crowds and are considered as a single instance of

Table 4.3: A comparison of instance detection performance of our approach against the baseline on the MS-COCO 2018 *Val* dataset. A comparison of recall is important as computation of precision, and by extension, the overall metric is affected by missing annotations in the dataset. As seen in the table, our approach successfully many instances that were missed by the baseline. The numbers shown in this table are for all 80 instance categories of the COCO dataset. (↑ = Higher is better, ↓ = Lower is better.)

Method	Backbone	*True positives* ↑	*False negatives* ↓	Recall ↑
Baseline	ResNet-50 FPN	20,599	15,504	0.5705
Ours	ResNet-50 FPN	21,121	14,982	0.5850
Improvement				+0.0145
Method	Backbone	*True positives* ↑	*False negatives* ↓	Recall ↑
Baseline	ResNet-101 FPN	22,535	13,568	0.6241
Ours	ResNet-101 FPN	23,308	12,795	0.6455
Improvement				+0.0214

the corresponding category. The proposed approach is trained on the Cityscapes dataset without applying attention as it performs better at detecting grouped instances. As seen in the table, the performance of the proposed approach is almost identical to the baseline.

4.8 Classwise Performance

In this section, we discuss the class-wise performance of our approach on the COCO dataset. The Microsoft COCO [48] dataset is a large scale dataset with pixel-wise annotations for both semantic and instance segmentation tasks. The collection of images represents of the complexity of natural scenes and includes scenarios encountered by vision systems in the wild. For the panoptic segmentation task in COCO, there are a total of 133 classes, consisting of 80 *thing* or object categories and 53 *stuff* categories. In this discussion, we show the class-wise breakdown of the results on the *Val* split of the dataset.

Table 4.4: Panoptic segmentation results on the Cityscapes *Val* dataset. Superscripts 'Th' and 'St' denote numbers for *thing* and *stuff* classes respectively. Input size, Multi-scale input and flip. (All values are shown as percentages. Higher is better.) (Λ: Computed using missing annotations.)

Method	Backbone	PQ	PQ^{Th}	PQ^{St}
Single Stage				
Hou et al. [33]	ResNet-50 FPN	58.8	52.1	63.7
Two Stage				
Panoptic FPN [35]	ResNet-101 FPN	58.1	52.0	62.5
SOGNet [84]	ResNet-50 FPN	60.0	56.7	62.5
SpatialFlow [8]	ResNet-101 FPN	59.6	55.0	63.1
TASCNet [43]	ResNet-50 FPN	59.3	56.3	61.5
Seamless [59]		60.3	56.1	63.3
OCFusion [41]	ResNet-50 FPN	59.3	53.5	63.6
UPSNet [81]	ResNet-50 FPN	58.7	53.2	62.6
$Ours^{\Lambda}$	ResNet-50 FPN	58.2	52.2	62.7

Figure 4.5 shows the *TP* and *FN* counts and recall for classes that see the most improvement when compared to the baseline. The *Person* and *Car* classes have the highest frequency in the dataset and have an improved recall of 2.59 and 2.99 percentage points respectively. Smaller sized instances (by area) of categories such as *Traffic Light*, *Bird*, *Book* and fruits see the largest gains in recall which indicates that our approach is able to detect instances with varied scales and is able to pick out instances even though they are closely packed (and often occluded) together. A situation where the proposed approach does not do so well is when objects with very similar textures are closely packed together. For instance, complex scenes in which animals such as the elephant, zebras and sheep that have very similar patterns across all instances overlap on another, the boundaries of instances become ambiguous, the model detects them as a single instance. This results in an increase in false negatives and affects negatively.

The features used for learning the parameters of the compositional model are extracted using ground truth bounding boxes and masks. During our experiments, the input images were used without the application of any data augmentation techniques for feature extraction to maintain fast convergence times. Further improvements to the performance of the compositional model can be expected by including augmented data for learning the parameters of the compositional model.

Table 4.5: A comparison of instance detection performance using the ResNet-50 backbone of our approach against the baseline on the MS-COCO 2018 *Val* dataset. A comparison of recall is important as computation of precision, and by extension, the overall metric is affected by missing annotations in the dataset. As seen in the table, our approach successfully many instances that were missed by the baseline. The numbers shown in this table are for instance categories of the COCO dataset.

Class	Baseline		Ours		Recall		*Improvement*
	TP	FN	TP	FN	Baseline	Ours	
Person	7884	2891	8163	2612	0.7317	0.7576	0.0259
Bicycle	131	183	134	180	0.4172	0.4268	0.0096
Car	1172	733	1229	676	0.6152	0.6451	0.0299
Motorcycle	211	156	216	151	0.5749	0.5886	0.0136
Airplane	108	35	109	34	0.7552	0.7622	0.0070
Truck	176	213	180	209	0.4524	0.4627	0.0103
Boat	196	228	205	219	0.4623	0.4835	0.0212
Traffic Light	349	285	371	263	0.5505	0.5852	0.0347
Parking Meter	35	25	37	23	0.5833	0.6167	0.0333
Bench	126	279	128	277	0.3111	0.3160	0.0049
Bird	187	239	204	222	0.4390	0.4789	0.0399
Cow	247	121	253	115	0.6712	0.6875	0.0163
Bear	55	16	56	15	0.7746	0.7887	0.0141
Backpack	78	282	85	275	0.2167	0.2361	0.0194
Umbrella	237	170	250	157	0.5823	0.6143	0.0319
Handbag	119	412	128	403	0.2241	0.2411	0.0169
Tie	62	190	65	187	0.2460	0.2579	0.0119
Suitcase	150	143	151	142	0.5119	0.5154	0.0034
Baseball Glove	78	70	82	66	0.5270	0.5541	0.0270
Bottle	553	456	578	431	0.5481	0.5728	0.0248
Wine Glass	143	195	155	183	0.4231	0.4586	0.0355
Cup	462	400	502	360	0.5360	0.5824	0.0464
Banana	114	256	140	230	0.3081	0.3784	0.0703
Apple	69	165	78	156	0.2949	0.3333	0.0385
Carrot	134	230	154	210	0.3681	0.4231	0.0549
Cake	158	149	167	140	0.5147	0.5440	0.0293
Chair	670	1082	757	995	0.3824	0.4321	0.0497
Potted Plant	153	187	154	186	0.4500	0.4529	0.0029
Book	287	841	362	766	0.2544	0.3209	0.0665
Scissors	12	23	13	22	0.3429	0.3714	0.0286
Toothbrush	16	41	17	40	0.2807	0.2982	0.0175

4.9 Analysis of Compositional Model

The compositional model used for object detection consists of a mixture of multivariate Gaussian distributions. We learn the parameters of the multivariate Gaussian component distributions in an unsupervised manner, using features at locations where an instance of *thing* class is present in the image. Upon convergence, the centers μ_k represent frequent occurring patterns that are characteristic to particular *thing* classes in the dataset.

In this section, we perform an analysis of the compositional model and show intermediate results generated during the inference phase. For each RoI, we retain the posterior values generated by the compositional model and consider the *argmax* at each position on the 2D lattice of the feature map after applying a threshold of 0.8. Then for each input image, RoIs predicted by the model are extracted and divided into $k \times k$ parts. Figures 4.6 and 4.7 show parts of the RoIs where the same component is the most active for a particular category. The first figure shows parts that map to the same component for while detecting the *person* and *bus* categories. As seen in the figure, facial regions mostly map to one of the components, while the second group shows parts near the headlights and doors correspond to a component while detecting *bus* instances. Another figure shows parts that correspond to the same component for while detecting the *dog* and *airplane* categories. As seen in the figure, facial regions of the dog mostly map to one of the components. For the *airplane* class, regions that contain the tail fin and the engine that map to a component are shown.

For each category, the COCO dataset also specifies a "supercategory" that indicates the broad group a category belongs to. For instance, *thing* classes such as *Car*, *Bicycle*, *Motorcycle*, *Truck* and *Bus* altogether form the *Vehicle* supercategory. Intestingly, classes in some of the supercategories also share some common features or patterns that are unique to the supercategory. Figure 4.8 shows parts that map to two components that are active while

detecting classes of the *Vehicle* supercategory. The first two rows show parts of wheels taken from instances that were classified as *Car*, *Motorcycle*, *Bus* and *Truck*, with all these parts having the same most active component of the compositional model. Similarly, the last two rows show parts that again map to the same component and extracted from instances of classes *Bicycle* and *Motorcycle*. As seen in the figure, these parts show the handle and seat areas of the instances. This shows that the compositional model is able to learn feature vectors that encode the common patterns observed on instances of the classes present in the dataset.

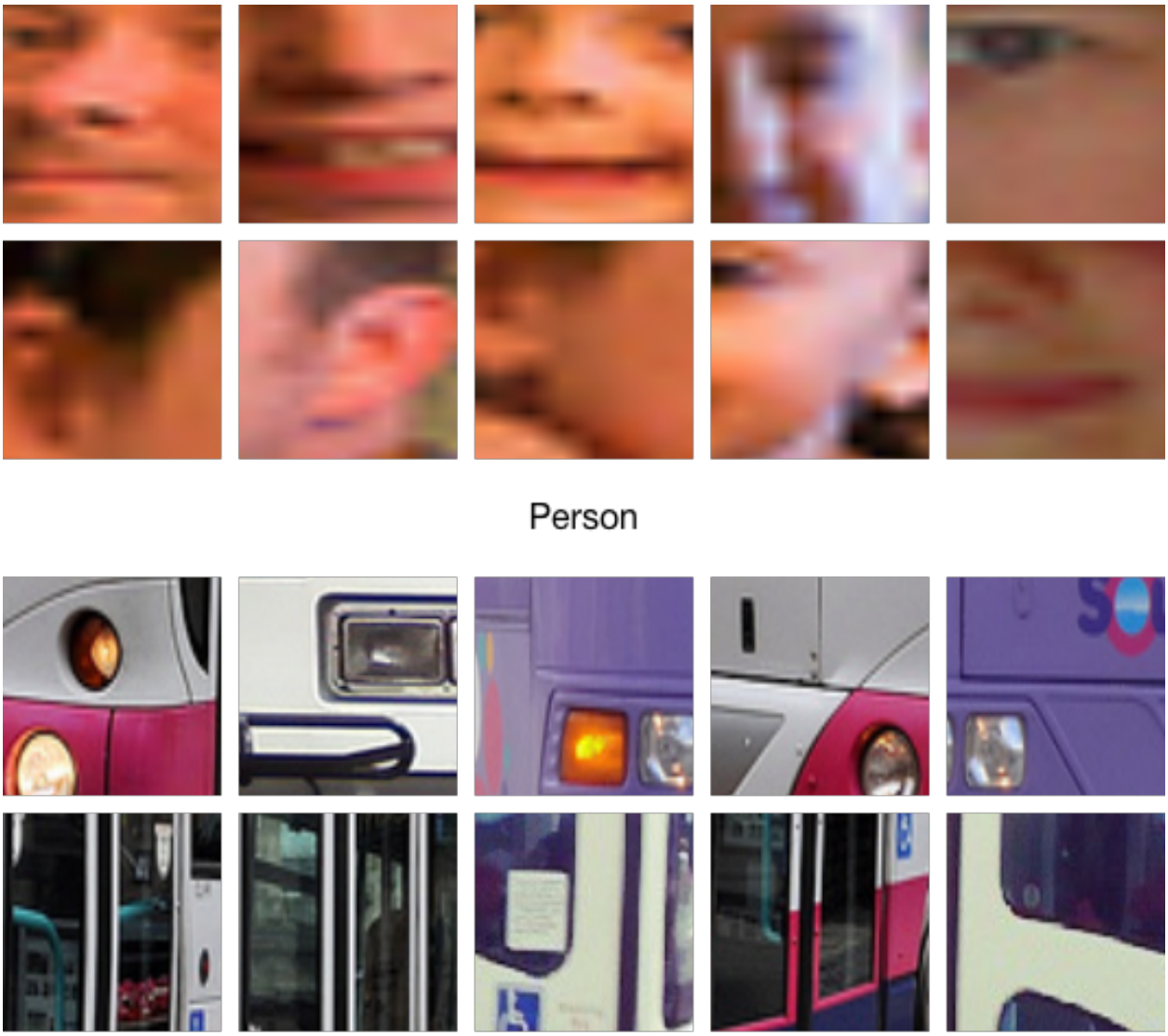

Figure 4.6: Analysis of the compositional model is shown in this figure. Parts that map to the same component for while detecting the *person* and *bus* categories are shown. As seen in the figure, facial regions mostly map to one of the components, while the second group shows parts near the headlights and doors correspond to a component while detecting *bus* instances.

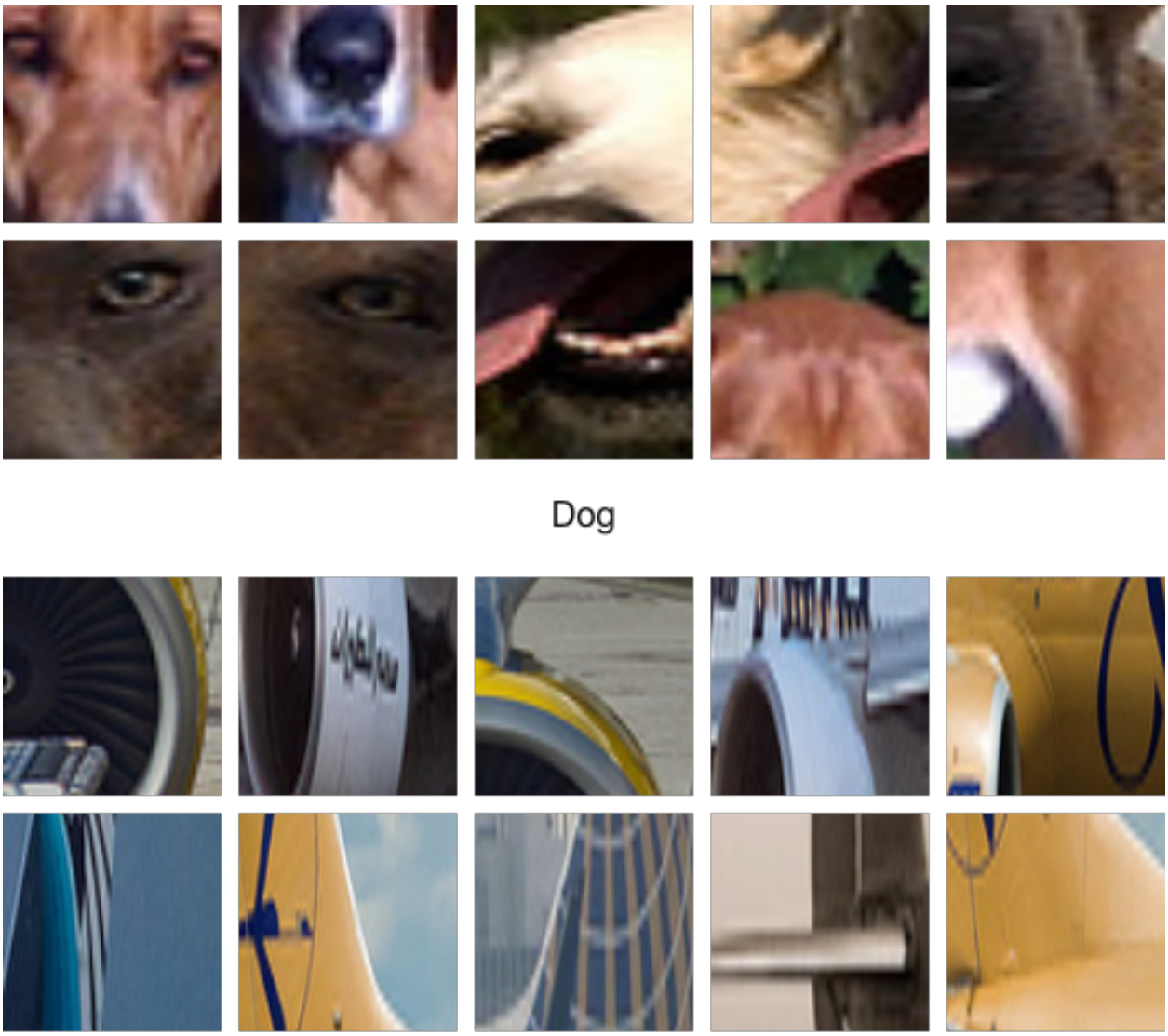

Figure 4.7: Another figure shows parts that correspond to the same component for while detecting the *dog* and *airplane* categories. As seen in the figure, facial regions of the dog mostly map to one of the components. For the *airplane* class, regions that contain the tail fin and the engine that map to a component are shown.

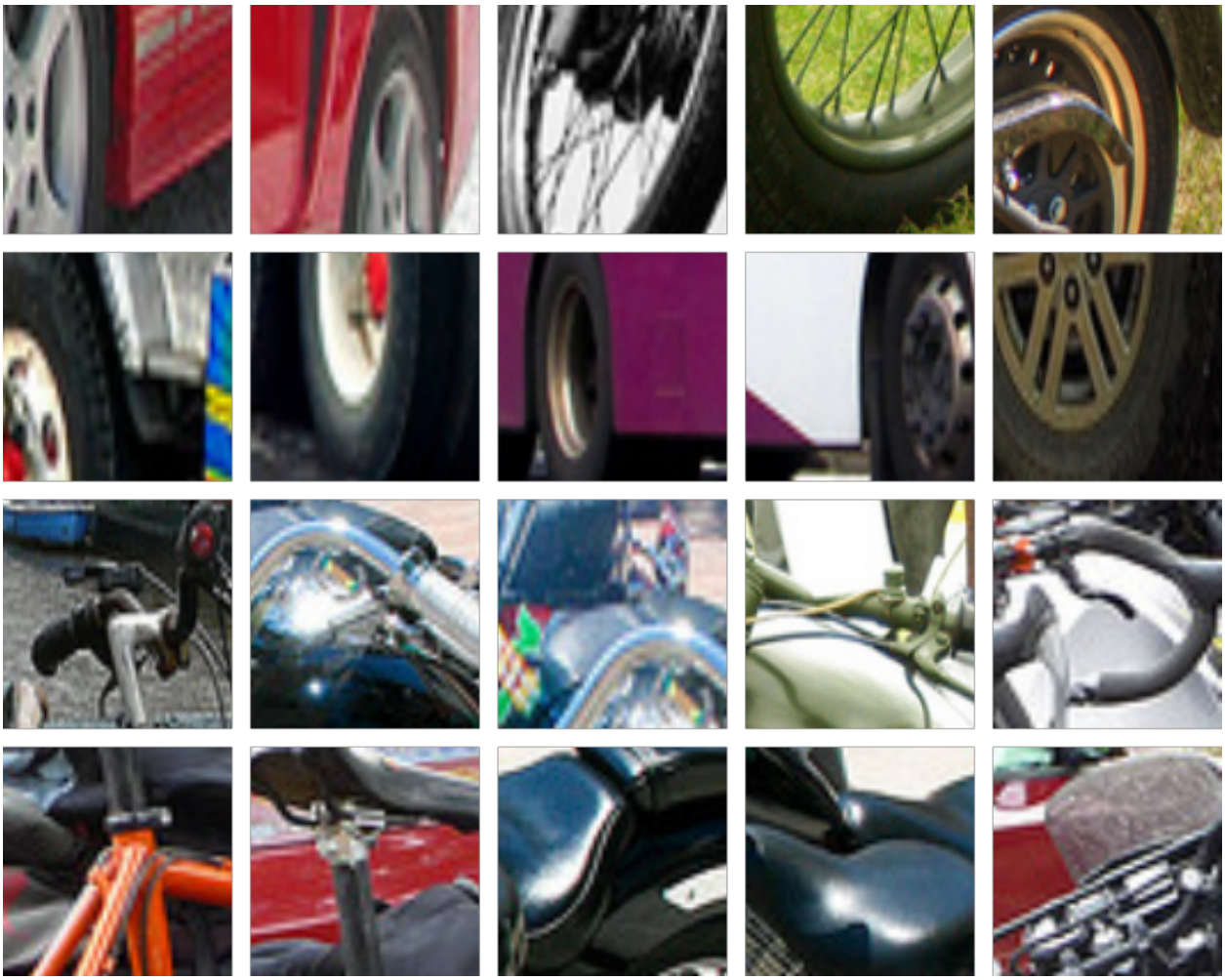

Figure 4.8: This figure shows parts that map to two components that are active while detecting classes of the *Vehicle* supercategory. The first two rows show parts of wheels taken from instances that were classified as *Car*, *Motorcycle*, *Bus* and *Truck*, with all these parts having the same most active component of the compositional model. Similarly, the last two rows show parts that again map to the same component and extracted from instances of classes *Bicycle* and *Motorcycle*. As seen in the figure, these parts show the handle and seat areas of the instances.

4.10 Ablation Studies

In this section, we present the ablation study and discuss a number of experiments that were performed to make decisions about various design choices while building the proposed

architecture. Unless otherwise specified, all experiments are performed using a ResNet-50 FPN backbone. The COCO *train* and *val* splits are used for training and evaluation respectively. To keep the comparison fair, all experiments in the same table have been run for the same number of epochs.

Number of clusters. The compositional model in our detect branch is formed using a mixture of multivariate Gaussian distributions. Each Gaussian component is defined using a set of parameters [π_k, μ_k and Σ_k] (for the kth mixture component). Upon convergence, the centres μ represent parts that are commonly observed on instances of *thing* classes. Determining the exact number of components required to model an entire class in the dataset is non trivial. Therefore, we chose to follow the setting used in [38] and set the count of mixtures to $C \times 4$, where C is the number of *thing* classes in the dataset. Following [26], we also included the *background* class in the group of *thing* classes. Table 4.6 shows results of experiments on the COCO *Val* dataset with varying sizes of the compositional model. Clusters represent the count of mixture components. As seen in the table, the RQ and Recall metrics improve as number of clusters are increased. To maintain speed of training, we show all results using a compositional model of 320 components ($C \times 4$ where C is the number of classes). However, performance improvements can be expected if size of the compositional model is increased.

Attention mechanism. The detect branch of our instance head is responsible for predicting class scores and bounding box offsets for each RoI generated by the Region Proposal Network (RPM). However, RoIs proposed by the RPN are rough estimates, with a majority of them having a low overlap with a *foreground* instance. Such RoIs, despite containing *foreground* object parts, are still classified as *background* instances. This reasoning for training the proposed ground truth model is not ideal. As the proposed compositional model makes

Table 4.6: Results of experiments with varying sizes of the compositional model. Clusters represent the count of mixture components. As seen in the table, the RQ and Recall metrics improve as number of clusters are increased. To maintain speed of training, we show all results using a compositional model of 320 components ($C \times 4$ where C is the number of classes). However, performance improvements can be expected if size of the compositional model is increased.

Clusters	Backbone	PQ	SQ	RQ	PQ^{Th}	SQ^{Th}	RQ^{Th}	Recall
160	ResNet-50 FPN	38.1	76.4	47.6	42.3	77.9	52.7	0.5289
320	ResNet-50 FPN	38.2	76.3	47.8	42.2	77.7	52.7	0.5319
640	ResNet-50 FPN	38.4	76.4	47.9	42.6	77.9	53.1	0.5334

Table 4.7: Results of experiments to compare the use of attention in performing classification of RoIs. As seen in the table, use of the attention mechanism improves detection performance by a large margin.

Attention	Backbone	PQ	SQ	RQ	PQ^{Th}	SQ^{Th}	RQ^{Th}	Recall
Without	ResNet-50 FPN	36.5	77.3	45.0	38.4	78.3	47.0	0.4591
With	ResNet-50 FPN	40.1	78.8	49.7	44.4	79.7	54.8	0.5456

class predictions by identifying object parts on the *RoI lattice*, an attention mechanism is introduced to determine if a position on the *RoI lattice* is a *foreground* object part, or simply the *background* region in the scene. The design of the attention mechanism is discussed in section 3.1.3. Table 4.7 shows the results of the ablation study of the attention mechanism. As seen in the table, use of the attention mechanism improves detection performance by a large margin.

Ground truth data. The instance detection pipeline uses a novel compositional model with an attention mechanism to make class score predictions. As the new design replaces the detection pipeline in Mask R-CNN [30], the ground truth data used for training is different and utilizes mask information as well. The new ground truth data consists of all boxes and

Table 4.8: Results of experiments to compare the data augmentation to increase training data size per image for the detection branch of the instance head. Including a higher count of *background* RoIs, allows the network to learn the large variation of *background* RoIs with different IoU thresholds. As seen in the table, showing more of these RoIs improves performance of the detection branch.

Augmentation	Backbone	PQ	SQ	RQ	PQ^{Th}	SQ^{Th}	RQ^{Th}	Recall
Without	ResNet-50 FPN	39.9	78.1	49.5	44.1	78.6	54.6	0.5357
With	ResNet-50 FPN	40.1	78.8	49.7	44.4	79.7	54.8	0.5456

masks of ground truth instances. The proposal boxes generated by the Region Proposal Network contain a high number of boxes that have low IoU with any *foreground* object. These are termed as *background* RoIs. Ideally, *background* RoIs should all be filtered out and only RoIs with maximum overlap with an object should be retained. To distinguish between *foreground* vs. *background* RoIs, RoIs with an IoU less than 0.5 are also added to the data. The ratio of *foreground* to *background* instances for training is set to 1:4, as inferred from the training data used for Faster R-CNN [62] (as both methods use RPN to generate proposals). It is important to learn this separation function well as assigning high confidence to RoIs with low overlap is not desirable. To include a high number of *background* proposals, the training data that only contains *foreground* instances is appended with shuffled versions of itself. This results in a larger count of *background* RoIs as $N \times 4$ *background* proposals are sampled (N is the count of *foreground* proposals). Table 4.8 shows the results of appending additional data to include more variations of *background* proposals. As seen in the table, showing more *background* RoIs to the detect branch during training improves performance of the detection branch.

Compositional model training. While learning parameters of the compositional model, feature vectors used for training are sampled using the procedure explained in section 3.2.2.

Features that map to *background* regions are also included in the training data to maintain some components that can recognize *background* regions in the RoI. We explore this design choice and perform A/B testing to determine if inclusion of *background* features improves recognition performance. Inclusion of *background* features helps the compositional model determine if some position on the *RoI lattice* corresponds to the background. The results of A/B testing that justify the inclusion of *background* features are shown in Table 4.9. As seen from the metrics, including *background* features improves detection performance.

Table 4.9: The compositional model is trained using features that are sampled from both *foreground* objects and *background* regions. Inclusion of *background* features helps the compositional model determine if some position on the *RoI lattice* corresponds to the background. The results of A/B testing that justify the inclusion of *background* features are shown in this table. As seen from the metrics, including *background* features improves detection performance.

Background	Backbone	PQ	SQ	RQ	PQ^{Th}	SQ^{Th}	RQ^{Th}	Recall
Without	ResNet-50 FPN	39.9	77.5	49.6	43.9	78.4	54.4	0.5440
With	ResNet-50 FPN	40.1	78.8	49.7	44.4	79.7	54.8	0.5456

Chapter 5

Conclusion

In this work, we introduced a novel object classification method that uses compositional models to perform part based detection. The new method showed promising results on the newly introduced panoptic segmentation task. The proposed approach also showed improved results on complex scenes and was able to reason about overlapping instances well. Through the analysis of the compositional model, it can be seen that object parts were correctly mapped to the corresponding class and helped in recognizing partially occluded objects.

This work presents many new research directions to improve performance of image understanding algorithms on partially occluded scenes, and by extension, on real world data as well. It would be interesting to explore the design of end-to-end learning algorithms that use compositional modelling. This will allow control over increased number of trainable parameters. Recognizing objects by looking for associated parts may improve detection performance of partially visible instances. The aspect of learning feature representations for objects may also pave the way for a new family of detection algorithms. An important task that is brought to attention in this work is the detection of instances with discontinuities. This occurs when an occluder severely blocks the object behind it in a way that it splits the occluded object into two or more segments. Learning to reason about multiple segments that are a part of the same instance is crucial to tackling the occlusion problem.

Bibliography

[1] Edward H. Adelson. On seeing stuff: the perception of materials by humans and machines. In *Human Vision and Electronic Imaging VI*, volume 4299, pages 1–12. International Society for Optics and Photonics, 2001.

[2] Pablo Arbeláez, Jordi Pont-Tuset, Jonathan T. Barron, Ferran Marques, and Jitendra Malik. Multiscale combinatorial grouping. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 328–335, 2014.

[3] Min Bai and Raquel Urtasun. Deep watershed transform for instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5221–5229, 2017.

[4] Elie Bienenstock, Stuart Geman, and Daniel Potter. Compositionality, MDL priors, and object recognition. In *Advances in Neural Information Processing Systems*, pages 838–844, 1997.

[5] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):834–848, 2017.

[6] Liang-Chieh Chen, Maxwell Collins, Yukun Zhu, George Papandreou, Barret Zoph, Florian Schroff, Hartwig Adam, and Jon Shlens. Searching for efficient multi-scale architectures for dense image prediction. In *Advances in Neural Information Processing Systems*, pages 8699–8710, 2018.

[7] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 801–818, 2018.

[8] Qiang Chen, Anda Cheng, Xiangyu He, Peisong Wang, and Jian Cheng. Spatialflow: Bridging all tasks for panoptic segmentation. *IEEE Transactions on Circuits and Systems for Video Technology*, 2020.

[9] Yi-Ting Chen, Xiaokai Liu, and Ming-Hsuan Yang. Multi-instance object segmentation with occlusion handling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3470–3478, 2015.

[10] Bowen Cheng, Maxwell D. Collins, Yukun Zhu, Ting Liu, Thomas S. Huang, Hartwig Adam, and Liang-Chieh Chen. Panoptic-deeplab. *arXiv preprint arXiv:1910.04751*, 2019.

[11] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The Cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3213–3223, 2016.

[12] Jifeng Dai, Kaiming He, Yi Li, Shaoqing Ren, and Jian Sun. Instance-sensitive fully convolutional networks. In *Proceedings of the European Conference on Computer Vision*, pages 534–549. Springer, 2016.

[13] Jifeng Dai, Kaiming He, and Jian Sun. Instance-aware semantic segmentation via multi-task network cascades. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3150–3158, 2016.

[14] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 764–773, 2017.

[15] Daan de Geus, Panagiotis Meletis, and Gijs Dubbelman. Panoptic segmentation with a joint semantic and instance segmentation network. *arXiv preprint arXiv:1809.02110*, 2018.

[16] Daan de Geus, Panagiotis Meletis, and Gijs Dubbelman. Fast panoptic segmentation network. *IEEE Robotics and Automation Letters*, 5(2):1742–1749, 2020.

[17] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of The Royal Statistical Society, Series B*, 39(1): 1–38, 1977.

[18] Markus Enzweiler, Angela Eigenstetter, Bernt Schiele, and Dariu M. Gavrila. Multi-cue pedestrian classification with partial occlusion handling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 990–997. IEEE, 2010.

[19] Alhussein Fawzi and Pascal Frossard. Measuring the effect of nuisance variables on classifiers. In Edwin R. Hancock Richard C. Wilson and William A. P. Smith, editors, *Proceedings of the British Machine Vision Conference (BMVC)*, pages 137.1–137.12. BMVA Press, September 2016. ISBN 1-901725-59-6. doi: 10.5244/C.30.137. URL https://dx.doi.org/10.5244/C.30.137.

[20] Voxel FiftyOne. Detection on COCO. https://voxel51.com/docs/fiftyone/tutorials/evaluate_detections.html, 2020.

[21] Naiyu Gao, Yanhu Shan, Yupei Wang, Xin Zhao, Yinan Yu, Ming Yang, and Kaiqi

Huang. Ssap: Single-shot instance segmentation with affinity pyramid. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 642–651, 2019.

[22] Naiyu Gao, Yanhu Shan, Xin Zhao, and Kaiqi Huang. Learning category-and instance-aware pixel embedding for fast panoptic segmentation. *arXiv preprint arXiv:2009.13342*, 2020.

[23] Stuart Geman, Daniel F. Potter, and Zhiyi Chi. Composition systems. *Quarterly of Applied Mathematics*, 60(4):707–736, 2002. ISSN 0033569X, 15524485. URL http://www.jstor.org/stable/43638480.

[24] Dileep George, Wolfgang Lehrach, Ken Kansky, Miguel Lázaro-Gredilla, Christopher Laan, Bhaskara Marthi, Xinghua Lou, Zhaoshi Meng, Yi Liu, Huayan Wang, et al. A generative vision model that trains with high data efficiency and breaks text-based captchas. *Science*, 358(6368), 2017.

[25] Ross Girshick. Fast R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1440–1448, 2015.

[26] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Region-based convolutional networks for accurate object detection and segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(1):142–158, 2015.

[27] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Simultaneous detection and segmentation. In *Proceedings of the European Conference on Computer Vision*, pages 297–312. Springer, 2014.

[28] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Hypercolumns for object segmentation and fine-grained localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 447–456, 2015.

[29] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[30] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2961–2969, 2017.

[31] Xuming He and Stephen Gould. An exemplar-based CRF for multi-instance object segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 296–303, 2014.

[32] Derek Hoiem, Andrew N. Stein, Alexei A. Efros, and Martial Hebert. Recovering occlusion boundaries from a single image. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE, 2007.

[33] Rui Hou, Jie Li, Arjun Bhargava, Allan Raventos, Vitor Guizilini, Chao Fang, Jerome Lynch, and Adrien Gaidon. Real-time panoptic segmentation from dense detections. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8523–8532, 2020.

[34] Zhaojin Huang, Lichao Huang, Yongchao Gong, Chang Huang, and Xinggang Wang. Mask scoring R-CNN. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6409–6418, 2019.

[35] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6399–6408, 2019.

[36] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, and Piotr Dollár.

Panoptic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9404–9413, 2019.

[37] Adam Kortylewski. *Model-based image analysis for forensic shoe print recognition*. PhD thesis, University_of_Basel, 2017.

[38] Adam Kortylewski, Ju He, Qing Liu, and Alan L. Yuille. Compositional convolutional neural networks: A deep architecture with innate robustness to partial occlusion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8940–8949, 2020.

[39] Adam Kortylewski, Qing Liu, Huiyu Wang, Zhishuai Zhang, and Alan Yuille. Combining compositional models and deep networks for robust object classification under occlusion. In *The IEEE Winter Conference on Applications of Computer Vision*, pages 1333–1341, 2020.

[40] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[41] Justin Lazarow, Kwonjoon Lee, Kunyu Shi, and Zhuowen Tu. Learning instance occlusion for panoptic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10720–10729, 2020.

[42] Anat Levin and Yair Weiss. Learning to combine bottom-up and top-down segmentation. In *Proceedings of the European Conference on Computer Vision*, pages 581–594. Springer, 2006.

[43] Jie Li, Allan Raventos, Arjun Bhargava, Takaaki Tagawa, and Adrien Gaidon. Learning to fuse things and stuff. *arXiv preprint arXiv:1812.01192*, 2018.

[44] Qizhu Li, Anurag Arnab, and Philip Torr. Weakly-and semi-supervised panoptic segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 102–118, 2018.

[45] Xiangtai Li, Li Zhang, Ansheng You, Maoke Yang, Kuiyuan Yang, and Yunhai Tong. Global aggregation then local distribution in fully convolutional networks. *arXiv preprint arXiv:1909.07229*, 2019.

[46] Yanwei Li, Xinze Chen, Zheng Zhu, Lingxi Xie, Guan Huang, Dalong Du, and Xingang Wang. Attention-guided unified network for panoptic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7026–7035, 2019.

[47] Yi Li, Haozhi Qi, Jifeng Dai, Xiangyang Ji, and Yichen Wei. Fully convolutional instance-aware semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2359–2367, 2017.

[48] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *Proceedings of the European Conference on Computer Vision*, pages 740–755. Springer, 2014.

[49] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2117–2125, 2017.

[50] Huanyu Liu, Chao Peng, Changqian Yu, Jingbo Wang, Xu Liu, Gang Yu, and Wei Jiang. An end-to-end network for panoptic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6172–6181, 2019.

[51] Huanyu Liu, Chao Peng, Changqian Yu, Jingbo Wang, Xu Liu, Gang Yu, and Wei Jiang. An end-to-end network for panoptic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6172–6181, 2019.

[52] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8759–8768, 2018.

[53] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. SSD: Single shot multibox detector. In *Proceedings of the European Conference on Computer Vision*, pages 21–37. Springer, 2016.

[54] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[55] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[56] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau,

M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[57] Andra Petrovai and Sergiu Nedevschi. Multi-task network for panoptic segmentation in automated driving. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 2394–2401. IEEE, 2019.

[58] Pedro O Pinheiro, Tsung-Yi Lin, Ronan Collobert, and Piotr Dollár. Learning to refine object segments. In *European Conference on Computer Vision*, pages 75–91. Springer, 2016.

[59] Lorenzo Porzi, Samuel Rota Bulo, Aleksander Colovic, and Peter Kontschieder. Seamless scene segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8277–8286, 2019.

[60] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[61] Mengye Ren and Richard S. Zemel. End-to-end instance segmentation with recurrent attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6656–6664, 2017.

[62] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, pages 91–99, 2015.

[63] Bernardino Romera-Paredes and Philip Torr. Recurrent instance segmentation. In *Proceedings of the European Conference on Computer Vision*, pages 312–329. Springer, 2016.

[64] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115 (3):211–252, 2015.

[65] Alexander Sergeev and Mike Del Balso. Horovod: fast and easy distributed deep learning in TensorFlow. *arXiv preprint arXiv:1802.05799*, 2018.

[66] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[67] Konstantin Sofiiuk, Olga Barinova, and Anton Konushin. Adaptis: Adaptive instance selection network. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7355–7363, 2019.

[68] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 23(5):828–841, 2019.

[69] Jian Sun, Yin Li, Sing Bing Kang, and Heung-Yeung Shum. Symmetric stereo matching for occlusion handling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 399–406. IEEE, 2005.

[70] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[71] Joseph Tighe, Marc Niethammer, and Svetlana Lazebnik. Scene parsing with object instances and occlusion ordering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3748–3755, 2014.

[72] Zhuowen Tu, Xiangrong Chen, Alan L. Yuille, and Song-Chun Zhu. Image parsing: Unifying segmentation, detection, and recognition. *International Journal of Computer Vision*, 63(2):113–140, 2005.

[73] Jonas Uhrig, Marius Cordts, Uwe Franke, and Thomas Brox. Pixel-level encoding and depth layering for instance-level semantic labeling. In *German Conference on Pattern Recognition*, pages 14–25. Springer, 2016.

[74] Abhinav Valada, Johan Vertens, Ankit Dhall, and Wolfram Burgard. Adapnet: Adaptive semantic segmentation in adverse environmental conditions. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4644–4651. IEEE, 2017.

[75] Abhinav Valada, Rohit Mohan, and Wolfram Burgard. Self-supervised model adaptation for multimodal semantic segmentation. *International Journal of Computer Vision*, pages 1–47, 2019.

[76] Haochen Wang, Ruotian Luo, Michael Maire, and Greg Shakhnarovich. Pixel consensus voting for panoptic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9464–9473, 2020.

[77] Jianyu Wang, Cihang Xie, Zhishuai Zhang, Jun Zhu, Lingxi Xie, and Alan Yuille. Detecting semantic parts on partially occluded objects. *arXiv preprint arXiv:1707.07819*, 2017.

[78] Xiaoyu Wang, Tony X. Han, and Shuicheng Yan. An hog-lbp human detector with partial occlusion handling. In *2009 IEEE 12th International Conference on Computer Vision*, pages 32–39. IEEE, 2009.

[79] Xinlong Wang, Tete Xiao, Yuning Jiang, Shuai Shao, Jian Sun, and Chunhua Shen.

Repulsion loss: Detecting pedestrians in a crowd. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7774–7783, 2018.

[80] Mark Weber, Jonathon Luiten, and Bastian Leibe. Single-shot panoptic segmentation. *arXiv preprint arXiv:1911.00764*, 2019.

[81] Yuwen Xiong, Renjie Liao, Hengshuang Zhao, Rui Hu, Min Bai, Ersin Yumer, and Raquel Urtasun. UPSNet: A unified panoptic segmentation network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8818–8826, 2019.

[82] Tien-Ju Yang, Maxwell D. Collins, Yukun Zhu, Jyh-Jing Hwang, Ting Liu, Xiao Zhang, Vivienne Sze, George Papandreou, and Liang-Chieh Chen. Deeperlab: Single-shot image parser. *arXiv preprint arXiv:1902.05093*, 2019.

[83] Yibo Yang, Hongyang Li, Xia Li, Qijie Zhao, Jianlong Wu, and Zhouchen Lin. Sognet: Scene overlap graph network for panoptic segmentation. *arXiv preprint arXiv:1911.07527*, 2019.

[84] Yibo Yang, Hongyang Li, Xia Li, Qijie Zhao, Jianlong Wu, and Zhouchen Lin. Sognet: Scene overlap graph network for panoptic segmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 12637–12644, 2020.

[85] J. Yao, S. Fidler, and R. Urtasun. Describing the scene as a whole: Joint object detection, scene classification and semantic segmentation. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 702–709, 2012. doi: 10.1109/CVPR.2012.6247739.

[86] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

[87] Xiaohang Zhan, Xingang Pan, Bo Dai, Ziwei Liu, Dahua Lin, and Chen Change Loy. Self-supervised scene de-occlusion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3784–3792, 2020.

[88] Zhishuai Zhang, Cihang Xie, Jianyu Wang, Lingxi Xie, and Alan L. Yuille. Deepvoting: A robust and explainable deep network for semantic part detection under partial occlusion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1372–1380, 2018.

[89] Ziyu Zhang, Sanja Fidler, and Raquel Urtasun. Instance-level segmentation for autonomous driving with deep densely connected MRFs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 669–677, 2016.

[90] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2881–2890, 2017.

[91] Hongru Zhu, Peng Tang, Jeongho Park, Soojin Park, and Alan Yuille. Robustness of object recognition under extreme occlusion in humans and computational models. *arXiv preprint arXiv:1905.04598*, 2019.

[92] Yan Zhu, Yuandong Tian, Dimitris Metaxas, and Piotr Dollár. Semantic amodal segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1464–1472, 2017.

Appendices

Appendix A

Additional Results

A comparison of the results of our approach with the baseline using the ResNet-101 FPN backbone is shown in figures A.1 and A.2. As seen from the figures, using a ResNet-101 backbone improves segmentation performance. Many examples also show the improvement of our approach over the baseline. Our approach makes a noticeable improvement in reasoning about instances in complex scenes over the baseline.

In Figure A.1, the first example shows a complex scene with many annotated instances. As seen, our approach reliably separates densely packed partially occluded instances. The same observation can be made for the second and third examples as well. Figure A.2 shows some more results of our approach using the ResNet-101 backbone. Rows 1, 3 and 5 show examples of crowded scenes where our approach shows an improvement over the baseline. It is interesting to note that our approach detects and segments many instances that are not even present in the ground truth labels. Row 3 shows an example that is segmented well by our approach despite having an unusual perspective with a partially visible *Zebra* instance occluding another instance of the same class.

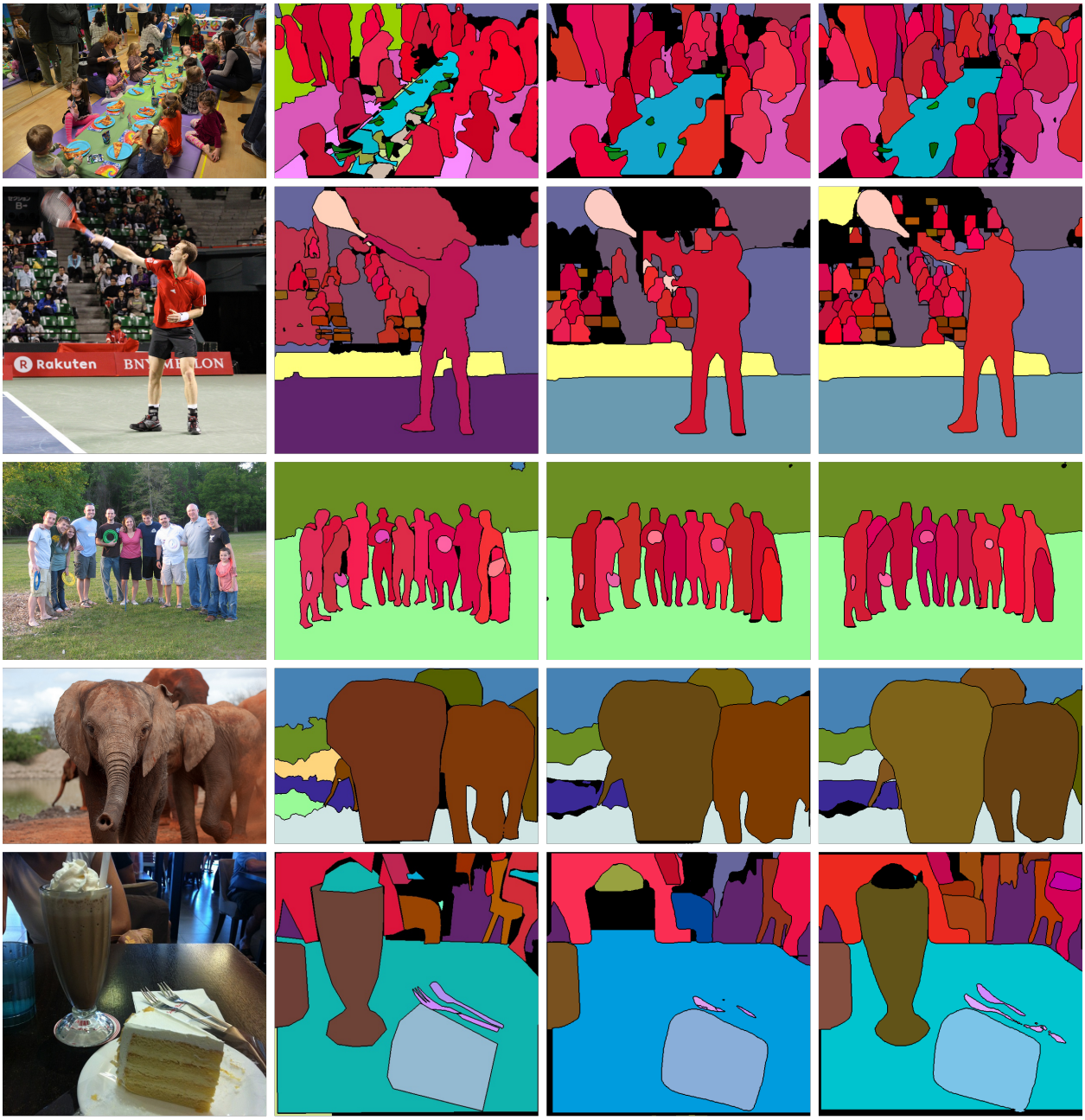

Figure A.1: Results of our approach using ResNet-101 on the COCO *Val* dataset are shown in this figure. The first example shows a complex scene with many annotated instances. As seen, our approach reliably separates densely packed partially occluded instances. The same observation can be made for the second and third examples as well.

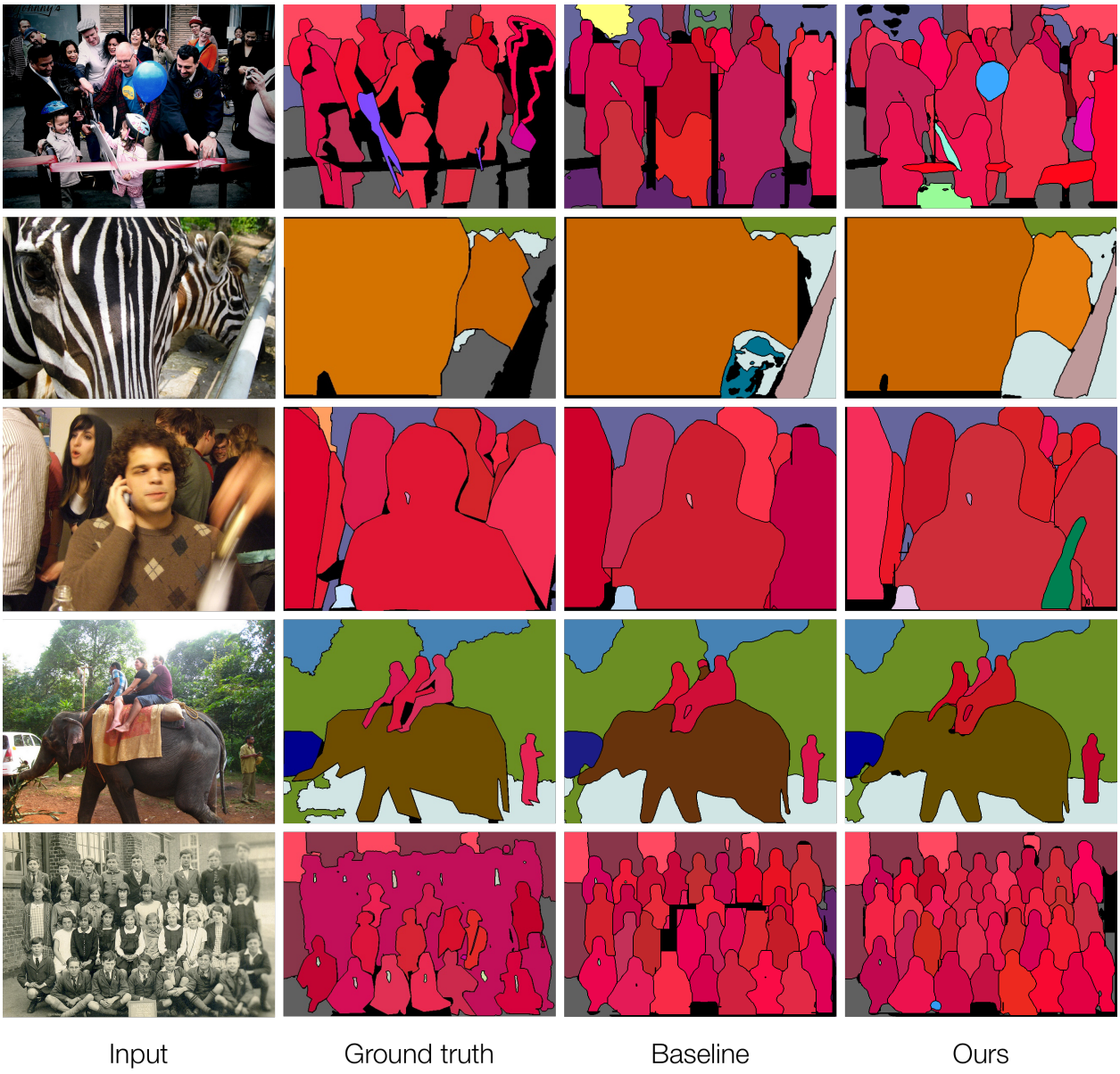

Figure A.2: This figure shows some more results of our approach using the ResNet-101 backbone. Rows 1, 3 and 5 show examples of crowded scenes where our approach shows an improvement over the baseline. It is interesting to note that our approach detects and segments many instances that are not even present in the ground truth labels. Row 3 shows an example that is segmented well by our approach despite having an unusual perspective with a partially visible *Zebra* instance occluding another instance of the same class.