Instance Segmentation with Occlusion Order Supervision: Two Problems

by

Cheuk-To Jeremy Yu

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2025
© Cheuk-To Jeremy Yu 2025

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Joint architectures for predicting instance masks with additional depth-related information have shown improvements in performance on both the original segmentation tasks and their corresponding depth-related tasks. Closely related to depth, occlusion can also provide strong cues for segmentation. Although occlusion ordering between instances provides less supervision than full pixel-wise depth, occlusions can be easily annotated for existing datasets. Motivated by the above, we propose two problems that incorporate occlusion information into standard segmentation tasks with their corresponding methods. For both methods, we explore the level of supervision occlusion provides in terms of segmentation accuracy by comparing our results with baseline segmentation approaches trained without occlusion supervision.

Firstly, we develop an end-to-end framework to perform instance segmentation and per-instance global occlusion order prediction simultaneously by appending a global occlusion order head to standard instance segmentation architectures. Our approach to occlusions differs from most prior work. Prior work performs occlusion estimation in a local pairwise manner: given two instances, classify one of them as the occluder. Due to locality, for scenes known not to have occlusion cycles, such approaches can (and do) produce occlusion cycles, which are errors. Unlike most prior work, we directly label instances with their occlusion-order labels, where an instance with a larger label occludes any neighboring instances with smaller labels. This approach is cycle-free by design. Using cross-entropy with occlusion-order labels fails as occlusion-order labels do not have a fixed semantic meaning. Therefore, we develop a novel regularized loss function for successful training. Our framework achieves high occlusion-order accuracy with improved performance in instance segmentation, likely due to the added supervision.

Secondly, we develop a new direction for occlusion-supervised amodal instance segmentation (AIS). AIS is an emerging task that segments the complete object instance, both the visible and occluded parts. All prior work for AIS can be divided into two groups: (i) methods constructing a synthetic amodal dataset; (ii) methods using human-annotated amodal masks. The drawback of methods in the first group is that constructing a realistic synthetic dataset is difficult. The drawback of methods in the second group is that human annotation is prone to error, as humans must reason about invisible regions. Our method requires neither a synthetic dataset nor human-annotated amodal masks, but instead uses ground-truth modal masks and the pairwise occlusion order between instances. By using the occlusion order to reason about the visible and invisible parts of the amodal mask, we develop effective loss functions for the visible and occluded parts of the predicted mask. We achieve comparable accuracies to those of the same architecture trained on human-annotated amodal masks, despite not using amodal masks for training.

Acknowledgment

I would like to offer my deepest and sincerest gratitude to my supervisor, Professor Olga Veksler, who has been a guiding light during my time at Waterloo. Her expertise and knowledge have been instrumental to my work and growth, and her kindness and patience have encouraged me to challenge myself and be introspective. Thank you for offering me an opportunity that I will cherish forever.

I am also deeply grateful to my thesis committee members, Professor Yuri Boykov and Professor Shlomi Steinberg. Thank you for spending time on my thesis; your feedback and evaluations are invaluable and deeply appreciated.

I would also like to thank Soroosh Baselizadeh, who was willing to spend time to guide me through his papers and allowed me to aid in his work despite my lack of experience during my early terms at Waterloo. You have given me the greatest crash course in segmentation research and have inspired me to work on occlusion ordering-related tasks.

Thank you to Lori Paniak, Indy Ng, Lucas Gamez, as well as other members of CSCF, for maintaining the WATGPU clusters, which have served as the primary computational resource throughout my time at Waterloo. You have been incredibly helpful, even during weekends, with my issues and inquiries.

Dedication

To my family and friends, for they are my motivation and inspiration, who allowed me to be who I am.

Contents

Author's Declaration ii

Abstract iii

Acknowledgment iv

Dedication v

1 Introduction 1
- 1.1 Thesis Organization . . . 5

2 Background 6
- 2.1 Image Classification . . . 6
  - 2.1.1 Convolutional Neural Networks . . . 7
  - 2.1.2 Vision Transformers . . . 9
  - 2.1.3 Training Loss . . . 11
- 2.2 Object Detection . . . 12
  - 2.2.1 R-CNN . . . 13
  - 2.2.2 DETR . . . 14
  - 2.2.3 Training Loss . . . 15
- 2.3 Semantic Segmentation . . . 18
- 2.4 Instance Segmentation . . . 19
  - 2.4.1 Mask R-CNN . . . 19
  - 2.4.2 Mask2Former . . . 20
  - 2.4.3 Training Loss . . . 21
- 2.5 Amodal Instance Segmentation . . . 22
- 2.6 Depth Estimation . . . 23
  - 2.6.1 Monocular Depth Estimation . . . 24
  - 2.6.2 Training Loss . . . 24

3 Related Works 26
- 3.1 Combining Depth and Segmentation . . . 26
- 3.2 Occlusion Estimation . . . 27
- 3.3 Combining Occlusion and Segmentation . . . 28
- 3.4 Amodal Mask Completion . . . 30
  - 3.4.1 Amodal Content Completion . . . 31
  - 3.4.2 Generative Adversarial Networks . . . 32

4 Datasets **35**
4.1 KINS . . . 35
4.2 COCOA . . . 36
4.3 Other Datasets . . . 36

5 Problem 1: End-to-End Occlusion Ordered Semantic Instance Segmentation **39**
5.1 Methodology . . . 41
5.2 Experimental Results . . . 43
5.2.1 Architecture and Training Details . . . 43
5.2.2 Joint Evaluation of Segmentation And Occlusion Order Accuracy . . . 43
5.2.3 Evaluation of Segmentation Accuracy . . . 44
5.2.4 Ablations . . . 46
5.2.5 Depth and Occlusion Correlation . . . 46
5.2.6 Amodal Instance Segmentation . . . 47
5.2.7 Qualitative Results . . . 47
5.3 Application: Improving Depth Maps . . . 49
5.3.1 Methodology . . . 49
5.3.2 Architecture and Training Details . . . 50
5.3.3 Depth Accuracy . . . 50
5.3.4 Qualitative Results . . . 51

6 Problem 2: Occlusion-Supervised Amodal Instance Segmentation **52**
6.1 Methodology . . . 54
6.1.1 Assumptions . . . 54
6.1.2 Visible Losses . . . 57
6.1.3 Occluded Loss . . . 58
6.2 Experimental Results . . . 59
6.2.1 Architecture and Training Details . . . 59
6.2.2 Segmentation Accuracy . . . 61
6.2.3 Ablations . . . 62
6.2.4 Alternative Generator Loss . . . 62
6.2.5 Mixing Amodal Supervision . . . 62
6.2.6 Qualitative Results . . . 63

7 Conclusion and Future Work **66**

Bibliography **67**

Appendices **78**

A Proofs **79**
A.1 Proof for Proposition 6.1.1 . . . 79
A.2 Proof for Proposition 6.1.2 . . . 81
A.3 Proof for Proposition 6.1.3 . . . 82

B Connection Between Eq. (6.4) and Volume Rendering **83**

C Approximation of Eq. (6.4) for Numerical Stability **85**
C.1 Numerical Stability of $\ln(1 + e^x)$ 86
C.2 Numerical Stability of $\ln(e^x - 1)$ 86
C.3 Numerical Stability of $\ln(\Pi_i(1 + e^{x_i}) - 1)$ 87

Chapter 1

Introduction



(a) Original image. (b) Instance segmentation masks.

Figure 1.1: Examples of instance segmentation annotation from the KITTI 2015 dataset [1], which marks out visible regions of common road scene objects such as cars and pedestrians.

Instance segmentation, also known as semantic instance segmentation, refers to the task of predicting the visible region of each object at the pixel level, often performed in conjunction with predicting the class (i.e., the semantic meaning) of the object. Building upon techniques for image classification or object detection, instance segmentation models have matured since 2017, with countless variations of the classic R-CNN-based [33] architectures [36], as well as the newer DETR-based [8] approaches [13, 61], which have quickly become the state-of-the-art in recent years.

To help push the envelope, many works have started to incorporate more supervision outside of simply relying on ground-truth instance masks, as an attempt to improve segmentation accuracy for their specific applications. For example, in *panoptic segmentation*, it combines both semantic and instance segmentation, aiming to predict "stuff" (i.e., uncountable regions such as the sky or ocean) in addition to just the "things" (i.e., countable instances) in standard instance segmentation [50, 12, 14, 13]. Whereas in [80, 116, 105], the authors rely on human joint locations (in the form of keypoints) to perform human pose

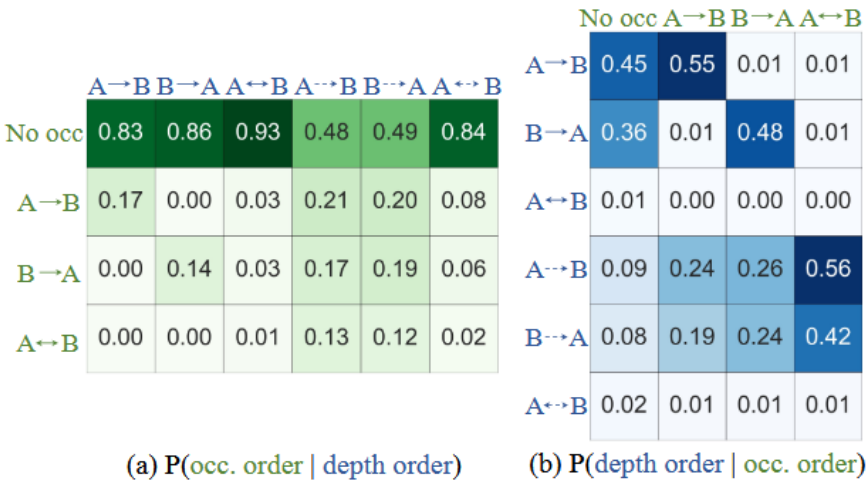
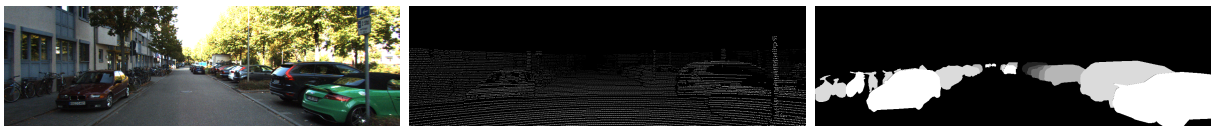


Figure 1.2: Correlation between pairwise occlusion and depth relations of instances. The arrow represents the direction of occlusion/depth order between instances A and B. The dotted arrows represent partial depth orders, where there can be overlaps between the minimum or maximum depth values from both instances. [59]

estimation and segmentation. These tasks tend to improve segmentation performance over training with only segmentation supervision. This might be unsurprising; in the case of panoptic segmentation, knowing the "stuff" helps with distinguishing the "things", whereas in pose estimation, knowing the exact pose of a person helps with outlining the person. In both tasks, the extra supervision encodes additional scene information, which in turn helps with the segmentation task at hand.

Over the past years, there has been some work that addresses instance segmentation with some form of depth prediction. In [118, 109, 29], the models jointly predict a per-pixel depth estimate along with panoptic or instance segmentation, while [119] predicts a per-instance depth ordering. Similar to panoptic segmentation and pose estimation, these methods showed improved performance on both the instance segmentation task and their corresponding depth estimation task, suggesting that these tasks are closely related, as segment boundaries usually indicate depth discontinuities.

What about *occlusion*? While occlusion is not exactly depth, [59] showed that pairwise occlusion and pairwise depth order are strongly correlated, suggesting that occlusion and depth are complementary to each other when describing a 3D scene. After all, both occlusion and depth encode some form of 3D scene information: "occlusion" refers to some area of one object in front of another object, whereas "depth" directly gives the distance of objects to the camera. Occlusion and depth also have a strong correlation, as shown in Fig. 1.2, with high covariance when the pairwise depth order respects the pairwise occlusion order (i.e., $P(A \rightarrow B | A \rightarrow B)$ and $P(B \rightarrow A | B \rightarrow A)$). Naturally, depth and occlusion are strong mutual cues, as occlusion boundaries are usually depth discontinuities, with the occlusion order determining the direction of depth discontinuities at these boundaries. This is also reflected experimentally in the form of InstaOrderNet [59]. InstaOrderNet is a classification model that takes 2 masks as inputs and outputs a pairwise depth and/or occlusion order between the masks (using the same categories as Fig. 1.2). When trained to output both depth and occlusion orders, the model outperforms those trained on either depth or occlusion orders in their respective metrics, suggesting that the depth and occlusion prediction



(a) Original image. (b) Depth from KITTI [31]. (c) Occlusion from KINS [82].

Figure 1.3: An example of annotations for a given image.

tasks can enhance each other, similar to how depth enhances segmentation training.

If occlusion helps with depth, and depth helps with segmentation, what about combining segmentation and occlusion for training? Occlusions are strong cues for segmentation after all, as occlusion boundaries *are* object boundaries. Admittedly, using occlusion to improve segmentation presents several obstacles compared to simply using depth. Occlusion-annotated datasets are far rarer than datasets with full pixel-wise depth annotations, and occlusion annotations between datasets do not have a very consistent format or definition (see Chapter 4). Depth annotations are also easier to use; the pixel-wise depth annotation can be more easily and intuitively incorporated into many standard segmentation models, which is reflected by the volume of work that combines depth and segmentation (see Section 3.1. Last but not least, pixel-wise depth annotations also provide significantly more 3D scene information than occlusion order annotations between objects. After all, pairwise occlusion order annotations are simply labels for each pair of objects. Occlusion order labeling is sparse; it can be common for objects to have few or no occlusion relations at all, depending on the dataset. In the case of the KINS [82] and COCOA [123] datasets, only around 8.6% and 8.5% of object pairs are in an occlusion order, respectively.

However, occlusion order does have its merits. Occlusion order can be easily annotated by humans without requiring any extra equipment (e.g., LiDAR scanners for depth, as shown in Fig. 1.3b). Occlusion order can also be annotated retrospectively to existing datasets quickly with very few errors, as humans are highly skilled at ordering objects in images [15]. Also, training for occlusion reasoning as opposed to depth might be sufficient depending on the applications, such as question answering [73], scene de-occlusion [113], and amodal instance segmentation [62]. Fig. 1.3 shows an example of depth annotation and occlusion annotation for a given image.

Motivated by the above, we propose two problems that incorporate occlusion information into standard segmentation tasks. To enable the use of pairwise occlusion annotations, our methods for each of the problems rely on two different sets of assumptions. In both of these problems, we explore the level of supervision occlusion provides in terms of segmentation accuracy, by comparing our results with baseline segmentation approaches without any occlusion supervision. For both problems, we train and evaluate on the KINS [82] and COCOA [123] dataset, which contains visible, amodal segmentation masks, as well as occlusion order annotations.

In this work, we propose a method for *End-to-End Occlusion-Ordered Semantic Instance Segmentation*, which addresses the same problem proposed in [4]. The model augments a standard instance segmentation architecture by outputting a global occlusion order per instance, in addition to the instance segmentation masks via appending an "occlusion order" head, which is side-by-side with the class prediction head. The occlusion order head is responsible for predicting a single real number representing the global occlusion order, which respects the occlusion order between instances: for neighboring instances, the instance with



(a) Original image. (b) Segmentation masks. (c) Occlusion graph.

Figure 1.4: An example from the InstaOrder dataset [59]. The arrows with two heads in (c) are examples of occlusion cycles.

the higher global occlusion order would be the occluder. To train the occlusion order head to follow this rule, we propose a simple ReLU-based regularized loss, inspired by [98, 97]. Here, we use Mask R-CNN [36] and Mask2Former [13], two very different architectures for instance segmentation, as our baselines for comparison.

The global occlusion order naturally assumes that there are no occlusion cycles within a scene. Of course, in reality, objects can be in an occlusion cycle; Fig. 1.4 shows an example of an occlusion cycle, in which the person's left leg occludes the horse, which occludes the right leg. However, the rarity of such cycles largely depends on the domain (i.e., what kind of scene) and object definitions (i.e., what counts as an object). For example, road scenes such as those in Fig. 1.3 can have little to no occlusion cycles, as vehicles tend to be distant from each other and relatively more "convex" compared to most objects. In the case of KINS and COCOA, neither dataset contains any occlusion cycles. With the predicted global occlusion order, we can recover the relative order between any neighboring instances. The applications include question answering [73], scene de-occlusion [113], and amodal instance segmentation [62].

We evaluate our approach on the OAIR metric proposed in [4], which jointly evaluates the accuracy of the predicted masks and occlusion order. Our method showed improved performance compared to [4] while being much more computationally efficient. Compared to baseline models without occlusion-order supervision, our method also shows an improvement in instance segmentation metrics, likely due to the added information from the ground-truth occlusion order. We also propose an additional application in the form of depth sharpening using a test-time training approach, based on the global occlusion order predictions of our architecture.

Secondly, we propose a method for *Occlusion-Supervised Amodal Instance Segmentation*, which addresses *amodal instance segmentation* with only supervision from ground-truth visible segmentation masks and occlusion order. Our method allows us to use existing single-stage, non-Region-of-Interest-based instance segmentation architectures for training, and relies on an assumption about the dataset: for a given image, the total area of amodal masks must be the total area of modal masks. We propose two effective losses that act upon the visible and invisible parts of each object, respectively.

Here, Mask2Former [13] is used as our baseline model. We evaluate our approaches via AP scores. Our approach results in performance close to that of the same architecture trained on human-annotated amodal masks, even though we only use modal masks for

training. Our approach can be extended to other methods of incorporating shape priors.

1.1 Thesis Organization

Chapter 2 contains the background and walks through some of the foundations, as well as any important models and architectures, notably Mask2Former, Mask R-CNN, ResNet, and GLPDepth, which will be used in some capacity for our experiments. In Chapter 3, we describe related works. Chapter 4 explains the choice of datasets. Chapter 5 details our end-to-end occlusion-ordered semantic instance segmentation approach, along with an application in the form of depth map test-time training. In Chapter 6, we detail a possible framework for occlusion-supervised amodal instance segmentation.

Chapter 2

Background

2.1 Image Classification

Perhaps one of the most well-studied and foundational problems within computer vision and image processing–image classification (or object recognition) refers to the task of assigning a label to a given image. Image classification has a vast number of applications, including face recognition, car/pedestrian recognition for parking/traffic systems, as well as biomedical imaging. It also serves as the basis of many branching problems within computer vision.

Traditionally, image classification builds upon previous techniques in low-level feature recognition. [103] proposed a framework for face recognition by generating an optimal series of Haar filters via AdaBoost. Designed to be computationally efficient, Haar filters are a specific type of full-image filter that consists only of rectangular regions of ± 1 weight. A positive response produced by applying the generated Haar filter to an image corresponds to a human face. Experimentally, the chosen filters correspond to common bright & dark features on faces; as shown in Fig. 2.1, the left filter corresponds to the dark eye regions and bright upper cheeks, whereas the right filter corresponds to the bright nose ridge.

In [72], Lowe proposed the *Scale-Invariant Feature Transform* (SIFT), which provides a scale and rotation-invariant algorithmic approach to low-level object recognition. Given two images of the same scene, SIFT extracts key locations that correspond to unique local

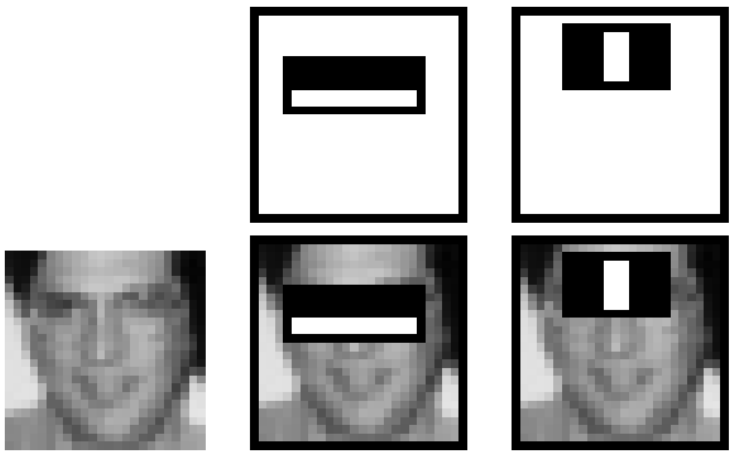


Figure 2.1: Haar filters. [103]

features. Scale-invariance is achieved by using scale pyramids by resizing the image in factors of 2, while rotation-invariance is achieved by selecting locations at the maxima and minima of the difference of Gaussian. The features between the two images are then matched. If there is a known object in one of the images, the object can then be recognized in the other image based on the location of the matched local features of the object.

Inspired by SIFT, [19] proposes using *Histograms of Oriented Gradients* (HOGs) to detect pedestrians. This is done by calculating the gradient orientation and magnitude at fixed locations on an image, and creating a histogram of the gradient orientation and magnitude over a given region. The resulting vector can then be fed through a classifier, which [19] implements via SVM as the baseline. While not completely translation/rotation-invariant, a small degree of invariance is allowed and can be controlled by adjusting the bin size, which proves sufficient as pedestrians are usually already in an upright position. HOGs showed strong performance in pedestrian detection while being significantly cheaper in computation compared to previous works.

While these methods performed well for their specific tasks, their robustness is far from ideal, as they require careful calibration in both the algorithm and the input images. [19, 103] requires humans/faces to be centered and cropped for their specific applications, whereas [72] requires multiple images of the same scene to perform feature matching. Naturally, the ever-increasing demand for robustness led to the creation of more intricate systems to tackle such challenges–or more specifically, the adoption of deep learning.

2.1.1 Convolutional Neural Networks

At its core, *neural networks* refer to systems with connected *neurons*, which aim to mimic their biological counterparts. These neurons are commonly organized in layers, with connections between neurons of neighboring layers. Each neuron processes signals from upstream connections, usually by calculating a weighted sum of these signals, and outputs them downstream. The signals between neurons commonly go through an *activation function*–a simple non-linear function that guarantees non-linearity for a multi-layer neural network based on the universal approximation theorem [18]. Such multi-layered neural networks are also known as deep neural networks (DNNs) or deep learning.

Up until the late 1980s, neural networks have been shown to generalize well on regression tasks [89]. This period also saw the advent and application of many modern techniques in the context of deep learning, such as backpropagation [92, 106], ReLU activation functions [27], and stochastic gradient descent [2].

Convolutional neural networks (CNNs) refer to neural networks that optimize training efficiencies by applying weights in the form of *kernels* to the input or signals between the layers. As opposed to fully connecting neurons between layers, each neuron only has access to signals from spatially nearby neurons from the upstream layer. Using kernels allows the neural networks to be *shift-invariant* in terms of the input, which can be desirable depending on the problem. The reduced number of connections and parameters also drastically improves training efficiency. Perhaps one of the earliest works on CNNs is Waibel's Time-Delay Neural Network (TDNN) [104], which performs phoneme recognition via a 1D CNN based on 1D audio inputs, with convolution along the "time" axis. Here, shift-invariance is useful as any slight misalignment of the audio file should not affect the model's performance.

Building upon Waibel's work, LeCun proposed the 2D CNN for handwritten zip code recognition [57], which is the first application of CNN in an image context. The model

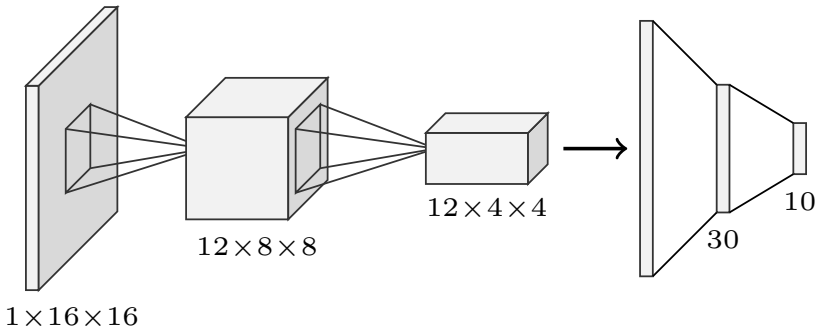


Figure 2.2: Model architecture for [57]. All convolution operations are performed with 5×5 kernels at a stride of 2.

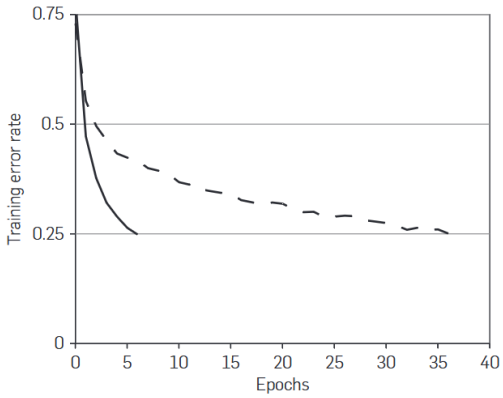


Figure 2.3: Performance of ReLU vs tanh activation function from AlexNet. Solid line: ReLU, dashed line: tanh. [53]

consists of 2 convolutional layers with 5×5 kernels on a 16×16 grayscale input. To reduce the number of parameters, convolution was performed with a stride of 2, thereby halving the dimension of the input at every layer. The outputs from the second convolutional layer were flattened and sent to a fully connected layer as shown in Fig. 2.2. The final class probabilities were derived using the softmax function, with cross-entropy as the training loss

Coming from 1989, LeCun's work already resulted in a very modern design of CNNs. Improving on the previous model, in 1998, LeCun introduced the LeNet-5 [58], which was larger with 7 convolutional layers. LeNet-5 also introduced pooling layers, or, as the paper calls them, sub-sampling layers.

Skipping to 2012, the next breakthrough comes in the form of AlexNet [53], which was the first to introduce ReLU activation functions to CNNs, which had predominantly been using tanh activation functions, resulting in training that was several times faster, as shown in Fig. 2.3. To combat overfitting, the model was the first to employ dropouts in CNNs, and suggested a few methods for data augmentation. AlexNet shows improvement against SIFT-based methods.

In 2015, Microsoft proposed *residual networks* (ResNets) [37]. One of the major problems of deep neural networks is that training naturally gets more difficult as more layers are added, even after applying existing techniques to guard against vanishing gradi-

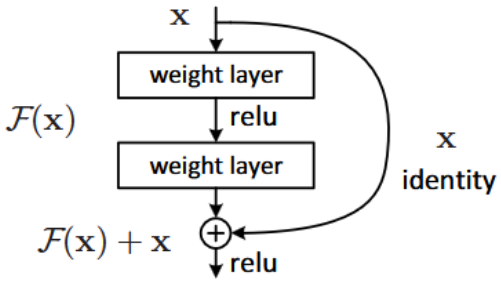


Figure 2.4: An example of a building block from ResNet. [37]

ent/overfitting. To address this issue, [37] proposed the *residual learning* framework. Under residual learning, given a block of neural network layers, the output of the block is trained to output the difference between the input and the desired output (hence "residual learning"), as shown in Fig. 2.4. In other words, it simply adds the input to the output of the block, thereby preserving many of the original input features at each layer. Not only does this help with vanishing gradient as this explicitly exposes output from the previous layer at each subsequent layer, this also allows the networks to more explicitly "memorize" the original image or lower level image features, which, in use cases such as image supersampling or denoising, can be very important. With residual learning, ResNet was able to create extremely deep CNNs for its time, with up to 152 layers, significantly improving the state-of-the-art classification on the CIFAR-10 dataset [52].

To this day, ResNet remains one of the most popular models to build upon for computer vision-related tasks, commonly used as a *backbone* for many current models. Unsurprisingly, it is the most cited computer vision paper as of April 2025 [101]. For our experiments in Chapter 5 and 6, we will be using ResNet-50 (denoting 50 layers) as the backbone.

2.1.2 Vision Transformers

Within the landscape of natural language processing, *transformer*-based architecture remains the most prominent type of model today. Transformers [102] rely on the *attention* mechanism, which explicitly trains a mapping from a set of input queries and key-value pairs to some output. In the context of the language tasks, these queries and keys usually come from embeddings of tokens for a given sequence. As for the token embeddings, a positional encoding is added (usually based on some sine/cosine functions) to inject some information on the position of the token in the sequence. Attention allows long-range correlations and dependencies within a sequence of tokens to be mapped more explicitly, which directly improves upon the local nature of convolutional neural networks. With the attention mechanism, their transformer (see Fig. 2.5) was able to provide far more interpretable results while outperforming recurrent neural networks (RNNs) in language translation tasks, which was the norm before the meteoric rise of transformer-based large language models that we are familiar with today.

Adapting the transformers of 1D language-related tasks to 2D vision-related tasks, [23] proposed the *vision transformer* (ViT) in 2020. Instead of tokens in a sequence, ViTs break up an image into flattened 2D patches to learn the embeddings, as shown in Fig. 2.6. As for positional embedding, experimentally, using a 2D focused positional embedding did not outperform using the standard 1D positional embedding by simply treating the patches as a 1D sequence. ViTs also allow for combination with CNNs by using CNN feature maps

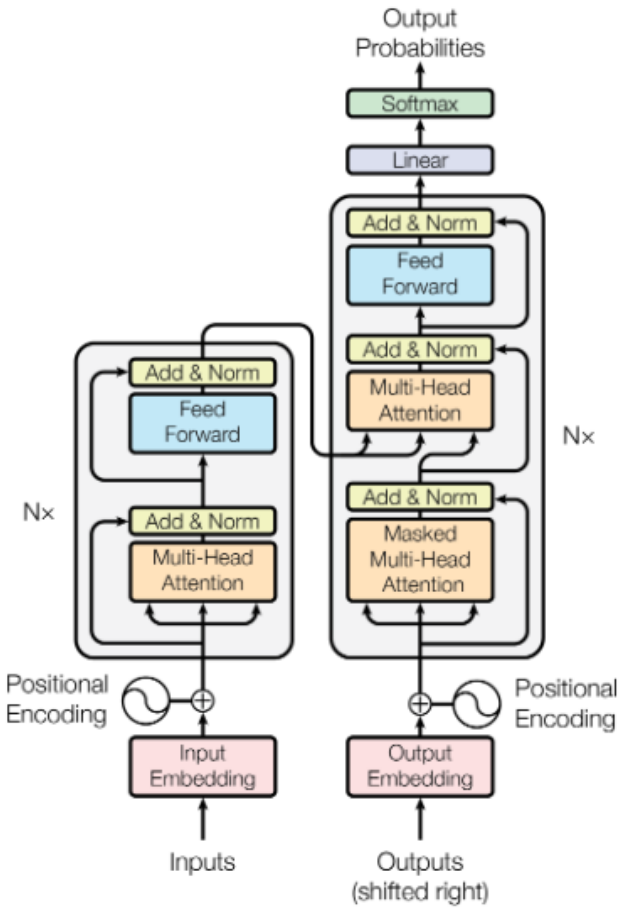


Figure 2.5: Transformer architecture. [102]

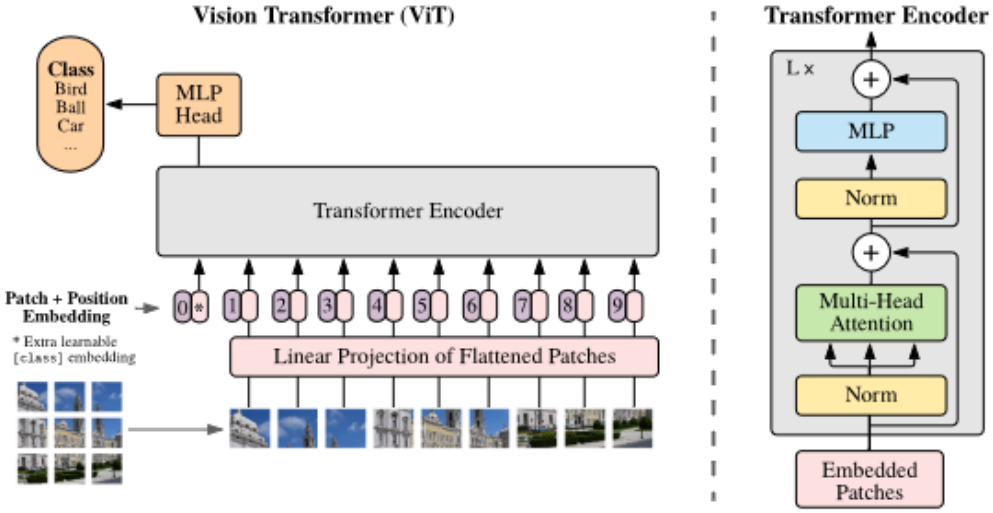
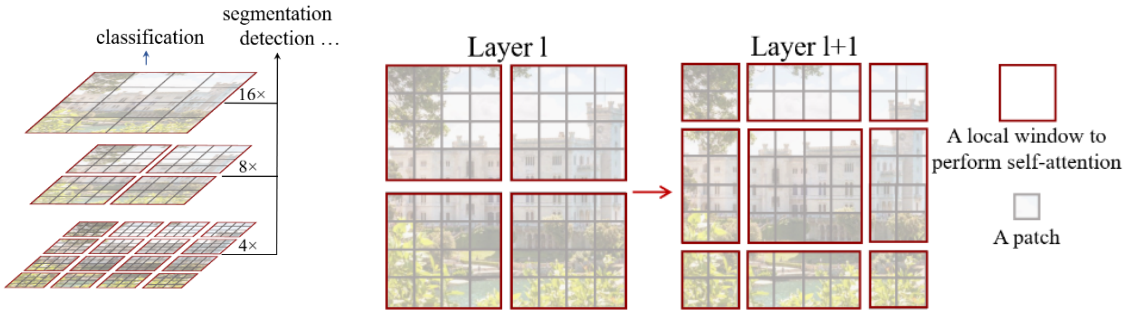


Figure 2.6: Vision transformer architecture. [23]



(a) Hierarchy. (b) Shifted window approach.

Figure 2.7: Swin Transformer characteristics. [70]

instead of the raw image as input. ViTs were able to achieve comparable performances to the then state-of-the-art CNNs in classification tasks.

Improving upon ViTs, [70] introduced a hierarchical transformer with *s*hifted *win*dows, also known as Swin Transformers. Swin Transformers improves upon ViTs by creating a hierarchical architecture through merging windows at increasing layers, as shown in Fig. 2.7a. Attention is done within each window at a fixed sequence size of 16 patches, which lowers the training runtime complexity to linear from the quadratic complexity of ViTs. The hierarchy also allows Swin Transformer to extend more easily outside of image classification problems, and can be modeled similarly to U-Nets or feature pyramid networks (FPN), which will be mentioned in Section 2.3 and 2.4. Swin Transformer also employed a shifted window approach between layers as shown in Fig. 2.7b, which makes the window boundary more arbitrary. Swin Transformer was able to outperform other ViT-based methods and ResNet in many computer vision-related tasks, resulting in yet another popular choice of backbone for many modern architectures.

2.1.3 Training Loss

The most commonly used loss for classification is the cross-entropy (CE) loss:

$$CE(y, y^*) = -\sum_{i}^{N}\sum_{c}^{K} y_i^{*c} \log(y_i^c) \tag{2.1}$$

Where N is the size of the dataset and K is the number of classes. For sample i, y_i^c represents the predicted probability for class c, and y_i^* represents the target class distribution. One common variant of the cross-entropy loss is the weighted cross-entropy:

$$WCE(y, y^*) = -\sum_{i}^{N}\sum_{c}^{K} w_c y^{*c}_i \log(y_i^c) \tag{2.2}$$

Which weights the cross-entropy with the class-based w_c. The weighted cross-entropy is commonly used to address class imbalance in the dataset. Another variant is the binary cross-entropy loss, which is a special case of cross-entropy when dealing with only 2 classes (i.e., binary classification):

$$BCE(y, y^*) = -\sum_{i}^{N} (y_i^* \log(y_i) + (1 - y_i^*) \log(1 - y_i)) \tag{2.3}$$

To convert from the final logits $z_i^c \in \mathbb{R}$ to the probabilities $y_i^c \in [0, 1]$, a softmax operator is applied:

$$y_i^c = \text{softmax}(z_i, c) = \frac{e^{z_i^c}}{\sum_{j=1}^{K} e^{z_i^j}} \tag{2.4}$$

In the case of binary classification, sigmoid, which is a special case of softmax for 2 classes, can be used instead:

$$y_i = \sigma(z_i) = \frac{1}{1 + e^{-z_i}} \tag{2.5}$$

Note that when combining softmax/sigmoid with the cross-entropy losses, it can result in taking the natural log of a probability; when the probability approaches 0, the log approaches $-\infty$, which can cause numerical issues in practice. This can be solved by combining the softmax/sigmoid and cross-entropy loss into one operation and taking advantage of the *log-sum-exp* trick, which is implemented in most popular machine learning libraries (e.g., `torch.nn.BCEWithLogitsLoss`).

2.2 Object Detection

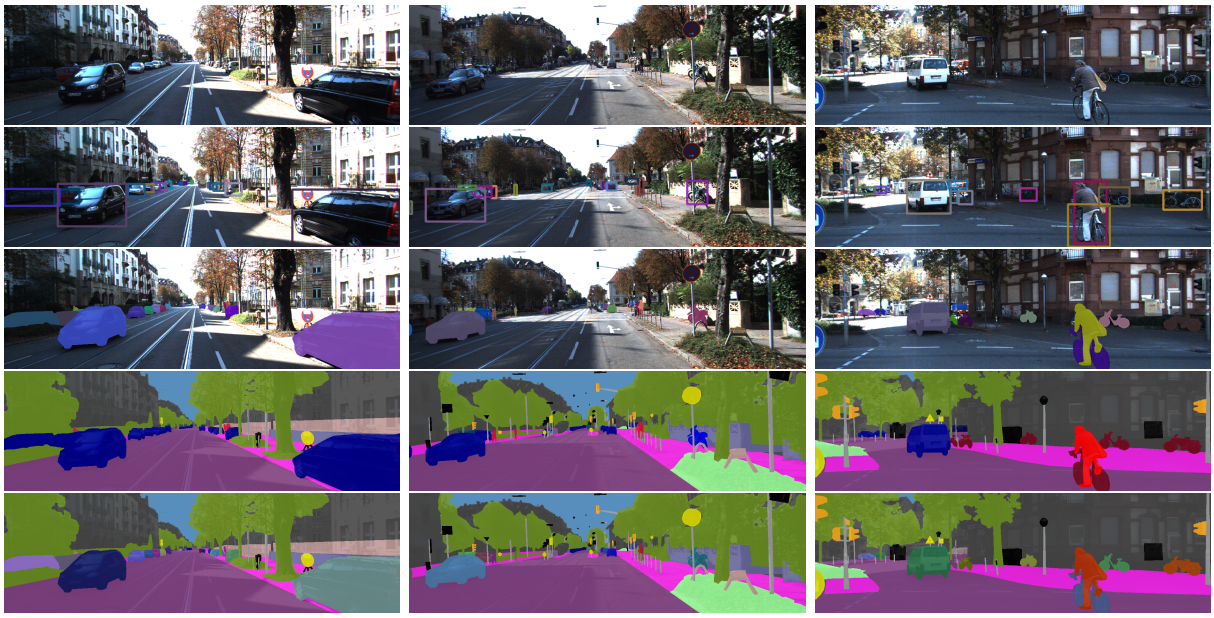
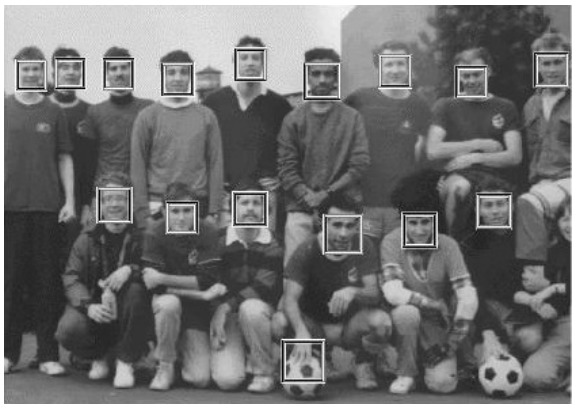


Figure 2.8: Various computer vision tasks. From top to bottom: original image, object detection, instance segmentation, semantic segmentation, panoptic segmentation. Taken from the KITTI 2015 dataset [1].

Within computer vision, object detection or localization refers to the task of locating object instances within a given image, and is commonly done in tandem with classification, where the class (e.g. person or car) of the object needs to be identified as well. This is commonly done via predicting the *bounding box* of each object, which consists of a center point (x, y) and the size (w, h), as shown in Row 2 of Fig. 2.8. Object detection encompasses a wide range of applications, including facial recognition [103], autonomous driving, medical imaging, and video surveillance.



(a) Haar filters applied via sliding windows. [103]



(b) CAMs. [121]

Figure 2.9: Examples from prior works.

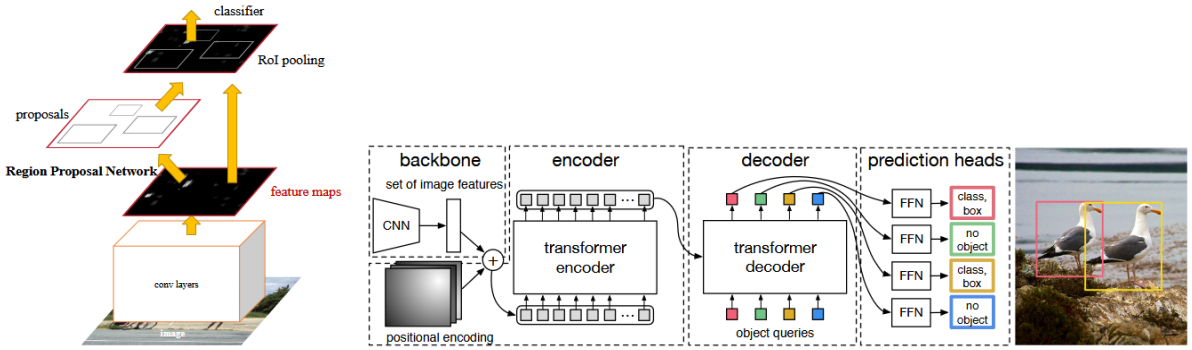
Unsurprisingly, object detection and image classification are closely tied together. In many image classification publications, the locations of objects classified have historically been a major consideration, almost treated as if they are two sides of the same coin. After all, one can always take an image classification model and apply it to an image via a sliding window to get a coarse position of the classified object (especially if the method is efficient and/or relies on low-level image features), which is the case and the proposed method in the original papers for Haar filters [103], as shown in Fig. 2.9a.

Some image classification architectures may also already contain information on the location of the target object. In SIFT-based methods [72] such as HOGs [19], this is somewhat explicit as the position of matched features is known, and can be used to extrapolate the general location of the entire object. This is also true for convolutional neural networks (CNNs). With the rise of CNNs, which worked well and robustly for image classification tasks, it naturally lends itself to semantic segmentation and object detection – with [121] even bridging the gap between image classification and object localization via *class activation maps* (CAMs). Given a model that uses *global average pooling* (GAP) at the final convolutional layer, CAMs are a way to visualize the region of an image that gives high responses for a given class during image classification by reverse engineering the final layers of the model. To calculate the CAM for a given class c, the class weight w_c is used as the weight to calculate the weighted sum of the last convolutional layer before the GAP layer. By visualizing regions of high class response, it is possible to locate the object, as shown in Fig. 2.9b.

In the following sections, we will explore a few notable deep-learning-based architectures that serve as a foundation to the baseline models that we will be using throughout our experiments.

2.2.1 R-CNN

Object detection has given rise to a myriad of deep learning-based architectures throughout the years, with some of them being the *Region of Interest*-based (RoI) approaches. Perhaps one of the most significant families of architectures is the *Region-based Convolutional Neural Networks* (R-CNN) [33], which introduces the task of extracting *region proposals*, which are the potential bounding boxes for instances (i.e. RoI). While the design of R-CNN is agnostic



(a) Faster R-CNN. [86] (b) DETR. [8]

Figure 2.10: Various architectures.

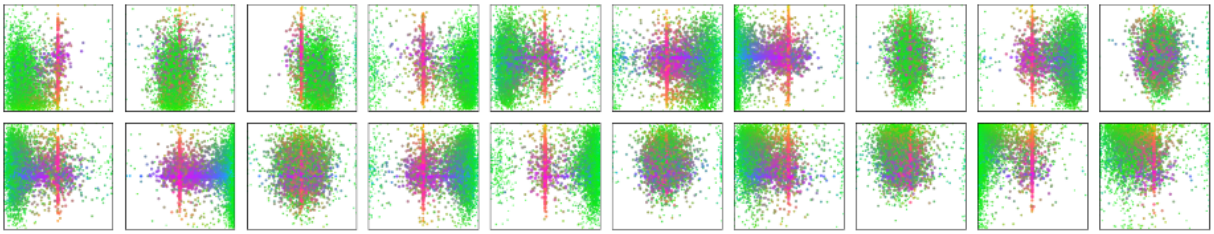


Figure 2.11: Visualization of DETR output corresponding to each object query. [8]

on the choice of extractors, the regions are extracted using the selective search algorithm proposed in [100], which remained as the algorithm of choice for future iterations of R-CNN (e.g. [32]). Selective search is designed to be exhaustive with a high recall – meaning that while it may produce a lot of false positives, the false negative rates are very low. After $\sim$ 2000 regions are extracted, the regions are then fed through a CNN classifier to determine what class each belongs to.

There have been multiple improvements over the design of R-CNN, notably Fast R-CNN [32] and Faster R-CNN [86]. Fast R-CNN introduces *RoIPool*; instead of feeding each region proposal into a CNN classifier, Fast R-CNN first feeds the entire image into a CNN backbone. It then performs selective search, and crops out the region proposals directly from the feature maps produced by the backbone and resizes to a pre-configured size (hence "*pool*"). The pooled feature vectors can then be fed through a CNN-based classifier. Fast R-CNN provides some much-needed efficiency to R-CNN and experimentally shows stronger performance on object detection.

Faster R-CNN [86] (see Fig. 2.10a) iterates upon Fast R-CNN by forgoing selective search, and implements an entirely CNN-based approach for extracting region proposals called the *Region Proposal Network* (RPN), which further improves performance by having the RPN share convolutional layers with Fast R-CNN.

2.2.2 DETR

With transformer-based architecture inspiring various designs for various tasks within computer vision, such as the Swin Transformers mentioned in Section 2.1.2, it is no surprise that transformers will find their footing within object detection. Indeed, the *DEtection TRansformer* (DETR, see Fig. 2.10b) proposed in [8] predates Swin Transformers, and

incorporates transformers in a completely different manner. DETR first takes the feature maps from an existing CNN backbone and flattens the result. The flattened feature vector is passed with a positional encoding into a transformer encoder, whose purpose is to introduce stronger correlations between long-range locations on the feature vector. The output is then passed to a transformer decoder, which takes a fixed number of *object queries* and attends to the encoder outputs. Each object query is trainable and corresponds to a possible object instance, with the number of queries N corresponding to the maximum number of detections for any given image. Each query was experimentally shown to map to specific regions of the image (see Fig. 2.11). The N outputs from the decoder are then passed into a feedforward network to predict the bounding box and class of each instance. By using the transformer decoder, the architecture enables the relation between instances to be reasoned more explicitly.

A notable extension of DETR is *DETR with Improved deNoising anchOr boxes* (DINO) from [114] (not to be confused with the self-supervised ViT model DINO [9]), which iterated upon DN-DETR [60] and DAB-DETR [68] with a multitude of efficiency-focused upgrades. One notable feature from both DINO and DN-DETR is that they add an auxiliary task to the transformer decoder to perform *contrastive denoising training*. The training supplies ground-truth bounding boxes and labels with noise as object queries to the decoder and trains the model to reconstruct the original data, which is experimentally shown to reduce the instability caused by bipartite matching (see Section 2.2.3).

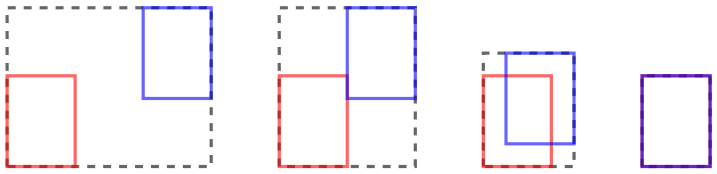
2.2.3 Training Loss

Usually, an object detection architecture predicts object instances and their respective classes for a given image in an arbitrary order. To calculate the total loss against the ground truth, the training algorithm must:

- match the predicted bounding boxes and ground-truth objects into pairs;
- handle the unmatched predictions; and
- calculate the loss for each matched pair.

In the case of R-CNN-based architectures, a match is achieved based on some IoU threshold. If multiple predictions map to the same object, *non-maximum suppression* is applied, which selects the best prediction based on some confidence score [33]. For example, in the case of Faster R-CNN, a match is achieved when the pair has an IoU of > 0.7. In cases where no match can be found, the pair with the highest IoU is matched [86]. As for DETR, *bipartite matching* is used to match predicted masks and ground truth masks using the Hungarian matching algorithm [54], where the matching score is based on a combination of standard bounding box regression loss and class loss [8].

As for the unmatched instances, which typically signify extra or unnecessary instances, the training algorithm needs to have a mechanism to suppress these results, which can be achieved in various ways. In Faster R-CNN, this is done by introducing a binary classification branch for each instance to determine whether a predicted instance is an object and is trained against whether each predicted instance was matched [86]. In contrast, DETR appends a "no class" (or commonly known as the "background" class) to the list of class labels with a custom class weight for the loss (see Eq. (2.2)) and uses that instead [8]. In either architecture, the unmatched instances are ignored when calculating the bounding-box regression losses.



GIoU	-0.62	-0.43	0.23	1
IoU	0	0	0.33	1

Figure 2.12: Examples of GIoU and IoU values. Red and blue rectangles represent the two bounding boxes A & B, while the black (dashed) rectangle shows the smallest enclosing box C. Notice that GIoU can extend into the negatives when IoU stays at 0.

For a matched pair, bounding-box regression losses commonly employ L_1/L_2-norms or variants of these costs between the predicted and ground-truth centers, widths, and heights (x, y, w, h). For R-CNN, L_2-norm is used [33], whereas for Fast R-CNN and Faster R-CNN, they use the smooth L_1 loss:

$$\text{smooth } L_1(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases} \tag{2.6}$$

Recently, the scale-invariant Generalized IoU (GIoU) [87] is another cost function that is gaining popularity:

$$\text{GIoU}(A, B) = \text{IoU} - \frac{|C \setminus A \cup B|}{|C|} \tag{2.7}$$

Where C is the smallest enclosing box that contains A and B. [87] argued that IoU-based losses are better than L_1/L_2-norms, as L_1/L_2-norms do not correlate well with IoU, which is commonly used in many evaluation metrics. GIoU improves upon IoU when there is little or no overlap between the two boxes, and adds differentiability in cases where there is no overlap, as IoU $= 0$ is not differentiable, as shown in Fig. 2.12. GIoU is used by DETR in combination with the L_1-norm.

As for training the classification loss, one possible approach is to use the same cross-entropy-based losses as mentioned in Section 2.1.3, which is the case for all of our baseline models in our experiments. However, *focal loss* [64] is also another popular choice for classification in the context of object detection, where there is usually a large imbalance between the foreground classes and the "background" class. Focal loss adds a weight to the standard cross-entropy loss $(1 - y_i^c)^\gamma$ with a tunable term $\gamma \geq 0$:

$$FL(y, y^*) = -\sum_i^N \sum_c^K (1 - y_i^c)^\gamma y_i^{*c} \log(y_i^c) \tag{2.8}$$

The added term reduces the loss on correct classifications, thus shifting the penalty more on difficult examples, as shown in Fig. 2.13.

For evaluation, object detection usually relies on *average precision* (AP) [25]. AP refers to the area under the precision-recall curve plotted based on varying confidence score thresholds. These confidence scores are entirely up to the segmentation architectures and are

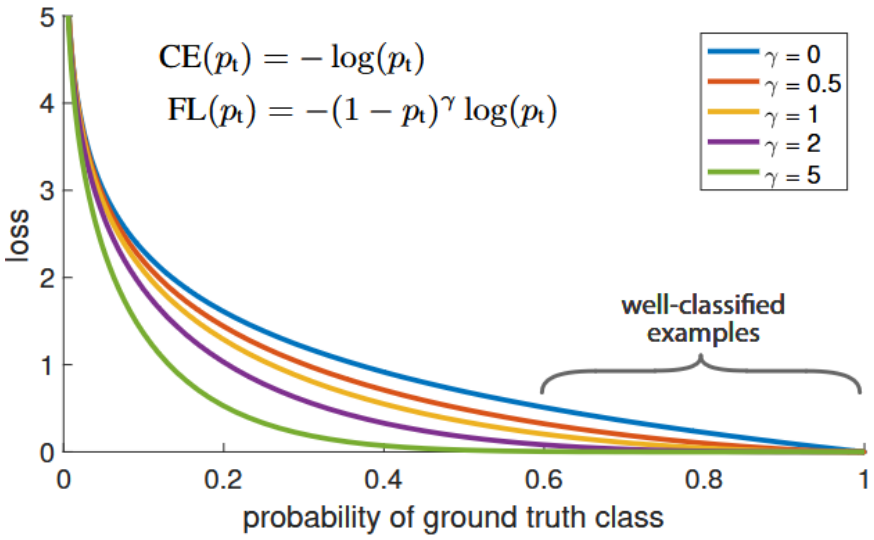


Figure 2.13: Focal loss across various γ values. [64]



(a) Original images. (b) Results.

Figure 2.14: Co-segmentation. [40]

calculated and returned for each predicted instance. As for the precision-recall curve:

$$\text{Recall} = \frac{\#\text{ of True Positive}}{\#\text{ of True Positive} + \#\text{ of False Negative}} \tag{2.9}$$

$$\text{Accuracy} = \frac{\#\text{ of True Positive}}{\#\text{ of True Positive} + \#\text{ of False Positive}} \tag{2.10}$$

AP scores are typically calculated per class and averaged across every class in the form of *mean AP* (mAP). What determines a match between predicted and ground-truth objects (thus what defines a true/false positive/negative sample) depends on an IoU threshold. The IoU threshold is typically denoted in the form of AP_x, where x is the IoU threshold. Typically, AP is calculated as mAP, averaged across IoU thresholds of 0.5 to 0.95 in increments of 0.05.

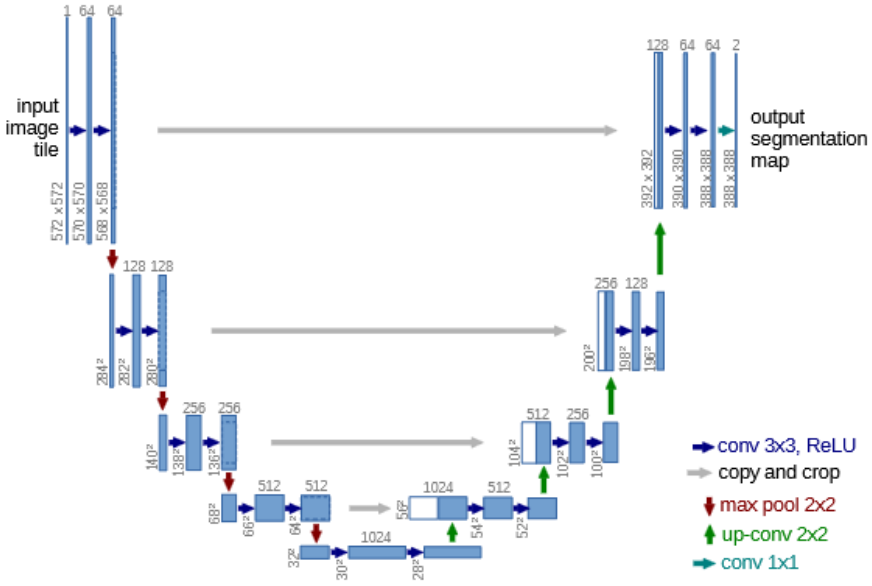


Figure 2.15: *U-Net* architecture. [88]

2.3 Semantic Segmentation

Traditionally, segmentation can be seen as a graph theory problem based on clustering pixels, with many popular and diverse approaches before the advent of deep learning. Early segmentation tasks revolve around separating the foreground from the background given an image. Techniques for such tasks commonly involve graph cuts [6, 5, 90] or clustering techniques that group pixels into super-pixels [16, 63]. *Co-segmentation* later gained popularity as one of the first tasks that requires no human input. Co-segmentation refers to segmenting a common object given multiple images of such object, as shown in Fig. 2.14, requiring a much more explicitly modeled prior knowledge, with techniques including [91] and [40].

With the arrival of deep learning, the task of *semantic segmentation* emerges as a mainstream problem in computer vision. Semantic segmentation aims to predict a per-pixel class label given an image, as shown in Row 4 of Fig. 2.8. One of the first to address such a problem with a pixel-to-pixel CNN approach would be [71], which proposed the *fully convolutional networks* (FCNs). Building upon FCNs, [88] proposed U-Net – an encoder-decoder CNN architecture with long-range *skip connections* as shown in Fig. 2.15. U-Net expanded upon FCNs by adding more layers to the upscaling (i.e., encoder) part, resulting in a symmetric encoder-decoder architecture that can produce more precise outputs. *Skip connections* refer to using outputs of the encoder layers as additional inputs to the corresponding symmetric decoder layers ("copy and crop" in Fig. 2.15). Without skip connections, the amount of information throughout the model would be bottlenecked by the smallest embedding size between the encoder and decoder. Skip connections return some of the information to the decoder layers, thus enhancing the pixel precision of such systems.

Another well-known CNN-based semantic segmentation model would be DeepLab [11]. DeepLab makes use of *dilated* (or *atrous*) convolution, which adds gaps in the convolutional kernel to allow for larger kernel sizes and receptive field at low computational cost. DeepLab also uses a fully connected *conditional random fields* (CRFs) to refine the produced segmentation masks.

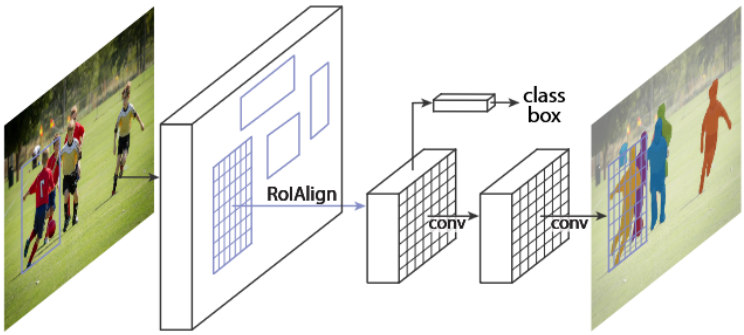


Figure 2.16: Architecture of Mask R-CNN. [36]

2.4 Instance Segmentation

Instance segmentation, or semantic instance segmentation, refers to the task of predicting the visible segment (or mask) of *each object* at the pixel level. This is commonly done by predicting N classes and binary segmentation masks $m_i \in [0,1]^{H \times W}$, where the value at each pixel corresponds to the probability that the pixel is part of the visible section of the instance; though other methods such as refining sets of 2D points as instance contours [117] also exist. Instance masks with precise boundaries are useful for applications such as medical imaging and image editing.

While semantic segmentation can easily be solved using encoder-decoder architectures, the same cannot be said about *instance segmentation*, as they are fundamentally very different problems and require different mindsets. In semantic segmentation, instances are not a concern; but in instance segmentation, different objects of the same class are segmented separately, and objects that cannot be quantified into instances (e.g., sky or grass) are ignored, as shown in Row 3 of Fig. 2.8. *Panoptic segmentation* [50] (see Row 5 of Fig. 2.8) is an emerging task that aims to combine both semantic and instance segmentation, aiming to predict both "things" (i.e. countable instances) and "stuff" (i.e., uncountable regions), with the *panoptic feature pyramid network* [49] being one of the earliest examples to address such task.

In this work, we focus on instance segmentation. For most of our experiments, we use Mask R-CNN [36] and Mask2Former [13] as the baseline architectures, which will be explained further in detail below.

2.4.1 Mask R-CNN

Recall from Section 2.2, object detection architectures commonly produce a bounding box for each instance with a center point and size (x, y, w, h). Since the segmentation mask of an object is constrained to be within the bounding box, it is unsurprising to find that many object detection architectures are augmented for instance segmentation purposes, usually via adding a new mask prediction branch. From DINO [114] we have Mask DINO [61], and from Faster R-CNN [86] we have one of the most popular models within the instance segmentation community – Mask R-CNN [36].

Recall from Section 2.2.1 that Faster R-CNN contains a CNN-based classification branch after RoIPool, Mask R-CNN introduces a mask prediction branch parallel to the classification branch (see Fig. 2.16) that takes the contents in each bounding box as input and

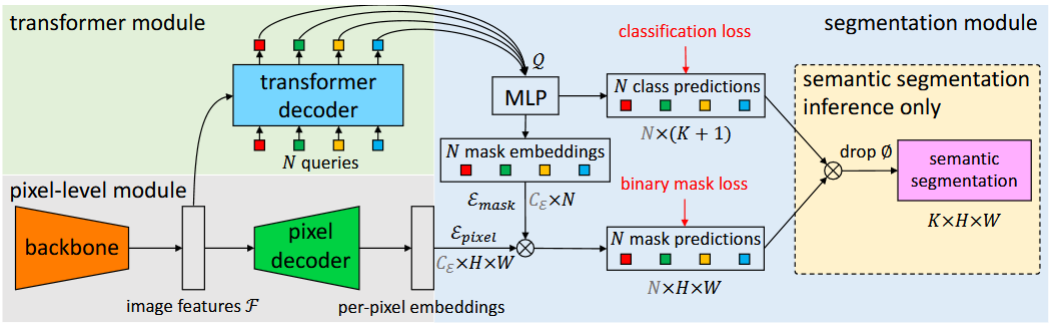


Figure 2.17: Architecture of *MaskFormer*. [14]

predicts a pixel-to-pixel mask for each bounding box. The added branch has a small overhead to Faster R-CNN at around 20% higher cost.

A critical change to enable Mask R-CNN to work is the introduction of RoIAlign, which is a modified version of the RoIPool algorithm. Recall that RoIPool resizes the RoI of a feature map to a pre-configured size (e.g., 7×7) before feeding it into the classification branch. RoIPool rounds (or quantizes) the four corners of the RoI to the original pixels from the feature map, and rounds again for each of the bins of the pre-configured size (each pixel in 7×7). This is not an issue and is efficient for Faster R-CNN, as the resized output is only meant for classification; small translation errors and misalignment have little effect on the results. However, the segmentation mask branch relies on a well-aligned RoI, which resulted in *RoIAlign*. RoIAlign uses bilinear interpolation without any rounding and experimentally shows significantly more accurate masks compared to just using RoIPool.

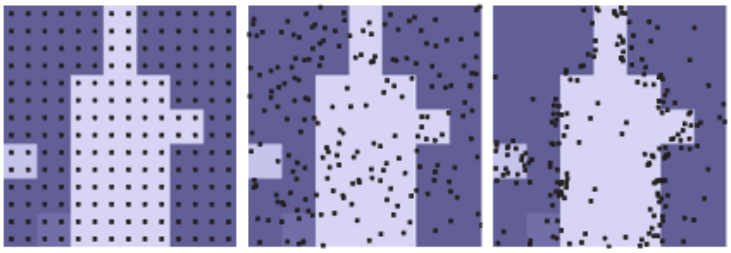
Approaches where the bounding box predictions (or object proposals) must come before the segmentation masks, like Mask R-CNN, are examples of two-stage methods. Mask R-CNN has given rise to many variants [42] and is commonly used as the baseline for new techniques and cost functions [51].

2.4.2 Mask2Former

Similarly, Mask2Former [13] can be seen as the instance segmentation counterpart for DETR [8]. Mask2Former originates from MaskFormer [14], which stems from DETR. Recall from Section 2.2.2 that the transformer decoder outputs a set of N output embeddings, which are then used for predicting the bounding box coordinates and classes. MaskFormer instead uses the N output embeddings to generate N *mask embeddings*. Then, it takes the dot product between the mask embeddings and a feature map of the image to generate the N segmentation masks (see Fig. 2.17).

Mask2Former iterates upon MaskFormer by introducing various adjustments in the transformer decoder architecture. Notably, it introduces *masked attention*. At each layer of the transformer decoder, a crude mask is obtained from the previous layer and used to mask away sections of the feature map that are outside the crude mask during the cross-attention step of the transformer. This allows the transformer at higher layers to converge more quickly by forcing them to attend to more localized regions, which is traditionally a weakness of transformer-based models.

Another important training technique that Mask2Former included is *Point-based Rendering* (PointRend) [51]. Normally, mask losses are calculated and summed over each pixel



(a) Regular grid. (b) Random. (c) *PointRend*.

Figure 2.18: Various ways points can be chosen to calculate mask loss. [51]

as shown in Fig. 2.18a. PointRend redistributes the points randomly, with a higher probability at which the masks are the most uncertain (i.e., around the segment boundaries in Fig. 2.18c), and uses those points to calculate the segmentation mask cost function instead. Since the random points are not discrete, bilinear interpolation is used to calculate the values at these points. See Section 2.4.3 for more information on loss calculations. For our experiments, PointRend is applied to Mask R-CNN as well for a fair comparison between Mask2Former and Mask R-CNN.

Since the bounding box prediction comes at the end and is not required for any region proposals, these are examples of one-stage approaches. Since object detection is not the primary task for MaskFormer and Mask2Former, they completely forgo the bounding box prediction branch; if needed, the bounding box can always be recalculated using the bounds of the segmentation masks. Since these architectures do not need to perform any RoIAlign or similar operations, they can be easily modified to perform other segmentation tasks, such as semantic segmentation (see the right section Fig. 2.17) and *panoptic segmentation* – the combination of semantic and instance segmentation. Other similar approaches include Mask DINO [61], which applies similar principles to the DINO object detection model to generate segmentation masks.

2.4.3 Training Loss

In general, the training process mimics each architecture's object detection counterpart, using similar losses for classification and AP scores for evaluation. For Mask R-CNN, it follows Faster R-CNN, while for Mask2Former, it uses the same bipartite matching process as DETR. For mask losses, there are 2 common cost functions associated with them. The BCE loss can be used to perform a pixel-wise BCE:

$$BCE(y, y^*) = -\sum_{p}^{\text{pixel}} \left(y_p^* \log(y_p) + (1 - y_p^*) \log(1 - y_p) \right) \tag{2.11}$$

The *Dice loss* [95] is another common cost function that models closer to IoU-based metrics and can be seen as the mask version of the GIoU loss:

$$Dice(y, y^*) = 1 - \frac{2 \cdot \sum_p^{\text{pixel}} (y_p \cdot y_p^*) + \epsilon}{\sum_p^{\text{pixel}} y_p + \sum_p^{\text{pixel}} y_p^* + \epsilon} \tag{2.12}$$

One advantage of Dice loss over traditional BCE is that for small instances, BCE averages over all pixels across the image, resulting in a smaller loss compared to large instances.

However, small instances are equally important. Dice loss re-weights the loss for each instance based on the size of the instance in the denominator, such that the loss is more invariant to sizes. An ϵ is added to avoid numerical instability. In the case of Mask2Former, a combination of BCE and Dice is used, both during bipartite matching and in the final training loss, with $\epsilon = 1$.

2.5 Amodal Instance Segmentation

Amodal instance segmentation is an emerging sub-task in instance segmentation that seeks to segment the whole object for a given image–both the visible and invisible parts. Applications include scene de-occlusion [113], 3D reasoning, image editing, etc. In this work, we propose a method for semi-supervised amodal instance segmentation with only supervision from visible segmentation masks and pairwise occlusion orders, which is the main focus of Chapter 6. See Fig. 4.2 and 4.3 for some examples of amodal instances.

At the formulation level, since amodal instance segmentation is just instance segmentation with overlaps allowed between masks from different instances, most of the mentioned architectures for standard instance segmentation still apply. Both Mask R-CNN and Mask2Former predict a binary mask for each instance and can easily be used to perform amodal instance segmentation as they have no inherent constraints that disallow overlaps between masks. However, amodal instance segmentation is far more demanding than standard visible instance segmentation when it comes to prior knowledge of shapes and occlusions. Even as humans, when given an image, mapping out the invisible regions of an object requires a lot more understanding of the object than just mapping out the visible object boundaries. As a result, many segmentation architectures that specialize in tackling the amodal task have emerged over the years and have shown far greater performance than baseline segmentation models like Mask R-CNN.

Occlusion R-CNN (ORCNN) [26] expands upon Mask R-CNN by adding an amodal mask prediction head parallel to the visible mask prediction head, as well as an occlusion mask prediction head that takes input from the two aforementioned heads to predict only the occluded/invisible regions without the visible part. The main benefit comes from training efficiency; the overhead from the new heads is small, and it can continue from existing pre-trained Mask R-CNN models by adding the new heads and duplicating the weights of the visible mask prediction head to the amodal one. Mask-based losses are then applied to all three mask prediction heads. ORCNN is the first to combine visible and amodal instance segmentation as a joint architecture, becoming a common theme for future models.

Bilayer Convolutional Network (BCNet) [46] is another two-stage approach to tackle amodal instance segmentation. BCNet predicts both the masks and contours of a given instance and the occluder of the instance. The contours are explicitly predicted as shown in Fig. 2.19, as the occlusion boundaries tend to be strong cues of segmentation masks. An important change that BCNet adopted is to replace standard fully convolutional networks with *graph convolution networks* (GCNs), which has shown improved performance in modeling long-range relationships within an image; this is important to predicting heavily occluded instances as the visible segmentation masks could contain disconnected regions due to the occlusion. While BCNet focuses on visible instance segmentation by utilizing the predicted mask from the occludee branch in Fig. 2.19 during inference, it can also be trained to produce full amodal segmentation masks in the lower branch, demonstrating

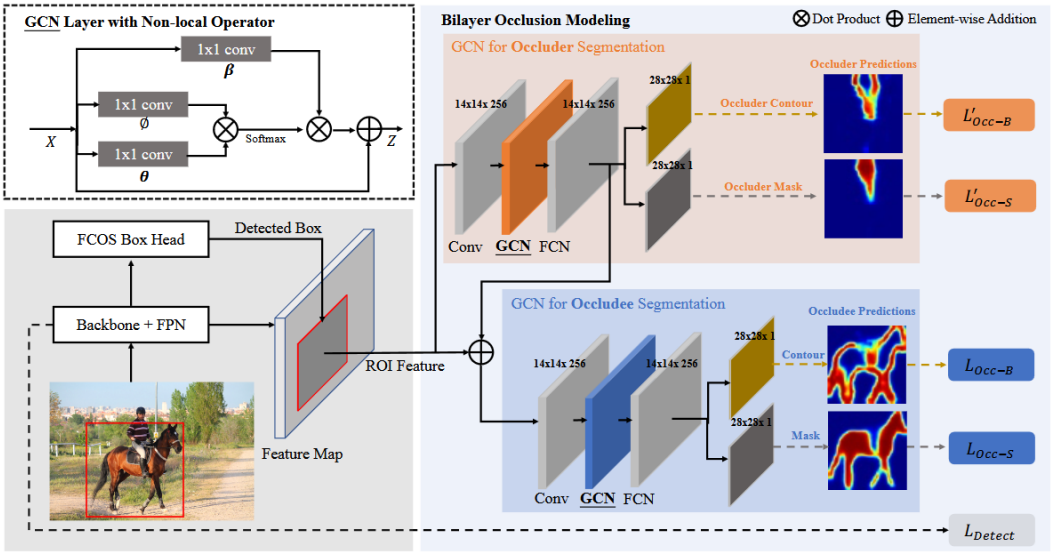


Figure 2.19: Architecture of BCNet. [46]

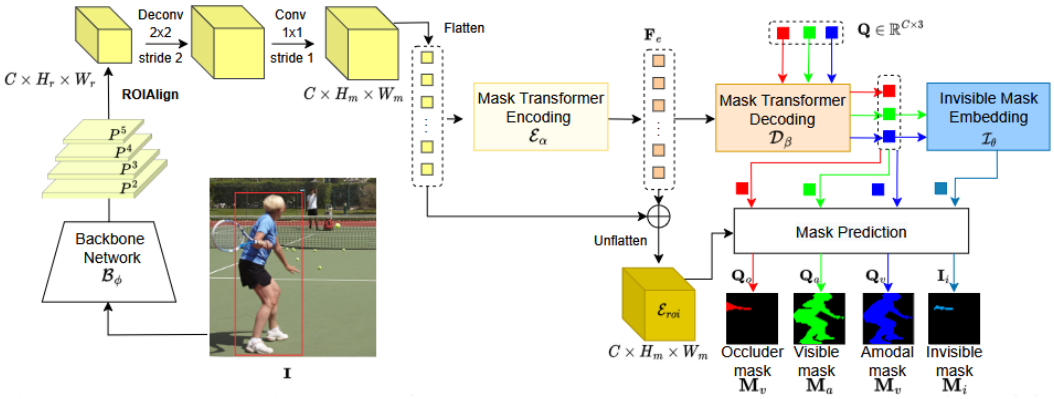


Figure 2.20: Architecture of AISFormer. [99]

strong performance in amodal tasks.

[99] proposes AISFormer, a transformer-based approach to amodal instance segmentation heavily inspired by DETR. Following the common theme of predicting multiple masks based on occlusion relations for a given instance, AISFormer uses a transformer decoder after RoIAlign, which takes three trainable queries as input. A dot product is taken with the feature map to produce the occluder, visible, and amodal masks (see Fig. 2.20).

Of course, all of the above works require amodal ground truth for training. Our work in Chapter 6 tackles a different requirement in only permitting the use of visible segmentation masks and pairwise occlusion orders.

2.6 Depth Estimation

Given an image (or a set of images of the same scene, i.e., multi-view), depth estimation refers to finding the distance from each pixel to the camera. A long-standing problem within computer vision, depth estimation is crucial for applications such as autonomous driving, scene understanding, and 3D reconstruction.

Within depth estimation, stereo matching (or stereo correspondence) is perhaps one of the earliest methods for depth estimation [74, 79]. Given a pair of left-right images, stereo matching refers to finding the distance between the corresponding pixels from each image (i.e., pixel disparity), which is inversely proportional to the depth. Many algorithms have been proposed to match pixels based on various criteria and cost functions [39], although more recent development sees a rise in deep-learning-based approaches such as those in [10, 94, 111].

Stereo matching is readily available in many computer vision code libraries such as OpenCV. It is also commonly used to generate ground truth depth maps for many datasets, such as the Cityscape [17] and Middlebury [93] datasets. It can also be used in conjunction with laser scan data to provide a more accurate and dense depth map, which is the case for the KITTI dataset [31].

2.6.1 Monocular Depth Estimation

While the above methods can be very reliable, laser scans or stereo images may not always be readily available during inference time. This gives rise to the problem of monocular depth estimation: estimating the pixel-wise depth from only a single image. Most methods employ an architecture consist of an encoder-decoder pair with skip-connections, with some notable examples as follows.

Godard *et al.* [34] devised an unsupervised technique that only requires stereo image pairs (instead of pre-calculated depth map via stereo matching) during training by reverse engineering stereo matching. This is achieved by taking one of the images from the image pair as input, and reconstructing the other image using the original image and the predicted pixel disparity. An image similarity loss can then be calculated between the reconstructed image and the actual image. As the whole process is fully differentiable, this results in an end-to-end unsupervised method for monocular depth estimation.

MiDaS [84] explored combining multiple datasets during training by proposing a training objective that generalizes any inconsistencies between different datasets. They were able to show improved performances, with the method generalizing well into unseen datasets.

GLPDepth [48] is a transformer-based encoder-decoder that extends CutDepth [43], which proposed a novel data augmentation technique that randomly replaces a rectangular window within the image with the actual depth map. GLPDepth fine-tuned the method by only allowing *vertical* CutDepths – instead of replacing a random window, GLPDepth replaces random vertical sections of the image (i.e., the top/bottom of the window must touch the top/bottom of the image). By preserving more vertical geometric details, the model can more easily extrapolate long-range depth information from other sections of the image, which suggests that the vertical depth information is far more important than the horizontal when it comes to depth estimation tasks [48]. In Section 5.3, we will use GLPDepth to demonstrate a test-time training application of the model.

2.6.2 Training Loss

One thing of note is the unique loss/error function commonly used in supervised depth estimation settings – the scale-invariant log-scale loss, first proposed in [24]. Let y_i, y_i^* be

the predicted and ground truth depth at pixel i in log-space respectively:

$$L = \frac{1}{n}\sum_i (y_i - y_i^*)^2 - \frac{\lambda}{n^2}\left(\sum_i y_i - y_i^*\right)^2 \tag{2.13}$$

When $\lambda = 0$, the loss simply becomes an MSE between the log-depths and penalizes as long as the y_is are not at the correct absolute depth. However, when $\lambda = 1$:

$$\begin{aligned}
L &= \frac{1}{n}\sum_i (y_i - y_i^*)^2 - \frac{\lambda}{n^2}\left(\sum_i y_i - y_i^*\right)^2 \\
&= \frac{1}{n}\sum_i (y_i - y_i^*)^2 - \frac{1}{n^2}\sum_{i,j}(y_i - y_i^*)(y_j - y_j^*) \\
&= \frac{1}{2n^2}\sum_{i,j}(y_i - y_i^*)^2 - \frac{1}{n^2}\sum_{i,j}(y_i - y_i^*)(y_j - y_j^*) + \frac{1}{2n^2}\sum_{i,j}(y_j - y_j^*)^2 \\
&= \frac{1}{2n^2}\sum_{i,j}\left((y_i - y_i^*)^2 - 2(y_i - y_i^*)(y_j - y_j^*) + (y_j - y_j^*)^2\right) \\
&= \frac{1}{2n^2}\sum_{i,j}\left((y_i - y_j) - (y_i^* - y_j^*)\right)^2
\end{aligned}$$

As shown above, the loss only penalizes if the relative predicted log-depth among pixels (i.e., $y_i - y_j$) is different from the ground truth (i.e., $y_i^* - y_j^*$), which results in a scale-invariant loss. This is useful as the global scale can be ambiguous given an image; the above allows us to shift the objective of the problem from the absolute depths to the relative depths among pixels by shifting λ from 0 to 1. A λ of 0.5 is commonly used to achieve a balance between absolute and relative depths [24].

Chapter 3

Related Works

3.1 Combining Depth and Segmentation

In the past few years, there has been some work to extend instance segmentation to also predict additional depth-related information by creating joint architectures. In PanopticDepth [29], the architecture jointly predicts a per-pixel depth estimate along with panoptic segmentation by predicting a depth map for each detected instance, as shown in Fig. 3.1. Each depth map from each instance is then stitched together based on the segmentation mask to create a full pixel-level depth map. As encoder-decoder architectures tend to produce over-smoothed depth maps, by having a depth map for each instance, PanopticDepth can model depth discontinuities at segmentation boundaries much more explicitly, thus improving depth accuracy.

[118] proposed *Task-Recursive Learning* (TRL) – a training framework that alternates between depth estimation and semantic segmentation tasks iteratively by passing results from previous iterations as input into a *Task-Attention Module* (TAM), then into the next task. This allows depth estimation and semantic segmentation to slowly refine themselves with help from their counterpart. A TAM is used at each stage, which is trained with the previous depth estimation and semantic segmentation results to boost information that is useful for both tasks (see Fig. 3.2).

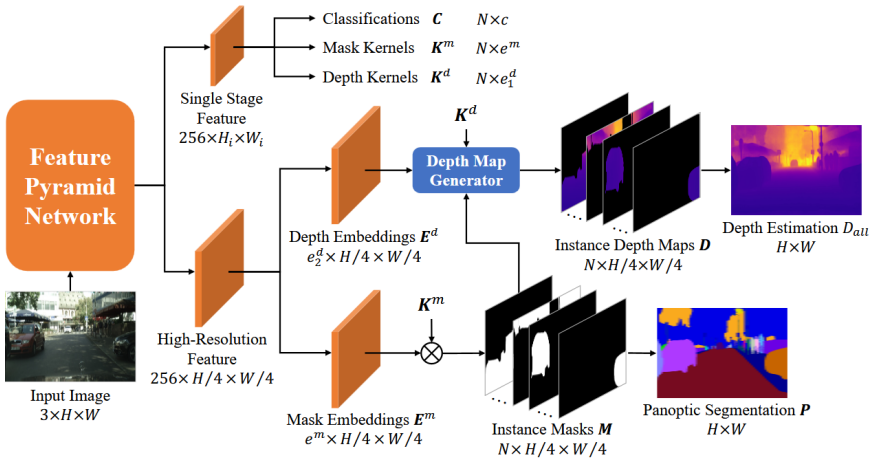


Figure 3.1: Architecture of PanopticDepth. [29]

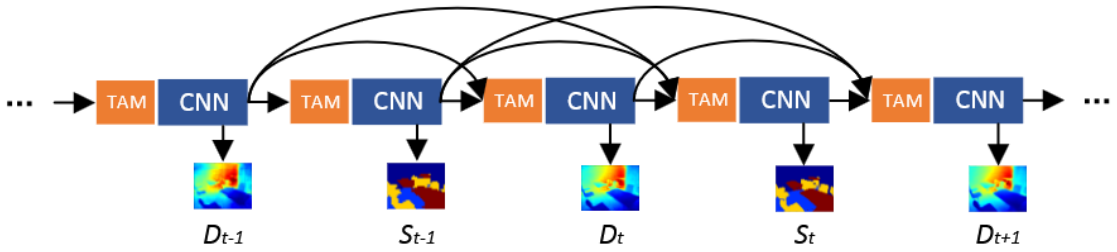


Figure 3.2: *Task-Recursive Learning.* [118]

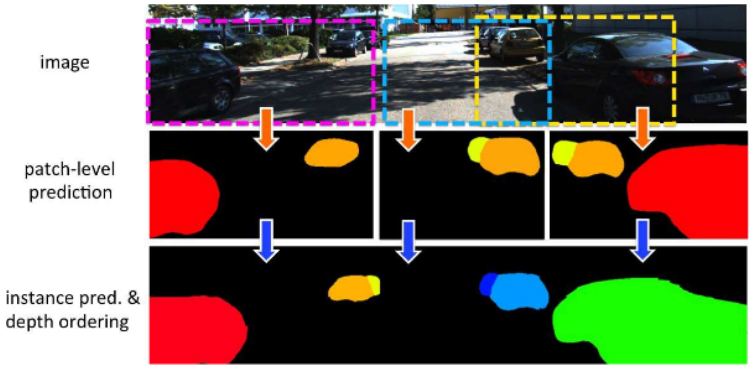


Figure 3.3: Zhang *et al.* [119]

[109] proposed a multi-task *prediction-and-distillation network* (PAD-Net) that first predicts 4 auxiliary tasks, namely depth estimation, surface normal estimation, semantic segmentation, and contour line prediction. The results from the auxiliary tasks are then used as input to distill a final prediction for depth and semantic segmentation. As surface normals and contour lines are low-level features/cues that are heavily related to depth and semantic segmentation, they can refine the final predictions, without needing additional annotation, as normals and contours can be inferred from the ground truth depth map and segments, respectively.

Instead of predicting pixel-level depths, Zhang *et al.* [119] predict a per-instance depth ordering along with instance segmentation using a *Markov random field* (MRF). In their work, each instance is predicted a label from 1 to 5, denoting the depth order of the instance in the given image. As an image can have more than 5 instances, by running the model against small patches of the image with ≤ 5 instances, the MRF can be used to stitch the results together into a global depth ordering for all the instances, as shown in Fig. 3.3.

These methods improve performance on both the instance segmentation task and the depth estimation task, suggesting that these tasks are closely related. This might be unsurprising, as depth and segmentation are strong mutual cues: segments are usually separated by depth, with the segment boundaries usually signifying discontinuities in depth.

3.2 Occlusion Estimation

Occlusion can also be a strong cue for segmentation; after all, segment boundaries *are* occlusion boundaries. Pairwise occlusion relations between instances are closely correlated to pairwise depth relations, as shown in Fig. 1.2 in Chapter 1. Although occlusion ordering contains less supervision than full pixel-wise depths, one major benefit of occlusion ordering

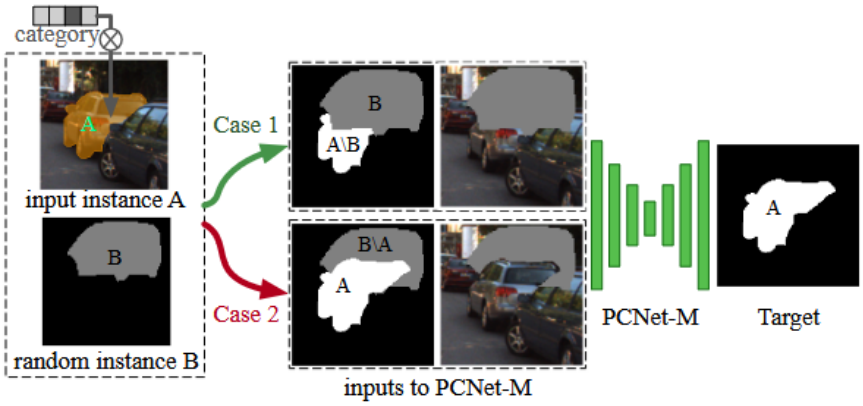


Figure 3.4: Architecture of PCNet-M. [113]

is that it can be easily human-annotated given an existing dataset as opposed to the more commonly used pixel-wise depth annotations.

There is prior work on predicting depth/occlusion order given a pair of instance masks [59, 113, 123]. In [123], they proposed OrderNet, and in [59], they proposed InstaOrderNet and InstaDepthNet for their corresponding pairwise order/depth tasks. OrderNet performs a binary classification based on the pairwise depth order, whereas InstaOrderNet and InstaDepthNet perform classification based on the various ordering types as seen in Fig. 1.2. One application of such models is that they can be used on unseen datasets for image captioning or annotating occlusion orders.

[113] proposed *Partial Completion Networks* (PCNets) as a way to tackle scene deocclusion – to regenerate the full occluded masks given the ground-truth masks. These models take a pair of random segmentation masks, overlap them to generate the visible sections of both masks, and use those as input to the training to reconstruct the original masks (see Fig. 3.4). Based on the reconstructed masks, the occlusion order can be inferred; if one of the masks remains the same, it is likely the occluder between the pair. This is an example of amodal mask completion, which is further explained in Section 3.4.

One major disadvantage of such models is that they rely on ground-truth segmentation masks as input and do not perform as well when they are applied to predicted masks, as shown in [4]. Applying such models to predicted masks is also computationally expensive. Our Mask2Former model is configured to predict at most 100 instances, which could require inference on a large number of pairs for any given image.

3.3 Combining Occlusion and Segmentation

Some works incorporate occlusion to improve panoptic, instance, or amodal segmentation, including [56, 112, 47]. In [56], they proposed OCFusion, which performs panoptic segmentation by adding an occlusion order prediction branch on top of Mask R-CNN. The branch takes feature maps from a pair of instances and performs binary classification using a standard CNN approach. A full panoptic output is then achieved via combining the instance masks with the occlusion orders together with the results from another "stuff" branch (see Fig. 3.5). To reduce computational cost, only instance pairs above a certain intersection area threshold are considered for the occlusion order branch.

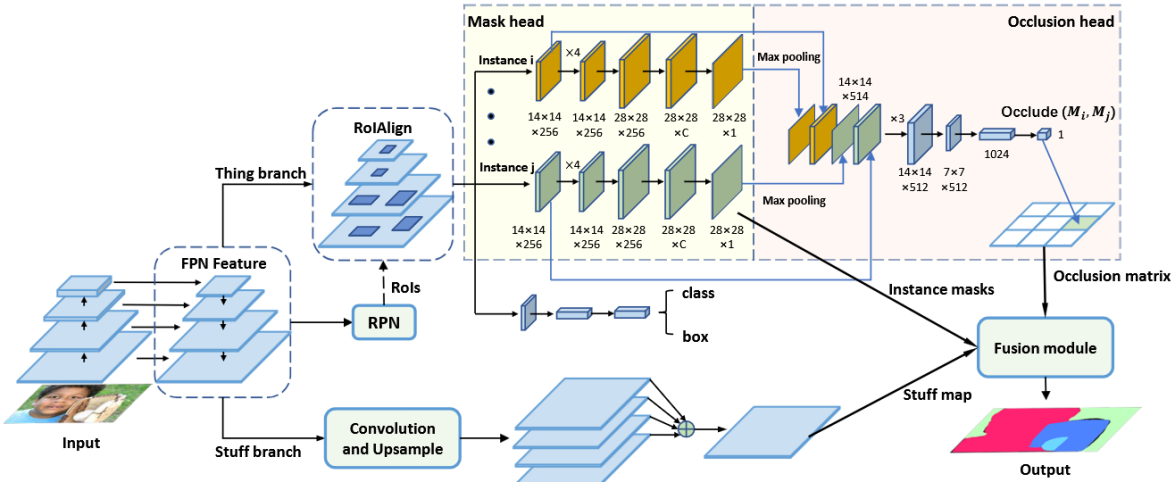


Figure 3.5: Architecture of OCFusion. [56]

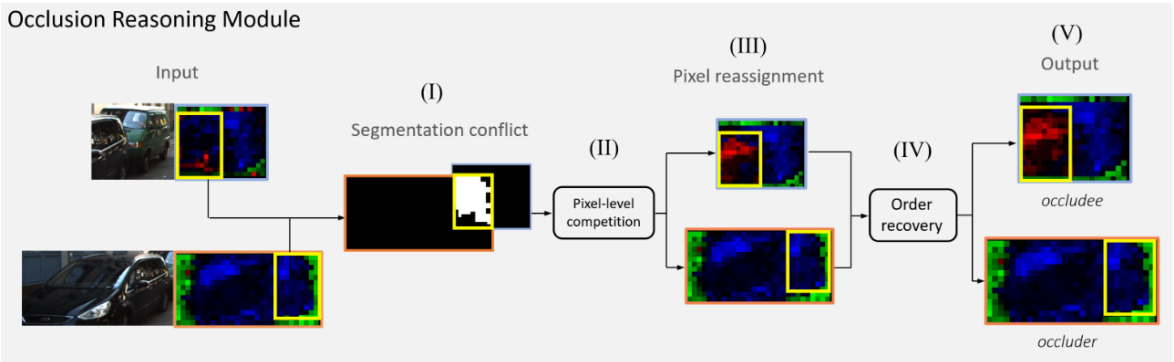


Figure 3.6: Architecture of ORM. [112]

[112] proposed the *Occlusion Reasoning Module* (ORM). The ORM takes segmentation maps of a pair of instances that contain segmentation conflict, which happens when the same pixel is defined as foreground (denoted by blue pixels in Fig. 3.6) for both instances. Such conflicts are resolved by reassigning pixels based on foreground probabilities. If a pixel is already occluded by other third-party instances, the pixel is assigned to neither instance from the pair. Otherwise, it is assigned to the instance with the higher foreground pixel probability. The occlusion order can then be inferred based on the number of pixels in the conflict region between both instances. The pixels in the conflict region are reassigned again to the occluder in an "all-or-nothing" manner.

Some amodal instance segmentation methods, such as BCNet [47], as mentioned in Section 2.5, contain mechanisms that use and output pairwise occlusion relations explicitly as well (see Fig. 2.19).

However, similar to the previous occlusion works in Section 3.2, these works do not establish a global occlusion order among instances. Instead, occlusions are estimated pairwise locally, resulting in the same problem of occlusion cycles, which are errors for our choice of datasets.

There is one prior work [4] where they do address the global occlusion order estimation of instances. First, [4] shows that treating the global occlusion order as labels for a classification task and training with cross-entropy on such labels fails. The failure is because occlusion-order labels do not have a consistent semantic meaning. Instead, they propose to extract instances with their occlusion-order labels via discrete optimization, avoiding a machine learning formulation altogether. This is computationally expensive because discrete optimization cannot be efficiently implemented on GPUs. Unlike [4], our approach in Chapter 5 is based entirely on deep learning and, therefore, is efficiently implemented on GPU: we append an occlusion order head to a standard instance segmentation network to learn a global occlusion order.

The goal of Chapter 5 is to extend instance segmentation to also predict the occlusion-order labeling of instances by creating a joint architecture for this task. We assume that a scene has a global occlusion order, i.e., each instance can be assigned an occlusion-order label that is larger than the occlusion-order label of any other instance it occludes, as shown in Fig. 5.1a. In Chapter 6, we extend such an assumption to perform amodal instance segmentation with only visible segmentation masks and the global occlusion order as supervision. The global occlusion order naturally assumes that there are no occlusion cycles. While occlusion cycles can exist (e.g., two objects occluding each other), in practice, they are very rare, with the COCOA [123] and KINS [82] datasets – two popular instance segmentation datasets with occlusion order annotations – having zero occlusion cycles [59].

3.4 Amodal Mask Completion

Amodal mask completion refers to the task of reconstructing amodal segmentation masks from visible masks. Frequently, such architectures rely on encoder-decoder approaches such as *variational auto-encoders* (VAEs) to capture the shape prior of amodal masks. One such example would be PCNet-M, as mentioned in Section 3.3 (see Fig. 3.4).

[66] proposed Amodal-VAE, which only requires visible segmentation masks for supervision. Amodal-VAE first trains a VAE using the full unoccluded masks. Synthetic occluded masks are then generated and used to train the VAE to reconstruct the full amodal masks;

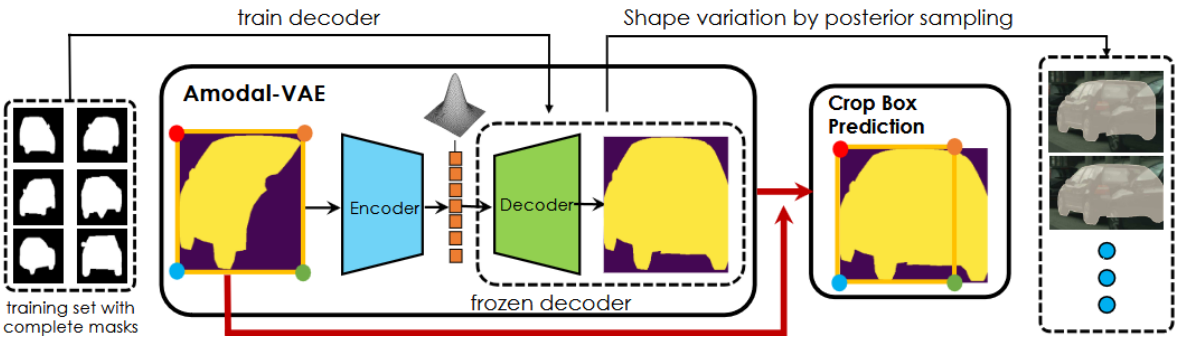


Figure 3.7: Architecture of Amodal-VAE. [66]

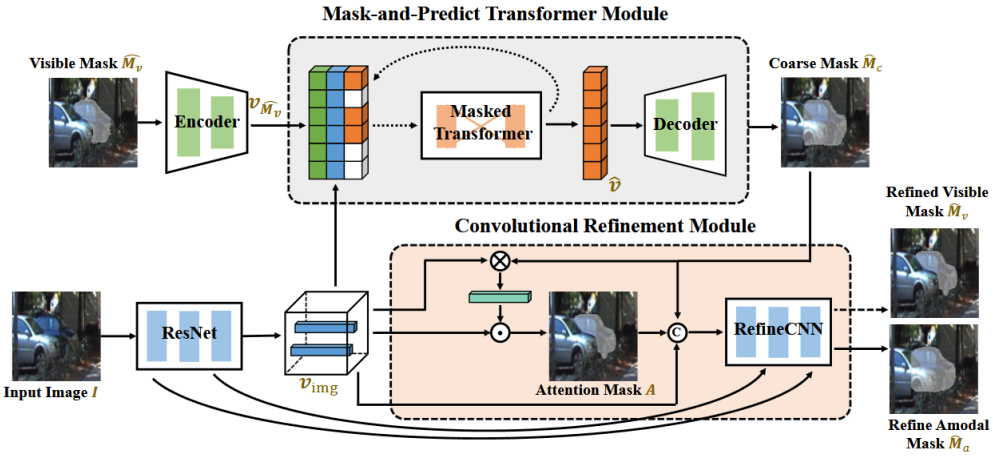


Figure 3.8: Architecture of C2F-Seg. [28]

this is achieved by freezing the decoder part of the VAE, and forcing the encoder to map the occluded masks to the amodal masks in the latent space of the embedding, as shown in Fig. 3.7. Amodal-VAE is then fine-tuned by training using real, non-synthetic visible masks to reconstruct the amodal masks by only applying reconstruction loss on the visible section of each mask, similar to our approach in Chapter 6. As amodal segmentation masks and visible segmentation masks have different bounding boxes, the scaling of the crop box could be different between the two and requires further resizing via a crop box prediction component. The parameters of the resize transform can be learned during the synthetic stage, where the training has access to both sizes of the visible/amodal bounding boxes.

While all of the above focus on just the task of amodal mask completion exclusively and require ground-truth masks during inference, amodal mask completion can also be seen as a sub-task/refinement step for amodal instance segmentation, which can be seen in a few architectures. One such example would be *Coarse-to-Fine Segmentation* (C2F-Seg) [28], in which predicted visible masks are passed into an encoder-decoder to generate coarse amodal masks for further refinement, as shown in Fig. 3.8.

3.4.1 Amodal Content Completion

Different from mask completion, amodal content completion (or simply amodal completion) refers to reconstructing the underlying image content of occluded instances, as opposed to just the masks. This can be seen as a generalization of amodal mask completion, requiring

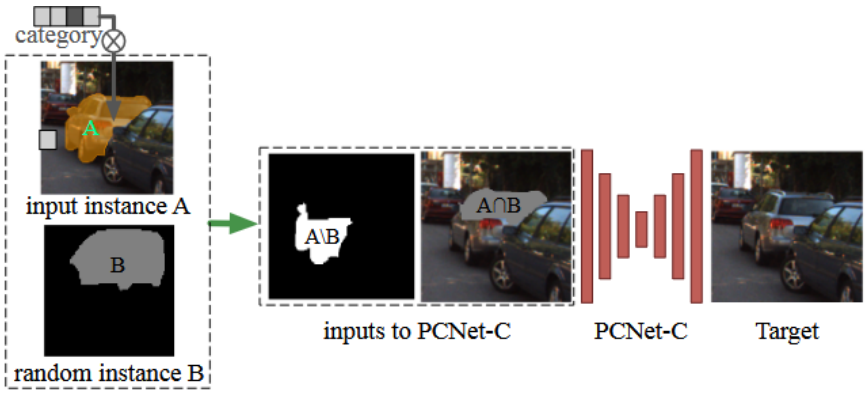


Figure 3.9: Architecture of PCNet-C. [113]

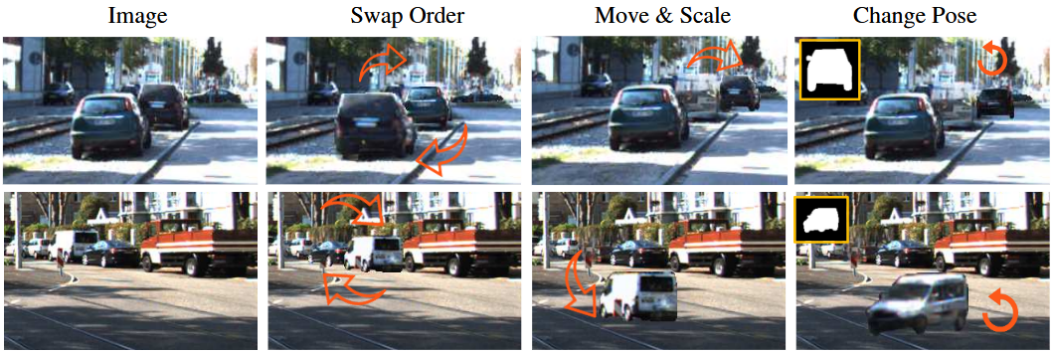


Figure 3.10: Types of image manipulation enabled by Amodal-VAE. [66]

more complex models as the output now requires full-range RGB values instead of binary masks.

Similar to mask completion, such architectures largely follow the encoder-decoder paradigm. [113], which proposed PCNets as mentioned in Section 3.3, also provides a way to achieve content completion by providing a separate model named PCNet-C – as shown in Fig. 3.9.

Similarly, for Amodal-VAE [66], the authors proposed a method for content completion via a convolutional in-painting model proposed in [67]. To push the application one step further, Amodal-VAE also enables instance manipulation, where the pose of each instance can be modified using a generative adversarial network (GAN)-based approach [81]. Combined with the inpainting, this allows each instance to be moved, resized, and rotated within an image, as shown in Fig. 3.10.

3.4.2 Generative Adversarial Networks

While *generative adversarial networks* (GANs) do not explicitly complete amodal masks by taking a visible mask as input and outputting the amodal mask, it is another way to impose shape priors based on amodal masks on an existing segmentation model. Traditionally, a GAN consists of a generator G and a discriminator D. The generator is responsible for generating data based on a target distribution (in our case, the distribution of amodal masks), whereas the discriminator aims to distinguish between generated outputs from G and real data. The objective function of the training is set up such that the generator is trained to fool the discriminator, while the discriminator is trained to take the generated/ground

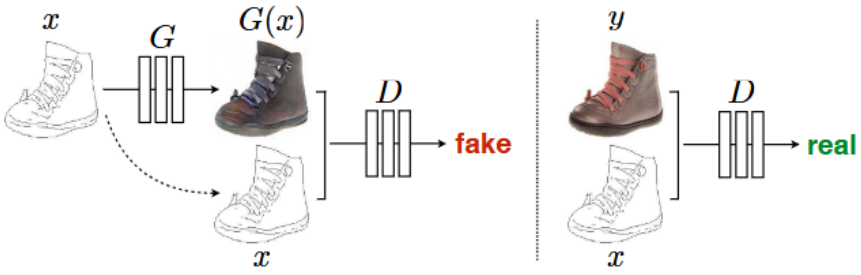


Figure 3.11: Example application of cGAN, where the contour lines of an image are provided to the model as x. [44]

truth data as input and perform binary classification based on whether the input is real or not. The competing nature of the generator and discriminator corresponds to a two-player minimax problem, with the generator converging to the target distribution and the discriminator converging to 0.5 output everywhere. Let x be the true data and z be the input noise variable for G, a GAN can be simplified to the following minimax problem [35]:

$$\min_G \max_D \mathbb{E}_x[\log D(x)] + \mathbb{E}_z[\log(1 - D(G(z)))] \tag{3.1}$$

When training, G and D can then be isolated from Eq. (3.1) into the following, which can be optimized in alternating steps as shown in the original GAN paper [35] or in a single step as shown in [78]:

$$L_G = \log(1 - D_{\text{frozen}}(G(z))) \tag{3.2}$$
$$L_D = -\log D(x) - \log(1 - D(G_{\text{frozen}}(z))) \tag{3.3}$$

Note that the objective function does not have to be log-based, as the f-GAN proposed in [78] generalizes the formula to allow for various measures of divergence.

Improving upon the original design, *deep convolutional generative adversarial networks* (DCGANs) [83] proposed a few guidelines for GANs, notably the inclusion of BatchNorms, and using LeakyReLU (with a slope of 0.2) in place of standard ReLU in the discriminator, which avoids the vanishing gradient issues when back-propagating through the discriminator.

Improving on GANs, [44] proposed the *conditional adversarial networks* (cGANs), which allows a GAN to be conditioned on an additional input y:

$$\min_G \max_D \mathbb{E}_x[\log D(x, y)] + \mathbb{E}_z[\log(1 - D(G(z, y), y))] \tag{3.4}$$

For example, from Fig. 3.11, the model is conditioned on the contour lines of an image; in the case of amodal masks, y could be the corresponding class labels. Experimentally, the conditioned discriminator showed improved performance over a standard unconditioned discriminator (i.e., Eq. (3.1)). In Chapter 6, we explore amodal mask completion by using a simple conditional discriminator to achieve amodal mask completion.

In the context of amodal mask completion, one of the earliest works to apply GANs is [110], which proposed the use of two discriminators for mask completion on vehicles: the object discriminator and the instance discriminator. Both discriminators aim to distinguish ground-truth masks as real. However, in addition to the ground-truth masks, synthetic

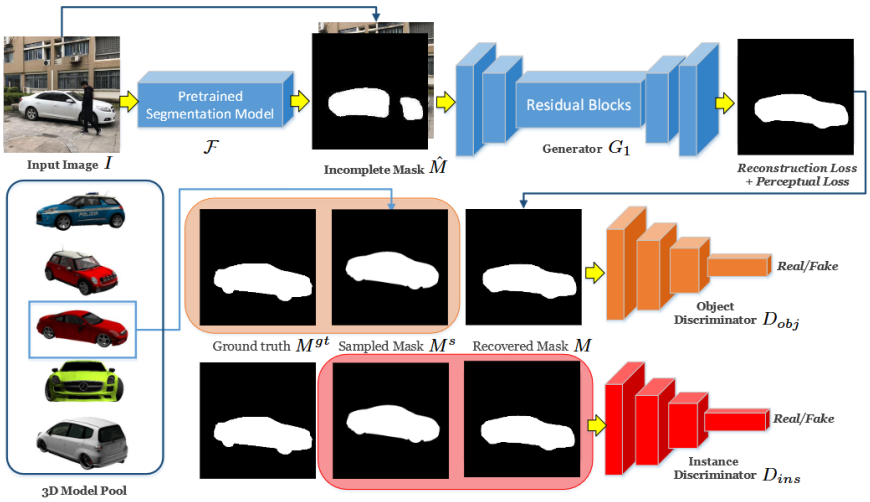


Figure 3.12: Architecture of the mask completion module from [110].

masks are also created based on a pool of 3D vehicle models, where the object discriminator treats them as real masks, while the instance discriminator treats them as fake, as shown in Fig. 3.12. The object discriminator only has access to the mask as input, whereas the instance discriminator has access to the mask and the original image as input. Experimentally, the two-discriminator approach outperforms using just a single discriminator. While such a method performs well, it requires ground-truth amodal masks and 3D models of vehicles, which we do not have access to in Chapter 6.

[122] and [115] proposed models for human de-occlusion by using a simple discriminator for masks. Both approaches require ground-truth/synthetic amodal masks for training. All of these models also have an additional module with a separate discriminator to achieve amodal content completion [110, 122, 115].

Chapter 4

Datasets

When it comes to instance segmentation datasets with occlusion annotations, there are only a few handful to choose from. In this work, we will be focusing on the KINS [82] and COCOA [123] datasets, which will be explained in further detail below.

4.1 KINS

KITTI INStance Segmentation (KINS) [82] is a subset of the KITTI dataset [30]. The KITTI dataset is a multi-purpose dataset consisting of stereo street view images taken with mounted vehicle cameras, and provides annotations for object detection as well as sparse depth information captured from LiDAR sensors. KINS expands upon the 2012 dataset with added segmentation masks for both visible and amodal segmentation, as well as occlusion relations annotations. In total, KINS contains 7,476/7,517 training/testing images and 8 classes: cyclist, pedestrian, rider, car, tram, truck, van, and misc, mostly dominated by cars and pedestrians. Combining both the training and testing set, 8.6% of instance pairs are occlusion pairs for KINS.

The annotation of occlusion relations is provided via an occlusion order number per instance within each cluster of connected (neighboring) instances. For example, from Fig. 4.1a, given the annotation in KINS, it is unknown whether instance 2 occludes instance 0. We treat all instance pairs within a cluster as occlusion pairs for KINS, with the higher-numbered segment occluding the other. Naturally, there are no occlusion cycles for such a formulation. See Fig. 4.2 for some examples.



(a) KINS annotation, single cluster. (b) COCOA annotation.

Figure 4.1: Differences in annotation between KINS [82] and COCOA [123].

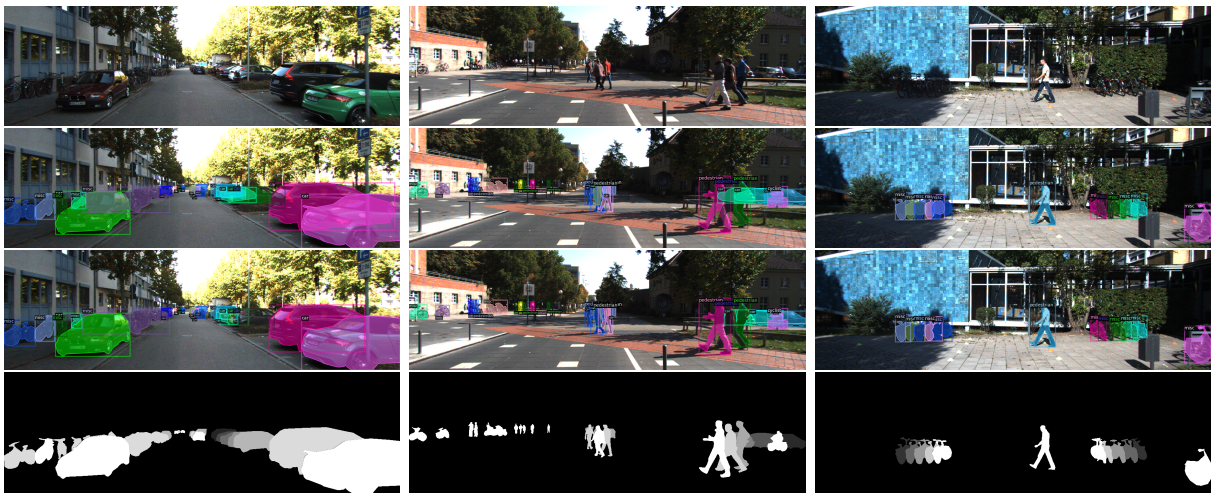


Figure 4.2: Examples from KINS. From top to bottom: original image, amodal instance segmentation, visible instance segmentation, occlusion order. The occlusion order corresponds to the occlusion order number per instance cluster as shown in Fig. 4.1a.

4.2 COCOA

COCO Amodal (COCOA) [123] is a subset of the COCO dataset [65]. COCO originally contains 80 classes with visible segmentation masks of a wide variety of images; COCOA *re-annotates* a subset of images of COCOA from scratch with visible and amodal segmentation masks as well as pairwise occlusion order. With dense annotation being one of COCOA's primary objectives, COCOA also included objects which are outside of COCO's 80 classes and are not present in COCO's annotations, but at the cost of removing class annotations altogether. In total, COCOA consists of 2,500/1,323 training/testing images with classless instances (i.e., only 1 class for all instances: "object"). Combining both the training and testing set, 8.5% of instance pairs are occlusion pairs for COCOA; all pairwise occlusion orders are explicitly annotated (see Fig. 4.1b) without any occlusion cycles [59]. See Fig. 4.3 for some examples.

4.3 Other Datasets

[59] proposes a new *InstaOrder* dataset with visible segmentation masks on a large number of instances for the task of occlusion ordering of instance pairs. However, due to resource constraints in annotation, only a maximum of 10 instances per image are annotated. While this limited instance annotation is not a problem for methods that seek to train only a pairwise occlusion classifier (as they do in [59], see Section 3.2), the sparse annotation drastically reduces the number of occurrences for occlusions, which is the primary subject of our experiments. The maximum limit on instances per image also results in instances of existing classes being ignored. As mentioned in Section 2.2.3, these instances will be assigned as non-objects/"no class", which can overfit and/or throw off our model predictions when evaluating. It makes this dataset unsuitable for our methods, which seek to segment *all* the instances; the lack of amodal masks also prevents us from evaluating our experiments for Chapter 6.

COCOA-cls [26] is another dataset with occlusion ordering. As the name would suggest,

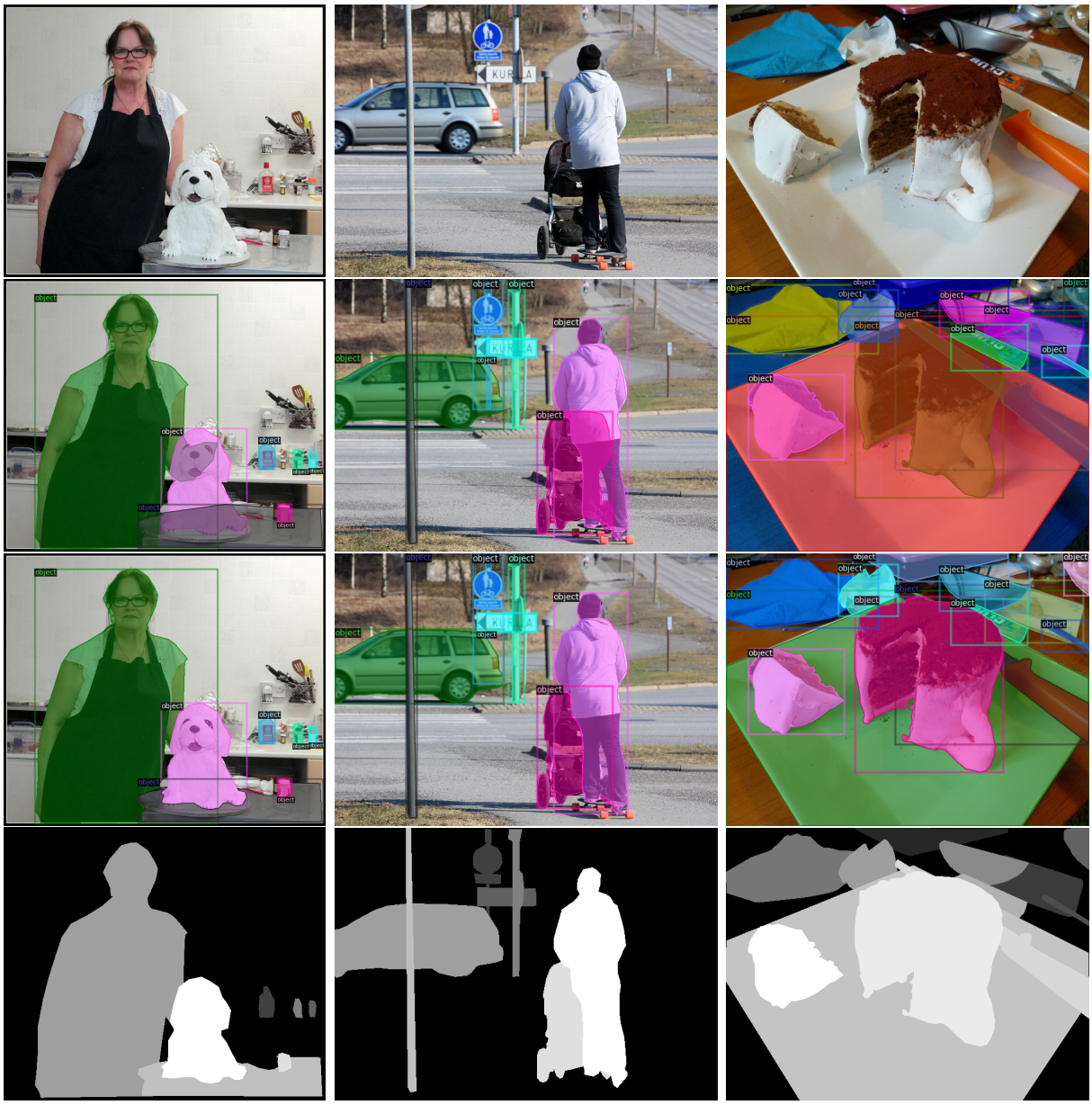


Figure 4.3: Examples from COCOA. From top to bottom: original image, amodal instance segmentation, visible instance segmentation, occlusion order. The occlusion order corresponds to the order of the provided instance, which is consistent with the global occlusion order.

COCOA-cls reintroduces class back to the COCOA dataset by matching annotations of segmentation masks between COCOA and COCO. The matching is done using an IoU threshold of 0.75, and the corresponding class from the matched instance in COCO is added back to COCOA. However, as mentioned in Section 4.2, COCOA is much denser and contains objects outside of the listed 80 classes from COCO, leading to *COCOA-cls* having a drastically reduced number of instances. The sparse nature of *COCOA-cls* would have a similar effect as *InstaOrder* – the lack of occlusion is ill-posed for comparing results with and without occlusion supervision for our methods.

Other less commonly used datasets with occlusion ordering includes *SAIL-VOS* [41], *Walt* [85], *Amodal Cityscapes* [7], *OLMD* [21], *CSD* [120]. However, all of these datasets are synthetic, created by either rendering simulated 3D scenes or superimposing instances on top of each other.

Chapter 5

Problem 1: End-to-End Occlusion Ordered Semantic Instance Segmentation

As mentioned in Section 3.1, there has been some work to extend instance segmentation to also predict additional depth-related information by creating joint architectures. In [118, 109, 29], the architecture jointly predicts a per-pixel depth estimate along with panoptic or instance segmentation. In [119], they predict a per-instance depth ordering. These methods showed improved performance on both the instance segmentation task and the depth estimation task, suggesting that these tasks are closely related, as segment boundaries usually indicate depth discontinuities.

Occlusion can also be a strong cue for segmentation, as pairwise occlusion relations between instances are closely correlated to pairwise depth relations, as shown in [59]. While occlusion ordering contains less supervision than full pixel-wise depths, occlusion orders benefit from the ease of human annotation, even allowing existing datasets to be retrospectively annotated with occlusion information. None of these applies to the more commonly used pixel-wise depth annotations.

The goal of our work is to extend instance segmentation to also predict the occlusion-order labeling of instances by creating a joint architecture for this task. We assume that a scene has a global occlusion order, i.e., each instance can be assigned an occlusion-order label which is larger than the occlusion-order label of any other instance it occludes. See



(a) Prior work. (b) Our approach.

Figure 5.1: Prior work vs. our approach to occlusions. In (a), prior work extracts pairwise occlusions between instances; an arrow points from the occluding instance to the occluded one. Such a formulation admits cyclic results (red arrows) even if a scene is known to be occlusion-cycle free, and any such cycle is erroneous. In (b), our approach extracts occlusions globally by assigning labels to instances. An instance with a larger label occludes a neighboring instance with a smaller label. Cycles are impossible by design.



(a) Input image. (b) Output masks and occlusion order.

Figure 5.2: An example input/output of occlusion ordered semantic instance segmentation. For (b), a darker color signifies a further global occlusion order.

Fig. 5.1b for an illustration. The global occlusion order naturally assumes that there are no occlusion cycles. While occlusion cycles can exist (e.g., two objects occluding each other), in practice, they are very rare, with the COCOA [123] and KINS [82] datasets – two popular instance segmentation datasets with occlusion order annotations – having zero occlusion cycles [59]. See Fig. 5.2 for an illustration of the input and output of our approach.

As mentioned in Section 3.3, some works incorporate occlusion [56, 112, 47] to improve panoptic, instance, or amodal segmentation. However, these works do not establish a global occlusion order among instances. Instead, occlusions are estimated locally in a pairwise fashion: given two neighboring instances, these methods estimate which of the instances is the occluder. When gathering such local pairwise orderings into a global order, the local nature of the estimations can cause numerous occlusion cycles in practice, as shown in Fig. 5.1a. Applying such models to naturally occlusion cycle-free scenes can cause many erroneous cycles in the estimates [4]. Our work assumes that all images are cycle-free by design (see Fig. 5.1b), which is the case for our choice of datasets.

There is one prior work [4] where they do address the global occlusion order estimation of instances. First, they show that training with cross-entropy on occlusion-order labels fails [4]. The failure is because occlusion-order labels do not have a consistent semantic meaning. Instead, they propose to extract instances with their occlusion-order labels via discrete optimization, avoiding a machine learning formulation altogether. This is computationally expensive because discrete optimization cannot be efficiently implemented on a GPU. Unlike [4], our approach is based entirely on deep learning, and, therefore, is efficiently implemented on GPU: we append an occlusion order head to a standard instance segmentation network to learn a global occlusion order.

As mentioned above, training cross-entropy on the occlusion-order labels fails [4], so a crucial component of our approach is the design of an effective loss function for training the occlusion order head. Inspired by regularized losses [98, 97] for weakly supervised semantic segmentation, we propose a loss function that, instead of encouraging some fixed occlusion-order label for an instance, encourages an instance to have an occlusion label larger than any other instance it occludes.

Global occlusion-ordering is a useful task in its own right, as it gives a coarse 3D information about the scene: given a monotonic chain of occlusion-ordered neighboring instances, we can recover the relative depth order between any two of them. For example, consider a monotonic chain of bicycles in Fig. 5.2 on the left (or on the right) of the pedestrian. Relative depth ordering between any two instances in the chain is known. Therefore, global occlusion ordering can provide information related to the depth ordering in the scene, for example, which of two (or more) given instances in an occlusion relation is closer to the

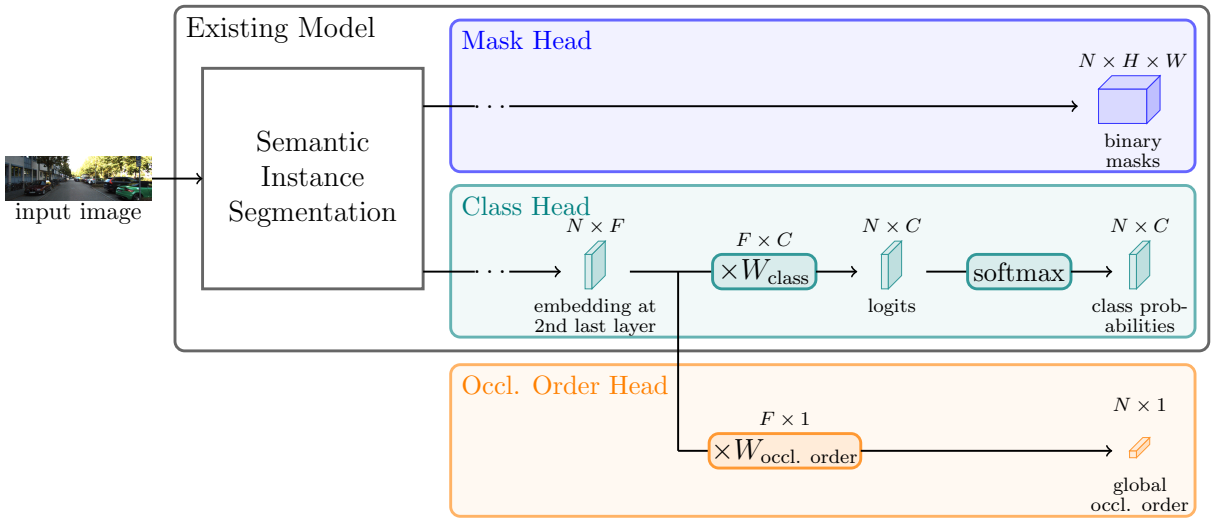


Figure 5.3: Model architecture. The shallow occlusion order head (orange) is added to an existing architecture. N is the number of segments predicted by the model, C is the number of classes, and F is the size of the embedding at the penultimate layer of the class head.

viewer from the camera viewpoint. The applications include question answering [73], scene de-occlusion [113], and amodal instance segmentation [62].

We evaluate our approach on the new OAIR curve metric introduced in [4]. This metric jointly evaluates the accuracy of instance segmentation masks and the accuracy of occlusion order. We perform better than [4], and also are much more computationally efficient. In addition, our method also provides an improvement in instance segmentation metrics, likely due to the added information from the ground-truth occlusion order, which is a richer supervision.

It is interesting to note that our model, trained on occlusions, not depth, still learns to a good degree how to order instances by depth. That is, our model can correctly order by depth many instance pairs which are not connected through a monotonic chain of occlusion relations, see Section 5.2.5 and 5.2.7.

To show the usefulness of our joint instance segmentation and occlusion-ordering approach, we also develop an application for improving depth maps during test-time training, which is detailed in Section 5.3.

5.1 Methodology

Let $i \gg j$ represent that instance i occludes instance j. Our goal is to develop an approach to extract instance masks together with their global occlusion order. Let o_i be the occlusion-order label assigned to instance i. The labels are interpreted as follows. For a pair of neighboring instances i, j, if $o_i > o_j$, then instance $i \gg j$. Unlike [4], our occlusion-order labels do not have to be integers; they can take any real value.

Almost all prior work that addresses occlusions for segmentation [56, 112, 47] is based on pairwise estimation: given a pair of instances, there is an occlusion branch that outputs an occlusion relationship between the input instances. The drawback of such approaches is that local pairwise occlusions are error-prone and can result in contradictory occlusion cycles for

scenes known not to contain occlusion cycles. This is because every pair of occlusions is estimated independently from all other occlusions.

Similarly to [4], we take a principally different approach. We directly assign occlusion-order labels to the instances. This is a more general approach than pairwise occlusion-order labeling (assuming a scene permits a global occlusion labeling), as the pairwise occlusion relations can be deduced from the global occlusion labels. Our approach is cycle-free by design due to the use of global occlusion-order labels.

Unlike [4], our approach is end-to-end and is based entirely on deep learning. Our design is simple and intuitive, and can be applied to any architecture. Given an architecture for instance segmentation, we add a shallow head directly next to the class prediction head to predict the per-instance global occlusion order. Precisely speaking, for a given instance, suppose that x is the embedding at the penultimate layer of the class prediction head and w_{class} are the class weights such that $x^T w_{\text{class}}$ are the logits for class predictions. We train additional weights $w_{\text{occl. order}}$ such that $x^T w_{\text{occl. order}}$ outputs a single number representing the global occlusion order for the given instance. See Fig. 5.3 for an illustration of the architecture.

For training, we use the same loss as in the original instance segmentation approach, plus an occlusion loss L_{occl} added with a weight λ. The main question is the design of the loss function L_{occl}. In [4] (Sec. 5.1), they tried direct training with cross-entropy on the occlusion-order labels as class labels, but such approaches fail as occlusion-order labels do not have a fixed semantic meaning. An instance with occlusion-order label, say 2, in one image may have nothing in common with an instance also labeled with an occlusion-order label 2 in another image. In fact, given a valid occlusion-order labeling of a scene, they can all be increased (or decreased) by the same amount, and the result is still a valid global occlusion-order labeling of the scene.

Our loss function is inspired by regularized losses [98] for weakly supervised semantic segmentation. Since pixel-precise labels are not available for training in a weakly supervised setting for all pixels, a regularized loss encourages equal semantic labels for nearby pixels, instead of encouraging a fixed label for each pixel. The word "regularized" reflects that we expect a certain regular property between neighboring pixels, without specifying the exact values for the pixels.

This leads us to design a regularized occlusion-order head loss for instances in an occlusion relation, given that the absolute value of the occlusion-order labels is not meaningful. Let i, j be a pair of ground-truth instances with ground-truth occlusion orders o_i, o_j. We propose two regularized losses to enforce the linear constraint i occludes j such that $o_i \geq o_j$:

$$L^1_{\text{occl}} = \frac{1}{N} \sum_{i \gg j} \text{ReLU}(o_j - o_i) \tag{5.1}$$

$$L^2_{\text{occl}} = \frac{1}{N} \sum_{i \gg j} \text{ReLU}(o_j - o_i + 1). \tag{5.2}$$

L^1_{occl} in Eq. (5.1) sums over all pairs of instances i, j in the ground truth for which the occlusion relation is known, and has a non-zero penalty whenever the occluded instance j is assigned to a label larger than that of the instance i that occludes it. N is the total number of ground-truth instances in a known occlusion relation.

Note that the loss in Eq. (5.1) is zero whenever $o_i \geq o_j$ for instances i, j with i occluding j, even if the difference is zero or a tiny number. This might be suboptimal during test time, where a larger difference between o_j and o_i could result in a more robust performance of occlusion order determination. Therefore, we develop the loss L^2_{occl} in Eq. (5.2) which enforces an additional gap (i.e., $o_i \geq o_j + 1$) for a more robust differentiation between different occlusion orders at test time. Mathematically, L^1_{occl} can be seen as enforcing $o_i \geq o_j$, whereas L^2_{occl} enforces the stricter $o_i > o_j$.

Since our regularized occlusion losses are based on instance pairs with an occlusion relation, we just need a dataset with pairwise occlusion relations for training. We do not need a dataset with a global occlusion ordering.

5.2 Experimental Results

5.2.1 Architecture and Training Details

As mentioned in Chapter 4, we use the We use the *KITTI INStance Segmentation* (KINS) dataset [82] and the *COCO Amodal* (COCOA) dataset [123].

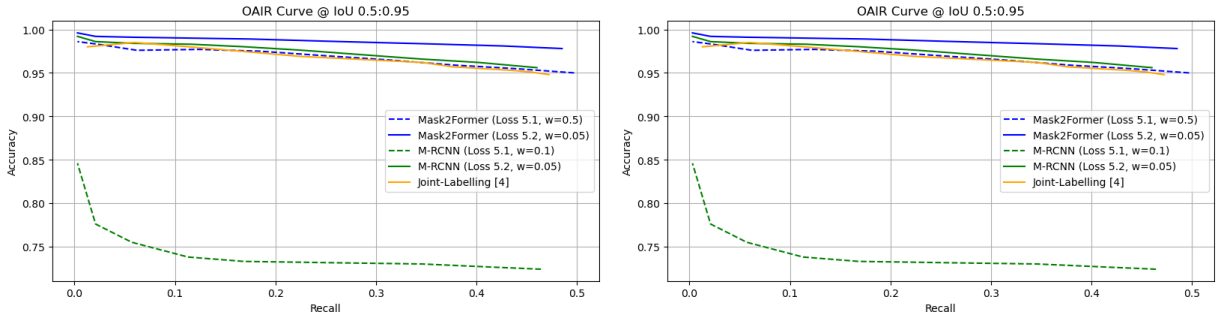
We evaluate our approach using Mask R-CNN and Mask2Former with ResNet-50 [37] backbone as baselines. Across all experiments for KINS, configurations and augmentations of each model mirror the default configuration for the *Cityscapes* dataset [17] – another dataset focused on road scenes. As for COCOA, the configuration mirrors the default configuration for the COCO dataset. For Mask R-CNN, the experiments are done at 8 images per batch$\times 45,000$ iterations at learning rate 0.01 ($\sim$ 7 hours); where as for Mask2Former, the experiments were run for 8 images per batch$\times 90,000$ for KINS and $\times 40,000$ for COCOA at learning rate 0.000075 ($\sim$ 22 hours for KINS and $\sim$ 7 hours for COCOA). All experiments are run on a single Nvidia RTX™ 6000 Ada Generation graphics card.

5.2.2 Joint Evaluation of Segmentation And Occlusion Order Accuracy

First, we evaluate the joint accuracy of instance masks and their occlusion order. In [4], they introduce a new metric called the *OAIR* curve, which is based on the Accuracy vs. Recall curve. The OAIR curve depends both on the accuracy of the instance masks and on the correctness of the estimated occlusion order.

We now describe the OAIR curve metric. First, following the standard practice, the obtained instance masks are matched with the ground truth masks based on IoU thresholds, from 0.5 to 0.95 at increments of 0.05. The obtained masks that do not match any ground truth mask are discarded. For any ground truth instance pair with a known occlusion relation, we say it is recovered if both instances in the pair are matched to the obtained masks. Let i^*, j^* be a recovered instance pair in the ground truth, and i, j be the corresponding matched instances in the predictions. Recall that o_i denotes the occlusion-order label of instance i. If i^* occludes j^*, we say the recovered instance pair has the correct order if $o_i > o_j$. The case j^* occludes i^* is handled symmetrically.

For each IoU threshold, we compute Recall and Accuracy as follows.



(a) For KINS. (b) For COCOA.

Figure 5.4: Occlusion OAIR curve comparison for our methods and the Joint-labeling method [4] on KINS and COCOA.

$$Recall = \frac{\#\text{ of recovered instance pairs}}{\#\text{ of ground truth occlusion instance pairs}} \tag{5.3}$$

$$Accuracy = \frac{\#\text{ of recovered instance pairs with correct order}}{\#\text{ of recovered instance pairs}} \tag{5.4}$$

Finally, we plot the accuracy/recall points on the OAIR curve. The x-axis represents the various recall levels from the various IoU thresholds, and the y-axis represents the occlusion accuracy at their corresponding recall level. Note that if the recall is low, the number of recovered instance pairs is also low, which is why the accuracy-recall curves can be slightly unstable at the start/left of the curves.

Figure 5.4 shows the OAIR curves for the KINS and COCOA datasets. For occlusion accuracy, Mask2Former outperforms Mask R-CNN across all settings, which can be attributed to Mask2Former's attention mechanism being more explicit in modeling relationships between segments, a crucial aspect for occlusion reasoning. However, with the correct losses, both Mask2Former and Mask R-CNN can achieve very high occlusion accuracy ($> 95\%$ for KINS) across all recall levels. Experimentally, Loss 5.1 outperforms Loss 5.2 in all settings.

In Fig. 5.4 we can also compare our OAIR curves to that of the most related method [4], which is called Joint-Labeling. For the KINS dataset, both Mask2Former and Mask R-CNN with Loss 2 outperform Joint-Labeling, with a significantly greater improvement for the Mask2Former method. For the COCOA dataset, our Mask2Former method outperforms Joint-Labeling [4] by a large margin. This is likely due to our method being end-to-end, unlike Joint-Labeling.

In [4], they also compare their method with multiple baselines based on Mask2Former [13], and Mask R-CNN [36] to extract instances and *InstaOrderNet* [59] and *OrderNet* [123] to order them. Joint-Labeling [4] outperforms baselines by a margin, and we outperform Joint-Labeling.

5.2.3 Evaluation of Segmentation Accuracy

Tables 5.1 and 5.2 are the instance segmentation accuracies for the KINS and COCOA datasets using standard metrics and the same parameters as for the evaluation in Fig. 5.4. It

Method	L_{occl}	L_{occl} Weight	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l	AR
Mask2Former	No Occlusion	N/A	21.04	37.38	21.10	7.28	28.41	54.03	14.77
Mask2Former	Loss 5.1	0.5	**21.37**	**37.64**	**21.38**	**7.58**	**28.79**	**54.26**	14.76
Mask2Former	Loss 5.2	0.05	21.11	37.31	21.05	7.44	28.77	53.92	**15.14**
M. R-CNN	No Occlusion	N/A	25.68	44.68	26.58	9.62	33.34	57.01	17.83
M. R-CNN	Loss 5.1	0.1	25.80	44.89	**26.80**	**9.73**	**33.74**	**57.56**	**18.31**
M. R-CNN	Loss 5.2	0.05	**25.83**	**44.89**	26.64	9.47	33.19	56.48	17.78

Table 5.1: Segmentation accuracy for KINS. Median across 3 runs for all experiments.

Method	L_{occl}	L_{occl} Weight	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l	AR
Mask2Former	No Occlusion	N/A	28.35	46.06	28.78	4.30	19.67	40.86	9.67
Mask2Former	Loss 5.1	0.5	28.49	46.35	28.88	**4.45**	**19.68**	40.71	**9.72**
Mask2Former	Loss 5.2	0.05	**28.70**	**46.70**	**29.17**	4.00	19.51	**41.07**	9.71
M. R-CNN	No Occlusion	N/A	22.07	38.62	22.53	3.66	**16.85**	29.22	8.32
M. R-CNN	Loss 5.1	0.5	22.34	38.82	**22.89**	**3.73**	16.74	29.55	**8.42**
M. R-CNN	Loss 5.2	0.05	**22.35**	**38.94**	22.68	3.36	16.74	**29.65**	8.31

Table 5.2: Segmentation accuracy for COCOA. Median across 3 runs for all experiments.

Method	L_{occl}	L_{occl} Weight	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l	AR
Mask2Former	No Occlusion*	N/A	21.04	37.38	21.10	7.28	28.41	54.03	14.77
Mask2Former	Loss 5.1	1.0	20.65	36.58	20.58	7.40	28.18	52.25	14.49
Mask2Former	Loss 5.1*	0.5	21.37	37.64	21.38	7.58	28.79	**54.26**	14.76
Mask2Former	Loss 5.1	0.1	21.35	37.94	21.42	7.61	29.01	53.83	15.23
Mask2Former	Loss 5.1	0.05	**21.72**	**38.11**	**21.91**	**7.79**	**29.19**	53.94	**15.43**
Mask2Former	Loss 5.2	1.0	20.42	36.52	20.10	7.53	27.58	51.88	14.87
Mask2Former	Loss 5.2*	0.05	21.11	37.31	21.05	7.44	28.77	53.92	15.14
Mask2Former	Loss 5.2	0.001	**21.59**	**38.03**	**21.68**	**7.59**	**29.03**	**54.41**	**15.39**

Table 5.3: Segmentation accuracy for KINS across various weights. *The selected weights for Table 5.1; median across 3 runs.

follows that having the occlusion head generally increases the AP score for both Mask2Former and Mask R-CNN. However, the margin of increase is small. Note that the experiments in Tables 5.1 and 5.2 are with the weight of L_{occl} optimized to obtain the best OAIR curves, not for evaluation of instance segmentation accuracy alone.

Table 5.3 shows the results for different occlusion-order loss weights, and more significant performance gains are obtained if the weight of L_{occl} is optimized for the instance segmentation task alone. Both Loss 5.1's and Loss 5.2's weight tuning show a significant improvement across most metrics compared to the result without occlusion loss (top line). Thus, if accurate instance segmentation is the goal (occlusion order accuracy is not important), using occlusions as an additional source of supervision can significantly improve instance masks.

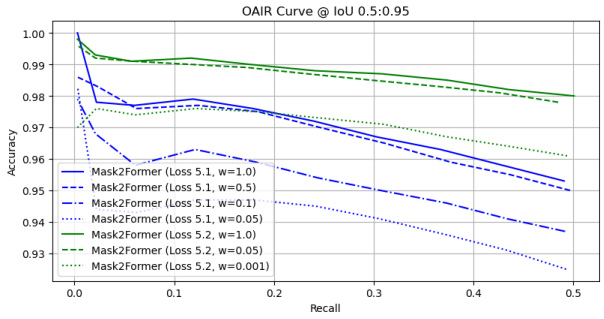


Figure 5.5: Occlusion OAIR curve for the KINS dataset across various weights.

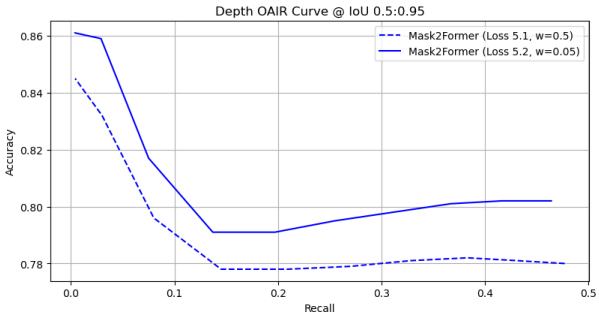


Figure 5.6: Depth OAIR curve for the KINS.

5.2.4 Ablations

Figure 5.5 and Table 5.3 show that varying weights for regularized loss can lead to large trade-offs between occlusion accuracies and segmentation accuracies. Although the regularized loss enforces a correlation between occlusion and segmentation task, an excessively high weight can cause segmentation performance to suffer, even falling below baseline performance without the regularized loss ('No Occlusion' line in Table 5.3). On the other hand, a low weight will provide a larger margin of increase in segmentation accuracies at the expense of lower occlusion accuracies. Depending on use cases/applications, this could be desirable in a strict segmentation setting where occlusion pairs are treated only as supplementary supervision.

5.2.5 Depth and Occlusion Correlation

Figure 5.6 shows the OAIR curve where instead of ground truth occlusions, we take the ground truth depth relation between *all* pairs of instances. Note that many of these pairs are not in an occlusion relation with each other and are not connected by a monotone chain of other instances in an occlusion relation with each other.

Pairwise depth orders between instances are obtained by taking the depth maps produced by LiDAR scans from the KITTI dataset. We then take the median depth value for each segment (if any) to calculate the pairwise depth orders between instances for a given image. Note that as LiDAR scans are sparse in nature with difficulties scanning distant objects, so far and/or small segments might not have any depth values associated with them and are thus omitted from calculation for the depth-based OAIR metric. The depth-based curve is evaluated on Mask2Former models trained on occlusion orders for KINS (same as the models from Table 5.1).

As shown in Fig. 5.6, our model trained on pairwise occlusion orders performs well on pairwise depth orders. Although our model does not produce the exact depth for each instance, it can provide an accurate relative depth ordering between instances, even for non-occluding instances that are isolated and far from each other. This shows a strong correlation between depth and occlusion information. Although pairwise occlusion order annotations are much less in volume (8.6% of instance pairs are occlusion pairs for KINS) compared to pairwise depth order, which exists between all instance pairs, our results suggest that occlusions themselves can provide the bulk of the information for the depth

Method	L_{occl}	L_{occl} Weight	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l	AR
Mask2Former	No Occlusion	N/A	22.65	40.53	22.36	**9.32**	30.07	53.62	15.27
Mask2Former	Loss 5.1	0.5	**23.05**	40.69	22.69	9.05	30.00	54.14	15.46
Mask2Former	Loss 5.2	0.05	23.00	**40.77**	**22.87**	9.15	**30.16**	**54.57**	**15.62**

Table 5.4: Amodal segmentation accuracy for KINS. Median across 3 runs for all experiments.

Method	L_{occl}	L_{occl} Weight	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l	AR
Mask2Former	No Occlusion	N/A	28.16	46.83	27.67	**5.06**	18.87	40.12	9.91
Mask2Former	Loss 5.1	0.5	28.27	**46.92**	27.98	4.62	**19.11**	40.30	9.90
Mask2Former	Loss 5.2	0.05	**28.31**	46.81	**28.04**	4.43	18.89	**40.44**	**9.94**

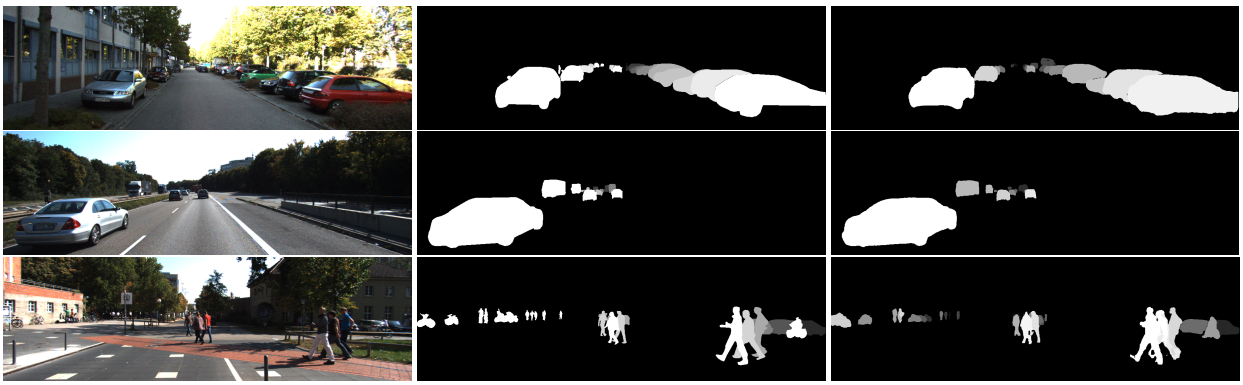
Table 5.5: Amodal segmentation accuracy for COCOA. Median across 3 runs for all experiments.

understanding of a scene.

5.2.6 Amodal Instance Segmentation

Tables 5.4 and 5.5 show the results when applying our methodology to amodal instance segmentation using Mask2Former, showing similar improvements as the standard instance segmentation in Section 5.2.3.

5.2.7 Qualitative Results



(a) (b) (c)

Figure 5.7: Examples from the KINS dataset: (a) Original image, (b) Ground masks and occlusion orders, (c) Predicted masks and occlusion orders. Prediction is done with Mask2Former using Loss 5.2.

Figures 5.7 and 5.8 show some example outputs on the KINS and COCOA datasets. The intensity for the ground truth occlusion order is based on the annotated occlusion order of each dataset: the order ID within each cluster for KINS (as shown in Fig. 5.2) and the



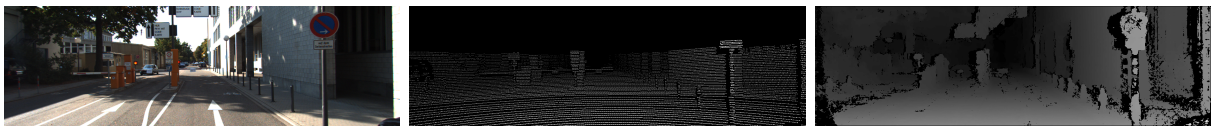
(a) (b) (c)

Figure 5.8: Examples from the COCOA dataset: (a) Original image, (b) Ground masks and occlusion orders, (c) Predicted masks and occlusion orders. Prediction is done with Mask2Former using Loss 5.2.

global occlusion order for COCOA, which is provided in addition to the pairwise occlusion order. The intensity of our predicted masks is the sorted order of the predicted global occlusion orders.

The global occlusion order is correct as long as for every pair of neighboring instances, the occluding instance has a larger label (appears brighter) than the instance it occludes. Therefore, for example, in Fig. 5.7 on the bottom right, the rightmost person on a bicycle in our prediction, although not as bright as in the ground truth, has a correct occlusion order as it is brighter than the two objects it occludes.

As mentioned in Section 5.2.5, despite training on occlusions, not depth, our approach learns to a large degree relative depths of objects. For example, in the bottom row of Fig. 5.7, the ground truth on the left has several isolated objects without any occlusion relations, and, therefore, they have the same label. For the same objects, our approach tends to label objects deeper in the scene with a smaller (less bright) label, even though they are not in occlusion relations with other objects.



(a) (b) (c)

Figure 5.9: Various ways depth could be annotated for an image: (a) Original image, (b) Raw LiDAR scans from KITTI [30], (c) Semi-global matching [38] results from stereo images.

5.3 Application: Improving Depth Maps

To demonstrate the usefulness of joint instance segmentation and occlusion ordering of instances, we develop an application that relies on this task to enhance depth maps during test-time training.

Object boundaries are traditionally a weakness when it comes to annotating ground-truth depth maps. One common approach to annotating depth maps is to use LiDAR scans (such as the raw depth data from the KITTI [30] dataset, see Fig. 5.9b); however, LiDAR scans are sparse and cannot capture object boundaries and distant objects. Another approach is to generate depth maps via stereo matching image pairs (such as the *Cityscape* [17] and *Middlebury* [93] datasets), but they can have uncertainties around one side of an object's boundary that is occluded from view from the other image, as shown from the left-side of objects in Fig. 5.9c. Many joint architectures have been created to predict per-pixel depth estimates along with segmentation ([118, 109, 29] as mentioned in Section 2 of the main paper), which have shown reduced noise around object boundaries and distant objects – after all, instance boundaries are strong cues for depth discontinuities.

However, when it comes to occlusion-ordered instances, they provide an additional important constraint, which is the direction of depth discontinuity among occluded objects. One simple application of our estimated occlusion-ordered instances is that they can be used to sharpen instance boundaries in depth maps, especially for distant objects, via an iterative test-time training approach [96] for heavily occluded scenes.

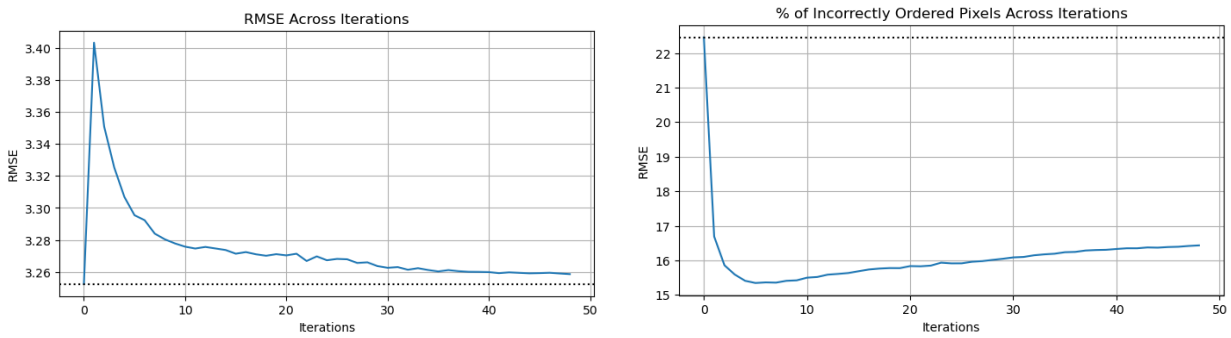
5.3.1 Methodology

Our experiment is set up as follows: we first train a baseline model for per-pixel depth estimation with a training dataset. Then, we further train the model for each test image with the occlusion order constraint. Given neighboring pixels p_i, p_j, where p_i, p_j are pixels of instances i, j respectively with $i \gg j$, we can apply the following losses:

$$L_o = \sum_{p_i, p_j} \text{ReLU}(d(p_i) - d(p_j)) \tag{5.5}$$

$$L_d = \sum_{p_i} \min(|d(p_i) - d_b(p_i)|, (d(p_i) - d_b(p_i))^2) \tag{5.6}$$

Where d is the predicted depth map at each step of the test-time training, and d_b is the original predicted depth using the baseline model. Similar to Loss 5.1, the loss L_o from Eq. (5.5) constrains the depths at pixels at occlusion boundaries to be in the correct order.



(a) (b)

Figure 5.10: Metrics across iterations: (a) Depth RMSE, (b) % of incorrectly ordered pixels among pixels at instance boundaries. Both are evaluated based on ground truth annotations averaged across all tested images. Lower is better.

Whereas L_d from Eq. (5.6) aims to discourage the training iteration from deviating too much from the original results by minimizing the distance between predicted depths from the current iteration and the baseline model.

Note that the occlusion boundaries for L_o are based on the predicted instance masks and occlusion orders taken from the Mask2Former [13] model using Loss 5.2; no ground truth information is used during the test time training.

The final loss combines the two with weighting λ and is applied to train iteratively:

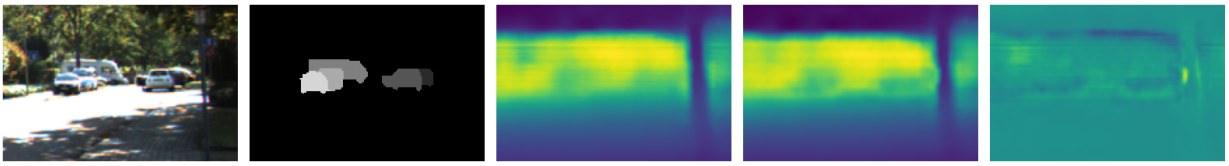
$$L_{\text{total}} = \lambda \cdot \frac{L_o}{HW} + \frac{L_d}{HW} \tag{5.7}$$

5.3.2 Architecture and Training Details

We use [108]'s implementation of *GLPDepth* [48] as mentioned in Section 2.6.1 with *Swin-v2-Base* [69] backbone to train a baseline model for per-pixel depth estimation on the KINS dataset using the default configurations for KITTI. The depth map for KINS is taken from LiDAR scans from the KITTI dataset. After training the baseline model, we further train the model for every fourth test image in KINS with the occlusion order constraint in a batch size of 6 (with only color augmentations) for 48 iterations at `max_lr`$= 3 \times 10^{-4}$ and `min_lr`$= 3 \times 10^{-5}$. Each image took $\sim$ 6 minutes on a single Nvidia RTX™ 6000 Ada Generation graphics card.

5.3.3 Depth Accuracy

As shown in Fig. 5.10, the metrics deviate strongly from depth in favor of occlusion ordering at the beginning, but stabilize and converge as the training converges. Note that the depth RMSE from Fig. 5.10a never improves beyond the baseline level – the evaluation is based on ground truth depth (LiDAR), which is sparse and cannot capture the improvements well on boundaries and distant objects. However, the results in Fig. 5.10b showed improvements on pixels at instance boundaries compared to the baseline level.



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

Figure 5.11: Test-time training on distant objects: (a) Original image, (b) Ground truth occlusion order, (c) Baseline depth estimation (before test-time training), (d) Test-time training depth estimation, (e) Difference between (c) & (d). For (c) & (d), brighter region signifies a higher depth; for (e), the test-time training pushed the depth further in bright regions, and pulled the depth closer in dark regions.



(a) Original image.

(b) Ground truth occlusion order.

(c) Difference between the baseline and test-time trained depth estimation.

Figure 5.12: Examples from test-time training results.

5.3.4 Qualitative Results

Figure 5.11 shows a zoomed-in example of distant objects that LiDAR was unable to capture accurately. The training slightly improves depth resolutions for faraway instances, with Fig. 5.11e showing a slight sharpening effect to the depth map produced after the test-time training. See Fig. 5.12 for more examples of the sharpening effect.

Chapter 6

Problem 2: Occlusion-Supervised Amodal Instance Segmentation

As mentioned in Section 2.5, *amodal* instance segmentation seeks to segment the whole extent of an object instance, i.e., both the visible and invisible (occluded) parts. Applications of AIS include autonomous driving and robot path planning [77], scene de-occlusion [113], 3D reasoning, image editing [66], etc. In this chapter, we will refer to the standard instance segmentation (visible part only) as *modal* segmentation.

Prior work on amodal instance segmentation can be divided into two groups. In the first group are the methods that generate synthetic datasets with amodal masks from modal mask datasets [62, 113, 77]. However, the construction of a realistic synthetic dataset can be difficult, often requiring extensive studies on distributions of sizes, locations, correlations, etc., among instance classes [85] (see Fig. 6.1). In the second group are the methods that use human-annotated amodal masks [123, 26]. However, amodal annotation inherently requires more effort compared to modal annotations, and such annotation is more error-prone and probabilistic, as the annotator must reason about the invisible object part.

In this chapter, we propose a new direction for amodal instance segmentation that differs from the two groups above. Our method does not require human amodal mask annotation *and* does not generate a synthetic dataset. Similar to Chapter 5, this method relies on ground-truth pairwise occlusion order together with modal instance masks for supervision. As mentioned before, compared to amodal annotation, occlusion order annotation is less labor-intensive and less error-prone, as humans are highly skilled at ordering objects in images [15]. Our method hinges on an important assumption about the dataset: for a given image, the total area of amodal masks must be the total area of modal masks. The



Figure 6.1: Examples of images from the synthetic *Amodal Cityscapes* dataset. Left: randomly inserting instances into an image. Right: a more plausible approach. [85]

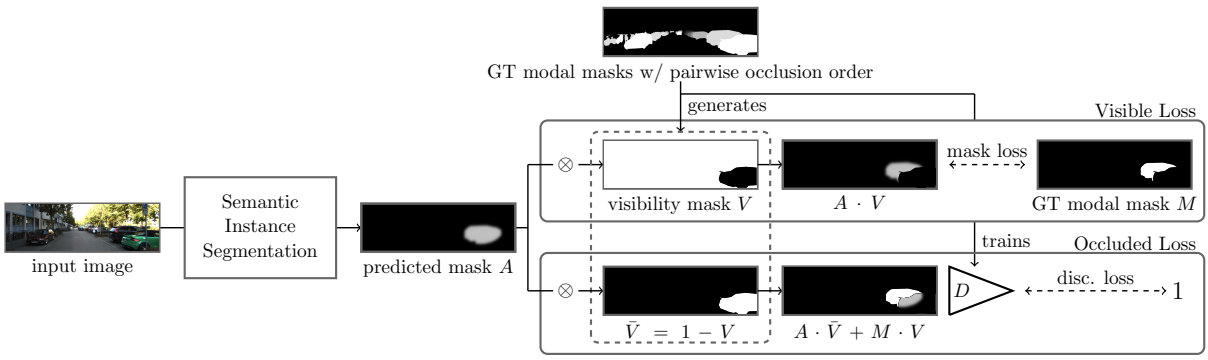


Figure 6.2: Illustration of our methodology. The visibility mask V allows us to incur corresponding losses on the predicted instance mask based on the visibility conditions inferred from the ground-truth occlusion order.

importance of this assumption will be explained in Section 6.1.1. Unlike Chapter 5, this method allows occlusion cycles and does not require a global occlusion order amongst the instances.

Our network outputs amodal masks. The most important component of our work is the design of an effective loss function for the amodal mask output based on the modal ground truth masks and their occlusion order. Our main idea is to break the loss into two components: the *visible* loss and the *occluded* loss. These losses are applied separately to each instance. The visible loss is applied to pixels that are visible or are not part of the instance. Visible loss can be implemented by cross-entropy with the modal ground truth, since it does not involve potentially occluded pixels. The occlusion annotation is essential to make the visible loss apply to the largest possible set of pixels. This is important because the visible loss, based on ground truth masks, is the more effective of our two loss components.

Visible loss does not apply to pixels that could be in the occluded (invisible) part of an instance. For these pixels, the modal ground truth mask cannot be used to guide the loss. For such pixels, an intuitive choice is to impose a shape prior. Therefore, our occluded loss is designed to encourage the shape of the amodal mask to correspond to an unoccluded object shape. We develop a novel approach to shape prior loss, inspired by GANs [35]. We treat the output of our network as the generator, and we train a discriminator between unoccluded and occluded masks. Since we have modal masks and pairwise occlusions, for discriminator training, we can determine which of the modal masks are occluded and which are unoccluded. See Fig. 6.2 for a visual overview of our methodology.

With our loss, our approach allows us to augment existing single-stage (i.e., non-Region-of-Interest-based) instance segmentation architectures to perform amodal instance segmentation. To test our approach, we use Mask2Former architecture [13]. Same as Chapter 5, we use COCOA [65] and KINS [82] datasets for evaluation, as these datasets have annotations for both pairwise occlusion (for training) and amodal segmentation masks (for evaluation). Our approach results in performance close to that of the same architecture trained on human-annotated amodal masks, even though we only use modal masks for training. Our approach can be extended to other architectures and other ways of incorporating shape prior.

6.1 Methodology

6.1.1 Assumptions

Let $A_i(p)$ and $M_i(p)$ be the amodal/modal mask probabilities of instance i at pixel p, where:

$$1 \geq A_i(p) \geq M_i(p) \geq 0 \tag{6.1}$$

As for instances $i \neq j$, let $j \gg_p i$ denote that j occludes i at pixel p, where:

$$j \not\gg_p i \implies M_j(p) \cdot A_i(p) = 0 \tag{6.2}$$

Note that occlusion is not guaranteed if $j \gg_p i$; only that there is no overlap between the modal mask of the closer instance and the amodal mask of the further instance if $j \not\gg_p i$. This generalizes the definition of "occlusion" between KINS and COCOA, which annotates occlusion differently as shown in Fig. 4.1 in Chapter 4. Now consider the following *assumption*, where the total area of amodal masks (denoted using the product operator as amodal masks could overlap each other) must be the same as the total area of modal masks:

$$\sum_i M_i(p) = 1 - \prod_i (1 - A_i(p)) \tag{6.3}$$

With the above assumption, the following holds:

Proposition 6.1.1. *Suppose instances are ordered from back to front such that for instances $i, j \in \{1, \cdots, N\}$, $j \gg_p i \implies j > i$ at pixel p. Then for an arbitrary instance i:*

$$M_i(p) = A_i(p) \prod_{j>i} (1 - A_j(p)) \tag{6.4}$$

Please refer to Appendix A.1 for a proof of Proposition 6.1.1. Eq. (6.4) enables us to turn amodal masks into modal masks, given that we know the global occlusion order at each pixel. The latter product operator part ($\prod_{j>i}(1 - A_j(p))$) can be seen as the *visibility condition* of instance i, which we will denote as the *visibility mask* $V_i(p)$ at some pixel p:

$$M_i(p) = A_i(p) \cdot V_i(p) \tag{6.5}$$

Where $V_i(p) = 1$ means that either pixel p belongs to instance i and is visible, or pixel p does not belong to instance i. $V_i(p) = 0$ means that if pixel p belongs to instance i, it must be occluded (invisible).

Another way to think about $V_i(p)$ in Eq. (6.4) is through volume rendering techniques in computer graphics [45]. $V_i(p)$ can be seen as the accumulated transmittance (or attenuation) at the given pixel, i.e., the probability that a ray travels until instance i without being "absorbed" by the previous instances, based on the probability of the previous amodal segmentation masks. See Fig. 6.3 for a visual illustration. Such parallels are described in detail in Appendix B.

With $A_i(p)$ and $V_i(p)$, we can then infer the probability of the modal segmentation mask at pixel p as $M_i(p)$ from Eq. (6.4). To provide an intuitive understanding:

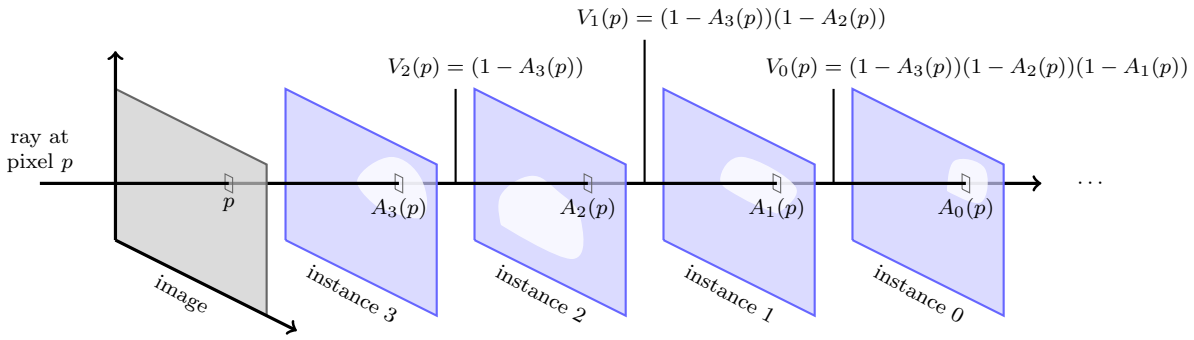


Figure 6.3: Visual representation of Eq. (6.4) as a ray at pixel p, where each $V_i(p)$ can be seen as the remaining amount of transmittance not "absorbed" by the previous occluding instances. As the ray travels through each instance, $A_j(p)$ of the ray is absorbed. The amodal segmentation masks are shown for each instance.

- If a pixel is not a part of the amodal mask (i.e., low $A_i(p)$), it will not be a part of the modal mask (i.e., low $M_i(p)$).
- If a pixel is part of the amodal mask but is heavily occluded (i.e., high $A_i(p)$ but low $V_i(p)$), it will not be a part of the modal mask (i.e., low $M_i(p)$).
- If a pixel is part of the amodal mask and is not occluded (i.e., high $A_i(p)$ and high $V_i(p)$), it will be a part of the modal mask (i.e., high $M_i(p)$).

Do note that Eq. (6.3) is an assumption, and is not true for most datasets. While this holds for KINS, in COCOA, certain instances can be occluded by objects not part of the image annotations. In such cases, a pixel can be part of the amodal mask, not be occluded by other annotated masks, yet not be part of the modal mask due to such occlusion. This happens for $\sim 10.7\%$ of "unoccluded" instances for the validation set in COCOA, and results in the area of the amodal masks being larger than that of the modal masks. However, if a dataset's annotations are dense enough, such inconsistencies should hopefully be minimal.

With that in mind, Eq. (6.4) allows us to impose mask loss between the ground-truth modal masks and predicted amodal masks by converting the predicted amodal masks $A_i(p)$ into predicted modal masks $M_i(p)$ based on a ground-truth occlusion order (or a predicted occlusion order as demonstrated in Chapter 5). This allows us to use the resultant $M_i(p)$ when calculating losses against ground-truth modal masks, with the visibility masks $V_i(p)$ representing the subset of pixels from $A_i(p)$ that can be trained against the ground-truth modal masks. Immediately, however, this method can be seen as quite unstable; bad predictions during early iterations for segmentation masks at the front can have large effects on masks behind. When back-propagating, more of the gradients are stacked towards the closer instances as the close instances all contribute to the calculation of the far instances and may require careful weighting of such gradients (or even freezing the latter product operator part) for stability. A significant amount of computational effort is also required to circumvent the numerical instabilities in gradients caused by the product operator during backpropagation, which are detailed in Appendix C.

Luckily, there are alternatives that let us better make use of the binary ground-truth modal masks. Consider the following, which does not make use of any explicit occlusion ordering:



(a) Instances (b) A_0 (c) M_0 (d) V_0 in Eq. 6.6 (e) V_0 in Eq. 6.7

Figure 6.4: Examples of ground-truth masks based on instance 0. White represents a value of 1 and black represents 0. Note that A_0 is for illustrative purposes only, as we do not have access to the ground-truth A_0 during training.

Proposition 6.1.2. *Suppose $M_i(p)$'s are binary. Then for an arbitrary instance i:*

$$M_i(p) = A_i(p)\left(1 - \sum_{j \neq i} M_j(p)\right) \tag{6.6}$$

Please refer to Appendix A.2 for a proof of Proposition 6.1.2. In the case of Eq. (6.6), visibility mask $V_i(p) = (1 - \sum_{j \neq i} M_j(p))$. All $V_i(p)$ does here is remove all areas occupied by another instance's modal mask from instance i's amodal mask, which should result in the modal mask of instance i. This works because modal masks represent visible sections of each instance, hence, each instance must be the front-most instance in its corresponding modal mask. With this, we can infer a predicted modal mask $M_i(p)$ based on the predicted amodal mask $A_i(p)$ and the ground-truth modal masks in $V_i(p)$. However, with access to ground-truth pair-wise occlusion order, we can do better:

Proposition 6.1.3. *Suppose $M_i(p)$'s are binary. Then for an arbitrary instance i:*

$$M_i(p) = A_i(p)\left(1 - \sum_{j \gg_p i} M_j(p)\right) \tag{6.7}$$

Please refer to Appendix A.3 for a proof of Proposition 6.1.3. Compared to Eq. (6.6), Eq. (6.7) excludes all instances that do not occlude instance i based on the ground-truth pairwise occlusions. While both formations are perfectly valid, Eq. (6.7) is superior when used for training, as its visibility mask $V_i(p) = (1 - \sum_{j \gg_p i} M_j(p))$ (Fig. 6.4e) encompasses a larger area compared to that of Eq. (6.6) (Fig. 6.4d). As $V_i(p)$ denotes the subset of pixels from $A_i(p)$ that can be trained against the ground-truth modal masks, this results in more of the predicted $A_i(p)$ showing up in $M_i(p)$ when calculating loss against the ground-truth modal masks, hence $A_i(p)$ can be better trained. See Fig. 6.4 for a visual illustration. Moving forward, all $V_i(p)$ will refer Eq. (6.7). Notation-wise, for convenience, we will use $\bar{V}_i(p) = 1 - V_i(p)$.

Since ground-truth modal masks are binary/one-hot (i.e. $M_i(p) \in \{0, 1\}, \forall i$), the visibility mask $V_i(p)$ can easily be generated for each instance as a binary mask as part of pre-processing, where imposing training losses (or any other cost calculations) on $M_i(p) = A_i(p) \cdot V_i(p)$ is akin to imposing losses on only the unoccluded (i.e. visible) regions for each instance $A_i(p)$ by masking away the occluded regions via binary mask $V_i(p)$. $V_i(p)$ can then be used as part of our proposed losses, as explained in the sections below.

The total segmentation mask loss for each instance i is a weighted combination of the following proposed losses:

$$L_{\text{Mask}} = L_{\text{Visible}} + L_{\text{Occluded}} \tag{6.8}$$

6.1.2 Visible Losses

For each instance i, the visible loss can be calculated by using Eq. (6.5), by taking a subset of the predicted amodal mask $M_i(p)$ based on pixels p s.t. $V_i(p) = 1$. As mentioned in Section 2.4.3, Mask2Former uses BCE and Dice loss when calculating the bipartite matching score and/or the training loss. Let ${M_i}^*(p)$ be the ground truth visible mask for instance i at pixel p. Then, plugging Eq. (6.5) into the BCE loss (see Eq. (2.11)), we get:

$$\begin{aligned} L_{\text{BCE}|V_i}(A_i, {M_i}^*) &= -\sum_p [{M_i}^*(p) \log(M_i(p)) + (1 - {M_i}^*(p)) \log(1 - M_i(p))] \\ &= -\sum_p [{M_i}^*(p) \log(A_i(p) \cdot V_i(p)) + (1 - {M_i}^*(p)) \log(1 - A_i(p) \cdot V_i(p))] \end{aligned}$$

However, when $V_i'(p) = 0$ (due to the one-hot nature of the ground truth modal masks), $\log(0)$ becomes undefined. Hence, in practice, we use the following to filter out the masked pixels:

$$L_{\text{BCE}|V_i}(A_i, {M_i}^*) = -\sum_p V_i(p) \cdot [{M_i}^*(p) \log(A_i(p)) + (1 - {M_i}^*(p)) \log(1 - A_i(p))] \tag{6.9}$$

Whereas plugging Eq. (6.5) into the Dice loss (see Eq. (2.12)), is more straightforward:

$$\begin{aligned} L_{\text{Dice}|V_i}(A_i, {M_i}^*) &= 1 - \frac{2 \cdot \sum_p (M_i(p) \cdot {M_i}^*(p)) + \epsilon}{\sum_p M_i(p) + \sum_p {M_i}^*(p) + \epsilon} \\ &= 1 - \frac{2 \cdot \sum_p (A_i(p) \cdot V_i(p) \cdot {M_i}^*(p)) + \epsilon}{\sum_p (A_i(p) \cdot V_i(p)) + \sum_p {M_i}^*(p) + \epsilon} \end{aligned} \tag{6.10}$$

One advantage of such losses is that they allow joint training using visible masks with pairwise occlusion order for some images, while using the full amodal masks for others. In the case of training using amodal masks, we can set $V_i = 1$ for all pixels, which will convert the above losses to the standard BCE/Dice losses. This is explored further in Section 6.2.5 and could be useful for combining various datasets with similar domains but varying levels of supervision.

As these losses are also used for the bipartite matching of Mask2Former, which requires the loss to be calculated between every pair of ground-truth and predicted masks, $\sum_p (A_i(p) \cdot V_i(p))$ at the denominator of the Dice loss (Eq. (6.10)) adds a small overhead during bipartite matching. As $V_i(p)$ is calculated using ground-truth modal masks via Eq. (6.7), an additional broadcasting step between the ground-truth dependent $V_i(p)$'s and the predicted $A_i(p)$'s is needed for bipartite matching.

With regards to *PointRend*, which is used to calculate the final loss for Mask2Former (see Section 2.4.2), *PointRend* can be constrained to only sample within the $V_i(p)$'s by randomly sampling points using $V_i(p)$ as a multinomial probability distribution.

The total visible loss is the weighted sum of the BCE and Dice-based losses:

$$L_{\text{V}} = \lambda_{\text{BCE}} L_{\text{BCE}|V_i}(A_i, {M_i}^*) + \lambda_{\text{Dice}} L_{\text{Dice}|V_i}(A_i, {M_i}^*) \tag{6.11}$$

6.1.3 Occluded Loss

While the visible loss above allows training by only imposing cost/loss functions on the visible areas $V_i(p)$, what happens in the invisible regions $\bar{V}_i(p)$ is entirely up to the architecture itself. Thus, training with the visible loss alone results in an amodal segmentation, but it is unlikely to be accurate. To obtain better results, we can impose a shape prior via an *occluded* loss L_{O} on the occluded areas $\bar{V}_i(p)$ explicitly.

Our method is agnostic to shape prior implementations. One possible implementation is to use encoder-decoder approaches, such as those mentioned in 3.4, to perform completion on the occluded sections of each amodal mask. However, such approaches are computationally expensive. In addition, explicitly performing amodal mask completion through encoder-decoders would result in doubled effort, as our model already generates masks. Finally, encoder-decoder approaches are usually used within a two-stage framework on cropped segments based on their bounding box to maximize the resolution of the decoded shape, which does not apply to our case, as we do not have amodal bounding boxes for supervision.

For our experiments, we instead opt for a more straightforward approach by using an additional conditional discriminator as part of a GAN. Recall from Section 3.4.2, in GANs, the generator G generates data based on a target distribution, and the discriminator D distinguishes between the outputs generated by G and the real data. In our case, we treat our model as the generator G, and it is responsible for generating the distribution of amodal masks. We add a lightweight discriminator D, which performs binary classification of whether an instance is unoccluded (based on the pairwise occlusion annotations), encouraging the amodal mask for each instance to have an unoccluded shape. Note that our method does not follow GAN exactly, as it is not adversarial in nature. Traditionally, GAN-based discriminators are trained to distinguish between real and generated inputs, with the generator trained to fool the discriminator. In our case, as we already have access to positive/negative inputs in the form of unoccluded/occluded instance masks, we can use them to train the discriminator directly with no need for adversarial training. A significant advantage of using discriminators is the relatively small overhead compared to other methods to impose shape priors. By treating the model itself as the generator as part of a GAN, using an additional discriminator is much closer to directly applying a regularization loss to the occluded regions without needing any decoders, which significantly lowers the computational effort. For amodal mask completion, [110, 122, 115] use discriminators for masks. However, all these methods require ground-truth or synthetic amodal masks for training, unlike our work.

Thus, our discriminator loss (L_{D}) and generator loss (L_{G}) are:

$$L_{\text{D}|V_i}(M_i^*, y_i^*) = \begin{cases} -\log D(M_i^*, y_i^*) & i \text{ is unoccluded} \\ -\log(1 - D(M_i^*, y_i^*)) & \text{if otherwise} \end{cases} \tag{6.12}$$

$$L_{\text{G}|V_i}(A_i, M_i^*, y_i^*) = \begin{cases} 0 & i \text{ is unoccluded} \\ \log(1 - D_{\text{frozen}}(A_i \cdot \bar{V}_i + M_i^* \cdot V_i, y_i^*)) & \text{if otherwise} \end{cases} \tag{6.13}$$

Where D and G denote the discriminator and generator, respectively, with y_i^* representing the true class label of instance i and used as part of our discriminator input, similar to a cGAN (see Section 3.4.2). Here, L_{D} aims to distinguish between ground-truth amodal masks and ground-truth visible masks; while L_{G} forces the generated masks to become amodal-like. See Fig. 6.5 for a visual illustration of these losses.

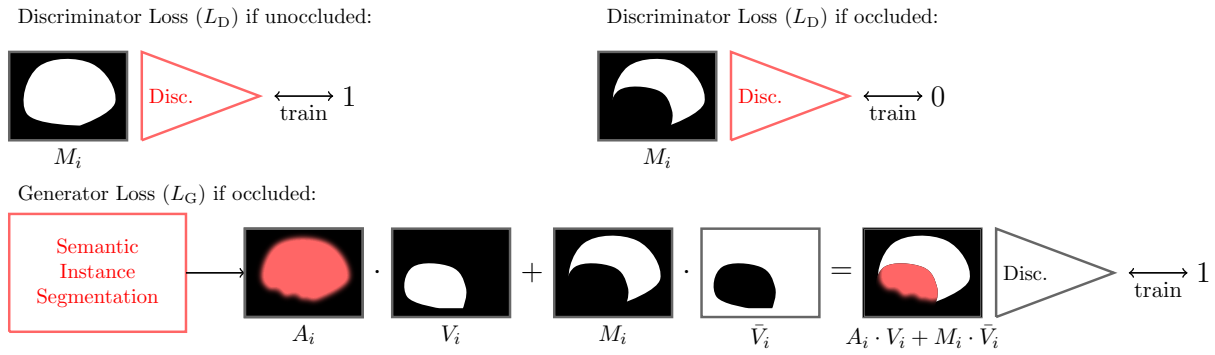


Figure 6.5: Illustration of our GAN-inspired occluded losses L_D and L_G. Red represents the trained/unfrozen components.

Note that instead of directly using A_i as the input to the discriminator in L_G, we use A_i in the occluded part (multiplication by $\bar{V}_i$) and the ground truth visible mask M_i in the visible part (multiplication by V_i). This effectively ensures that our occluded loss influences only the occluded part of A_i, as for the unoccluded part of A_i, the visible loss developed in Section 6.1.2 is more effective. This also improves training efficiency. Please refer to the supplementary materials for comparisons between the above and directly using A_i.

As the ground truth visible masks are discrete, our discriminator is trained using discrete data. However, when it comes to the input to the discriminator in the generator loss L_G, binarizing the continuous probabilistic masks A_i does not work as the operation cannot be back-propagated. Additional mechanisms would be required to handle the gradient. Such "discrete GAN" is little researched, with most of the studied cases of discrete GANs focused on NLP or other discrete sequence problems [55, 20], and relies on Monte Carlo-based gradient estimates [76], such as the REINFORCE method [107]. REINFORCE adds some form of random noise to the discretized generator output and directly copies the gradient from the discretized outputs to the original continuous outputs when back-propagating. However, "random noise" is difficult to define in the context of segmentation masks. We opt for a simpler approach by directly passing the continuous probabilistic masks in the generator loss L_G. This works well despite the domain gap with the discrete data that the discriminator is trained on.

The final occluded loss is the weighted sum of the two losses:

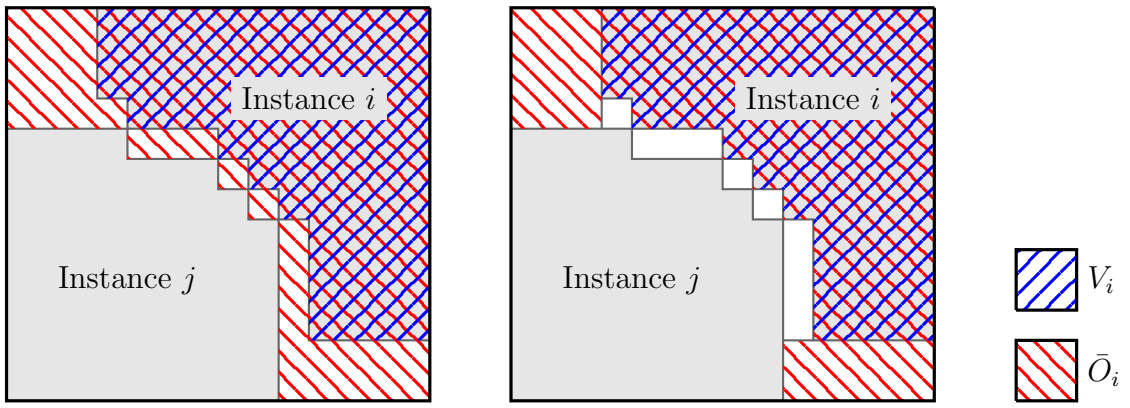
$$L_O = \lambda_G L_{G|V_i}(M_i^*, y_i^*) + \lambda_D L_{D|V_i}(M_i^*, y_i^*) \tag{6.14}$$

6.2 Experimental Results

6.2.1 Architecture and Training Details

As mentioned in Chapter 4, we use the We use the *KITTI INStance Segmentation* (KINS) dataset [82] and the *COCO Amodal* (COCOA) dataset [123].

Let $M_i \in \{0,1\}^{W \times H}$ be the modal segmentation masks for instance i. The *visibility mask* $V_i \in \{0,1\}^{W \times H}$ is first created by combining the modal binary segmentation masks of all occluders of instance i. Note that ground truth annotations with small gaps between modal segmentation masks (see Fig. 6.6a) can cause the model to train such regions as not



(a) Bad ground truth annotation. (b) Modified V_i'. (c) Legend.

Figure 6.6: Here instance j occludes instance i (i.e., $j > i$). Notice that the erroneous gap caused by bad ground truth annotation is resolved.

part of the amodal segmentation masks. As segmentation models tend to produce smooth segmentation masks, such gaps can cause the model to fit for the modal mask of instance i, despite the lack of training loss on the region occupied by the occluding instance j. As a result, to avoid small 1-pixel gaps due to small errors in annotation, a *closing* morphology transformation is applied to the occlusion mask via dilation and erosion with a kernel size of 3×3:

$$V_i \leftarrow 1 - (\text{erode}(\text{dilate}((1 - V_i) \vee M_i)) \wedge \neg\, \text{erode}(\text{dilate}(M_i))) \tag{6.15}$$

The above closes all gaps between V_i and M_i, resulting in the new V_i as shown in Fig. 6.6b. Handling the gap is also vital for *PointRend*; if the gap is unresolved, *PointRend* is likely to sample points from the gap as it is designed to sample points from uncertain areas around the boundary, which intensifies the error.

To improve the performance of the generator loss at early iterations of the training and reduce computational effort between experiments, we pre-train the discriminator network using $L_{\text{D}|V_i}$ (see Eq. (6.12)) as the loss requires only ground-truth masks.

We use a simple cGAN consisting of 6/7 (for KINS/COCOA) convolutional layers and 2 more fully connected layers. The one-hot class labels are concatenated with the flattened output from the convolutional layers, which is then passed as input to the fully connected layers. Our cGAN follows all conventions as suggested in [83], with BatchNorms and LeakyReLUs (with a slope of 0.2).

Similar to Chapter 5, for our main segmentation model, we use Mask2Former with ResNet-50 [37] backbone. For all experiments of KINS, the configuration mimics the default configurations for *Cityscapes* [17], another road scenes dataset. The configuration for COCOA mirrors the default configuration for COCO. All experiments use the standard weights of $\lambda_{\text{BCE}} = \lambda_{\text{Dice}} = 5$ provided by Mask2Former for the aforementioned datasets, and are run on a single Nvidia RTX™ 6000 Ada Generation graphics card.

Similar to Chapter 5, the pretrained discriminators were run with 8 images per batch$\times 90,000$ for KINS and $\times 40,000$ for COCOA at learning rate 0.000075 with $\lambda_{\text{D}} = 1$ ($\sim$ 7 hours for KINS and $\sim$ 4 hours for COCOA). The pretrained discriminators were then frozen and attached to the main segmentation models (if required). The main models were run at the same number of iterations and learning rate as the pretrained discriminators ($\sim$ 26 hours

Losses	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l	AR
Modal Baseline*	16.68	32.51	15.36	6.16	21.66	51.00	13.96
L_{V}	21.65	40.00	20.57	8.78	27.97	53.09	15.50
$L_{\mathrm{V}} + L_{\mathrm{O}}$ ($\lambda_{\mathrm{G}} = 0.1$)	22.08	40.66	21.22	8.68	28.71	52.78	16.01
Amodal Baseline	22.95	40.72	22.86	9.31	29.94	54.01	15.49

Table 6.1: Amodal segmentation accuracies for KINS. Median across 3 runs. *Trained using ground-truth modal masks and evaluated on amodal masks.

Losses	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l	AR
Modal Baseline*	23.59	40.85	22.68	4.27	16.58	32.65	9.28
L_{V}	25.53	43.84	24.52	4.22	17.44	35.47	9.63
$L_{\mathrm{V}} + L_{\mathrm{O}}$ ($\lambda_{\mathrm{G}} = 0.2$)	25.87	44.11	25.13	4.45	17.64	35.91	9.65
Amodal Baseline	28.46	47.44	28.06	5.04	19.42	40.19	9.94

Table 6.2: Amodal segmentation accuracies for COCOA. Median across 3 runs. *Trained using ground-truth modal masks and evaluated on amodal masks.

for KINS and $\sim$ 13 hours for COCOA).

6.2.2 Segmentation Accuracy

We first evaluated the amodal baseline (trained and evaluated against amodal ground truth) and the modal baseline (trained with modal ground truth but evaluated against amodal ground truth. As shown in Tables 6.1 and 6.2, all of our results sit between the baselines, with a sizable performance increase from the modal baseline.

Using the visible loss L_{V} alone is able to boost the amodal performance by a large margin. Even without incurring any losses on the occluded regions of each instance, Mask2Former is able to extend the predicted masks beyond the visible regions. Adding in L_{G} is able to push the performance slightly further, despite the rudimentary nature of our shape prior.

Note the difference in performance increase between KINS and COCOA. Our methods are able to push the performance much closer to the amodal baseline for KINS than for COCOA. This can be attributed to the more diverse and complex shapes within COCOA, as opposed to the car-centric street views of KINS, as well as errors in the training dataset when training the discriminator for COCOA. Note that in COCOA as mentioned in 6.1.1, an "unoccluded" instance can sometimes be occluded by an unlabeled object. This introduces errors in our training occluded/unoccluded dataset, with such errors happening for $\sim$ 10.7% of "unoccluded" instances for COCOA. This is reflected when evaluating the binary classification performance of the pretrained discriminator against ground truth unoccluded/occluded masks, with the discriminator for KINS achieving a classification accuracy of 86.54% as opposed to COCOA's 68.30%.

Losses	λ_G	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l	AR
$L_V + L_O$	0.01	21.68	39.79	20.74	8.39	28.70	53.42	15.12
$L_V + L_O$	0.05	22.05	39.69	**21.40**	8.54	28.52	**53.71**	15.31
$L_V + L_O$*	0.1	**22.08**	**40.66**	21.22	**8.68**	**28.71**	52.78	**16.01**

Table 6.3: Amodal segmentation accuracies for KINS across various L_G weights. *The selected weights for Table 6.1; median across 3 runs.

Losses	λ_G	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l	AR
$L_V + L_O$	0.1	25.41	43.57	24.55	3.94	17.36	35.46	9.40
$L_V + L_O$*	0.2	**25.87**	**44.11**	**25.13**	4.45	**17.64**	**35.91**	**9.65**
$L_V + L_O$	0.5	25.48	43.26	24.98	3.84	17.59	35.52	9.51
$L_V + L_O$	4.0	24.46	41.77	23.72	3.82	17.02	33.66	9.58

Table 6.4: Amodal segmentation accuracies for COCOA across various L_G weights. *The selected weights for Table 6.2; median across 3 runs.

6.2.3 Ablations

Tables 6.3 and 6.4 shows various weights for L_G and their segmentation accuracies. In particular, excessively large losses (e.g., $L_G = 4$ in Table 6.4) can lead to a large drop in performance.

6.2.4 Alternative Generator Loss

Tables 6.5 and 6.6 compare the performance between the generator loss from Eq. 7 of the main paper and the following:

$$L_{G2}(A_i, M_i, y_i) = \begin{cases} 0 & i \text{ is unoccluded} \\ \log(1 - D_{\text{frozen}}(A_i, y_i)) & \text{if otherwise} \end{cases} \tag{6.16}$$

L_{G2} applies the generator loss to the entire predicted amodal mask A_i, as opposed to only the occluded regions, achieved by replacing the visible regions with ground truth in Eq. 7 of the main paper. In general, L_G outperforms L_{G2}. For KINS, the performance difference is minimal. However, for COCOA, applying the generator loss to the entire mask causes a slight drop in performance. This could be attributed to the difference in performance of the discriminator, as mentioned in the main paper, which affects the performance of the generator loss. The lower performance of the generator loss naturally generates more inaccuracies, which is magnified when applied to the whole predicted amodal mask A_i.

6.2.5 Mixing Amodal Supervision

A major advantage of our method is that it is possible to jointly train using visible masks with pairwise occlusion order for some images, while using the full amodal masks for others. To test this direction, we train on KINS with 50% of the images using amodal ground truth masks and the other 50% using modal ground truth masks with pairwise occlusion order.

Losses	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l	AR
$L_{\mathrm{V}} + L_{\mathrm{O}}$ ($\lambda_{\mathrm{G}} = 0.1$)	22.08	40.66	21.22	8.68	28.71	52.78	16.01
$L_{\mathrm{V}} + L_{\mathrm{O}}$ ($\lambda_{\mathrm{G2}} = 0.1$)	22.06	40.34	20.99	8.54	28.65	54.41	15.69

Table 6.5: Amodal segmentation accuracies for KINS between L_{G} and L_{G2} (Eq. (6.16)). Median across 3 runs.

Losses	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l	AR
$L_{\mathrm{V}} + L_{\mathrm{O}}$ ($\lambda_{\mathrm{G}} = 0.2$)	25.87	44.11	25.13	4.45	17.64	35.91	9.65
$L_{\mathrm{V}} + L_{\mathrm{O}}$ ($\lambda_{\mathrm{G2}} = 0.2$)	25.44	43.7	24.44	4.17	17.42	35.27	9.77

Table 6.6: Amodal segmentation accuracies for COCOA between L_{G} and L_{G2} (Eq. (6.16)). Median across 3 runs.

For this experiment, we use only visible loss L_{V}, omitting the occluded loss L_{O} as the amodal ground truth already directly provides a form of shape prior. The results are in Table 6.7. As expected, the performance of such "mixed" model sits between our original models and the amodal baseline. This could be useful for combining various datasets with similar domains but varying levels of supervision.

6.2.6 Qualitative Results

Fig. 6.7 shows two examples of predicted outputs for a single instance in KINS and COCOA. For KINS, the result produced by training only with the visible loss L_{V} is able to extend into the occluded region from the modal mask, while training with L_{O} creates a more natural shape within the unoccluded region.

However, for COCOA, the result produced by training only with the visible loss L_{V} largely remains modal with very slight extensions into the occluded areas (e.g., tip of the fork). Training with L_{O} produces more plausible amodal masks, but a large amount of noise can be seen (e.g., around the cake area). This could be attributed to the performance of the discriminator, which generalized better for the simpler shapes in KINS. Please refer to Figs. 6.8 and 6.9 for examples of predicted outputs for Chapter 6.

Losses	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l	AR
L_{V}*	21.65	40.00	20.57	8.78	27.97	53.09	15.50
$L_{\mathrm{V}} + L_{\mathrm{O}}$* ($\lambda_{\mathrm{G}} = 0.1$)	22.08	40.66	21.22	8.68	28.71	52.78	16.01
$L_{\mathrm{V}} + 50\%$ Amodal	22.56	40.51	22.20	9.31	29.64	54.05	15.01
Amodal Baseline*	22.95	40.72	22.86	9.31	29.94	54.01	15.49

Table 6.7: Amodal segmentation accuracies for KINS. *Median across 3 runs.

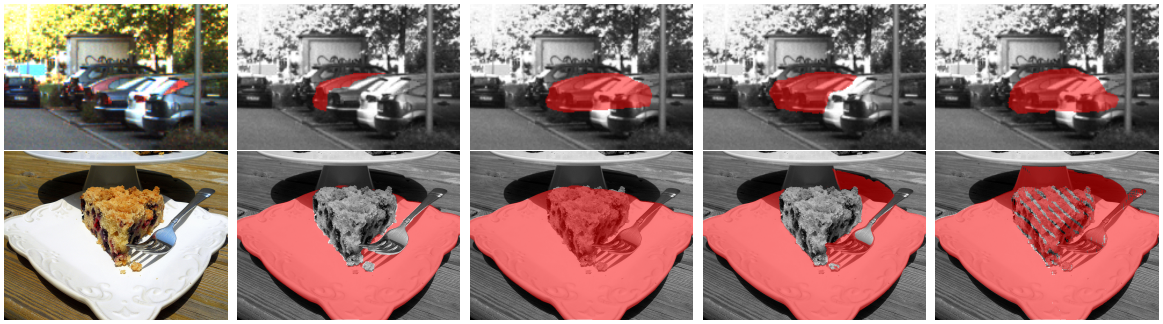


Figure 6.7: From top to bottom: an example from *KIN* and from *COCOA*. From left to right: original image, ground-truth modal mask, ground-truth amodal mask, predicted amodal mask trained with only L_{V}, predicted amodal mask trained with $L_{\mathrm{V}} + L_{\mathrm{O}}$.

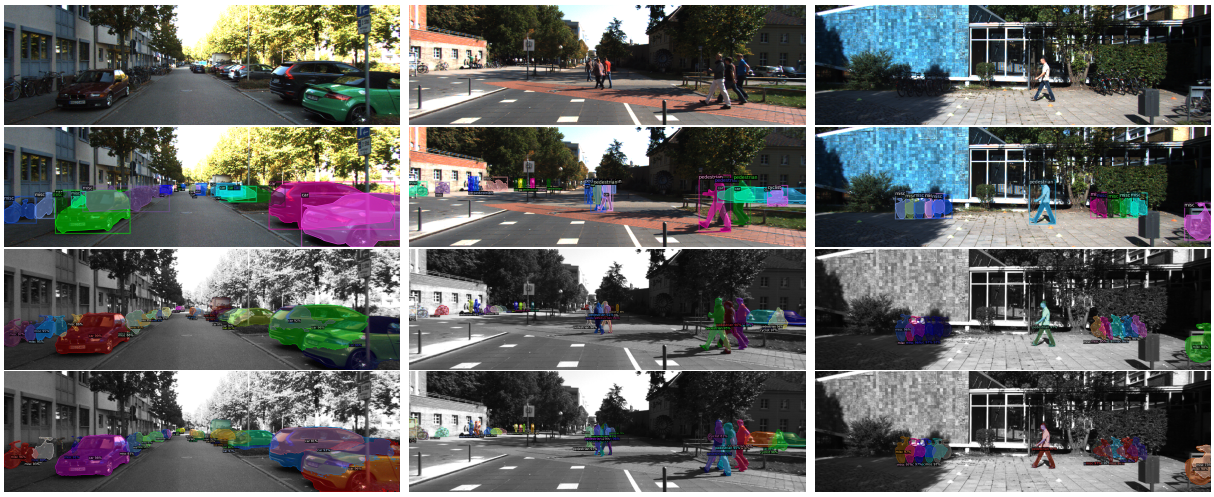


Figure 6.8: Results from *KINS*. From top to bottom: original image, amodal ground truth masks, predicted masks trained with only L_{V}, predicted masks trained with both L_{V} and L_{O}.

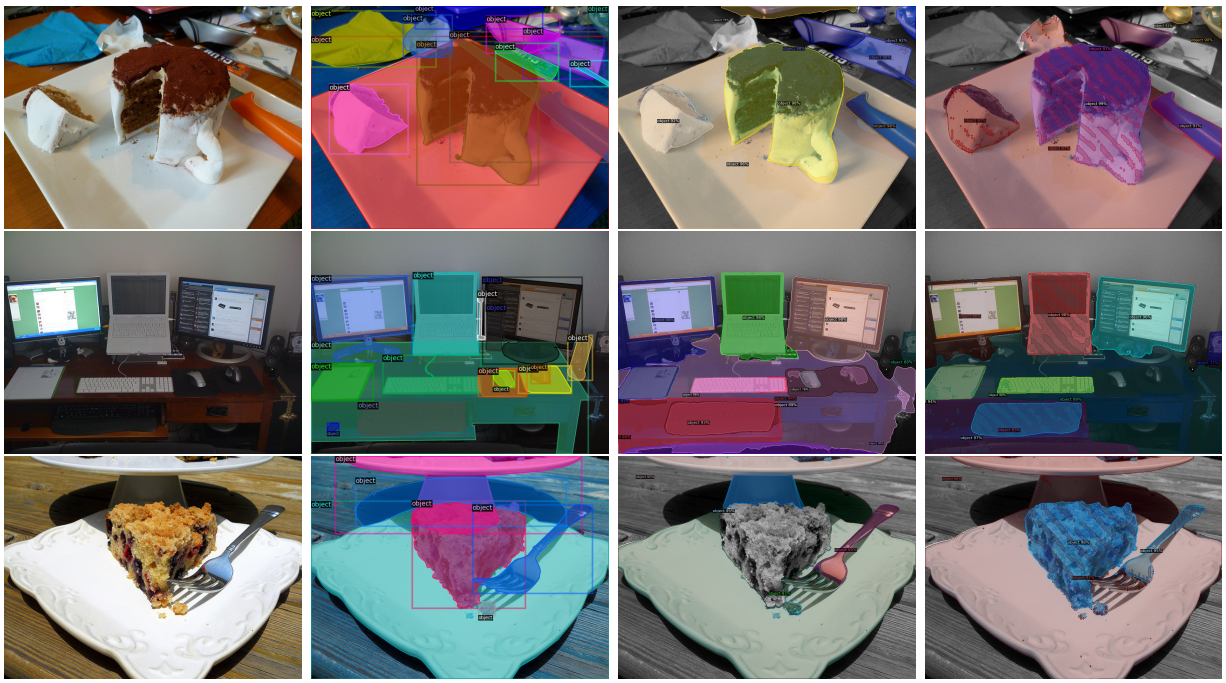


Figure 6.9: Results from *COCOA*. From left to right: original image, amodal ground truth masks, predicted masks trained with only L_{V}, predicted masks trained with both L_{V} and L_{O}.

Chapter 7

Conclusion and Future Work

In Chapter 5, we developed an end-to-end method for joint instance segmentation and occlusion ordering of instances. Our formulation has an advantage over prior work when it comes to how occlusion is modelled: our approach is based on global labeling and is naturally free of occlusion cycles. We outperform prior work on the joint task of instance segmentation and occlusion ordering using the OAIR metric [4]. Our method also shows improvements in standard segmentation tasks after fine-tuning, likely due to the added information from the ground-truth occlusion order, which is a richer supervision. Our model, trained on occlusions, can also learn to a good degree how to order instances by depth from our depth-based OAIR metrics. We provide an application for sharpening depth maps during test-time training using our results, which is detailed in Section 5.3.

As the purpose of our training loss is to enforce a set of simple inequality constraints ($o_i > o_j$ for all $i \gg j$), one possible future direction to improve the efficiency and/or accuracy of the predicted occlusion order is to model these constraints as standard optimization problems and incorporate some well-studied deep learning constraint optimization methods, such as [3] and [22]. Another possible application is generating synthetic occlusion order ground truths: using the predicted global occlusion orders to annotate other existing datasets, which enables other uses for many existing, large-scale datasets without occlusion ordering, including our occlusion-supervised amodal instance segmentation.

In Chapter 6, we developed a new direction for amodal instance segmentation using modal ground truths and pairwise occlusion orders, which does not rely on generating synthetic ground truths. Our method can be applied to existing single-stage (non-RoI-based) segmentation architectures and is agnostic to the choice of shape prior implementation. Extension of our approach to RoI-based architectures, such as Mask R-CNN [42], is a subject of future work. One major obstacle between our approach and RoI-based methods is the bounding box regression for the RoIs. As we do not have access to ground-truth amodal segmentation masks, the amodal bounding boxes, which are larger and inherently different from the visible bounding boxes, are also out of reach for our training. The bounding boxes are important when training using RoI-based methods, as they define where the mask is and where the mask loss backpropagates; without this information, our method cannot be easily applied to RoI-based architectures. Of course, as mentioned in Section 3.4, there are existing methods that resize visible bounding boxes into amodal bounding boxes, such as C2F-Seg [28] as shown in Fig. 3.8; however, they rely on synthetic approaches by artificially overlaying instances on top of each other.

Bibliography

[1] Hassan Abu Alhaija, Siva Karthik Mustikovela, Lars Mescheder, Andreas Geiger, and Carsten Rother. Augmented reality meets computer vision: Efficient data generation for urban driving scenes. *International Journal of Computer Vision*, 126:961–972, 2018.

[2] Shunichi Amari. A theory of adaptive pattern classifiers. *IEEE Transactions on Electronic Computers*, EC-16(3):299–307, 1967.

[3] Brandon Amos and J Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning*, pages 136–145. PMLR, 2017.

[4] Soroosh Baselizadeh. Occlusion-ordered semantic instance segmentation. Master's thesis, University of Waterloo, 2023.

[5] Yuri Boykov and M-P Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in nd images. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 1, pages 105–112, 2001.

[6] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, 2001.

[7] Jasmin Breitenstein and Tim Fingscheidt. Amodal cityscapes: a new dataset, its generation, and an amodal semantic segmentation challenge baseline. In *IEEE Intelligent Vehicles Symposium*, pages 1018–1025, 2022.

[8] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Proceedings of the European Conference on Computer Vision*, pages 213–229. Springer, 2020.

[9] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVR International Conference on Computer Vision*, pages 9650–9660, 2021.

[10] Jia-Ren Chang and Yong-Sheng Chen. Pyramid stereo matching network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5410–5418, 2018.

[11] 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.

[12] Bowen Cheng, Maxwell D Collins, Yukun Zhu, Ting Liu, Thomas S Huang, Hartwig Adam, and Liang-Chieh Chen. Panoptic-deeplab: A simple, strong, and fast baseline for bottom-up panoptic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12475–12485, 2020.

[13] Bowen Cheng, Ishan Misra, Alexander G Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1290–1299, 2022.

[14] Bowen Cheng, Alex Schwing, and Alexander Kirillov. Per-pixel classification is not all you need for semantic segmentation. *Advances in Neural Information Processing Systems*, 34:17864–17875, 2021.

[15] Forrester Cole, Kevin Sanik, Doug DeCarlo, Adam Finkelstein, Thomas Funkhouser, Szymon Rusinkiewicz, and Manish Singh. How well do line drawings depict shape? *ACM Transactions on Graphics*, 28(3):1–9, 2009.

[16] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.

[17] 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.

[18] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, 1989.

[19] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 886–893, 2005.

[20] Nicola De Cao and Thomas Kipf. Molgan: An implicit generative model for small molecular graphs. *arXiv preprint arXiv:1805.11973*, 2018.

[21] Helisa Dhamo, Nassir Navab, and Federico Tombari. Object-driven multi-layer scene decomposition from a single image. In *Proceedings of the IEEE/CVR International Conference on Computer Vision*, pages 5369–5378, 2019.

[22] Priya L Donti and David Rolnick. Dc3: A learning method for optimization with hard constraints. In *International Conference on Learning Representations*, 2021.

[23] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[24] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. *Advances in Neural Information Processing Systems*, 27, 2014.

[25] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88:303–338, 2010.

[26] Patrick Follmann, Rebecca König, Philipp Härtinger, Michael Klostermann, and Tobias Böttger. Learning to see the invisible: End-to-end trainable amodal instance segmentation. In *IEEE Winter Conference on Applications of Computer Vision*, pages 1328–1336, 2019.

[27] Kunihiko Fukushima. Visual feature extraction by a multilayered network of analog threshold elements. *IEEE Transactions on Systems Science and Cybernetics*, 5(4):322–333, 1969.

[28] Jianxiong Gao, Xuelin Qian, Yikai Wang, Tianjun Xiao, Tong He, Zheng Zhang, and Yanwei Fu. Coarse-to-fine amodal segmentation with shape prior. In *Proceedings of the IEEE/CVR International Conference on Computer Vision*, pages 1262–1271, 2023.

[29] Naiyu Gao, Fei He, Jian Jia, Yanhu Shan, Haoyang Zhang, Xin Zhao, and Kaiqi Huang. Panopticdepth: A unified framework for depth-aware panoptic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1632–1642, 2022.

[30] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[31] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361, 2012.

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

[33] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014.

[34] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 270–279, 2017.

[35] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in Neural Information Processing Systems*, 27, 2014.

[36] 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.

[37] 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.

[38] Heiko Hirschmuller. Accurate and efficient stereo processing by semi-global matching and mutual information. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 807–814, 2005.

[39] Heiko Hirschmuller and Daniel Scharstein. Evaluation of cost functions for stereo matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[40] Dorit S Hochbaum and Vikas Singh. An efficient algorithm for co-segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 269–276, 2009.

[41] Yuan-Ting Hu, Hong-Shuo Chen, Kexin Hui, Jia-Bin Huang, and Alexander G Schwing. Sail-vos: Semantic amodal instance level video object segmentation-a synthetic dataset and baselines. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3105–3115, 2019.

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

[43] Yasunori Ishii and Takayoshi Yamashita. Cutdepth: Edge-aware data augmentation in depth estimation. *arXiv preprint arXiv:2107.07684*, 2021.

[44] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1125–1134, 2017.

[45] James T Kajiya and Brian P Von Herzen. Ray tracing volume densities. *ACM SIGGRAPH computer graphics*, 18(3):165–174, 1984.

[46] Lei Ke, Yu-Wing Tai, and Chi-Keung Tang. Deep occlusion-aware instance segmentation with overlapping bilayers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4019–4028, 2021.

[47] Lei Ke, Yu-Wing Tai, and Chi-Keung Tang. Occlusion-aware instance segmentation via bilayer network architectures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(8):10197–10211, 2023.

[48] Doyeon Kim, Woonghyun Ka, Pyungwhan Ahn, Donggyu Joo, Sehwan Chun, and Junmo Kim. Global-local path networks for monocular depth estimation with vertical cutdepth. *arXiv preprint arXiv:2201.07436*, 2022.

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

[50] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, and Piotr Dollár. Panoptic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9404–9413, 2019.

[51] Alexander Kirillov, Yuxin Wu, Kaiming He, and Ross Girshick. Pointrend: Image segmentation as rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9799–9808, 2020.

[52] Alex Krizhevsky. Learning multiple layers of features from tiny images. Master's thesis, University of Toronto, 2009.

[53] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25, 2012.

[54] Harold W Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955.

[55] Matt J Kusner and José Miguel Hernández-Lobato. Gans for sequences of discrete elements with the gumbel-softmax distribution. *arXiv preprint arXiv:1611.04051*, 2016.

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

[57] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[58] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 2002.

[59] Hyunmin Lee and Jaesik Park. Instance-wise occlusion and depth orders in natural scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21210–21221, 2022.

[60] Feng Li, Hao Zhang, Shilong Liu, Jian Guo, Lionel M Ni, and Lei Zhang. Dn-detr: Accelerate detr training by introducing query denoising. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13619–13627, 2022.

[61] Feng Li, Hao Zhang, Huaizhe Xu, Shilong Liu, Lei Zhang, Lionel M Ni, and Heung-Yeung Shum. Mask dino: Towards a unified transformer-based framework for object detection and segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3041–3050, 2023.

[62] Ke Li and Jitendra Malik. Amodal instance segmentation. In *Proceedings of the European Conference on Computer Vision*, pages 677–693. Springer, 2016.

[63] Zhengqin Li and Jiansheng Chen. Superpixel segmentation using linear spectral clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1356–1363, 2015.

[64] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2980–2988, 2017.

[65] 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.

[66] Huan Ling, David Acuna, Karsten Kreis, Seung Wook Kim, and Sanja Fidler. Variational amodal object completion. *Advances in Neural Information Processing Systems*, 33:16246–16257, 2020.

[67] Guilin Liu, Fitsum A Reda, Kevin J Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro. Image inpainting for irregular holes using partial convolutions. In *Proceedings of the European Conference on Computer Vision*, pages 85–100. Springer, 2018.

[68] Shilong Liu, Feng Li, Hao Zhang, Xiao Yang, Xianbiao Qi, Hang Su, Jun Zhu, and Lei Zhang. Dab-detr: Dynamic anchor boxes are better queries for detr. In *International Conference on Learning Representations*, 2022.

[69] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, et al. Swin transformer v2: Scaling up capacity and resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12009–12019, 2022.

[70] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVR International Conference on Computer Vision*, pages 10012–10022, 2021.

[71] 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.

[72] David G Lowe. Object recognition from local scale-invariant features. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157, 1999.

[73] Mateusz Malinowski and Mario Fritz. A multi-world approach to question answering about real-world scenes based on uncertain input. *Advances in Neural Information Processing Systems*, 27, 2014.

[74] Gerard Medioni and Ramakant Nevatia. Segment-based stereo matching. *Computer Vision, Graphics, and Image Processing*, 31(1):2–18, 1985.

[75] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *Proceedings of the European Conference on Computer Vision*, pages 405–421. Springer, 2020.

[76] Shakir Mohamed, Mihaela Rosca, Michael Figurnov, and Andriy Mnih. Monte carlo gradient estimation in machine learning. *Journal of Machine Learning Research*, 21(132):1–62, 2020.

[77] Khoi Nguyen and Sinisa Todorovic. A weakly supervised amodal segmenter with boundary uncertainty estimation. In *Proceedings of the IEEE/CVR International Conference on Computer Vision*, pages 7396–7405, 2021.

[78] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. *Advances in Neural Information Processing Systems*, 29, 2016.

[79] Yuichi Ohta and Takeo Kanade. Stereo by intra-and inter-scanline search using dynamic programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7(2):139–154, 1985.

[80] George Papandreou, Tyler Zhu, Liang-Chieh Chen, Spyros Gidaris, Jonathan Tompson, and Kevin Murphy. Personlab: Person pose estimation and instance segmentation with a bottom-up, part-based, geometric embedding model. In *Proceedings of the European Conference on Computer Vision*, pages 269–286. Springer, 2018.

[81] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2337–2346, 2019.

[82] Lu Qi, Li Jiang, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Amodal instance segmentation with kins dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3014–3023, 2019.

[83] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *International Conference on Learning Representations*, 2022.

[84] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(3):1623–1637, 2020.

[85] N Dinesh Reddy, Robert Tamburo, and Srinivasa G Narasimhan. Walt: Watch and learn 2d amodal representation from time-lapse imagery. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9356–9366, 2022.

[86] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in Neural Information Processing Systems*, 28, 2015.

[87] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 658–666, 2019.

[88] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015.

[89] Frank Rosenblatt. Principles of neurodynamics. *Perceptrons and the Theory of Brain Mechanisms*, 1962.

[90] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. "grabcut" interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics*, 23(3):309–314, 2004.

[91] Carsten Rother, Tom Minka, Andrew Blake, and Vladimir Kolmogorov. Cosegmentation of image pairs by histogram matching-incorporating a global constraint into mrfs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 993–1000, 2006.

[92] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[93] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47:7–42, 2002.

[94] Xiao Song, Xu Zhao, Hanwen Hu, and Liangji Fang. Edgestereo: A context integrated residual pyramid network for stereo matching. In *Asian Conference on Computer Vision, Revised Selected Papers, Part V 14*, pages 20–35. Springer, 2019.

[95] Carole H Sudre, Wenqi Li, Tom Vercauteren, Sebastien Ourselin, and M Jorge Cardoso. Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. In *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, pages 240–248. Springer, 2017.

[96] Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *International Conference on Machine Learning*, pages 9229–9248. PMLR, 2020.

[97] Meng Tang, Abdelaziz Djelouah, Federico Perazzi, Yuri Boykov, and Christopher Schroers. Normalized cut loss for weakly-supervised cnn segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1818–1827, 2018.

[98] Meng Tang, Federico Perazzi, Abdelaziz Djelouah, Ismail Ben Ayed, Christopher Schroers, and Yuri Boykov. On regularized losses for weakly-supervised cnn segmentation. In *Proceedings of the European Conference on Computer Vision*, pages 507–522. Springer, 2018.

[99] Minh Tran, Khoa Vo, Kashu Yamazaki, Arthur Fernandes, Michael Kidd, and Ngan Le. Aisformer: Amodal instance segmentation with transformer. In *British Machine Vision Conference*, page 712, 2022.

[100] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 104:154–171, 2013.

[101] Richard Van Noorden. These are the most-cited research papers of all time, 2025. https://www.nature.com/articles/d41586-025-01124-w.

[102] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

[103] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages I–I, 2001.

[104] Alexander Waibel, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin J Lang. Phoneme recognition using time-delay neural networks. In *Backpropagation*, pages 35–61. Psychology Press, 2013.

[105] Fangyun Wei, Xiao Sun, Hongyang Li, Jingdong Wang, and Stephen Lin. Point-set anchors for object detection, instance segmentation and pose estimation. In *Proceedings of the European Conference on Computer Vision*, pages 527–544. Springer, 2020.

[106] Paul J Werbos. Applications of advances in nonlinear sensitivity analysis. In *System Modeling and Optimization*, pages 762–770. Springer, 1982.

[107] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

[108] Zhenda Xie, Zigang Geng, Jingcheng Hu, Zheng Zhang, Han Hu, and Yue Cao. Revealing the dark secrets of masked image modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14475–14485, 2023.

[109] Dan Xu, Wanli Ouyang, Xiaogang Wang, and Nicu Sebe. Pad-net: Multi-tasks guided prediction-and-distillation network for simultaneous depth estimation and scene parsing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 675–684, 2018.

[110] Xiaosheng Yan, Feigege Wang, Wenxi Liu, Yuanlong Yu, Shengfeng He, and Jia Pan. Visualizing the invisible: Occluded vehicle segmentation and recovery. In *Proceedings of the IEEE/CVR International Conference on Computer Vision*, pages 7618–7627, 2019.

[111] Gengshan Yang, Joshua Manela, Michael Happold, and Deva Ramanan. Hierarchical deep stereo matching on high-resolution images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5515–5524, 2019.

[112] Xiaoding Yuan, Adam Kortylewski, Yihong Sun, and Alan Yuille. Robust instance segmentation through reasoning about multi-object occlusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11141–11150, 2021.

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

[114] Hao Zhang, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel M Ni, and Heung-Yeung Shum. Dino: Detr with improved denoising anchor boxes for end-to-end object detection. In *International Conference on Learning Representations*, 2022.

[115] Qian Zhang, Qiyao Liang, Hong Liang, and Ying Yang. Removal and recovery of the human invisible region. *Symmetry*, 14(3):531, 2022.

[116] Song-Hai Zhang, Ruilong Li, Xin Dong, Paul Rosin, Zixi Cai, Xi Han, Dingcheng Yang, Haozhi Huang, and Shi-Min Hu. Pose2seg: Detection free human instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 889–898, 2019.

[117] Tao Zhang, Shiqing Wei, and Shunping Ji. E2ec: An end-to-end contour-based method for high-quality high-speed instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4443–4452, 2022.

[118] Zhenyu Zhang, Zhen Cui, Chunyan Xu, Zequn Jie, Xiang Li, and Jian Yang. Joint task-recursive learning for semantic segmentation and depth estimation. In *Proceedings of the European Conference on Computer Vision*, pages 235–251. Springer, 2018.

[119] Ziyu Zhang, Alexander G Schwing, Sanja Fidler, and Raquel Urtasun. Monocular object instance segmentation and depth ordering with cnns. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2614–2622, 2015.

[120] Chuanxia Zheng, Duy-Son Dao, Guoxian Song, Tat-Jen Cham, and Jianfei Cai. Visiting the invisible: Layer-by-layer completed scene decomposition. *International Journal of Computer Vision*, 129:3195–3215, 2021.

[121] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2921–2929, 2016.

[122] Qiang Zhou, Shiyin Wang, Yitong Wang, Zilong Huang, and Xinggang Wang. Human de-occlusion: Invisible perception and recovery for humans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3691–3701, 2021.

[123] 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

Proofs

A.1 Proof for Proposition 6.1.1

For all of the following proofs, we suppose instances are ordered from back to front such that for instances $i, j \in \{1, \cdots, N\}$, $j \gg_p i \implies j > i$ at pixel p.

Lemma A.1.1. *For an arbitrary instance i:*

$$M_i(p) \leq A_i(p) \prod_{j>i} (1 - A_j(p)) \tag{A.1}$$

Proof. If $M_i(p) = 0$:

$$M_i(p) = 0 \leq A_i(p) \prod_{j>i} (1 - A_j(p))$$

If $M_i(p) > 0$:

$$j > i \implies i \not\gg_p j \implies M_i(p) \cdot A_j(p) = 0 \implies A_j(p) = 0$$

Hence:

$$\prod_{j>i} (1 - A_j(p)) = 1$$

Thus:

$$M_i(p) \leq A_i(p) = A_i(p) \cdot 1 = A_i(p) \prod_{j>i} (1 - A_j(p))$$

□

Lemma A.1.2. *For an arbitrary instance i:*

$$\sum_{j>i} M_j(p) \leq 1 - \prod_{j>i} (1 - A_j(p)) \tag{A.2}$$

Proof. We prove by induction. Note that the base case is true for $i = N$. Now, suppose the statement is true for some $i = k > 0$. Then:

$$\sum_{j>k} M_j(p) \leq 1 - \prod_{j>k}(1 - A_j(p))$$
$$\sum_{j>k-1} M_j(p) \leq 1 - \prod_{j>k}(1 - A_j(p)) + M_k(p)$$
$$\sum_{j>k-1} M_j(p) \leq 1 - \prod_{j>k}(1 - A_j(p)) + A_k(p)\prod_{j>k}(1 - A_j(p))$$
$$\sum_{j>k-1} M_j(p) \leq 1 - \prod_{j>k}(1 - A_j(p)) + (1 - (1 - A_k(p)))\prod_{j>k}(1 - A_j(p))$$
$$\sum_{j>k-1} M_j(p) \leq 1 - (1 - A_k(p))\prod_{j>k}(1 - A_j(p))$$
$$\sum_{j>k-1} M_j(p) \leq 1 - \prod_{j>k-1}(1 - A_j(p))$$

By induction, the statement is true. □

Lemma A.1.3. *For an arbitrary instance i:*

$$\sum_{j>i} M_j(p) \geq 1 - \prod_{j>i}(1 - A_j(p)) \tag{A.3}$$

Proof. We prove by induction. Note that from Eq. (6.3), the base case is true for $i = N$. Now, suppose the statement is true for some $i = k > 0$. Then if $M_{k+1}(p) = 0$:

$$\sum_{j>k} M_j(p) = 1 - \prod_{j>k}(1 - A_j(p))$$
$$\sum_{j>k+1} M_j(p) = 1 - \prod_{j>k}(1 - A_j(p))$$
$$\sum_{j>k+1} M_j(p) = 1 - (1 - A_{k+1}(p))\prod_{j>k+1}(1 - A_j(p))$$
$$\sum_{j>k+1} M_j(p) = 1 - \prod_{j>k+1}(1 - A_j(p)) + (1 - (1 - A_{k+1}(p)))\prod_{j>k+1}(1 - A_j(p))$$
$$\sum_{j>k+1} M_j(p) = 1 - \prod_{j>k+1}(1 - A_j(p)) + A_{k+1}(p)\prod_{j>k+1}(1 - A_j(p))$$
$$\sum_{j>k+1} M_j(p) \geq 1 - \prod_{j>k+1}(1 - A_j(p))$$

If $M_{k+1}(p) > 0$:

$$j > k+1 \implies k+1 \not\gg_p j \implies M_{k+1}(p) \cdot A_j(p) = 0 \implies A_j(p) = 0, \text{ and } M_j(p) = 0$$

Thus:

$$\sum_{j>k+1} M_j(p) = 0 = 1 - \prod_{j>k+1}(1 - A_j(p))$$

By induction, the statement is true. □

Together with Lemma A.1.2, we have the following for arbitrary i:

$$\sum_{j>i} M_j(p) = 1 - \prod_{j>i}(1 - A_j(p)) \tag{A.4}$$

Now to prove Eq. (6.4) in Proposition 6.1.1:

Proof.

$$\begin{aligned}
M_i(p) &= \sum_{j>i-1} M_j(p) - \sum_{j>i} M_j(p) \\
M_i(p) &= 1 - \prod_{j>i-1}(1 - A_j(p)) - 1 + \prod_{j>i}(1 - A_j(p)) \\
M_i(p) &= (1 - (1 - A_i(p))) \prod_{j>i}(1 - A_j(p)) \\
M_i(p) &= A_i(p) \prod_{j>i}(1 - A_j(p))
\end{aligned}$$

□

A.2 Proof for Proposition 6.1.2

Proof. For all pixels p, suppose the modal masks are binary for all instances i:

$$M_i(p) \in \{0, 1\}$$

Notice that for arbitrary i:

$$\begin{aligned}
\prod_i (1 - A_i(p)) = 1 &\implies A_i(p) = 0 \quad \forall i \\
\prod_i (1 - A_i(p)) < 1 &\implies 1 - \sum_i M_i(p) < 1 \\
&\implies \sum_i M_i(p) > 0 \\
&\implies \sum_i M_i(p) = 1 \\
&\implies \prod_i (1 - A_i(p)) = 0
\end{aligned}$$

Thus:

$$A_i \prod_j (1 - A_j(p)) = 0 \tag{A.5}$$

Also:

$$A_i(p) \geq M_i(p) \implies M_i(p) = A_i(p) M_i(p) \tag{A.6}$$

Then:

$$\begin{aligned} M_i(p) &= A_i(p)M_i(p) \\ &= A_i(p)\prod_j (1 - A_j(p)) + A_i(p)M_i(p) \\ &= A_i(p)\left(1 - \sum_j M_j(p)\right) + A_i(p)M_i(p) \\ &= A_i(p)\left(1 - \sum_{j\neq i} M_j(p)\right) \end{aligned} \tag{A.7}$$

As required. □

A.3 Proof for Proposition 6.1.3

Proof. Continuing from Section A.2, for arbitrary i:

$$\begin{aligned} M_i(p) &= A_i(p)\left(1 - \sum_{j\neq i} M_j(p)\right) \\ &= A_i(p)\left(1 - \sum_{j\gg i} M_j(p) - \sum_{j\not\gg i} M_j(p)\right) \\ &= A_i(p)\left(1 - \sum_{j\gg i} M_j(p)\right) - \left(\sum_{j\not\gg i} A_i(p)M_j(p)\right) \\ &= A_i(p)\left(1 - \sum_{j\gg i} M_j(p)\right) \end{aligned} \tag{A.8}$$

As required. □

Appendix B

Connection Between Eq. (6.4) and Volume Rendering

From volumetric rendering in ray-tracing [45], following the notations from [75], the function for the accumulated transmittance (or attenuation) of a ray up until a given position $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ is:

$$T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds\right) \tag{B.1}$$

Where:

- t_n is the near bound, and
- $\sigma(\mathbf{r}(s))$ represents the volume density at $\mathbf{r}(s)$.

Strong parallels can be drawn between the above and $V_i(p)$ in Eq. (6.4). In our case, since we are not working in the actual 3D space, we can instead consider the "occlusion order"-space, defined by the global occlusion orders o_i from Chapter 5 for instance $i \in \{1, \cdots, N\}$. With this, we can discard the vector $\mathbf{r}(s)$ to use directly s in our "occlusion order"-space and remove the near bound, so we can simplify to:

$$T(t) = \exp\left(-\int_{-\infty}^{t} \sigma(s)ds\right) \tag{B.2}$$

Now consider the following choice of σ at some pixel p:

$$\sigma(t) = \sum_{i=1}^{N} -\ln(1 - A_i(p)) \cdot \delta(t - o_i) \tag{B.3}$$

Since we have no concept of a volumetric density function of instance, we can try to use the Dirac delta function δ instead to denote the "density function" of a segment at each of their o_i's where each instance layer is infinitely thin. To factor in the probabilities $A_i(p)$, we can use the negative log likelihood such that if $A_i(p) = 0$, $-\ln(1 - A_i(p)) = 0$, and if $A_i(p) = 1$, $-\ln(1 - A_i(p)) \to \infty$ which completely blocks out the ray at such infinity density. Turns

out when combining Eq. (B.2) and Eq. (B.3), it yields $V_i(p)$ in Eq. (6.4):

$$
\begin{aligned}
T(t) &= \exp\left(-\int_{-\infty}^{\infty} \sigma(s)ds\right) \\
&= \exp\left(-\int_{-\infty}^{t} \sum_{i=1}^{N} -\ln(1 - A_i(p)) \cdot \delta(s - o_i)ds\right) \\
&= \exp\left(-\sum_{i=1}^{N} \int_{-\infty}^{t} -\ln(1 - A_i(p)) \cdot \delta(s - o_i)ds\right) \\
&= \exp\left(-\sum_{i=1}^{N} -\ln(1 - A_i(p)) \cdot [o_i < t])\right) \\
&= \exp\left(\sum_{i=1}^{N} \ln(1 - A_i(p)) \cdot [o_i < t])\right) \\
&= \prod_{i=1}^{N} \exp(\ln(1 - A_i(p)) \cdot [o_i < t])) \\
&= \prod_{i=1}^{N} (1 - A_i(p))^{[o_i < t]}
\end{aligned}
$$

Suppose the i's are ordered in descending order (near-to-far). If $t = o_j$ for some instance j, we get the attenuation up until instance j, as shown in Eq. (6.4):

$$T(t) = \prod_{i>j}^{N} (1 - A_i(p)) = V_i(p) \tag{B.4}$$

Appendix C

Approximation of Eq. (6.4) for Numerical Stability

Numerical instability can easily creep up in many machine learning settings from the widespread use of logarithmic functions. Perhaps the most notable of such cases would be when taking the natural log of a probability; when the probability approaches 0, the log approaches $-\infty$, which, when not handled correctly, can "NaN" out the gradients, as mentioned in Section 2.1.3. Luckily, as most of these probabilities come from combinations of sigmoid/softmax functions, we can work directly in logit-space and handle such issues explicitly by applying approximations if the magnitudes of the logits become too large (i.e. $\to \pm\infty$). For example, in the case of BCE, most common machine learning libraries provide a numerically stable approach to calculate BCE by working in logit-space (e.g. `torch.nn.BCEWithLogitsLoss`).

In this section, we will detail all the numerical approximations needed to calculate Eq. (6.4) in logit-space to avoid numerical instabilities. Let x_i be the logit of $A_i(p)$ at pixel p (i.e., $A_i(p) = \sigma(x_i)$). Then:

$$\begin{aligned}
\sigma^{-1}\left(A_i(p)\prod_{j>i}(1-A_j(p))\right) &= \sigma^{-1}\left(\sigma(x_i)\prod_{j>i}(1-\sigma(x_j))\right) \\
&= \sigma^{-1}\left((1+e^{-x_i})^{-1}\prod_{j>i}(1+e^{x_j})^{-1}\right) \\
&= \ln\left(\frac{(1+e^{-x_i})^{-1}\prod_{j>i}(1+e^{x_j})^{-1}}{1-(1+e^{-x_i})^{-1}\prod_{j>i}(1+e^{x_j})^{-1}}\right) \\
&= -\ln\left(\left((1+e^{-x_i})\prod_{j>i}(1+e^{x_j})\right)-1\right)
\end{aligned}$$

The above can then be broken down by combining variable approximations as shown in the following sections. The calculated result in logit-space can then be plugged into standard functions such as the BCE approximations as mentioned above.

C.1 Numerical Stability of $\ln(1+e^x)$

For some $x \in \mathbb{R}, x \to \infty$:

$$\begin{aligned}\ln(1+e^x) &= \ln(e^x(e^{-x}+1)) \\ &= x + \ln(1+e^{-x}) \\ &= x + e^{-x} + \mathcal{O}((e^{-x})^2) \\ &\approx x\end{aligned}$$

For some $x \in \mathbb{R}, x \to -\infty$:

$$\begin{aligned}\ln(1+e^x) &= e^x + \mathcal{O}((e^x)^2) \\ &\approx 0\end{aligned}$$

Hence for some thresholds $\alpha_1 < 0 < \beta_1$:

$$\ln(1+e^x) = \begin{cases} 0 & x < \alpha_1 \\ x & x > \beta_1 \\ \ln(1+e^x) & \text{otherwise} \end{cases} \tag{C.1}$$

We will keep in mind that for $x \to -\infty$, $\ln(1+e^x) = e^x + \mathcal{O}((e^x)^2)$ (instead of just 0) as it will be useful later.

C.2 Numerical Stability of $\ln(e^x-1)$

For some $x \in (0,\infty), x \to \infty$:

$$\begin{aligned}\ln(e^x-1) &= \ln(e^x(1-e^{-x})) \\ &= x + \ln(1-e^{-x}) \\ &= x - e^{-x} + \mathcal{O}((-e^{-x})^2) \\ &\approx x\end{aligned}$$

For some $x \in (0,\infty), x \to 0^+$:

$$\begin{aligned}\ln(e^x-1) &= \ln(x + x^2/2 + \mathcal{O}(x^3)) \\ &= \ln(x(1 + x/2 + \mathcal{O}(x^2)) \\ &= \ln x + \ln(1 + x/2 + \mathcal{O}(x^2)) \\ &= \ln x + x/2 + \mathcal{O}(x^2) + \mathcal{O}((x/2 + \mathcal{O}(x^2))^2) \\ &\approx \ln x\end{aligned}$$

Hence for some thresholds $0 < \alpha_2 < \beta_2$:

$$\ln(e^x-1) = \begin{cases} \ln x & x < \alpha_2 \\ x & x > \beta_2 \\ \ln(e^x-1) & \text{otherwise} \end{cases} \tag{C.2}$$

C.3 Numerical Stability of $\ln(\Pi_i(1+e^{x_i})-1)$

For some x_is $\in \mathbb{R}$, if $\forall i, x_i \to -\infty$:

$$\begin{aligned}
\ln\left(\prod_i (1+e_i^x)-1\right) &= \ln\left(e^{\sum_i \ln(1+e_i^x)}-1\right) \\
&= \ln\left(e^{\sum_i (e^{x_i}+\mathcal{O}((e^{x_i})^2))}-1\right) \\
&= \ln\left(\sum_i (e^{x_i}+\mathcal{O}((e^{x_i})^2))\right) \\
&= t + \ln\left(\sum_i (e^{x_i-t}+\mathcal{O}((e^{x_i})^2))\right) \\
&\approx t + \ln\left(\sum_i (e^{x_i-t})\right)
\end{aligned}$$

Where:

$$t = \max_i(x_i)$$

For some x_is $\in \mathbb{R}$, if $\exists k, x_k \to \infty$:

$$\begin{aligned}
\ln\left(\prod_i (1+e_i^x)-1\right) &= \ln\left(e^{\sum_i \ln(1+e_i^x)}-1\right) \\
&\approx \sum_i \ln(1+e_i^x)
\end{aligned}$$

Hence using some thresholds $0 < \alpha_3 < \beta_3$:

$$\ln\left(\prod_i (1+e_i^x)-1\right) = \begin{cases} t + \ln\left(\sum_i (e^{x_i-t})\right) & \sum_i \ln(1+e_i^x) < \alpha_3 \\ \sum_i \ln(1+e_i^x) & \sum_i \ln(1+e_i^x) > \beta_3 \\ \ln\left(e^{\sum_i \ln(1+e_i^x)}-1\right) & \text{otherwise} \end{cases} \tag{C.3}$$

Where $\sum_i \ln(1+e_i^x)$ can first be calculated using Eq. (C.1), with the last case of Eq. (C.3) calculated with Eq. (C.2).