UNIVERSITY OF CALIFORNIA SAN DIEGO

Generative Modeling with Introspective Networks, Amodally-informed Vision Algorithms, and Regression-based Instance Segmentation

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Computer Science

by

Justin Lazarow

Committee in charge:

Professor Zhuowen Tu, Chair
Professor Hao Su, Co-Chair
Professor Virginia de Sa
Professor David Kriegman
Professor Lawrence Saul

2022

The dissertation of Justin Lazarow is approved, and it is acceptable in quality and form for publication on microfilm and electronically.

University of California San Diego

2022

DEDICATION

To my family and their unrelenting support

EPIGRAPH

We must know — we will know

—David Hilbert

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I am deeply grateful to my advisor, Professor Zhuowen Tu, for taking a leap of faith on myself as a Master's student. After almost 6 years of mentorship, there is no doubt to the amount of knowledge and tools he has bestowed upon me alongside priceless connections to others within academia and industry. Furthermore, I thank my committee as a whole: co-advisor Professor Hao Su, Professor David Kriegman, Professor Lawrence Saul, and Professor Virginia de Sa for their time and support.

The mIPC lab throughout the years (with many alumni): Saining Xie, Long Jin, Zeyu Chen, Kwonjoon Lee, Weijian Xu, Yifan Xu, Kunyu Shi have been tremendous mentors, colleagues, and friends during my time at UC San Diego. I am eternally grateful for their support and comradery throughout my studies.

I have been fortunate enough to intern at many great companies: Amazon, Google, and Facebook, each which has shaped some aspect of my advancement as a researcher. In particular, I would like to thank Xinlei Chen and Saining Xie for the tremendous amount of support they gave me and for believing in my abilities to approach hard problems.

Finally, my family and friends have and continue to be a stronghold of support and motivation throughout both the good and difficult times one inevitably faces during a PhD.

Chapter 2, in full, is a reprint of the material as it appears in the Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2017. (Justin Lazarow, Long Jin, Zhuowen Tu, "Introspective neural networks for generative modeling"). The dissertation author was the co-primary investigator and author of this paper.

Chapter 3, in full, is a reprint of the material as it appears in the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. (CVPR), 2020. (Justin Lazarow, Kwonjoon Lee, Kunyu Shi, Zhuowen Tu, "Learning instance occlusion for panoptic segmentation"). The dissertation author was the co-primary investigator and author of this paper.

Chapter 4, in full, has been submitted for publication of the material by Justin Lazarow

and Zhuowen Tu. This dissertation author was the primary investigator and author of this paper.

Chapter 5, in full, has been submitted for publication of the material by Justin Lazarow, Weijian Xu, Zhuowen Tu. This dissertation author was the primary investigator and author of this paper.

VITA

2013	Bachelor of Science in Computer Science, University of Texas at Austin
2013	Bachelor of Science in Mathematics, University of Texas at Austin
2017	Master of Science, University of California San Diego
2022	Doctor of Philosophy, University of California San Diego

PUBLICATIONS

Justin Lazarow*, Long Jin*, Zhuowen Tu, "Introspective neural networks for generative modeling", *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017. (* indicates equal contribution)

Long Jin, Justin Lazarow, "Introspective classification with convolutional nets", *Advances in Neural Information Processing Systems*, 2017.

Justin Lazarow*, Kwonjoon* Lee*, Kunyu Shi*, "Learning instance occlusion for panoptic segmentation", *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. (* indicates equal contribution)

Jeng-Hau Lin, Justin Lazarow, Yunfan Yang, Dezhi Hong, Rajesh Gupta, Zhuowen Tu "Local binary pattern networks", *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2020.

Kamal Gupta, Justin Lazarow, Alessandro Achille, Larry Davis, Vijay Mahadevan, Abhinav Shrivastava, "LayoutTransformer: Layout Generation and Completion with Self-attention", *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

ABSTRACT OF THE DISSERTATION

Generative Modeling with Introspective Networks, Amodally-informed Vision Algorithms, and Regression-based Instance Segmentation

by

Justin Lazarow

Doctor of Philosophy in Computer Science

University of California San Diego, 2022

Professor Zhuowen Tu, Chair
Professor Hao Su, Co-Chair

Visual perception is a fundamental task of computer vision. Subtasks within perception can be decomposed in two types: reasoning about the generative process of images or phenomena themselves (i.e., a prior on the sensory input we expect to perceive) as well as discriminating high-level structural information contained within them. In this dissertation, we first explore how this decomposition is not as decoupled as it might appear. In particular, we show how the act of discrimination is sufficient to produce a model of generative ability. Furthermore, while vision algorithms are required at minimum to discriminate about visible aspects of a scene captured,

it is often useful to reason about *what cannot be seen* within a 2D observation. We explore a practical method for eliciting so-called 2.1D information on two popular, large-scale datasets and demonstrate how this generative information can improve certain scene-level segmentation tasks. Next, we explicitly build a model of scene layout which relies on an amodal understanding (both what can and cannot be seen) of so-called "stuff" (background classes) to accurately place "things" (objects) in a scene. Finally, we revisit the de facto method of instance segmentation within the modern age of computer vision — binary mask prediction — and question whether other approaches (*i.e.*, boundary-based regression) can be suitable alternatives. We validate for the first time that continuous, boundary-based regression can match mask-based prediction with respect to a variety of notions of parity. We believe this opens up further research into the segmentation algorithms of the future.

Chapter 1

Introduction

Humans have a remarkable set of abilities to perceive and act upon the visual stimuli that surround them. The almost effortless capacity for the brain to perceive and explain the visual world serves as the principal target for the field of computer vision — to endow computers with similar potential. However, the sheer magnitude of this goal is daunting. While animals have had millions of years to fine-tune their perception of the visual world, computers have hardly existed in their modern form for half a century. Fortunately, a resurgence in efforts to apply machine learning techniques to solve computer vision tasks has led to a revolution in research and progress. The usage of machine learning allows us to divide these tasks into two main paradigms: generative models and discriminative models. Generative models attempt to explain the observation themselves (*e.g.*, image formation) whereas discriminative ones often try to derive structural information (*e.g.*recognition) from observations. However, both have found renewed interest in recent time.

We focus on applying both generative and discriminative ideas with respect to 2D images in this thesis. We depend on the strong advances from deep learning techniques to revisit approaches and methodology.

1.1 Generative Modeling

Machine learning begins with observations of some phenomena through a random variable X. Within unsupervised learning, estimating a probability distribution $p(x = X)$ (implicitly or explicitly) from these observations is desirable for many reasons. For computer vision and graphics, where X usually corresponds to some subspace of natural images, the ability to sample faithfully from this distribution allows for production of novel examples and provides the ability to impose priors on inputs to vision algorithms. While this seemingly simple task is quite difficult, it is also rich in history.

Early models focused on modeling simple textures and often used primitives and deformations as part of a general image pattern theory [Gre93] describing the generation process of an image. Significant work relied on MiniMax entropy theory and Markov random fields [ZWM97, ZWM98] to select a filter bank which can capture characteristic features of a given texture. By using Gibbs sampling from the resulting maximum entropy probability distribution, new examples of textures can be synthesized. Additionally, more general algorithms (often trained with maximum likelihood estimation) like wake-sleep [HDFN95] , fields of experts [RB05], and deep belief nets [HOT06] are all generative models which can be applied to the domain of images. These models are often hindered by an inability to directly maximize the log-likelihood criterion of the data for sufficiently complex distributions.

However, the resurgence of convolutional neural networks and deep learning has led to significant progress within generative models. Variational autoencoders (VAEs) [KW13, RMW14] extend the task of auto-encoding and rely on variational techniques to allow a neural network to predict parameters of a Gaussian which governs an approximate posterior over the bottleneck. After sampling from the predicted Gaussian, an additional feedforward process (the decoder) realizes samples from the bottleneck in the observation (*e.g.*, image) space and altogether, a *lower bound* of the log-likelihood of the data is maximized.

In contrast, generative adversarial nets (GANs) [GPAM$^+$14] along with numerous variants [ACB17, MLX$^+$17] define two adversarial components: a generator network and a discriminator network which are optimized according to a minimax protocol where the generator attempts to "fool" the discriminator by eventually producing convincing samples from the underlying probability distribution. The generator allows for an implicit model of the distribution with a feedforward explicit function defining the sampler. GANs generally display superior image fidelity [BDS18, KLA19] compared to VAEs but without additional techniques [SGZ$^+$16] can often be hard to train and less interpretable.

Flow-based models [DKB14, RM15] attempt to transform a standard distribution (*e.g.*, a Gaussian) directly to the observation space using a sequence of invertible functions. This allows for tractable log-likelihood computation and thus has a traditional maximum-likelihood training criterion. The exact (invertible) transformations govern the ability to model effectively and thus much work has gone into defining what these should be. Despite a simple training regime, flow-based models of images can produce very high quality results [KD18].

While most of the aforementioned generative models ultimately estimate a function that produces samples given random noise and a feedforward neural network, a large amount of recent work [JLT17a, LJT17, LXFT18, GWJ$^+$19, DM19, SE19] (including some presented in this thesis) relies on designing an iterative process from which samples can be drawn. Often, these methods rely on estimation of some form of the *gradient* of the underlying distribution coupled with the remarkable ability of Stochastic Gradient Langevin Dynamics [WT11] to reliably follow this distribution from areas of low density (i.e. noise) to modes of high density. Furthermore, rather than learning or modeling this gradient, an iterative "denoising" update rule can be learned directly by training a neural network on sequences of noise perturbations made to examples from the data distribution. These processes are now able to challenge and outperform [DN21] the standard models of the past e.g. GANs and VAEs and often have well-defined theoretical properties.

1.2 Image Segmentation

Image segmentation [MFTM01] is a natural extension of coarse localization methods like object detection. Rather than being concerned with predicting boxes which only roughly bound the extent of objects, segmentation aims to produce a *dense* understanding of the occupancy of an object or class within an image. Convolutional neural networks (CNNs) [LBD+89a] have ushered in enormous progress in image classification [KSH12] and similarly within segmentation tasks [LSD15, HGDG17] which rely on both effective localization and classification. *What a segmentation represents* can be more succinctly described based on the depth of understanding desired: semantic, instance, and panoptic segmentations are common paradigms in which to segment an image with varying levels of structural outputs. We note that we (and the literature) will often reference segmentations as being synonymous with predicting pixel-wise masks denoting membership, however, we are careful to establish that masks are simply *one* manner (albeit immensely popular) in which to denote occupancy in an image.

Semantic segmentation requires only a class-wise labeling of an image. In the context of masks, this means that each pixel must be labeled with a class and no instance-wise distinction for is required. For countable objects (*e.g.*persons), therefore, there is no need to denote that two person-labeled pixels may correspond to two distinct people. This allows the semantic segmentation task to be considered as a per-pixel classification task. The most successful architectures for semantic segmentation implement this as a "fully convolutional" [LSD15] approach which mostly uses convolutional and de-convolutional operations in a respective encoder and decoder component to output a pixel grid having size proportional to the input image. Within this grid, each cell predicts a probability distribution over classes from which a predicted label can be derived. Recent work [CPK+18] has shown the need to effectively model spatial context. Dilated operations [YK16] are often employed to accomplish this at an acceptable computational cost. In recent work, Transformers [VSP+17, CMS+20, XWY+21] have been employed as an

effective alternative to convolutional operations that inherently model long range depedencies.

Instance segmentation removes the requirement to label "background" classes (e.g. grass) and other non-countable phenomena. However, it introduces the need to differentiate between distinct instance within the same class (*e.g.*, Person 1 versus Person 2). Historically, this required significantly different architectures (R-CNN) and departures from semantic segmentation to achieve the best performance. Grouping approaches [LJFU17, AT17], which aim to produce an instance-level understanding from a semantic segmentation were quickly surpassed by object-centric, region-based models. The R-CNN [Gir15, RHGS15] family of object detection and eventual instance segmentation [HGDG17] approaches are often considered the de facto baselines. They constitute a well-engineered sequence of tasks: foreground-background classification, foreground classification and regression, and finally mask prediction on "regions of interest" which altogether serve as a robust model of instance segmentation. *Most* competitive models [CV19, CWHL20, KWHG20, QCY21] today are still heavily derived from this initial approach. Only recently have region-free approaches [TSC20, CSK21] to instance segmentation matched or exceeded the performance of Mask R-CNN [HGDG17].

Lastly, we remark that while this is not a definition *per se*, instance segmentation predictions are judged *independently* of one another in recent benchmarks [LMB$^+$14]. This means that both the ground-truth and predictions can be annotated and predicted in a manner such that two pixels might be occupied by the same instance. This appears to have been due to individual annotator bias and might be attributable to the ease at which humans perform amodal completion. There is some evidence that removing such overlap [LPY$^+$19] can reduce overall performance. Recently, this ambiguity has been resolved by the introduction of *panoptic segmentation* which requires combining both semantic segmentations of background objects and instance-wise segmentations of foreground objects within a single image output.

Panoptic segmentation [TCYZ05, KHG$^+$19] hinges on a dichotomy of categories/classes into both "things" and "stuff". Thing classes usually correspond to foreground objects with a

countable nature. Stuff classes are associated with background classes (road, grass, etc) that are usually amorphous in extent. Given this organization, one is tasked with assigning each and every pixel in an image to either a thing or stuff class. Thing classes must be additionally distinguished into distinct instances. Overall, this can be seen as a combination of semantic segmentation (of stuff) and an instance segmentation of things. Critically, since panoptic segmentation requires a single image output, we no longer have the ambiguity that might be present within instance segmentation. Thus, determining unique ownership of a pixel is a requisite task.

Approaches to panoptic segmentation initially replicated the combination of semantic and instance segmentation [KGHD19] by merging the outputs of a strong semantic segmentation model and a strong instance segmentation model. A greedy algorithm [KHG+19] can be constructed to assign pixels to instances by their classification confidence values (denoting uncertainty) and subsequently merge the "stuff" from the semantic segmentation to the remaining unclaimed pixels. As we discuss later in this disseration, this approach is not optimal and often results in "missing" objects in the final panoptic output that were successfully predicted by the instance segmentation component. However, recent work has sometimes removed the need for this heuristic-driven merging process. Recently, fully convolutional approaches [TSC20, LZQ+21] allow for predicting dynamic kernels (functional parameters) which unify the prediction of masks for both things and stuff. Furthermore, end-to-end Transformer-based approaches [CMS+20] appear to provide significantly improved performance — possibly due to the ability to better model global context within an image.

1.3 2D Scene Layout

Unlike the tasks of detection and segmentation which aim to predict structural elements from images, e.g. dense segmentations or object detections, 2D scene layout is a generative task which aims to *produce* structural information in the form of a layout, *i.e.*a set of box and class

pairs describing where objects plausibly should be within an image. Practically, this is realized by producing a generative model (e.g. a VAE [JDH+19], GAN [LYH+19], Autoregressive [GLA+21]) over the annotations (*i.e.*, without the corresponding image) of common object detection datasets [LMB+14, COR+16a]. However, predicting where objects are is inherently ill-posed without further conditioning. In particular, it is necessary to understand where the background classes are before one can have any sort of confidence in placing the foreground. A common approach is to rely on semantic segmentation (or panoptic) annotations to derive *bounding boxes* for background classes or segments. Bounding boxes are used since they provide generally amodal (but coarse) cues of where the background classes are since dense amodal segmentations of the background classes are not generally available.

An accurate model of scene layout can be useful. For instance, in the area of scene generation, an interpretable component would ideally sample a plausible scene layout before continuing to realize shapes within it. Altogether, this resulting panoptic segmentation can then be synthesized [PLWZ19] into an RGB image directly. Furthermore, depending on the underlying properties of the generative model, one could use this model as a prior on structural components like object detection systems *e.g.*, to detect unlikely layouts [LYH+19, GLA+21]. While we focus on 2D layouts here, similar ideas can be extended to 3D [GLA+21].

1.4 Overview of This Dissertation

In this dissertation, we study a myriad of instances of both generative and discrmina-tive tasks. These tasks range from generative models of images, panoptic segmentation with 2.1D [NM90] reasoning, scene layout using amodal information, and finally revisit instance segmentation as a regression problem rather than classification.

In Chapter 2, we consider generative models of images. Rather than relying on traditional methods or adversarial (having two components) methods, we show the ability to discriminate

between true observations of natural images and false observations is sufficient to produce a generative model of the natural images. This discriminative ability in the form of a sequence of deep classifiers provides direction to a gradient-based sampling algorithm to produce high quality, novel samples starting from random noise.

In Chapter 3, we consider the modal nature of vision alongside the amodal nature of perception. Within 2D images, we often observe the 3D world only partially (modally) since one entity might occlude or obscure another. Nonetheless, this does not change the *amodal* properties of the object which could be recovered by removing any occluders from the scene. While full recovery of the amodal nature of an object is difficult, it is often useful and possible the *relative depth ordering*. This form of 2.1D information becomes applicable to the task of panoptic segmentation. When two foreground instances might claim to own a subset of the same pixels, the relative depth ordering can be consulted to resolve these conflicts. We explore how to recover approximate ground-truth supervision of this relative depth ordering and build a classifier to learn instance occlusion within the panoptic segmentation task.

In Chapter 4, we consider similar ideas in the context of the generative task of *laying out* objects in a scene. To effectively produce layouts, understanding affinity between objects and the background extent of a scene can be informative. This is intuitive since in order to know where pedestrians and cars might be in an image, it is necessary to know the where the road and sidewalks are. Representing *where* the road and sidewalks are usually comes in the form of a coarse bounding box. However, we show it is a dense representations that provides more accurate conditioning and better overall performance. Producing a dense representation of the background requires high degrees of amodal knowledge of the world — roads are almost exclusively obscured by the cars that drive upon them, while sidewalks are crowded by people as they walk upon them. Therefore, for both the panoptic and scene layout tasks, we require amodal knowledge of the world. We find that this knowledge can often be derived "for free" from common datasets and work to understand how best to incorporate it into each respective task.

In Chapter 5, we revisit the task of instance segmentation. In the deep learning era, this has almost entirely been synonymous with directly predicting dense binary mask representations. However, this representation is inherently discrete and relies on *classification* to estimate the likelihood of a pixel belonging to an instance. However, we argue for continuous regression-based models of instance segmentation that sparsely predict points along the boundary of a polygon in a fully differentiable manner. While this poses numerous challenges, we present a model and training paradigm that requires no additional supervision than mask-based counterparts and performs equally as well as established baselines. We hope that this can spurn new directions in regression-based segmentation and provide a framework for future applications.

Chapter 2

Introspective Neural Networks for Generative Modeling

2.1 Introduction

Supervised learning techniques have made a substantial impact on tasks that can be formulated as a classification/regression problem [Vap95, FS97, Bre01, LBD+89b]. Unsupervised learning, where no task-specific labeling/feedback is provided on top of the input data, still remains one of the most difficult problems in machine learning but holds a bright future since a large number of tasks have little to no supervision.

Popular unsupervised learning methods include mixture models [DHS00], principal component analysis (PCA) [Jol02], spectral clustering [SM00], topic modeling [BNJ03], and autoencoders [BL07, Bal12]. In a nutshell, unsupervised learning techniques are mostly guided by the minimum description length principle (MDL) [Ris78] to best reconstruct the data whereas supervised learning methods are primarily driven by minimizing error metrics to best fit the input labeling. Unsupervised learning models are often generative and supervised classifiers are often discriminative; generative model learning has been traditionally considered to be a much harder task than discriminative learning [FHT01] due to its intrinsic learning complexity, as well as many assumptions and simplifications made about the underlying models.

Generative and discriminative models have traditionally been considered distinct and complementary to each other. In the past, connections have been built to combine the two families [FHT01, LJ08, TND+08, Jeb12]. In the presence of supervised information with a large amount of data, a discriminative classifier [KSH12] exhibits superior capability in making robust classification by learning rich and informative representations; unsupervised generative models do not require supervision but at a price of relying on assumptions that are often too ideal in dealing with problems of real-world complexity. Attempts have previously been made to learn generative models directly using discriminative classifiers for density estimation [WZH02] and image modeling [Tu07]. There is also a wave of recent development in generative adversarial networks (GAN) [GPAM+14, RMC15, SGZ+16, ACB17] in which a discriminator helps a

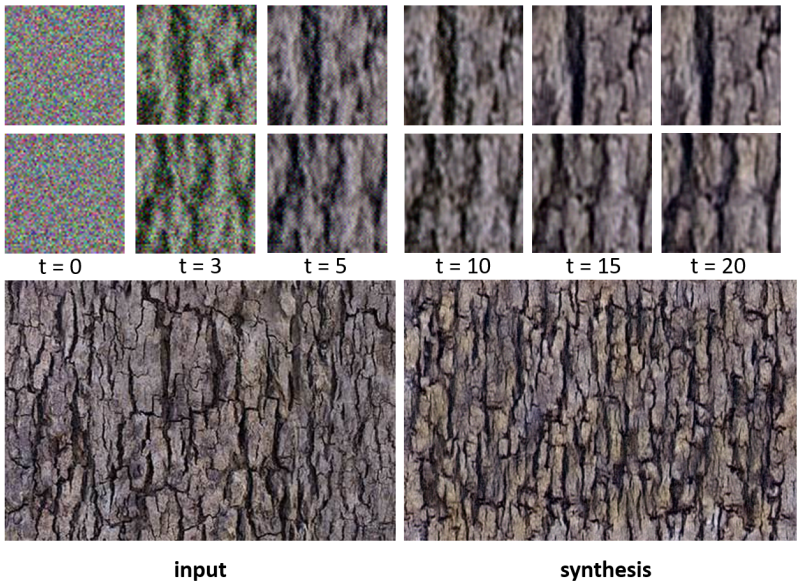


Figure 2.1: The first row shows the development of two 64×64 pseudo-negative samples (patches) over the course of the training process on the "tree bark" texture at selected stages. We can see the initial "scaffold" created and then refined by the networks in later stages. The input "tree bark" texture and a synthesized image by our INNg algorithm are shown in the second row. This texture was synthesized by INNg using 20 CNN classifiers each with 4 layers.

generator try not to be fooled by "fake" samples. We will discuss in detail the relations and connections of our model with these existing literature in the later sections.

In [WZH02], a self supervised boosting algorithm was proposed to train a boosting algorithm by sequentially adding features as weak classifiers on additionally self-generated negative samples. Furthermore, the generative discriminative modeling work (GDL) in [Tu07] generalizes the concept that a generative model can be successfully modeled by learning a sequence of discriminative classifiers via self-generated pseudo-negatives.

Inspired by the prior work on generative modeling [ZWM97, WZH02, Tu07] and development of convolutional neural networks [LBD$^+$89b, KSH12, GEB15], we develop an image modeling algorithm, introspective neural networks for generative modeling (INNg) that can be used simultaneously as a generator and a discriminator, consisting of two critical stages during training: (1) pseudo-negative sampling (synthesis) — a generation of samples considered to be positive examples and (2) a CNN classifier learning stage (classification) for self-evaluation and

model updating from the previous synthesis. There are a number of interesting properties about INNg worth highlighting:

- **CNN classifier as generator**: No special conditions on the CNN architecture are needed in INNg and existing CNN classifiers can be directly made into generators, if trained properly.
- **End-to-end self-evaluation and learning**: Perform end-to-end "introspective learning" to self-classify between synthesized samples (pseudo-negatives) and the training data, to approach the target distribution.
- **All backpropagation**: Our synthesis-by-classification algorithm performs efficient training using backpropagation in both stages: the sampling stage for the input images and the classification training stage for the CNN parameters.
- **Model-based anysize-image-generation**: Since we model the input image, we can train on images of a given size and generate an image of a larger size while maintaining coherence of the entire image.
- **Agnostic to various vision applications**: Due to its intrinsic modeling power being at the same time generative and discriminative, INNg can be adopted to many applications in computer vision. In addition to the applications shown here, extension of the objective (loss) function within INNg is expected to work for other tasks such as "image-to-image translation" [IZZE17].

2.2 Significance and related work

Our introspective neural networks generative modeling (INNg) algorithm has connections to many existing approaches including the MinMax entropy work for texture modeling [ZWM97], and the self-supervised boosting algorithm [WZH02]. It builds on top of convolutional neural networks [LBD$^+$89b] and we are particularly inspired by two lines of prior algorithms: the generative modeling via discriminative approach method (GDL) [Tu07], and the DeepDream code [MOT15] and the neural artistic style work [GEB15]. Parallels can be drawn to ideas elaborated in [GH10] where parameters of a distribution are learned using a (single) classifier between noise and training data. Additionally, the use of "negative" examples to bridge the gap

between an unsupervised task into a supervised one is also seen in [HOWT06], although this focuses on the training of the *weights* of the network for classification rather than synthesis. The work of [MOT15, GEB15], along with the Hybrid Monte Carlo literature [WT11], motivates us to significantly improve the time-consuming sampling process in [Tu07] by an efficient stochastic gradient descent (SGD) process via backpropagation (the reason for us to say "all backpropagation"). Next, we review some existing generative image modeling work, followed by detailed discussions about GDL [Tu07]; comparisons to generative adversarial networks (GAN) [GPAM$^+$14] will be provided in Section 2.3.7.

The history of generative modeling on image or non-image domains is extremely rich, including the general image pattern theory [Gre93], deformable models [YHC92], inducing features [DPDPL97], wake-sleep [HDFN95], the MiniMax entropy theory [ZWM97], the field of experts [RB05], Bayesian models [YK06], and deep belief nets [HOT06]. Each of these pioneering works points to some promising direction in unsupervised generative modeling. However the modeling power of these existing frameworks is still somewhat limited in computational and/or representational aspects. In addition, not too many of them sufficiently explore the power of discriminative modeling. Recent works that adopt convolutional neural networks for generative modeling [XLZW16b] either use CNNs as a feature extractor or create separate paths [XLZW16a, ULVL16]. The neural artistic transferring work [GEB15] has demonstrated impressive results on the image transferring and texture synthesis tasks but it is focused [GEB15] on a careful study of channels attributed to artistic texture patterns, instead of aiming to build a generic image modeling framework. The self-supervised boosting work [WZH02] sequentially learns weak classifiers under boosting [FS97] for density estimation, but its modeling power was not adequately demonstrated.

Relationship with GDL [Tu07]

The generative via discriminative learning framework (GDL) [Tu07] learns a generator through a sequence of boosting classifiers [FS97] using repeatedly self-generated samples, called **pseudo-negatives**. Our INNg algorithm takes inspiration from GDL, but we also observe a number of limitations in GDL that will be overcome by INNg: GDL uses manually specified feature types (histograms and Haar filters), which are fairly limited; the sampling process in GDL, based on Markov chain Monte Carlo (MCMC), is a big computational bottleneck. Additional differences between GDL and INNg include: (1) the adoption of convolutional networks in INNg results in a significant boost to feature learning. (2) introducing SGD based sampling schemes to the synthesis process in INNg makes a fundamental improvement to the sampling process in GDL that is otherwise slow and impractical. (3) two compromises to the algorithm, namely INNg-single (see Fig. 2.4) and INNg-compressed, are additionally proposed to maintain a single classifier or subset of classifiers, respectively.

Introspective Discriminative Networks [JLT17b]

In the sister paper [JLT17b], the formulation is extended to focus on the discriminative aspect — improvement of existing classifiers. Additional key differences are: **a)** the model in [JLT17b] is usually composed of a *single* classifier with a new formulation for training a softmax multi-class classification and **b)** it is less concerned with human perceivable quality of its syntheses and instead focuses on their impact within the classification task.

2.3 Method

We describe below the introspective neural networks generative modeling (INNg) algorithm. We discuss the main formulation first, which bears some level of similarity to GDL [Tu07] with the replacement of the boosting algorithm [FS97] by convolutional neural networks

[LBD+89b]. As a result, INNg demonstrates significant improvement over GDL in terms of both modeling and computational power. Whereas GDL relies on manually crafted features, the use of CNNs within INNg provides for automatic feature learning and tuning when backpropagating on the network parameters as well as an increase in computational power. Both are motivated by a formulation from the Bayes theory.

2.3.1 Motivation

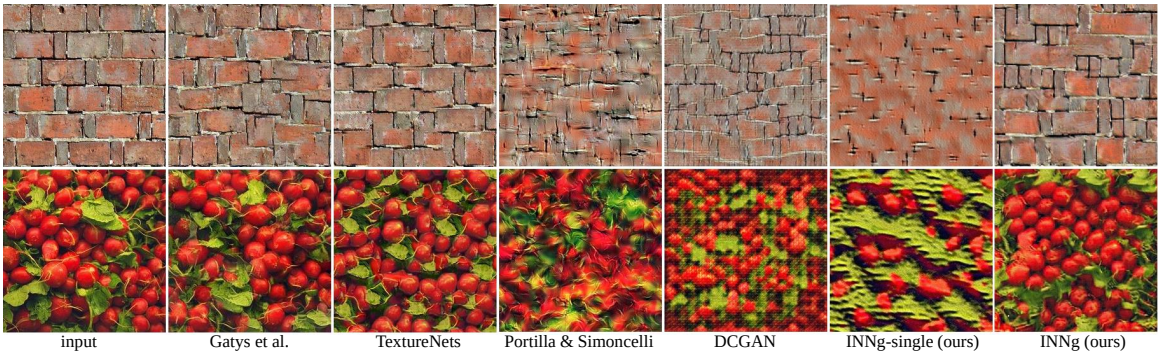


Figure 2.2: Texture synthesis algorithm comparison. Gatys et al. [GEB15], Texture Nets [ULVL16], Portilla & Simoncelli [PS00], and DCGAN [RMC15] results are from [ULVL16].

We start the discussion by borrowing notation from [Tu07]. Suppose we are given a set of training images (patches): $S = \{\mathbf{x}_i \mid i = 1..n\}$ where we assume each $x_i \in \mathcal{R}^m$ e.g. $m = 64 \times 64$ for 64×64 patches. These will constitute **positive** examples of the patterns/targets we wish to model. To introduce the supervised formulation of studying these patterns, we introduce class labels $y \in \{-1, +1\}$ to indicate negative and positive examples, respectively. With this, a generative model computes $p(y, \mathbf{x}) = p(\mathbf{x}|y)p(y)$, which captures the underlying generation process of $\mathbf{x}$ for class y. A discriminative classifier instead computes $p(y|\mathbf{x})$. Under Bayes rule, similar to [Tu07]:

$$p(\mathbf{x}|y=+1) = \frac{p(y=+1|\mathbf{x})p(y=-1)}{p(y=-1|\mathbf{x})p(y=+1)} p(\mathbf{x}|y=-1), \tag{2.1}$$

which can be further simplified when assuming equal priors $p(y=+1)=p(y=-1)$:

$$p(\mathbf{x}|y=+1)=\frac{p(y=+1|\mathbf{x})}{1-p(y=+1|\mathbf{x})}p(\mathbf{x}|y=-1). \tag{2.2}$$

Based on Eq. (2.2), a generative model for the positive samples (patterns of interest) $p(\mathbf{x}|y=+1)$ can be fully represented by a generative model for the negatives $p(\mathbf{x}|y=-1)$ and a discriminative classifier $p(y=+1|\mathbf{x})$, if both $p(\mathbf{x}|y=-1)$ and $p(y=+1|\mathbf{x})$ can be accurately obtained/learned. However, this seemingly intriguing property is circular. To faithfully learn the positive patterns $p(\mathbf{x}|y=+1)$, we need to have a representative $p(\mathbf{x}|y=-1)$, which is equally difficult, if not more. For clarity, we now use $p^-(\mathbf{x})$ to represent $p(\mathbf{x}|y=-1)$. In the GDL algorithm [Tu07], a solution was given to learning $p(\mathbf{x}|y=+1)$ by using an iterative process starting from an initial reference distribution of the negatives $p_0^-(\mathbf{x})$, e.g. a Gaussian distribution $U(\mathbf{x})$ on the entire space of $\mathbf{x}\in\mathcal{R}^m$:

$$p_0^-(\mathbf{x})=U(\mathbf{x}),$$

$$p_t^-(\mathbf{x})=\frac{1}{Z_t}\frac{q_t(y=+1|\mathbf{x})}{q_t(y=-1|\mathbf{x})}\cdot p_{t-1}^-(\mathbf{x}),\ t=1..T \tag{2.3}$$

where $Z_t=\int\frac{q_t(y=+1|\mathbf{x})}{q_t(y=-1|\mathbf{x})}p_{t-1}^-(\mathbf{x})d\mathbf{x}$. Our hope is to gradually learn $p_t^-(\mathbf{x})$ by following this iterative process of Eq. 2.3:

$$p_t^-(\mathbf{x})\overset{t=\infty}{\rightarrow}p(\mathbf{x}|y=+1), \tag{2.4}$$

such that the samples drawn $\mathbf{x}\sim p_t^-(\mathbf{x})$ become indistinguishable from the given training samples. The samples drawn from $\mathbf{x}\sim p_t^-(\mathbf{x})$ are called **pseudo-negatives**, following a definition in [Tu07] to indicate examples considered by the current iteration of the *model* to be positives but are, in reality, negative examples. Next, we present the practical realization of ideas from Eq. 2.3, namely INNg (consisting of a sequence of CNN classifiers composed to produce the process seen in Fig. 2.3) and, additionally, the extreme case of INNg-single that maintains a sequence

consisting of single CNN classifier as seen in Fig. 2.4.

2.3.2 INNg Training

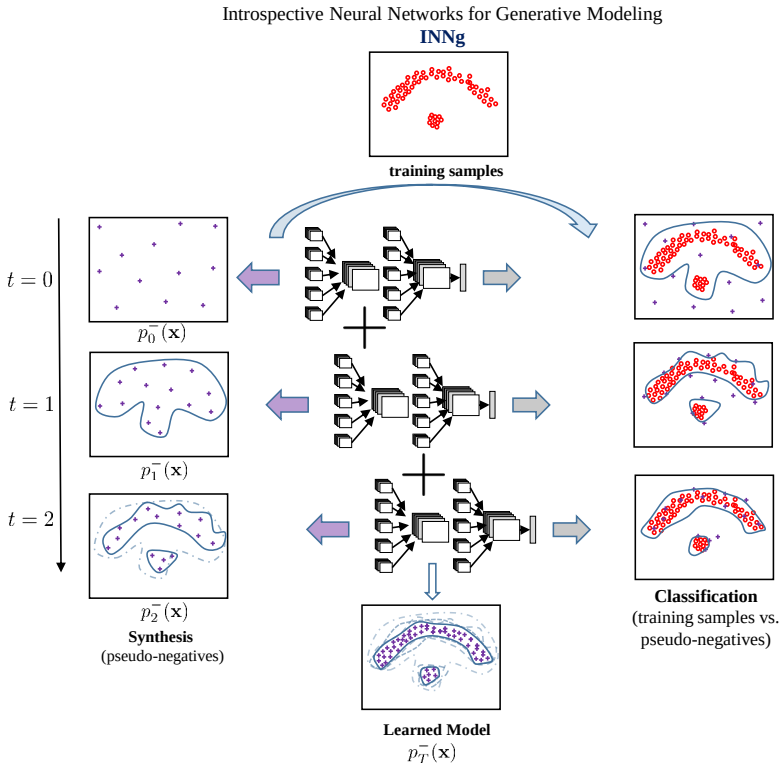


Figure 2.3: Schematic illustration of the pipeline of INNg. The top figure shows the input training samples shown in red circles. The bottom figure shows the pseudo-negative samples drawn by the learned final model. The left panel displays pseudo-negative samples drawn at each time stamp t. The right panel shows the classification by the CNN on the training samples and pseudo-negatives at each time stamp t.

As defined in the previous section, we are given an unlabeled training set by $S = \{\mathbf{x}_i \mid i = 1..n\}$ as our positive examples. Since pseudo-negatives will be added, we refer to this initial positive set as $S_+ = \{(\mathbf{x}_i, y_i = +1) \mid i = 1..n\}$ within the discriminative formulation. Additionally, we must consider the initial set of negatives bootstrapped from noise (also referred to as the initial

Algorithm 1 Outline of the INNg algorithm.

Input: Given a set of training data $S_+ = \{(\mathbf{x}_i, y_i = +1), i = 1..n\}$ with $\mathbf{x} \in \Re^m$.

Initialization: obtain an initial distribution e.g. Gaussian for the pseudo-negative samples: $p_0^-(\mathbf{x}) = U(\mathbf{x})$. Create $S_-^0 = \{(\mathbf{x}_i, -1), i = 1, ..., l\}$ with $\mathbf{x}_i \sim p_0^-(\mathbf{x})$

For t=1..T

1. Classification-step: Train CNN classifier C^t on $S_+ \cup S_-^{t-1}$, resulting in $q_t(y = +1|\mathbf{x})$.

2. Update the model: $p_t^-(\mathbf{x}) = \frac{1}{Z_t} \frac{q_t(y=+1|\mathbf{x})}{q_t(y=-1|\mathbf{x})} p_{t-1}^-(\mathbf{x})$.

3. Synthesis-step: sample l pseudo-negative samples $\mathbf{x}_i \sim p_t^-(\mathbf{x}), i = 1, ..., l$ from the current model $p_t^-(\mathbf{x})$ using a SGD-based sampling procedure (backpropagation on the input) to obtain $S_-^t = \{(\mathbf{x}_i, -1), i = 1, ..., l\}$.

4. $t \leftarrow t + 1$ and go back to step 1 until convergence (e.g. indistinguishable to the given training samples).
End

pseudo-negative set) denoted:

$$S_-^0 = \{(\mathbf{x}_i, -1) \mid i = 1, ..., l\}$$

where $\mathbf{x}_i \sim p_0^-(\mathbf{x}) = U(\mathbf{x})$ according to a Gaussian distribution. Since each stage of the algorithm will refine this set, we define the working set for stage $t = 1..T$ as

$$S_-^{t-1} = \{(\mathbf{x}_i, -1) \mid i = 1, ..., l\}.$$

to include the pseudo-negative samples (l of them) self-generated by the model after stage t. We then train the model at each stage t to obtain

$$q_t(y = +1|\mathbf{x}), \;\; q_t(y = -1|\mathbf{x}) \tag{2.5}$$

over $S_+ \cup S_-^t$ resulting in the classifier C^t. Note that q is an approximation to the true p due to limited samples drawn from $\Re^m$. At each time t, we then compute an approximation to (elaborated in Section 2.3.2)

$$p_t^-(\mathbf{x}) = \frac{1}{Z_t} \frac{q_t(y=+1|\mathbf{x})}{q_t(y=-1|\mathbf{x})} p_{t-1}^-(\mathbf{x}), \tag{2.6}$$

where $Z_t = \int \frac{q_t(y=+1|\mathbf{x})}{q_t(y=-1|\mathbf{x})} p_{t-1}^-(\mathbf{x}) d\mathbf{x}$. Then, we can draw new samples

$$\mathbf{x}_i \sim p_t^-(\mathbf{x})$$

to produce the stages's pseudo-negative set:

$$S_-^{t+1} = \{(\mathbf{x}_i, -1), i = 1, ..., l\}. \tag{2.7}$$

Algorithm 1 describes the learning process. The pipeline of INNg is shown in Fig. 2.3, which consists of: (1) a synthesis step and (2) a classification step. A sequence of CNN classifiers is progressively learned. With the pseudo-negatives being gradually generated, the classification boundary gets tightened and approaches the target distribution.

Classification-step

The classification-step can be viewed as training a classifier on the training set $S_+ \cup S_-^t$ where $S_+ = \{(\mathbf{x}_i, y_i = +1), i = 1..n\}$. $S_-^t = \{(\mathbf{x}_i, -1), i = 1, ..., l\}$ for $t \geq 1$. In practice, we also keep a subset of pseudo-negatives from earlier stages to increase stability. We use a CNN as our base classifier. When training a classifier C^t on $S_+ \cup S_-^t$, we denote the parameters to be learned in C^t by a high-dimensional vector $\mathsf{W}_t = (\mathbf{w}_t^{(0)}, \mathbf{w}_t^{(1)})$ which might consist of millions of parameters. $\mathbf{w}_t^{(1)}$ denotes the weights on the top layer combining the features $\phi(\mathbf{x}; \mathbf{w}_t^{(0)})$ and $\mathbf{w}_t^{(0)}$ carries all the internal representations. Without loss of generality, we assume a sigmoid function

for the discriminative probability

$$q_t(y|\mathbf{x};\mathsf{W}_t) = 1/(1+\exp\{-y < \mathbf{w}_t^{(1)}, \phi(\mathbf{x};\mathbf{w}_t^{(0)}) >\}). \tag{2.8}$$

Both $\mathbf{w}_t^{(1)}$ and $\mathbf{w}_t^{(0)}$ can be learned by the standard stochastic gradient descent algorithm via backpropagation to minimize a cross-entropy loss with an additional term on the pseudo-negatives:

$$\mathcal{L}(\mathsf{W}_t) = -\sum_{(\mathbf{x}_i,+1)\in S_+}^{i=1..n} \ln q_t(+1|\mathbf{x}_i;\mathsf{W}_t) - \sum_{(\mathbf{x}_i,-1)\in S_-^t}^{i=1..l} \ln q_t(-1|\mathbf{x}_i;\mathsf{W}_t)$$

Synthesis-step

In the classification step, we obtain $q_t(y|\mathbf{x};\mathsf{W}_t)$ which is then used to update $p_t^-(\mathbf{x})$ according to Eq. (2.6):

$$p_t^-(\mathbf{x}) = \prod_{a=1}^{t} \frac{1}{Z_a} \frac{q_a(y=+1|\mathbf{x};\mathsf{W}_a)}{q_a(y=-1|\mathbf{x};\mathsf{W}_a)} p_0^-(\mathbf{x}). \tag{2.9}$$

In the synthesis-step, our goal is to draw fair samples from $p_t^-(\mathbf{x})$. By using SGD-based sampling, we carry out backpropagation with respect to $\mathbf{x}$ (the image space) using the CNN models making up Eq. 2.9. Note that the partition function (normalization) Z_a is a constant that is not dependent on the sample $\mathbf{x}$. Let

$$g_a(\mathbf{x}) = \frac{q_a(y=+1|\mathbf{x};\mathsf{W}_a)}{q_a(y=-1|\mathbf{x};\mathsf{W}_a)} = \exp\{< \mathbf{w}_a^{(1)}, \phi(\mathbf{x};\mathbf{w}_a^{(0)}) >\}, \tag{2.10}$$

and take its ln, which is nicely turned into the logit of $q_a(y=+1|\mathbf{x};\mathsf{W}_a)$

$$\ln g_a(\mathbf{x}) =< \mathbf{w}_a^{(1)}, \phi(\mathbf{x};\mathbf{w}_a^{(0)}) > . \tag{2.11}$$

Starting from an initialization of $\mathbf{x}$, the process allows us to directly increase $\sum_{a=1}^{t} < \mathbf{w}_a^{(1)}, \phi(\mathbf{x}; \mathbf{w}_a^{(0)}) >$ using gradient ascent on $\mathbf{x}$ via backpropagation to obtain fair samples subject to Eq. (2.9). Injecting the noise to the sampler results in a general family of stochastic gradient Langevin dynamics [WT11]: $\Delta\mathbf{x} = \nabla(\sum_{a=1}^{t} \ln g_a(\mathbf{x})) + \eta$ where $\eta \sim N(0, \varepsilon)$ is a Gaussian distribution. In practice, to reduce the time and memory complexity, we initialize $\mathbf{x}$ drawn from $p_{t-1}^{-}(\mathbf{x})$, allowing us to primarily focus on the most recent model q_t with SGD sampling for $\ln g_t(\mathbf{x}) =< \mathbf{w}_t^{(1)}, \phi(\mathbf{x}; \mathbf{w}_t^{(0)}) >$. Therefore, generating pseudo-negative samples does not need a large overhead.

To ensure this follows Langevin dynamics as elaborated in [WT11], Gaussian noise with an annealed variance would be added, however, we did not observe a big difference in the quality of samples in practice. Additionally, recent work in [CFG14, MHB17] also show the connections and equivalence between MCMC and SGD-based sampling schemes, where the sampling bias and variance are worth further studying but pose no particular disadvantages here. We have also recently experimented on using different SGD sampling schemes (eg. early-stopping, long steps, perturbations) but did observe significant differences. This is likely partially compensated for by the inherent stochasticity introduced by the random selection of the image patches (and to what extent due to overlap) during synthesis.

Overall model

The overall INNg model after T stages of training becomes:

$$\begin{aligned} p_T^{-}(\mathbf{x}) &= \frac{1}{Z} \prod_{a=1}^{T} \frac{q_a(y=+1|\mathbf{x}; \mathsf{W}_a)}{q_t(y=-1|\mathbf{x}; \mathsf{W}_a)} p_0^{-}(\mathbf{x}) \\ &= \frac{1}{Z} \prod_{a=1}^{T} \exp\{< \mathbf{w}_a^{(1)}, \phi(\mathbf{x}; \mathbf{w}_a^{(0)}) >\} p_0^{-}(\mathbf{x}), \end{aligned} \tag{2.12}$$

where $Z = \int \prod_{a=1}^{T} \exp\{< \mathbf{w}_a^{(1)}, \phi(\mathbf{x}; \mathbf{w}_a^{(0)}) >\} p_0^-(\mathbf{x}) d\mathbf{x}$. INNg shares a similar cascade aspect with GDL [Tu07] where the convergence of this iterative learning process to the target distribution was shown by the following theorem in [Tu07].

Theorem 1 *$KL[p(\mathbf{x}|y = +1)||p_{t+1}^-(\mathbf{x})] \leq KL[p(\mathbf{x}|y = +1)||p_t^-(\mathbf{x})]$ where KL denotes the Kullback-Leibler divergences, and $p(x|y = +1) \equiv p^+(x)$.*

This implies that after sufficiently many stages of INNg training, the distribution of the positives should be well approximated by the resulting cascade of classifiers modeled in Eq. 2.12.

2.3.3 INNg synthesis

While the previous section describes the training process of an INNg model (which itself uses a synthesis step in it to generate pseudo-negatives for the next stage), we consider the synthesis of a *new* sample from the fully trained model. Since a fully trained model consists of T saved classifiers, we attempt to sample through this sequence in a similar fashion to the training. Starting with some $\mathbf{x} \sim U(\mathbf{x})$, we perform gradient ascent with respect to $\mathbf{x}$ until, based off the current classifier C^t, $\mathbf{x}$ crosses the decision boundary of C^t — to now be considered a positive example. Then, C^{t+1} is loaded and the process is repeated again on the resulting $\mathbf{x}$ until it has passed through all classifiers (through C^T) in a similar manner to produce the sample. Note that we perform early stopping within each stage i.e. it was not seen to be effective to produce a transformation of $\mathbf{x}$ that has near 1.0 probability of being considered a positive, only one that is a narrow margin over this boundary. This additionally allows for a more efficient runtime of the synthesis process.

2.3.4 An alternative: INNg-single

We briefly present the INNg-single algorithm and show the pipeline of INNg-single is in Fig. 2.4. Note that we maintain a single CNN classifier throughout the entire learning process in INNg-single.

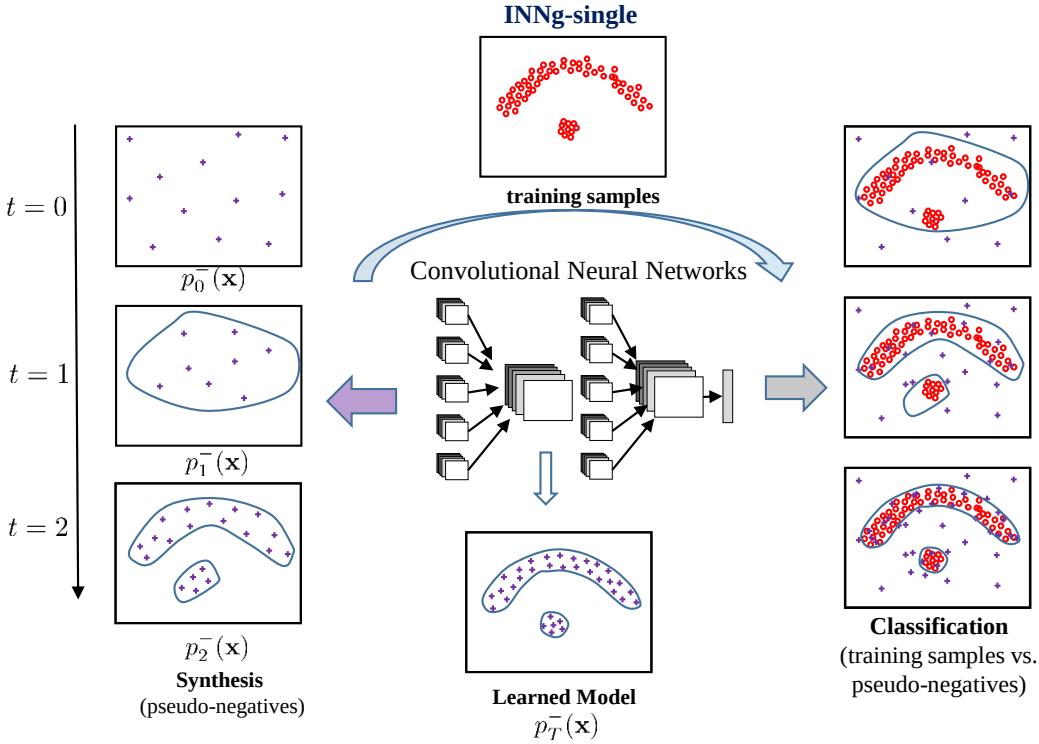


Figure 2.4: Schematic illustration of the pipeline of INNg-single.

In the classification step, we obtain $q_t(y|\mathbf{x};\mathsf{W}_t)$ (similar as Eq. 2.8) which is then used to update $p_t^-(\mathbf{x})$ according to Eq. (2.13):

$$p_t^-(\mathbf{x}) = \frac{1}{Z_t}\frac{q_t(y=+1|\mathbf{x};\mathsf{W}_t)}{q_t(y=-1|\mathbf{x};\mathsf{W}_t)}p_0^-(\mathbf{x}). \tag{2.13}$$

In the synthesis-step, we draw samples from $p_t^-(\mathbf{x})$. The overall INNg-single model after T stages of training becomes:

$$p_T^-(\mathbf{x}) = \frac{1}{Z_T}\exp\{< \mathbf{w}_T^{(1)}, \phi(\mathbf{x};\mathbf{w}_T^{(0)}) >\}p_0^-(\mathbf{x}), \tag{2.14}$$

where $Z_T = \int \exp\{< \mathbf{w}_T^{(1)}, \phi(\mathbf{x};\mathbf{w}_T^{(0)}) >\}p_0^-(\mathbf{x})d\mathbf{x}$.

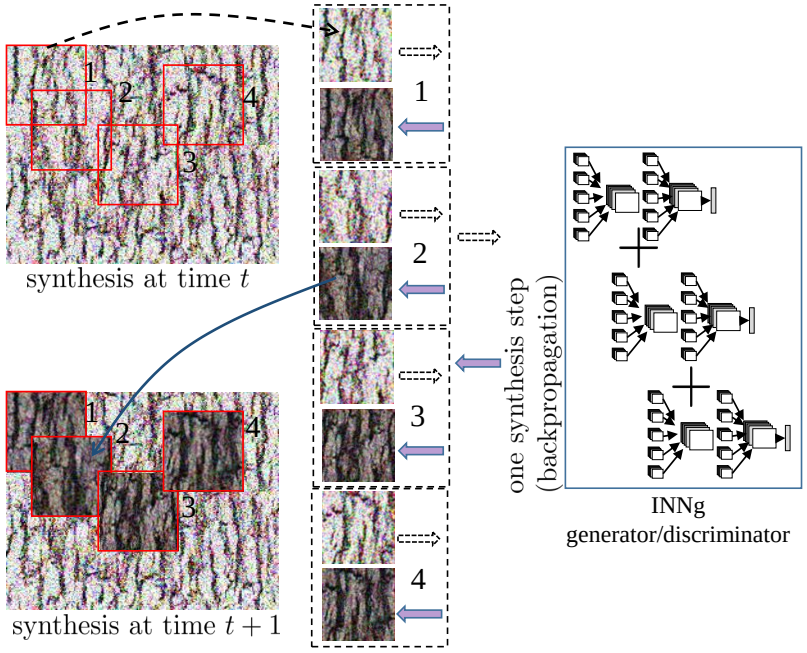


Figure 2.5: Illustration of model-based anysize-image-generation strategy.

2.3.5 Model-based anysize-image-generation

Given a particularly sized image, anysize-image-generation within INNg allows one to generate/synthesize an image much larger than the given one. Patches extracted from the training images are used in the training of the discriminator. However, their position within the training (or pseudo-negative) image is not lost. In particular, when performing synthesis using backpropagation, updates to the pixel values are made by considering the average loss of all patches that overlap a given pixel. Thus, up to stage T, in order to consider the updates to the patch of $\mathbf{x}(i,j)$ centered at position (i,j) for image $\mathbf{I}$ of size $m_1 \times m_2$, we perform backpropagation on the patches to increase the probability:

$$p_T(\mathbf{I}) \propto \prod_{a=1}^{T} \prod_{i=1}^{m_1} \prod_{j=1}^{m_2} g_a(\mathbf{x}(i,j)) p_0^-(\mathbf{x}(i,j)) \tag{2.15}$$

where $g_a(\mathbf{x}(i,j))$ (see Eq. 2.10) denotes the score of the patch of size e.g. 64×64 for $\mathbf{x}(i,j)$ under the discriminator at stage a. Fig. 2.5 gives an illustration for one stage of sampling. This allows

us to synthesize much larger images by being able to enforce the coherence and interactions surrounding a particular pixel. In practice, we add stochasticity and efficiency to the synthesis process by randomly sampling these set of patches.

2.3.6 Model size reduction

As mentioned in Section 2.3.2 and 2.3.3, the process of INNg relies on keeping a snapshot of the classifier after each stage $t = 1 \dots T$. This can pose a problem for both space and time efficiency. However, it may not be all necessary to keep each classifier around. Instead, one can pick some factor b such that the classifier is only saved every b stages. We accomplish this by: i) only saving the model to disk every b stages ii) initializing the sampling process at stage $b \times i + k$ ($i \in \mathbb{N}$ and $k < b$) from those of stage $b \times i$. We still, however, keep the pseudo-negatives around for training purposes to stabilize the process. We do find that later stages within each "mini stage" should be allowed more backpropagation steps during the synthesis step in order to compensate for the more complex optimization landscape. The image seen in Fig. 2.1 was generated from a reduced model ($b = 3$ and $T = 60$), resulting in only 20 classifiers. One can view this as a cascade of multiple of INNg-singles.

2.3.7 Comparison with GAN [GPAM$^+$14]

Next we compare INNg with a very interesting and popular line of work, generative adversarial neural networks (GAN) [GPAM$^+$14]. We summarize the key differences between INNg and GAN. Other recent algorithms [GPAM$^+$14, RMC15, ZML16, BLRW16] share similar properties with it.

- *Unified generator/discriminator vs. separate generator and discriminator.* INNg maintains a single model that is simultaneously a generator and a discriminator. The INNg generator therefore is able to self-evaluate the difference between its generated samples (pseudo-negatives) against the training data. This gives us an integrated framework to achieve competitive results in unsupervised and fully

supervised learning with the generator and discriminator helping each other internally (not externally). GAN instead creates two convolutional networks, a generator and a discriminator.

- *Training*. Due to the internal competition between the generator and the discriminator, GAN is known to be hard to train [ACB17]. INNg instead carries out a straightforward use of backpropagation in both the sampling and the classifier training stage, making the learning process direct. For example, all the textures by INNg shown in the experiments Fig. 2.2 and Fig. 2.6 are obtained under the identical setting without hyper-parameter tuning.
- *Speed*. GAN performs a forward pass to reconstruct an image, which is generally faster than INNg where synthesis is carried out using backpropagation. INNg is still practically feasible since it takes about 10 seconds to synthesize a batch of 50 images of 64×64 and around 30 seconds to synthesize a texture image of size 256×256, excluding the time to load the models.
- *Model size*. Since a sequence of CNN classifiers $(10 - 60)$ are included in INNg, INNg has a much larger model complexity than GAN. This is an advantage of GAN over INNg. Our alternative INNg-single model maintains a single CNN classifier but its generative power is worse than those of INNg and GAN.

2.4 Experiments

We evaluate both INNg and INNg-single. In each method, we adopt the discriminator architecture of [RMC15] which takes an input size of $64 \times 64 \times 3$ in the RGB colorspace by four convolutional layers using 5×5 kernel sizes with the layers using 64, 128, 256 and 512 channels, respectively. We include batch normalization after each convolutional layer (excluding the first) and use leaky ReLU activations with leak slope 0.2. The classification layer flattens the input and finally feeds it into a sigmoid activation. This serves as the discriminator for the 64×64 patches we extract from the training image(s). Note that it is a general purpose architecture with no modifications made for a specific task in mind.

In texture synthesis and artistic style, we make use of the "anysize-image-generation" architecture by adding a "head" to the network that, at each forward pass of the network, randomly

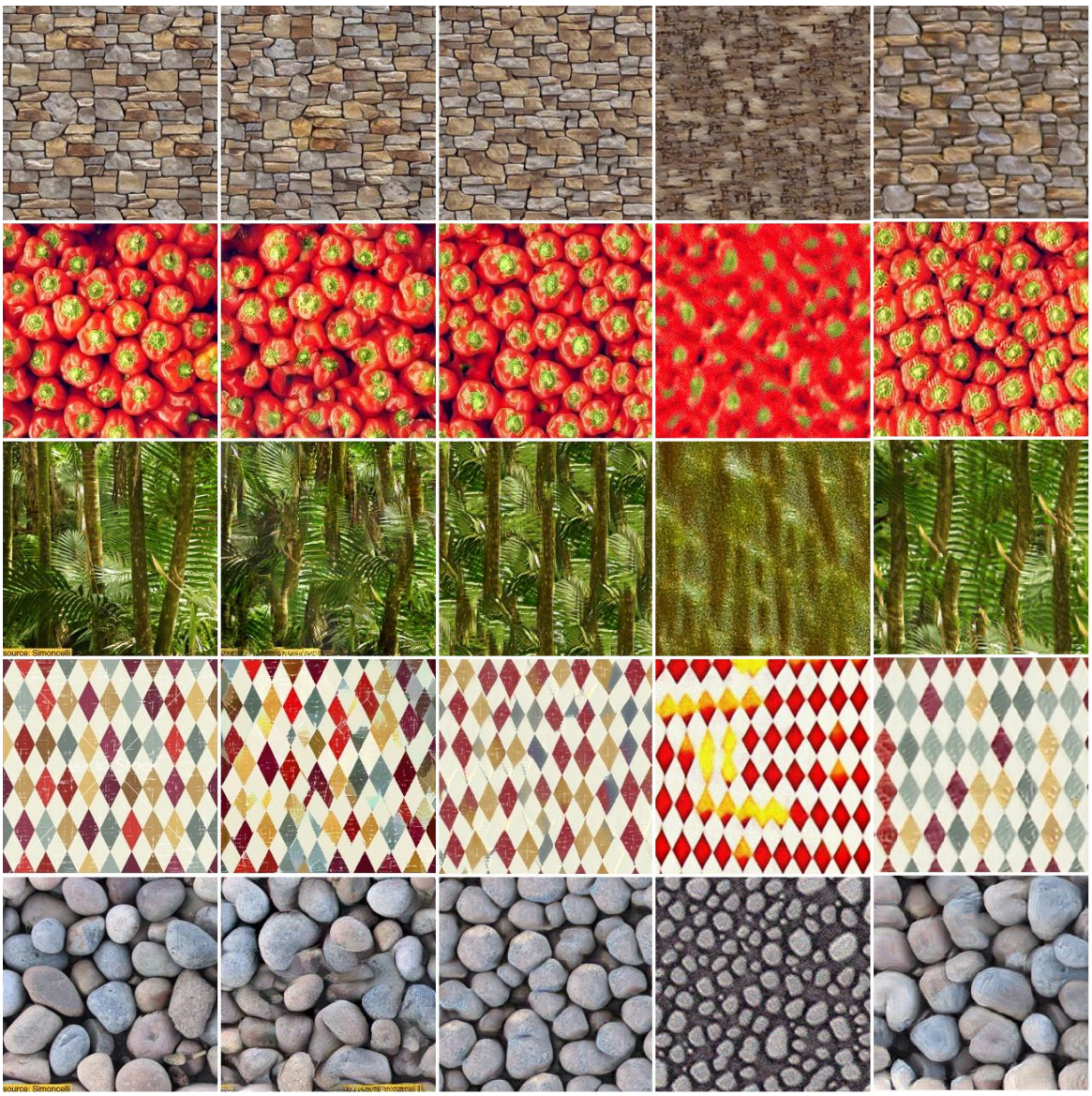


Figure 2.6: More texture synthesis results. Gatys et al. [GEB15] and Texture Nets [ULVL16] results are from [ULVL16].

selects some number (equal to the desired batch size) of 64×64 random patches (possibly overlapping) from the full sized images and passes them to the discriminator. This allows us to retain the whole space of patches within a training image rather than select some subset of them in advance to use during training.

2.4.1 Texture modeling

Texture modeling/rendering is a long standing problem in computer vision and graphics [HB95, ZWM97, EL99, PS00]. Here we are interested in statistical texture modeling [ZWM97, XLZW16a], instead of just texture rendering [EL99]. We train similar textures to [ULVL16]. Each source texture is resized to 256×256, used as the "positive" image in the training set; a set of 200 negative images are initially sampled from a normal distribution with $\sigma = 0.3$ of size 320×320 after adding padding of 32 pixels to each spatial dimension of the image to ensure each pixel of the 256×256 center has equal probability of being extracted in some patch. $1,000$ patches are extracted randomly across the training images and fed to the discriminator at each forward pass of the network (during training and synthesis stages) from a batch size of 100 patches — 50 random positives and negatives when training and 100 pseudo-negatives during synthesis. At each stage, our classifier is finetuned using stochastic gradient descent with learning rate 0.01 from the previous stage's classifier. Pseudo-negatives from more recent stages are chosen in mini-batches with higher probability than those of earlier stages in order to ensure the discriminator learns from its most recent mistakes as well as provide for more efficient training when the set of accumulated negatives has grown large in later stages. During the synthesis stage, pseudo-negatives are synthesized using the previous stage's pseudo-negatives as their initialization. Adam is used with a learning rate of 0.1 and $\beta = 0.5$ and stops early when the average probability of the patches under the discriminator becomes positive (across some window of steps, usually 20). We find this sampling strategy to attain a good balance in effectiveness and efficiency.

New textures are synthesized under INNg by: initializing from the normal distribution with $\sigma = 0.3$ followed by SGD based sampling via backpropagation using the saved networks for each stage, and feeding the resulting synthesis to the next stage. The number of patches is decided based on the image size to be synthesized, typically 10 patches when synthesizing a 256×256 image since this matches the average number of patches extracted per image during training. For INNg-single which consists of a single CNN classifier, SGD-based sampling is performed directly using this CNN classifier to transform the initial normal distribution to a desired texture.

Considering the results in Fig. 2.2, we see that INNg (60 CNN classifiers each with 4 layers) generates images of similar quality to [ULVL16], but of higher quality than those by Gatys et al. [GEB15], Portilla & Simoncelli [PS00], and DCGAN [RMC15]. In general, synthesis by INNg is usually more faithful to the structure of the input images.

More texture modeling results are provided in Fig. 2.6 using the same model and CNN setting as in Fig. 2.2. We make an interesting observation that the "diamond" texture (the fourth row) generated by INNg shows to preserve the near-regular patterns much better than the other methods. In the bottom row of Fig. 2.6, the "pebbles" synthesis of INNg captures the size variation as well as the variation in color and shading, better than TextureNets [ULVL16] and Gatys et al. [GEB15].

2.4.2 Artistic style transfer

We also attempt to transfer artistic style as shown in [GEB15]. However, our architecture makes no use of additional networks for content and texture transferring task uses a loss functions during synthesis to minimize

$$-\ln p(\mathbf{I}_{style} \mid \mathbf{I}) \propto \gamma \cdot ||\mathbf{I}_{style} - \mathbf{I}||_2 - (1-\gamma) \cdot \ln p^{-}_{style}(\mathbf{I}_{style}),$$

Figure 2.7: Artistic style transfer results using the “Starry Night” and “Scream” style on the image from Amsterdam.

where $\mathbf{I}$ is an input image and $\mathbf{I}_{style}$ is its stylized version, and $p^{-}_{style}(\mathbf{I})$ denotes the model learned from the training style image. We include a L_2 fidelity term during synthesis, weighted by a parameter γ, making $\mathbf{I}_{style}$ not too far away from the input image $\mathbf{I}$. We choose $\gamma = 0.3$ and average the L_2 difference between the original content image and the current stylized image at each step of synthesis. Two examples of the artistic style transfer are shown in Fig. 2.7.

2.4.3 Face modeling

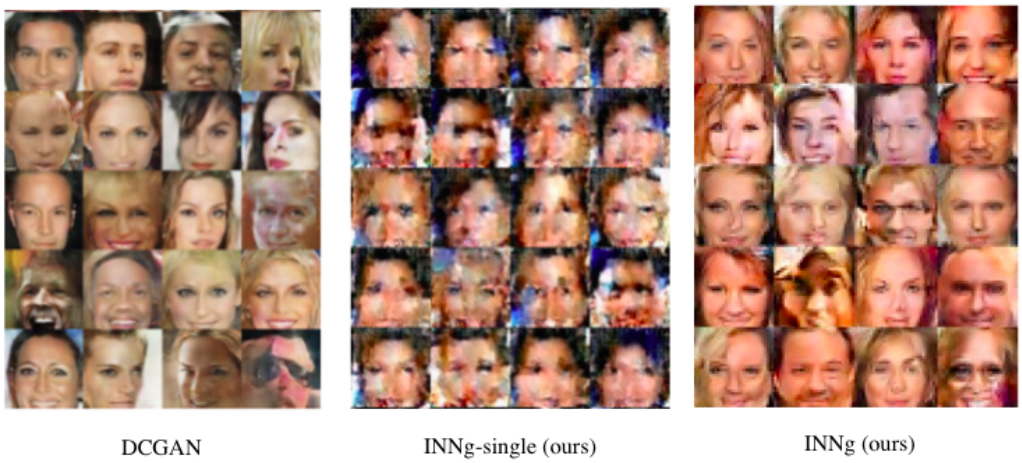


Figure 2.8: Generated images learned on the CelebA dataset. The first, the second, and the third column are respectively results by DCGAN [RMC15] (using tensorflow implementation [Kim16]), INNg-single (1 CNN classifier with 4 layers), and INNg (12 CNN classifiers each with 4 layers).

INNg is also demonstrated on a face modeling task. The CelebA dataset [LLWT15] is used in our face modeling experiment, which consists of $202,599$ face images. We crop the center 64×64 patches in these images as our positive examples. For the classification step, we use stochastic gradient descent with learning rate 0.01 and a batch size of 100 images, which contains 50 random positives and 50 random negatives. For the synthesis step, we use the Adam optimizer with learning rate 0.02 and $\beta = 0.5$ and stop early when the pseudo-negatives cross the decision boundary. In Fig. 2.8, we show some face examples generated by the DCGAN model [RMC15], our INNg-single model (1 CNN classifier with 4 conv layers), and our INNg model (12 CNN

classifiers each with 4 conv layers).

2.4.4 SVHN unsupervised learning

Figure 2.9: Generated images learned on the SVHN dataset. The first, the second, and the third column are respectively results by DCGAN [RMC15] (using tensorflow implementation [Kim16]), INNg-single (1 CNN classifier with 4 layers), and INNg (10 CNN classifiers each with 4 layers).

The SVHN [NWC+11] dataset consists of color images of house numbers collected by Google Street View. The training set consists of $73,257$ images, the extra set consists of $531,131$ images, and the test set has $26,032$ images. The images are of the size 32×32. We combine the training and extra set as our positive examples for unsupervised learning. Following the same settings in the face modeling experiments, we shown some examples generated by the DCGAN model [RMC15], INNg-single (1 CNN classifier with 4 conv layers), and INNg (10 CNN classifiers each with 4 conv layers).

2.4.5 Unsupervised feature learning

We perform the unsupervised feature learning and semi-supervised classification experiment by following the procedure outlined in [RMC15]. We first train a model on the SVHN

training and extra set in an unsupervised way, as in Section 2.4.4. Then, we train an L2-SVM on the learned representations of this model. The features from the last three convolutional layers are concatenated to form a 14336-dimensional feature vector. A $10,000$ example held-out validation set is taken from the training set and is used for model selection. The SVM classifier is trained on 1000 examples taken at random from the remainder of the training set. The test error rate is averaged over 100 different SVMs trained on random 1000-example training sets. Within the same setting, our INNg model achieves the test error rate of 32.81% and the DCGAN model achieves 33.13% (we ran the DCGAN code [Kim16] in an identical setting as INNg for a fair comparison).

2.5 Conclusion

Generative modeling using introspective neural networks points to an encouraging direction for unsupervised image modeling that capitalizes on the power of discriminative deep convolutional neural networks. It can be adopted for a wide range of problems in computer vision.

This chapter is based on material as it appears in the Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2017. (Justin Lazarow, Long Jin, Zhuowen Tu, "Introspective neural networks for generative modeling"). The dissertation author was the co-primary investigator and author of this paper.

Chapter 3

Learning Instance Occlusion for Panoptic Segmentation

3.1 Introduction

Image understanding has been a long standing problem in both human perception [Bie87] and computer vision [Mar82]. The *image parsing* framework [TCYZ05] is concerned with the task of decomposing and segmenting an input image into constituents such as objects (text and faces) and generic regions through the integration of image segmentation, object detection, and object recognition. Scene parsing is similar in spirit and consists of both non-parametric [TNL14] and parametric [ZSQ+17] approaches.

After the initial development, the problem of image understanding was studied separately as object detection (or extended to instance segmentation) and semantic segmentation. Instance segmentation [PCD15, PLCD16, DHS16, LLW+18, HGDG17, RSD+12, ZFU16, JCT16] requires the detection and segmentation of each *thing* (countable object instance) within an image, while semantic segmentation [SWRC06, Tu08, EVGW+10, LSD15, CPK+18, ZJRP+15, ZSQ+17] provides a dense per-pixel classification without distinction between instances within the same *thing* category. Kirillov *et al.*[KHG+19] proposed the panoptic segmentation task that combines the strength of semantic segmentation and instance segmentation. In this task, *each pixel* in an image is assigned either to a background class (*stuff*) or to a specific foreground object (an *instance* of *things*).

A common approach for panoptic segmentation has emerged in a number of works [KGHD19, LCZ+19, XLZ+19] that relies on combining the strong baseline architectures used in semantic segmentation and instance segmentation into either a separate or shared architecture and then *fusing* the results from the semantic segmentation and instance segmentation branches into a single panoptic output. Since there is no expectation of consistency in proposals between semantic and instance segmentation branches, conflicts must be resolved. Furthermore, one must resolve conflicts *within* the instance segmentation branch as it proposes segmentations independent of each other. While a pixel in the panoptic output can only be assigned to a single

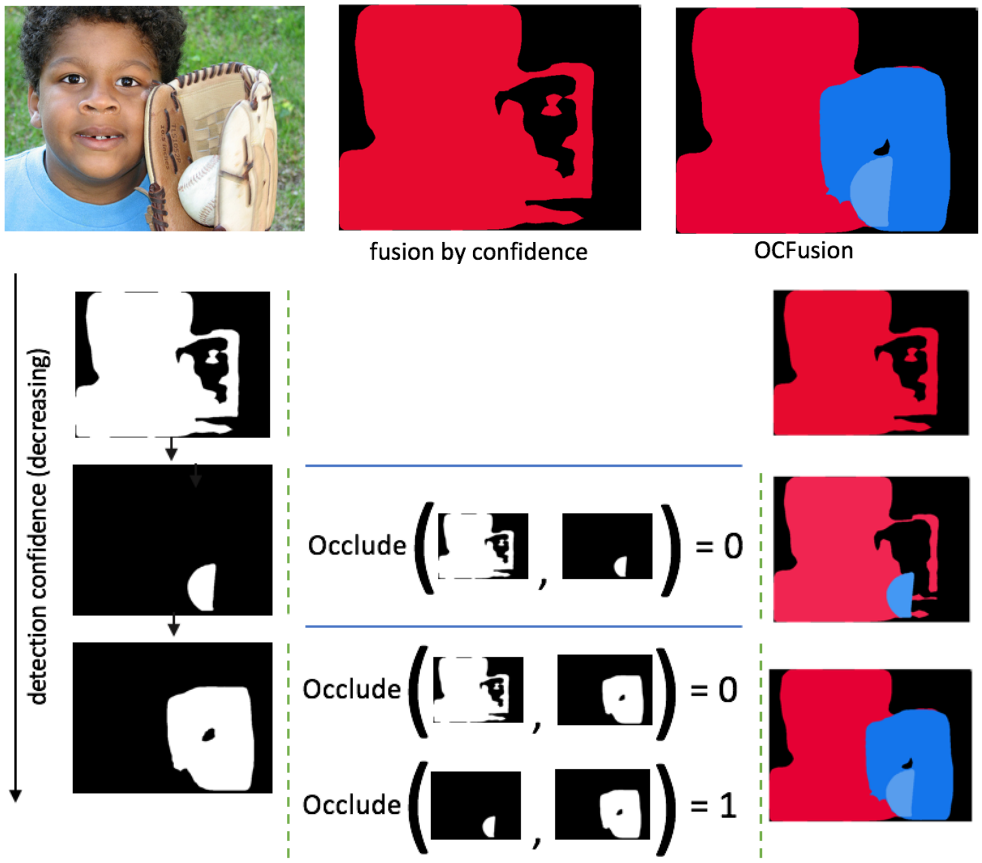


Figure 3.1: **An illustration of fusion using masks sorted by detection confidence alone [KHG+19] vs. with the ability to query for occlusions (OCFusion; ours).** Occlude$(A,B) = 0$ in occlusion head means mask B should be placed on top of mask A. Mask R-CNN proposes three instance masks listed with decreasing confidence. The heuristic of [KHG+19] occludes all subsequent instances after the "person", while our method retains them in the final output by querying the occlusion head.

class and instance, instance segmentation proposals are often overlapping.

To handle these issues, Kirillov *et al.*[KHG+19] proposed a fusion process similar to non-maximum suppression (NMS) that favors instance proposals over semantic proposals. However, we observe that occlusion relationships between different objects do not correlate well with object detection confidences used in this NMS-like fusion procedure [KHG+19], which therefore generally leads to poor performance when an instance that overlaps another (*e.g.*, a tie on a shirt

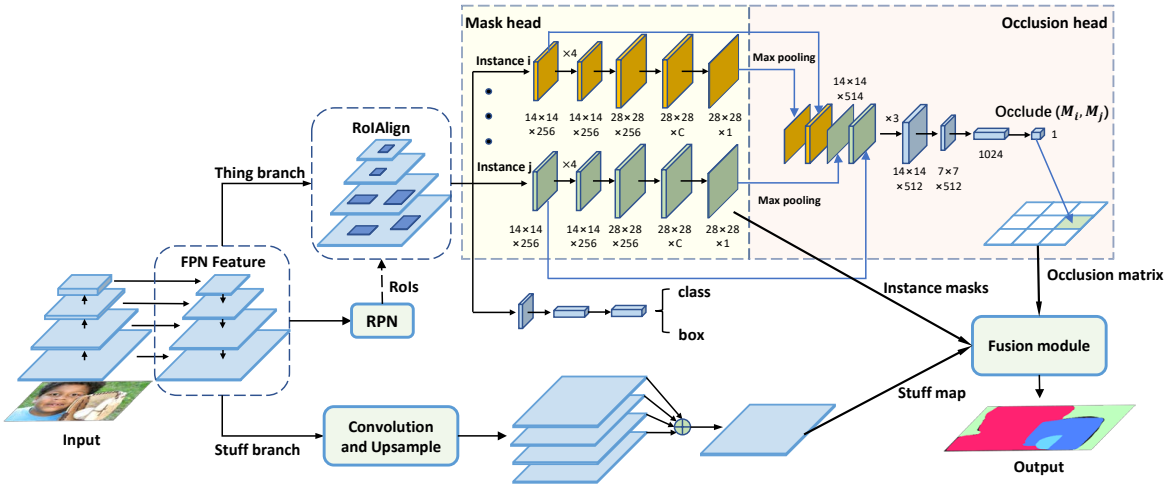


Figure 3.2: **Illustration of the overall architecture.** The FPN is used as a shared backbone for both thing and stuff branches. In thing branch, Mask R-CNN will generate instance mask proposals, and the occlusion head will output binary values $Occlude(M_i, M_j)$ (Equation 3.1) for each pair of mask proposals M_i and M_j with *appreciable* overlap (larger than a threshold) to indicate occlusion relation between them. Occlusion head architecture is described in Section 3.2.4. Fusion process is described in 3.2.3.

in Figure 3.3a) has lower detection confidence than the instance it should occlude. This can cause a large number of instances that Mask R-CNN *successfully* proposes fail to exist in the panoptic prediction (shown in Figure 3.1).

In this work, we focus on enriching the fusion process established by [KHG+19] with a binary relationship between *instances* to determine occlusion ordering. We propose adding an additional branch (occlusion head) to the instance segmentation pipeline tasked with determining which of two instance masks should lie on top of (or below) the other to resolve occlusions in the fusion process. The proposed occlusion head can be fine-tuned easily on top of an existing Panoptic Feature Pyramid Networks (FPNs) [KGHD19] architecture with minimal difficulty. We call our approach fusion with occlusion head (OCFusion). OCFusion brings significant performance gains on the COCO and Cityscapes panoptic segmentation benchmarks with low computational cost.

3.2 Learning Instance Occlusion for Panoptic Fusion

We adopt the coupled approach of [KGHD19] that uses a shared Feature Pyramid Network (FPN) [LDG$^+$17b] backbone with a top-down process for semantic segmentation branch and Mask R-CNN [HGDG17] for instance segmentation branch.

In this section, we first discuss the instance occlusion problem arising within the fusion heuristic introduced in [KHG$^+$19] and then introduce OCFusion method to address the problem. The overall approach is shown in Figure 3.2.

3.2.1 Fusion by confidence

The fusion protocol in [KHG$^+$19] adopts a greedy strategy during inference in an iterative manner. Instance proposals are first sorted in order of decreasing detection confidence. In each iteration, the proposal is skipped if its intersection with the mask of all already assigned pixels is above a certain ratio of τ. Otherwise, pixels in this mask that have yet to be assigned are assigned to the instance in the output. After all instance proposals of some minimum detection threshold are considered, the semantic segmentation is merged into the output by considering its pixels corresponding to each "stuff" class. If the number of pixels exceeds some threshold after removing already assigned pixels, then these pixels are assigned to the corresponding "stuff" category. Pixels that are unassigned after this entire process are considered void predictions and have special treatment in the panoptic scoring process. We denote this type of fusion as *fusion by confidence*.

Softening the greed. The main weakness of the greedy fusion process is the complete reliance on detection confidences (*e.g.*for Mask R-CNN, those from the box classification score) for a tangential task. Detection scores not only have little to do with mask quality (*e.g.*, [HHG$^+$19]), but they also do not incorporate any knowledge of *layout*. If they are used in such a way, higher detection scores would imply a more foreground ordering. Often this is detrimental since Mask

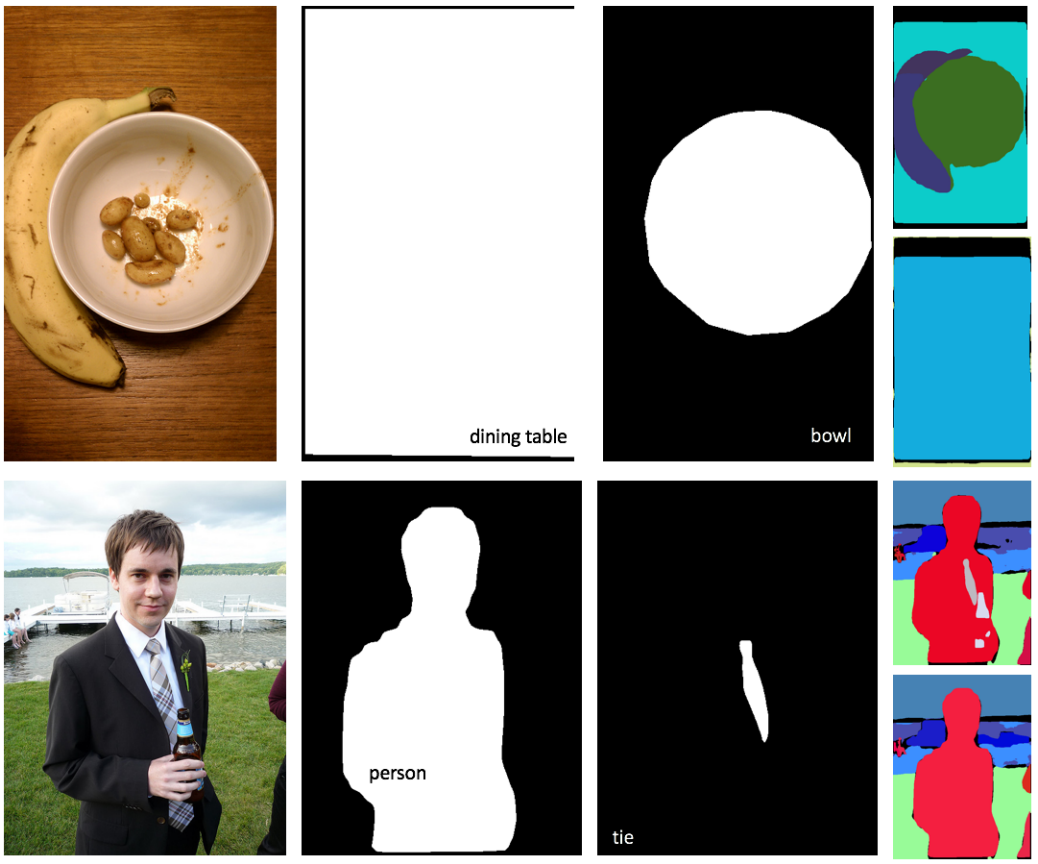
R-CNN exhibits behavior that can assign near-maximum confidence to very large objects (*e.g.*see dining table images in Figure 3.3b) that are both of poor mask quality and not truly foreground. It is common to see images with a significant number of true instances suppressed in the panoptic output by a single instance with large area that was assigned the largest confidence.

Our approach softens this greedy fusion process with an occlusion head that is dedicated to predicting the binary relation between instances with appreciable overlap so that instance occlusions can be properly handled.

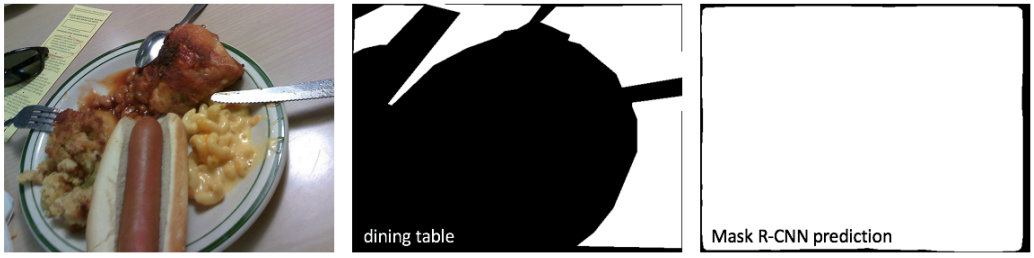
3.2.2 Occlusion head formulation

Consider two masks M_i and M_j proposed by an instance segmentation model, and denote their intersection as $I_{ij} = M_i \cap M_j$. We are interested in the case where one of the masks is heavily occluded by the other. Therefore, we consider their respective intersection ratios $R_i = \text{Area}(I_{ij})/\text{Area}(M_i)$ and $R_j = \text{Area}(I_{ij})/\text{Area}(M_j)$ where $\text{Area}(M)$ denotes the number of "on" pixels in mask M. As noted in Section 3.2.1, the fusion process considers the intersection of the current instance proposal with the mask consisting of all already claimed pixels. Here, we are looking at the intersection between two masks and denote the threshold as ρ. If either $R_i \geq \rho$ or $R_j \geq \rho$, we define these two masks as having appreciable overlap. In this case, we must then decide which instance the pixels in I_{ij} should belong to. We attempt to answer this by learning a binary relation $\text{Occlude}(M_i, M_j)$ such that whenever M_i and M_j have appreciable intersection:

$$\text{Occlude}(M_i, M_j) = \begin{cases} 1 \text{ if } M_i \text{ should be placed on top of } M_j \\ 0 \text{ if } M_j \text{ should be placed on top of } M_i. \end{cases} \tag{3.1}$$



(a)



(b)

Figure 3.3: **Images and ground truth masks from the COCO dataset. (a)** is an example where even predicting the ground truth mask creates ambiguity when attempting to assign pixels to instances in a greedy manner. The **baseline fusion process** [KHG+19] is unable to properly assign these as shown in the **2nd and 4th** images of the rightmost column whereas **our method** is able to handle the occlusion relationship present as shown in the **1st and 3rd** images of the rightmost column. **(b)** is an example where Mask R-CNN baseline produces an instance prediction that occludes the entire image and creates the same ambiguity in (a) despite an unambiguous ground truth annotation.

3.2.3 Fusion with occlusion head

We now describe our modifications to the inference-time fusion heuristic of [KHG+19] that incorporates Occlude(M_i, M_j) in Algorithm 2.

Algorithm 2 Fusion with Occlusion Head.

P is $H \times W$ matrix, initially empty.
ρ is a hyperparameter, the minimum intersection ratio for occlusion.
τ is a hyperparameter.

for each proposed instance mask M_i **do**
  $C_i = M_i - P$ ▷ pixels in M_i that are not assigned in P
  for $j < i$ **do** ▷ each already merged segment
    I_{ij} is the intersection between mask M_i and M_j.
    $R_i = \text{Area}(I_{ij})/\text{Area}(M_i)$.
    $R_j = \text{Area}(I_{ij})/\text{Area}(M_j)$.
    if $R_i \geq \rho$ or $R_j \geq \rho$ **then** ▷ significant intersection
      if Occlude$(M_i, M_j) = 1$ **then**
        $C_i = C_i \bigcup (C_j \bigcap I_{ij})$.
        $C_j = C_j - I_{ij}$.
      end if
    end if
  end for
  if $\text{Area}(C_i)/\text{Area}(M_i) \leq \tau$ **then**
    continue
  else
    assign the pixels in C_i to the panoptic mask P.
  end if
end for

After the instance fusion component has completed, the semantic segmentation is then incorporated as usual, only considering pixels assigned to stuff classes and determining whether the number of unassigned pixels corresponding to the class in the current panoptic output exceeds some threshold, *e.g.*, 4096. The instance fusion process is illustrated in Figure 3.1.

3.2.4 Occlusion head architecture

We implement Occlude(M_i, M_j) as an additional "head" in Mask R-CNN [HGDG17]. Mask R-CNN already contains two heads: a box head that is tasked with taking region proposal

network (RPN) proposals and refining the bounding box as well as assigning classification scores, while the mask head predicts a fixed size binary mask (usually 28×28) for all classes independently from the output of the box head. Each head derives its own set of features from the underlying FPN. We name our additional head, the "occlusion head" and implement it as a binary classifier that takes two (soft) masks M_i and M_j along with their respective FPN features (determined by their respective boxes) as input. The classifier output is interpreted as the value of $\text{Occlude}(M_i, M_j)$.

The architecture of occlusion head is inspired by [HHG+19] as shown in Figure 3.2. For two mask representations M_i and M_j, we apply max pooling to produce a 14×14 representation and concatenate each with the corresponding RoI features to produce the input to the head. Three layers of 3×3 convolutions with 512 feature maps and stride 1 are applied before a final one with stride 2. The features are then flattened before a 1024 dimensional fully connected layer and finally projected to a single logit.

3.2.5 Ground truth occlusion

We use ground truth panoptic mask along with ground truth instance masks to derive ground truth occlusion relation. We pre-compute the intersection between all pairs of masks with appreciable overlap. We then find the pixels corresponding to the intersection of the masks in the panoptic ground truth. We determine the instance occlusion based on which instance owns the majority of pixels in the intersection. We store the resulting "occlusion matrix" for each image in an $N_i \times N_i$ matrix where N_i is the number of instances in the image and the value at position (i, j) is either -1, indicating no occlusion, or encodes the value of $\text{Occlude}(i, j)$.

3.2.6 Occlusion head training

During training, the occlusion head is designed to first find pairs of predicted masks that match to different ground truth instances. Then, the intersection between these pairs of masks is computed, and the ratio of the intersection to mask area taken. A pair of masks is added for consideration when one of these ratios is at least as large as the pre-determined threshold ρ. We then subsample the set of all pairs meeting this criterion to decrease computational cost. It is desirable for the occlusion head to reflect the consistency of Occlude, therefore we also invert all pairs so that $\text{Occlude}(M_i, M_j) = 0 \iff \text{Occlude}(M_j, M_i) = 1$ whenever the pair (M_i, M_j) meets the intersection criteria. This also mitigates class imbalance. Since this is a binary classification problem, the overall loss L_o from the occlusion head is given by the binary cross-entropy over all subsampled pairs of masks that meet the intersection criteria.

3.3 Related work

Next, we discuss in detail the difference between OCFusion and the existing approaches for panoptic segmentation, occlusion ordering, and non-maximum suppression.

Panoptic segmentation. The task of panoptic segmentation is introduced in [KHG+19] along with a baseline where predictions from instance segmentation (Mask R-CNN [HGDG17]) and semantic segmentation (PSPNet [ZSQ+17]) are combined via a heuristics-based fusion strategy. A stronger baseline based on a single Feature Pyramid Network (FPN) [LDG+17b] backbone followed by multi-task heads consisting of semantic and instance segmentation branches is concurrently proposed by [LCZ+19, LRB+18, KGHD19, XLZ+19]. On top of this baseline, attention layers are added in [LCZ+19] to the instance segmentation branch, which are guided by the semantic segmentation branch; a loss term enforcing consistency between things and stuff predictions is then introduced in [LRB+18]; a parameter-free panoptic head which computes the

final panoptic mask by pasting instance mask logits onto semantic segmentation logits is presetned in [XLZ+19]. These works have been making steady progress in panoptic segmentation, but their focus is not to address the problem for explicit reasoning of instance occlusion.

Occlusion ordering and layout learning. Occlusion handling is a long-studied computer vision task [WHY09, EESG10, SLKS05, HSEH07]. In the context of semantic segmentation, occlusion ordering has been adopted in [TNL14, CLY15, ZTMD17]. A repulsion loss is added to a pedestrian detection algorithm [WXJ+18] to deal with the crowd occlusion problem, but it focuses on detection only, without instance segmentation.

In contrast, we study the occlusion ordering problem for instance maps in panoptic segmentation. Closest to our method is the recent work of [LPY+19], which proposes a panoptic head to resolve this issue in a similar manner to [XLZ+19] but instead with a learnable convolution layer. Since our occlusion head can deal with two arbitrary masks, it offers more flexibility over these approaches which attempt to "rerank" the masks in a linear fashion [XLZ+19, LPY+19]. Furthermore, the approach of [LPY+19] is based off how a *class* should be placed on top of *another class* (akin to semantic segmentation) while we explicitly model the occlusion relation between arbitrary *instances*. This allows us to leverage the *intra-class occlusion relations* such as "which of these two persons should occlude the other?", and we show this leads to a gain in Figure 3.7 and Table 3.9. In a nutshell, we tackle the occlusion problem in a scope that is more general than [LPY+19] with noticeable performance advantage, as shown in Table 5.1 and Table 3.3.

Learnable NMS. One can relate resolving occlusions to non-maximum suppression (NMS) that is applied to *boxes*, while our method tries to suppress intersections between masks. Our method acts as a *learnable* version of NMS for instance masks with similar computations to the analogous ideas for boxes such as [HBS17].

3.4 Experiments

3.4.1 Implementation details

We extend the Mask R-CNN benchmark framework [MG18], built on top of PyTorch, to implement our architecture. Batch normalization [IS15] layers are frozen and not fine-tuned for simplicity. We perform experiments on the COCO dataset [LMB+14] [KHG+19] as well as the Cityscapes dataset [COR+16b] with panoptic annotations.

We find the most stable and efficient way to train the occlusion head is by fine-tuning with all other parameters frozen. We add a single additional loss only at fine-tuning time so that the total loss during panoptic training is $L = \lambda_i(L_c + L_b + L_m) + \lambda_s L_s$ where L_c, L_b, and L_m are the box head classification loss, bounding-box regression loss, and mask loss while L_s is the semantic segmentation cross-entropy loss. At fine-tuning time, we only minimize L_o, the classification loss from the occlusion head. We subsample 128 mask occlusions per image.

During fusion, we only consider instance masks with detection confidence of at least 0.5 or 0.6 and reject segments during fusion when their overlap ratio with the existing panoptic mask (after occlusions are resolved) exceeds $\tau = 0.5$ on COCO and $\tau = 0.6$ on Cityscapes. Lastly, when considering the segments of *stuff* generated from the semantic segmentation, we only consider those which have at least 4096 pixels remaining after discarding those already assigned on COCO and 2048 on Cityscapes.

Semantic head. On COCO, repeat the combination of 3×3 convolution and $2\times$ bilinear upsampling until $1/4$ scale is reached, following the design of [KGHD19]. For the model with ResNeXt-101 backbone, we replace each convolution with deformable convolution [DQX+17]. For ResNet-50 backbone, we additionally add one experiment that adopts the design from [XLZ+19] which uses 2 layers of deformable convolution followed by a bilinear upsampling to the $1/4$ scale. On Cityscapes, we adopt the design from [XLZ+19].

COCO. The COCO 2018 panoptic segmentation task consists of 80 *thing* and 53 *stuff*

classes. We use 2017 dataset which has a split of 118k/5k/20k for training, validation and testing respectively.

Cityscapes. Cityscapes consists of 8 *thing* classes and 11 *stuff* classes. We use only *fine* dataset with a split of 2975/500/1525 for training, validation and testing respectively.

COCO training

We train the FPN-based architecture described in [KGHD19] for 90K iterations on 8 GPUs with 1 image per GPU. The base learning rate of 0.02 is reduced by 10 at both 60k and 80k iterations. We then proceed to fine-tune with the occlusion head for 2500 more iterations. We choose $\lambda_i = 1.0$ and $\lambda_s = 0.5$ while for the occlusion head we choose the intersection ratio ρ as 0.2. For models with ResNet-50 and ResNet-101 backbone, we use random horizontal flipping as data augmentation. For model with ResNeXt-101 backbone, we additionally use scale jitter (with scale of shorter image edge equals to $\{640, 680, 720, 760, 800\}$).

Cityscapes training

We randomly rescale each image by 0.5 to 2$\times$ (scale factor sampled from a uniform distribution) and construct each batch of 16 (4 images per GPU) by randomly cropping images of size 512 $\times$ 1024. We train for 65k iterations with a base learning rate of 0.01 with decay at 40k and 55k iterations. We fine-tune the occlusion head for 5000 more iterations. We choose $\lambda_i = \lambda_s = 1.0$ with intersection ratio ρ as 0.1. We do not pretrain on COCO.

Panoptic segmentation metrics

We adopt the panoptic quality (PQ) metric from [KHG+19] to measure panoptic segmentation performance. This single metric captures both segmentation and recognition quality. PQ can be further broken down into scores specific to *things* and *stuff*, denoted PQ^{Th} and PQ^{St}, respectively.

Multi-scale testing

We adopt the same scales as [XLZ+19] for both COCO and Cityscapes multi-scale testing. For the stuff branch, we average the multi-scale semantic logits of semantic head. For the thing branch, we average the multi-scale masks and choose not to do bounding box augmentation for simplicity.

Table 3.1: **Comparison to our implementation of Panoptic FPN [KGHD19] baseline model on the MS-COCO *val* dataset**.

Method	Backbone	PQ	PQ^{Th}	PQ^{St}
Baseline	ResNet-50	39.5	46.5	29.0
OCFusion	ResNet-50	41.3	49.4	29.0
relative improvement		+1.8	+3.0	+0.0
Baseline	ResNet-101	41.0	47.9	30.7
OCFusion	ResNet-101	43.0	51.1	30.7
relative improvement		+2.0	+3.2	+0.0

3.4.2 COCO panoptic benchmark

We obtain state-of-the-art results on COCO Panoptic Segmentation validation set with and without multi-scale testing as is shown in 5.1. We also obtain single model state-of-the-art results on the COCO test-dev set, as shown in Table 3.3. In order to show the effectiveness of our

Table 3.2: **Comparison to prior work on the MS-COCO *val* dataset.** m.s. stands for multi-scale testing. *Used deformable convolution.

Method	Backbone	m.s. test	PQ	PQ^{Th}	PQ^{St}
JSIS-Net [dGMD18]	ResNet-50		26.9	29.3	23.3
Panoptic FPN [KGHD19]	ResNet-50		39.0	45.9	28.7
Panoptic FPN [KGHD19]	ResNet-101		40.3	47.5	29.5
AUNet [LCZ+19]	ResNet-50		39.6	49.1	25.2
UPSNet* [XLZ+19]	ResNet-50		42.5	48.5	33.4
UPSNet* [XLZ+19]	ResNet-50	✓	43.2	49.1	34.1
OANet [LPY+19]	ResNet-50		39.0	48.3	24.9
OANet [LPY+19]	ResNet-101		40.7	50.0	26.6
AdaptIS [SBK19]	ResNet-50		35.9	40.3	29.3
AdaptIS [SBK19]	ResNet-101		37	41.8	29.9
AdaptIS [SBK19]	ResNeXt-101		42.3	49.2	31.8
OCFusion	ResNet-50		41.3	49.4	29.0
OCFusion*	ResNet-50		42.5	49.1	32.5
OCFusion	ResNet-101		43.0	51.1	30.7
OCFusion*	ResNeXt-101		45.7	53.1	34.5
OCFusion	ResNet-50	✓	41.9	49.9	29.9
OCFusion*	ResNet-50	✓	43.3	50.0	33.8
OCFusion	ResNet-101	✓	43.5	51.5	31.5
OCFusion*	ResNeXt-101	✓	**46.3**	**53.5**	**35.4**

Table 3.3: **Comparison to prior work on the MS-COCO *test-dev* dataset.** m.s. stands for multi-scale testing. *Used deformable convolution.

Method	Backbone	m.s. test	PQ	PQ^{Th}	PQ^{St}
JSIS-Net [dGMD18]	ResNet-50		27.2	29.6	23.4
Panoptic FPN [KGHD19]	ResNet-101		40.9	48.3	29.7
OANet [LPY+19]	ResNet-101		41.3	50.4	27.7
AUNet [LCZ+19]	ResNeXt-152	✓	46.5	**55.9**	32.5
UPSNet* [XLZ+19]	ResNet-101	✓	46.6	53.2	36.7
AdaptIS [SBK19]	ResNeXt-101		42.8	50.1	31.8
OCFusion*	ResNeXt-101	✓	**46.7**	54.0	**35.7**

method, we compare to our baseline model in Table 3.1, and the results show that our method consistently provides significant gain on PQ^{Th} as well as PQ.

3.4.3 Cityscapes panoptic benchmark

We obtain competitive results on the Cityscapes validation set and the best results among models with a ResNet-50 backbone, shown in Table 3.5. Table 3.4 shows our strong relative improvement over the baseline on PQ^{Th} as well as PQ.

Table 3.4: **Comparison to our implementation of Panoptic FPN [KGHD19] baseline model on the Cityscapes *val* dataset**. All results are based on a ResNet-50 backbone.

Method	**PQ**	**PQ^{Th}**	**PQ^{St}**
Baseline	58.6	51.7	63.6
OCFusion	59.3	53.5	63.6
relative improvement	+0.7	+1.7	+0.0

Table 3.5: **Comparison to prior work on the Cityscapes *val* dataset.** All results are based on a ResNet-50 backbone. m.s. stands for multi-scale testing. *Used deformable convolution.

Method	**m.s. test**	**PQ**	**PQ^{Th}**	**PQ^{St}**
Panoptic FPN [KGHD19]		57.7	51.6	62.2
AUNet [LCZ+19]		56.4	52.7	59.0
UPSNet* [XLZ+19]		59.3	54.6	62.7
UPSNet* [XLZ+19]	✓	60.1	55.0	63.7
AdaptIS [SBK19]		59.0	**55.8**	61.3
OCFusion*		59.3	53.5	63.6
OCFusion*	✓	**60.2**	54.0	**64.7**

3.4.4 Occlusion head performance

In order to better gauge the performance of the occlusion head, we determine its classification accuracy on both COCO and Cityscapes validation dataset at $\rho = 0.20$ with ResNet-50

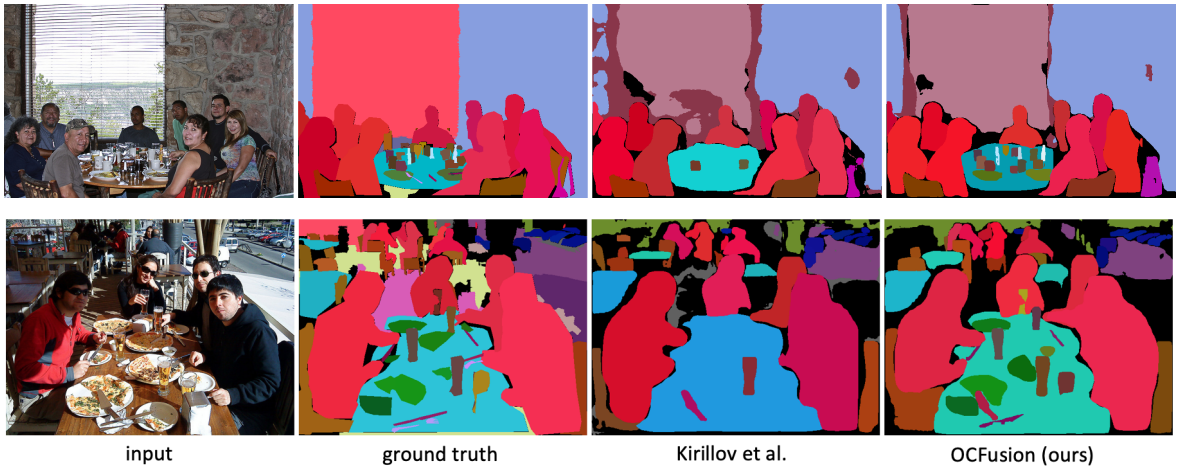


Figure 3.4: **Comparison against Kirillov et al. [KGHD19] which uses fusion by confidence.**

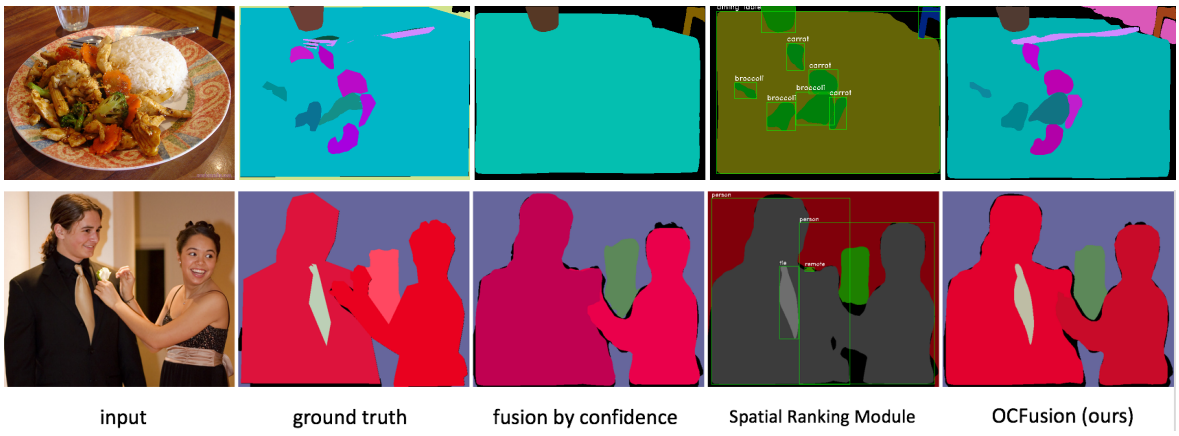


Figure 3.5: **Comparison against Spatial Ranking Module [LPY+19].**

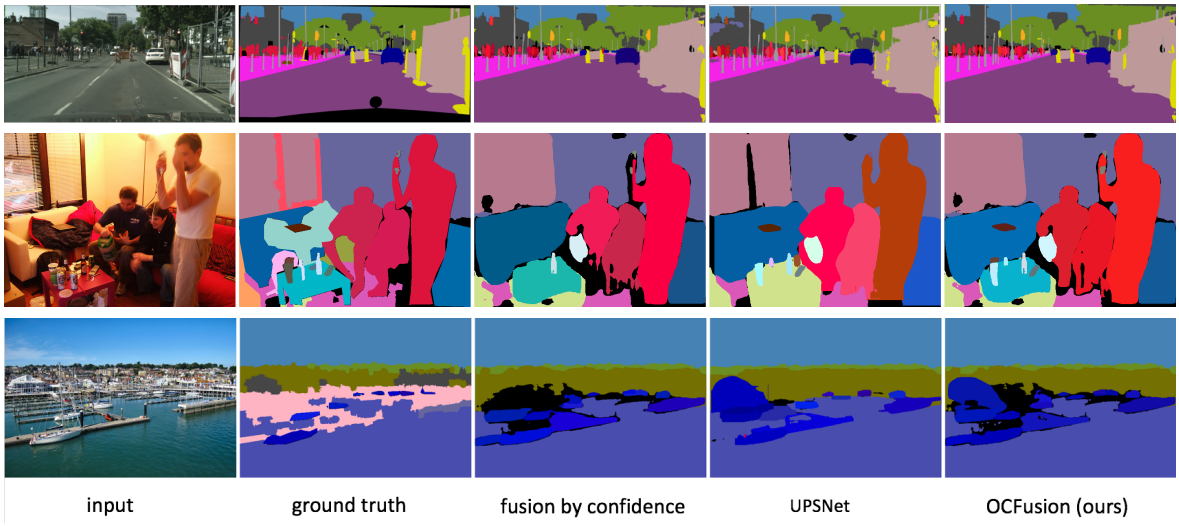


Figure 3.6: **Comparison against UPSNet [XLZ+19].**

backbone. We measure the accuracy of the occlusion head in predicting the true ordering given ground truth boxes and masks. The occlusion head classification accuracy on COCO and Cityscapes is 91.58% and 93.60%, respectively, which validates the effectiveness of OCFusion.

3.4.5 Inference time analysis

We analyze the computational cost of our method and empirically show the inference time overhead of our method compared to the baseline model. While our method incurs an $O(n^2)$ cost in order to compute pairwise intersections, where n is the number of instances, this computation is only needed for the subset of masks whose detection confidence is larger than a threshold (0.5 or 0.6 usually) as dictated by the Panoptic FPN [KGHD19] baseline. This filtering greatly limits the practical magnitude of n. Furthermore, only the subset of remaining mask pairs that have appreciable overlap (larger than ρ) requires evaluation by the occlusion head. We measure this inference time overhead in Table 3.6. OCFusion incurs a modest 2.0% increase in computational time on COCO and 4.7% increase on Cityscapes.

Table 3.6: **Runtime (ms) overhead per image.** Runtime results are averaged over the entire COCO and Cityscapes validation dataset. We use a single GeForce GTX 1080 Ti GPU and Xeon(R) CPU E5-2687W CPU.

Method	COCO	Cityscapes
Baseline	153	378
OCFusion	156	396
Change in runtime (ms)	+3	+18

3.4.6 Visual comparisons

Since panoptic segmentation is a relatively new task, the most recent papers offer only comparisons against the baseline presented in [KHG+19]. We additionally compare with a few other recent methods [LPY+19, XLZ+19].

We first compare our method against [KGHD19] in Figure 3.4 as well as two recent works: UPSNet [XLZ+19] in Figure 3.6 and the Spatial Ranking Module of [LPY+19] in Figure 3.5. The latter two have similar underlying architectures alongside modifications to their fusion process. We note that except for comparisons between [KGHD19], the comparison images shown are those *included in the respective papers and not of our own choosing*. Overall, we see that our method is able to preserve a significant number of instance occlusions lost by other methods while maintaining more realistic fusions, *e.g.*, the arm is entirely above the man versus sinking behind partly as in "fusion by confidence".

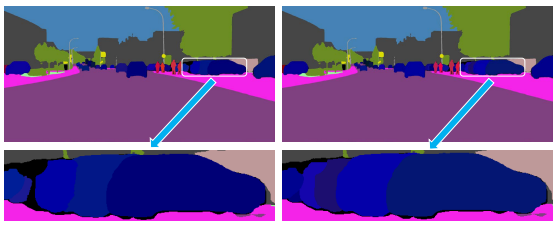


Figure 3.7: **Comparison for w/o (left) or w/ (right) intra-class capability enabled.** Best viewed in color.

Table 3.7: **COCO Hyperparameter Ablation: PQ**

(τ, ρ)	0.05	0.10	0.20
0.4	41.27 (Th: 49.43, St: 28.97)	41.22 (Th: 49.33, St: 28.97)	41.20 (Th: 49.30, St: 28.97)
0.5	41.20 (Th: 49.32, St: 28.95)	41.15 (Th: 49.23, St: 28.95)	41.24 (Th: 49.29, St: 29.10)
0.6	41.09 (Th: 49.15, St: 28.93)	41.03 (Th: 49.03, St: 28.93)	41.02 (Th: 49.02, St: 28.93)
N	192,519	157,784	132,165

Table 3.8: **Cityscapes Hyperparameter Ablation: PQ**

(τ, ρ)	0.05	0.10	0.20
0.4	58.76 (Th: 52.10, St: 63.62)	59.15 (Th: 53.00, St: 63.62)	59.07 (Th: 52.80, St: 63.63)
0.5	59.18 (Th: 53.09, St: 63.61)	59.26 (Th: 53.28, St: 63.61)	59.22 (Th: 53.19, St: 63.61)
0.6	59.21 (Th: 53.17, St: 63.61)	59.33 (Th: 53.46, St: 63.60)	58.70 (Th: 51.96, St: 61.60)
N	33,391	29,560	6,617

3.4.7 Ablation experiments

We study the sensitivity of our method to the hyperparameters τ and ρ in Table 3.7 for COCO and Table 3.8 for Cityscapes. We also include the number of examples of occlusions we are able to collect at the given ρ denoted as N. Naturally, a larger ρ leads to less spurious occlusions but decreases the overall number of examples that the occlusion head is able to learn from.

Intra-class instance occlusion in Cityscapes is a challenging problem, also noted in [HGDG17]. Since we can enable inter-class or intra-class occlusion query ability independently, we show ablation results in Table 3.9 that highlight the importance of being able to handle intra-class occlusion on. We believe this sets our method apart from others, *e.g.*, [LPY$^+$19] which simplifies the problem by handling inter-class occlusion only. Additionally, Figure 3.7 shows a visual comparison between resulting panoptic segmentations when intra-class occlusion handling is toggled on Cityscapes. Only the model with intra-class handling enabled can handle the occluded cars better during the fusion process.

Table 3.9: **Ablation study on different types of occlusion on the Cityscapes *val* dataset.** ✓means capability enabled.

Inter-class	Intra-class	PQ	PQ^{Th}	PQ^{St}
		58.6	51.7	63.6
✓		59.2 (+0.5)	53.0 (+1.3)	63.6 (+0.0)
✓	✓	59.3 (+0.7)	53.5 (+1.7)	63.6 (+0.0)

3.5 Conclusion

We have introduced an *explicit* notion of instance occlusion to Mask R-CNN so that instances may be effectively fused when producing a panoptic segmentation. We assemble a dataset of occlusions already present in the COCO and Cityscapes datasets and then learn an additional head for Mask R-CNN tasked with predicting occlusion between two masks. Adding occlusion head on top of Panoptic FPN incurs minimal overhead, and we show that it is effective even when trained for few thousand iterations. In the future, we hope to explore how further understanding of occlusion, including relationships of *stuff*, could be helpful.

This chapter is based on material as it appears in the Proceedings of the IEEE CVF Conference on Computer Vision and Pattern Recognition. (CVPR), 2020. (Justin Lazarow, Kwonjoon Lee, Kunyu Shi, Zhuowen Tu, "Learning instance occlusion for panoptic segmentation"). The dissertation author was the co-primary investigator and author of this paper.

Chapter 4

Scene Layout from Amodal Context

4.1 Introduction

Our ability to perceive the world requires us to understand scene layout. Scene layout conveys high-level abstractions by capturing the spatial relationships between participants within a scene. These participants constitute a dichotomy of "things" (countable object instances such as pedestrians and cars) and "stuff" (amorphous background regions such as sky and road). Significant progress has been made to infer these participants from natural images [LMB+14], [COR+16a] through semantic, instance, and now panoptic segmentation [CPK+18], [HGDG17], [KHG+19]. However, less work has been devoted to *producing scene layout* from generative models as a task within itself. While segmentation algorithms focus on producing structure *from* images, we focus on producing plausible structure from which novel images can be synthesized.

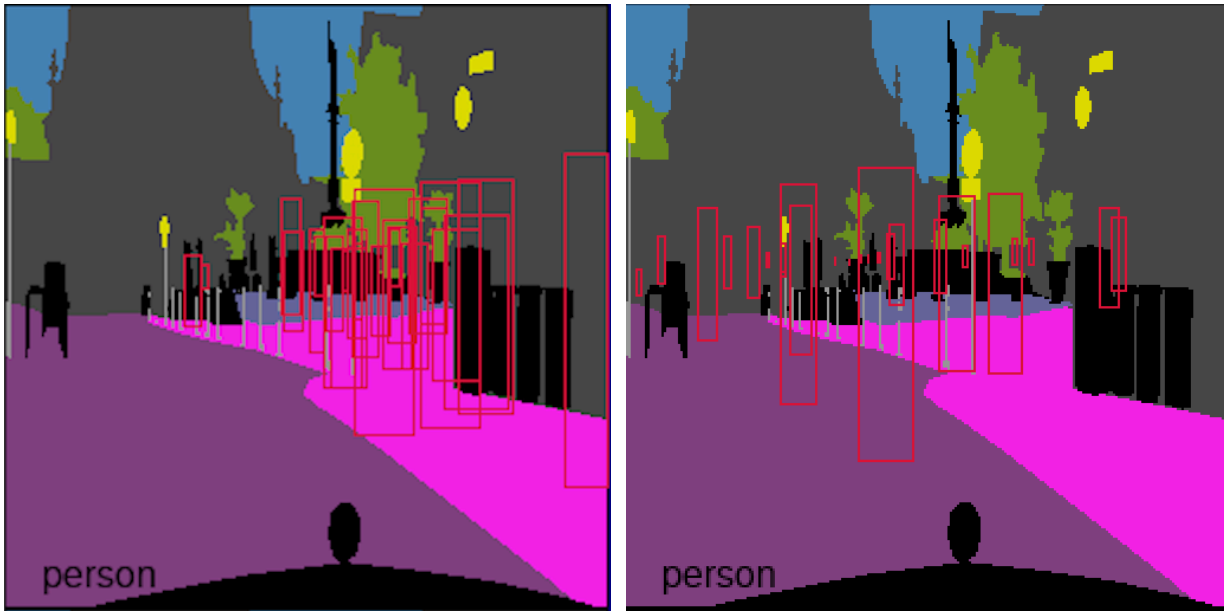


Figure 4.1: An illustration of our model capturing the spatial associations between "things" and "stuff" when attempting to hypothesize locations of persons. We call our method scene layout from amodal context (SLAC); LayoutVAE refers to the method in [JDH+19]. The predictions of SLAC shown on the left demonstrates its ability to accurately localize the spatial extent of amorphous background regions by giving a diverse set of locations of persons — all constrained to the sidewalk.

Learning to lay out scenes requires us to consider the nature of a scene. While "things" have a well-defined count, kind, position and scale, the "stuff" is amorphous in extent and remains more or less static over time. People and vehicles inherently have a sense of velocity and transience, while buildings, sidewalks, and roads do not generally find themselves in motion. Furthermore, a scene exists without "things", but it is less realistic to imagine a scene without "stuff" to frame it. Additionally, the "stuff" is highly informative for valid placements of objects. For example, a car should be *on* a road, and a pedestrian should likely be close to the safety of a sidewalk. Therefore, we adopt the view [JDH+19] that in order to lay out things in a sequential manner, we should condition on stuff.

While such an ordering of a scene is natural, the representation we pick for this conditioning requires care. Previous work [JDH+19] considers a coarse, bounding box based representation for both the "things" and "stuff". Representations of "things" through bounding boxes has been explored and validated within object detection. However, object detection is *only* concerned with "things". When understanding of "stuff" is desired, semantic segmentation algorithms are produced at the pixel-level to accommodate their amorphous nature. Therefore, we should adopt conditioning of stuff during layout in the same manner by using a representation at the pixel-level.

We contrast the box versus pixel-level views in Figure 4.2. In this example, it is unclear how to derive the true dense layout from bounding boxes alone. The bounding box for the road contains the entirety of the sidewalk and significant parts of the buildings. In contrast, the pixel-level semantic segmentation has clear boundaries and localization.

If we are to build an autoregressive (sequential) layout model conditioned on semantic segmentation, we must have the ability to condition each step on the history of both stuff and things that have already been placed. At the extreme, this implies we must have a view of the scene without *any things*. This view initially represents thecomplete background segmentation (or amodal segmentation of stuff) and in subsequent steps contains a partial number of modal instance segmentations of "things". While amodal segmentation has been considered previously

[LM16], [ZTMD17], [QJL+19] at the prediction level, our approach to *amodality* considers it within a generative model. Additionally, we consider *amodal instance segmentation* [LM16], [QJL+19] as the extent of a thing despite occlusion (from itself or others) within our application to scene synthesis.

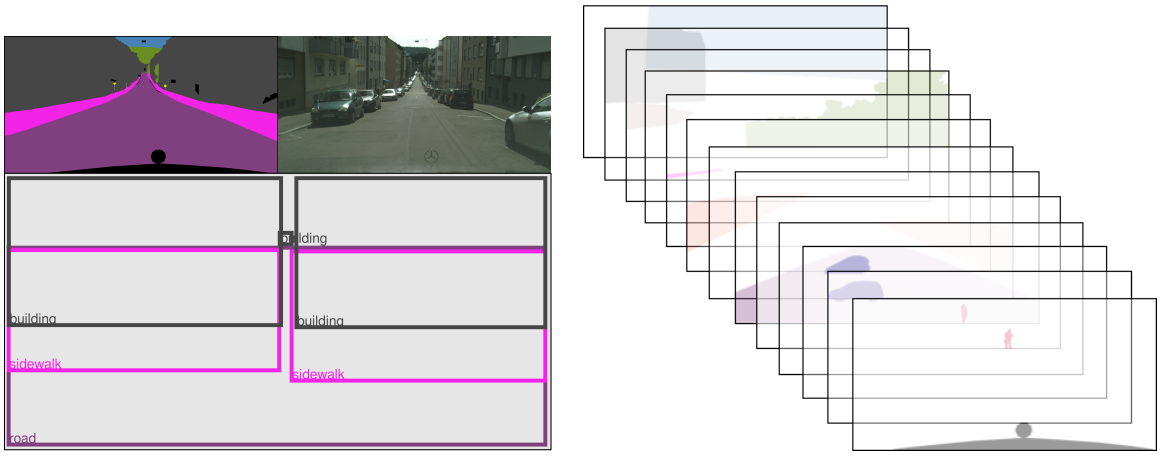
Overall, we face two problems:

- How can we produce amodal stuff segmentations to condition our model on?
- How should we build a model of layout that can condition and predict with respect to a pixel-wise segmentation?

To answer these questions, we contribute the following:

- We introduce a form of layout that is conditioned, constrained, and grounded by the amodal stuff within a scene. Our model conditions on pixels rather than on coordinates like previous work. We use a pixel-wise *amodal* segmentation of the stuff in a scene as a means to ground our model, as well as removing the need for memory units. Furthermore, we place it within a variational framework to support a diverse set of plausible layouts.
- We show that not only can we learn over a small-scale synthetic dataset [ZP13] and explicit amodally segmented datasets [Bil06], [ZTMD17], but that we can derive reasonable amodal stuff segmentations within the Cityscapes [COR+16a] dataset to enable a complex evaluation setting.
- We propose a scenario where we ask a system for "guesses" of where objects might be in a scene given only an amodal segmentation of the stuff in a scene. We show that our model can achieve significant recall of instances when we treat these guesses as layout hypotheses — being able to *recall over 55% of cars* in the Cityscapes dataset when given 500 guesses and showing efficiency when given a smaller number of opportunities.

  We refer to this as "scene layout from amodal context" or **SLAC**.



(a) (b)

Figure 4.2: (a). A comparison of representations for the "stuff" in terms of bounding boxes to one based off of a dense pixel-wise segmentation. Our approach uses a pixel-wise segmentation (top left) as conditioning while still being able to predict bounding boxes. (b). An illustration of how Cityscapes is annotated. We use this to generate near ground-truth amodal segmentations of the stuff in a scene by not merging layers corresponding to "things".

4.2 Related work

Layout and context. Context [OT07] plays an important role in visual perception. Context modeling has been adopted in various computer vision applications such as object categorization [RVG+07], semantic labeling [Tu08], and object detection [CDXH13]. While the concept of context is rather broad, layout [HHF] provides context in the form of spatial configurations of object shapes and locations. Learning a discriminative layout model [DRF11] has shown to be effective in producing improved multi-class object detection. Here, we study a generative model of layout that allows us to synthesize new layouts of multiple objects — obtaining faithful geometric and spatial configuration in a scene.

Amodal segmentation. Few works have investigated producing amodal segmentations of a scene. Zhu *et al.*[ZTMD17] produce amodal segmentations of a small subset of images within the COCO dataset and extend the SharpMask framework to produce amodal masks. Li and Malik [LM16] paste new instances into images to produce occlusions in order to train an amodal

segmentation algorithm. Recently, Qi *et al.*[QJL+19] released a dataset consisting of amodal instance segmentations of the KITTI dataset along with extensions to modern segmentation pipelines to produce amodal predictions.

Sentence-conditioned image generation. A number of works [RAM+16], [RAY+16] use natural language as grounding for synthesizing images. Most relevant to ours is Hong *et al.*[HYCL18] which uses natural language to infer a semantic layout of "things" from which a natural image consistent with it can be generated.

Inserting instances into scenes. Given a semantic segmentation, Lee *et al.*[LLG+18] consider inserting a new instance in a plausible manner by modeling location and shape. However, because they rely on *inserting* a new instance rather than reconstructing the layout, they are forced to rely on GAN-based techniques. The architecture is only validated on two classes (person and car) within the Cityscapes dataset and requires retraining for each class. Additionally, this setting is also smaller in scope than ours as we aim to generate an entire layout from conditioning.

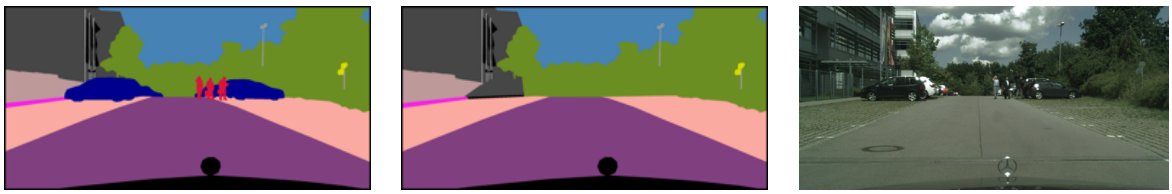
General layout. Recent work including [LYH+19] and [JDH+19] focuses on layout of simple objects e.g. points and boxes without conditioning. LayoutGAN [LYH+19] using a series of attentional blocks to transform a random parameterization of a shape or layout into plausible one along with a basic differentiable renderer as an adversarial loss. It directly produces samples from the joint distribution of a fixed count layout. LayoutVAE [JDH+19] is an autoregressive VAE most relevant to ours, however, it focuses on the problem in coordinate space where their model treats the representation of both "stuff" and "things" as normalized bounding boxes.

4.3 Humans as Amodal Annotators

The basic premise of our model requires amodal segmentations of the "stuff" within a scene. Except for synthetic datasets [ZP13], this hinges on the ability for humans to perceive the visual world amodally [Kan79], [KTCM15]. We note previous attempts to annotate scenes in

such a way [Bil06], [GH12], [ZTMD17] have been made in the past and we apply our model to each in this work. However, we consider the Cityscapes dataset [COR+16a] as an ideal setting for scene understanding — having basic spatial relationships but also having exceedingly complex scenes (often having over 60 objects in a scene). While this dataset has never before been applied in the context of amodal vision, we aim to show its promise and applicability to our problem here by taking advantage of *human annotator bias*.

We note an insight from [GH12] — that human annotators are very natural in their attempts to want to label segmentations amodally. Previous attempts to take advantage of this include [LLT19], where they elicit ordering relationships due to amodal annotator bias. Here, we examine the Cityscapes [COR+16a] dataset to see whether the same intuition can be applied to *stuff*. While this is not a new dataset, we note that the provided label images are most commonly used, however, in [COR+16a], they remark that annotators were asked to annotate *back to front* using polygons. We hypothesized that if we were to follow this sequential process for merging each annotation layer, we might simply be able to *ignore* layers corresponding to instances and be left with a scene that might sufficiently characterize the amodal segmentation of the stuff within the scene. We illustrate this idea in Figure 4.2.



(a) Original segmentation (b) Amodal segmentation (c) Input image

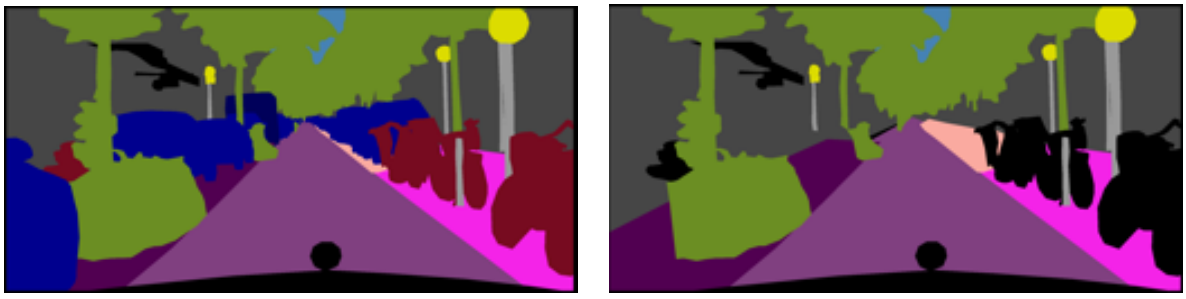
Figure 4.3: An illustration of the resulting amodal stuff segmentation compared to the original segmentation of the image in Figure 4.2.b

To verify this, we manually examined each image within the Cityscapes training and validation datasets. We simultaneously compared the amodal stuff segmentation of the scene (as generated by ignoring the "thing" layers) to both the ground truth semantic segmentation

of the entire scene and ground truth RGB image. We look for resulting segmentations where the annotator reasonably annotated the stuff behind an instance. In particular, it should not be possible to determine any characteristics (kind, extent, or shape) of an object that was removed. We find 2,707 of 2,975 images (over 90%) meet our criteria in the training set and 358 of 500 in the validation set. We provide examples of the process (including examples we reject) in Figures 4.3 and 4.4.

4.4 Scene Layout From Amodal Context

We present our model that learns scene layout of *things* by amodal context (**SLAC**). We emphasize the key distinctions between our model and previous work: we perform layout in the image-space, we do not require memory units, and we explicitly encode an object with its local context.



(a) Example segmentation (b) Rejected amodal segmentation

Figure 4.4: An example we reject from the Cityscapes dataset. We clearly see missing stuff annotations in (b) when compared to the original (a).

4.4.1 Layout within image-space

We aim to both condition and predict layout within the image space rather than regressing normalized coordinates in the range $[0, 1]$. This allows us to take advantage of well-established architectures and recent techniques within scene understanding. However, predicting bounding

boxes becomes less straightforward in the fixed, discrete space of an $N \times N$ image. We note that CornerNet [LD18], a recent work in object detection, found success by producing bounding box detections through keypoint detection. In particular, they produce an output of two feature maps whose activations are used to infer the top left and bottom right corner of a bounding box. We adopt this by building a generative model of "corners" from which we can then infer bounding boxes.

4.4.2 Segmentation as memory

Traditional sequence models rely on some form memory to capture long-term dependencies within the sequence. Rather than relying on memory, we extend our conditioning to provide history to the model. Recalling that our model relies on a dense segmentation of the "stuff" as grounding, we progressively merge the ground truth segmentation of a "thing" for subsequent steps. Subsequent steps are then immediately informed of both the class and extent of an preceding thing. A scene-level segmentation can capture occupancy in a straightforward manner as well as class relationships when predicting layout. Furthermore, since the only temporal dependency becomes the conditioning itself, teacher-forcing permits that each training step is based off of the conditioning alone whereas other models must past gradients through their memory units — costly for large layouts.

4.4.3 Encoding local context

While a dense segmentation provides a full spatial conditioning to ground the predictions of the model, we find it is unnecessary for the encoding process. Rather, we choose to encode the local context of an object. This significantly reduces the feature dimension of the conditioning for the encoder and allows us to encode at a variety of scales by extracting features at a subset of the conditioning network's feature maps. We extract features by performing bilinear interpolation on the deeper feature maps at the center of the object's bounding box. This produces 1×1 feature

slices which we can concatenate across the levels. We ablate the choice of context in Table 4.2.

4.4.4 Architecture

We implement our model of layout as an autoregressive (conditional [SLY15]) VAE, as done in [JDH+19]. Each autoregressive step predicts a single objects' position conditional on both the preceeding instances and background. We discuss the three high-level components of our conditional VAE.

Conditioning. Conditioning serves to aid both the encoding of an object and the decoding process. While previous work relies on coarse box information to condition on preceding objects, we rely on *dense* conditioning. This conditioning comes in the form of a pixel-wise segmentation of the scene at the current autoregressive step. Initially, this is exactly the background segmentation (no instances) of the scene. After each autoregressive step, we apply teacher forcing by merging the ground truth instance segmentation of the previous instance. We ensure this merging respects the depth ordering of the ground-truth instances in the scene. Our model has no explicit memory, instead, we find the (partially) merged segmentation sufficient to act as both conditioning and history. The conditioning is applied in different ways to the encoder and the decoder of the VAE as mentioned in each respective section.

In our implementation, we use a CNN as a conditioning network. The input is a one-hot encoding of the current scene segmentation. The CNN consists of 5 layers of stride 2 convolutions with LeakyReLU activations after. Initially, there are 32 filters with a doubling of the filter size after each convolution until the second to last layer where the depth is repeated at 256 feature maps.

Encoder. The encoder network E of our VAE serves to encode an object's bounding box, class information, and conditioning. We encode a bounding-box represented by top-left and bottom-right coordinates (xyxy). In order to encode local object context, we perform bilinear interpolation on the final three feature maps of the conditioning network at the center point of the

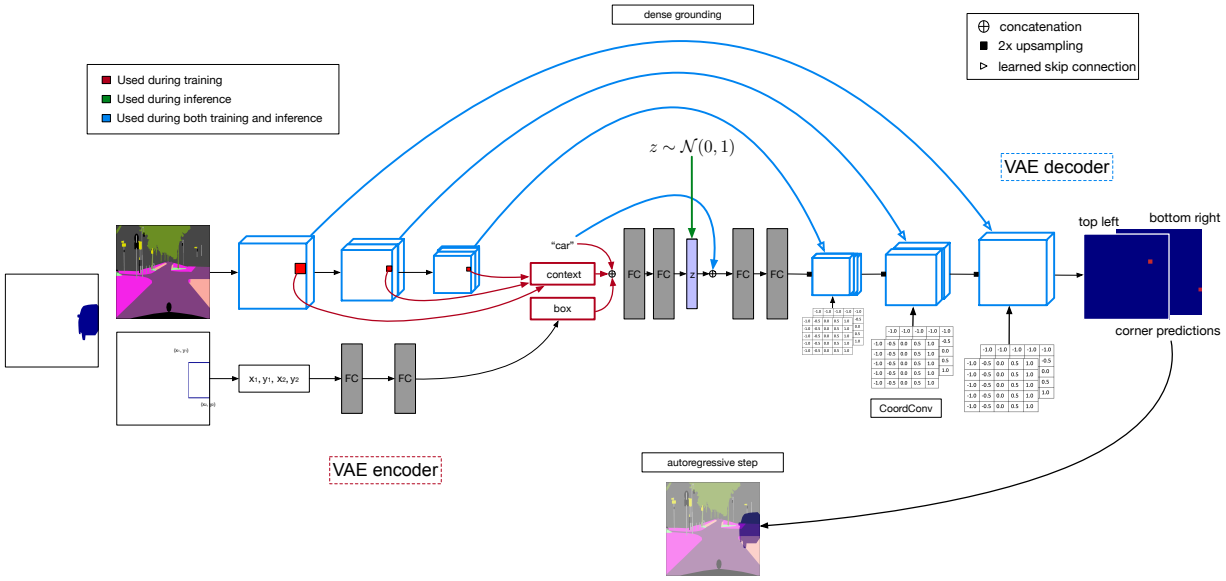


Figure 4.5: An illustration of the general architecture of our model. The autoregressive nature of the model is captured by the arrow emanating from the output to the input where we merge the ground truth segmentation to use as conditioning for subsequent object placements

object's bounding box. We concatenate the one-hot label of the current instance before projecting into the latent representation z. We denote the approximate posterior induced by E as q_ϕ with prediction of mean and variance parameters μ_ϕ and σ^2_ϕ.

Decoder. The decoder network D is a CNN that uses bilinear upsampling (2x) between layers of convolution. While the input to the encoder is in coordinate space, the decoder outputs in image space in order to retain the spatial nature of the conditioning features. To accomodate this gap, we use CoordConv [LLM+18] at each convolution to explicitly provide the discretized nature of a feature map. Additionally, while the encoder is privy to the position of an object, the decoder must make use of *global* conditioning information to ground its prediction. To provide this global information, we use learned skip connections from the conditioning network. Therefore, our "decoding" process mimics that of a U-Net [RFB15] where the skip connections can aid the localization of the corner keypoints. Finally, we output two full resolution probability maps. We denote the induced Bernoulli distribution over the output space as p_θ.

An illustration of the architecture is given in Figure 4.5.

4.4.5 Training

We follow standard VAE training procedures. Because we are only attempting to predict two pixels (one for each corner) with a positive label, we encounter a large amount of imbalance in an ordinary cross entropy loss formulation. Like previous work in keypoint-based detection [LD18], we use a focal loss [LGG+17] as supervision of the decoder's outputs. For each bounding box, we extract the top left and bottom right corners and encode each as coordinates. The decoder produces two heatmaps in image space for each such corner and this output is supervised using the focal loss. We use the procedure of [LD18] which chooses an adaptive radius (corresponding to the standard deviation of a Gaussian) based off of the size of the bounding box. Formally, the loss for a single step i in our model becomes:

$$\mathcal{L}^i = E_{q_\phi(z_i|x_i,c_i,A_i)}\left[\log p_\theta(h_i \mid z_i, c_i, A_i)\right] + \beta * \mathrm{KL}\left(q_\phi(z_i \mid x_i, c_i, A_i) \middle\| p_\theta(z_i \mid c_i)\right)$$

where A is the conditioning segmentation, x is the coordinate encoding of the bounding box, h are the heatmaps defining the top left and bottom right corners, and c is the class label of the instance at the current step. We average this loss over each step in a given layout.

We find $\beta = 0.1$ better balances the reduced magnitude of the focal loss.

4.4.6 Inference

At inference time, we compute the conditioning from the amodal stuff segmentation. We draw from a unit Gaussian and concatenate the flattened conditioning with it and a one-hot encoding of the label of the current object. From this, the decoder produces two full-sized feature maps. We find the peak activation in the first map and use the image coordinates as the top left corner of the bounding box. We then search in the second map down and to the right from this corner to find the bottom right corner. In order to condition on the prediction, we must place a

mask at the predicted box. On Abstract Scenes, we can project the true mask of the object. On other datasets, we fill the predicted bounding box with pixels as a mask.

4.5 Experiments

We perform experiments on the Abstract Scenes dataset [ZP13], the COCO Amodal dataset [ZTMD17], CBCL StreetScenes [Bil06], Cityscapes dataset [COR+16a]. Because our model requires both amodal segmentations and a clear separation of "things" and "stuff", we elaborate on the accomodations necessary for each dataset in their respective section. All models are trained for 100 epochs except for COCO Amodal which is trained for 50 epochs. The model with best validation (IoU) loss chosen for evaluation. We optimize using Adam [KB14] with a learning rate of 0.0001.

4.5.1 Baselines

We compare against LayoutVAE [JDH+19] as a strong baseline. LayoutVAE is a purely bounding box based approach that similarly uses an autoregressive VAE but requires an LSTM conditioning component. We relax it into a conditional sequence model by initially providing it all "stuff" bounding boxes and then supervising its predictions of bounding boxes for subsequent things. We follow their ordering procedure of boxes by sorting from leftmost to rightmost. An implementation of LayoutVAE is not available from the authors so all experiments are done using our own implementation.

For the "knowing where to look" evaluation, we additionally explore an extension to LayoutVAE that we name "LayoutVAE + Dense" where we provide equal conditioning to that of SLAC by augmenting it to accept segmentation masks within its conditioning module. At each step, we provide the corresponding binary mask of the stuff or thing segment of interest and encode it using a CNN similar to the conditioning network of SLAC.

We do not compare against LayoutGAN [LYH+19] nor [LLG+18]. LayoutGAN directly models the joint distribution of a fixed-length layout and therefore does not factorize conditionally like an autoregressive model. This makes comparisons of likelihood as difficult as evalauting the log-likelihood of a GAN. Additionally, the authors provide no implementation nor is a successful reproduction available. The model from [LLG+18] is only validated by the authors on "person" and "car" classes where they require a *separate model* to be trained for each.

4.5.2 Abstract Scenes

The Abstract Scenes dataset [ZP13] is made up of "clip art" scenes where users on MechanicalTurk were given seed images and asked to place some number of objects within it. The set of objects is fixed and within each type of object (e.g. boy), there are a predefined set of instantiations given in image files. We have the ability to both generate true segmentations and synthesize novel images, because the placement of each object within the scene is described in a procedural manner.

The dataset does not come with a predefined classification of its objects into "things" and "stuff". We treat grass, sky, animals, trees, sun, clouds, slides, sandboxes, grills, swings, tents, and shelters as "stuff". We treat hat, food, boy, girl, ball, plane, rocket, balloon, and glasses as the "things" we want to lay out. We remove a small number of objects where the corresponding object is not easily represented as a polygon (e.g. a cloud with rain droplets from it) for simplicity. We use these to generate semantic segmentations of the scenes as required by our model. We reserve 20% of the dataset for validation.

4.5.3 COCO Amodal

The COCO amodal dataset was introduced in [ZTMD17] and consists of a small subset of the COCO dataset with amodal annotations of the scene. However, annotators were given

freedom to assign category names to each annotation which results in an unwieldy number of categories. We reconcile this by considering the panoptic annotation [KHG+19] of each image and assigning an amodally annotated region to a COCO category by finding the category of the segment in the panoptic ground truth that its modal region most overlaps. We require at least 0.33 IoU to assign a segment in such a manner. However, there are numerous cases were annotators neglected to annotate certain amorphous regions of the image. This lack of annotation can result in sparse conditioning for our models. On the other hand, the panoptic segmentation is much more complete in terms of "stuff". Therefore, we additionally require that at least 1/3 of the known areas of stuff from the panoptic segmentation are covered in the amodal stuff regions. This results in 1,086 training images and 626 validation images over the 80 "thing" and 53 "stuff" categories in COCO Panoptic.

4.5.4 CBCL StreetScenes

The CBCL StreetScenes dataset [Bil06] consists of street scenes around Boston, MA. Annotators were given strict rules regarding occlusion and amodal annotation of surfaces. It has been used in the past [GH12] as a setting for amodal surface prediction. The dataset includes 9 categories which we divide into "stuff" and "things". The former includes building, tree, road, sky, sidewalk, and store while the latter is made up of car, pedestrian, and bicycle. After applying the process in Section 4.3, we retain 2,616 training images and 600 validation images.

4.5.5 Cityscapes

We adopt the Cityscapes dataset [COR+16a] as described in Section 4.3 and retain a training subset 2,707 images and validation subset of 358 images. We divide the set of classes into things and stuff using the same split as panoptic segmentation [KGHD19]. Thing classes include person, rider, car, truck, train, and bicycle while stuff classes include ground, road, sidewalk

parking, building, wall, fence, pole, traffic light, vegetation, terrain, and sky. We ignore all other classes and set their corresponding labels to "void" in the segmentations. For all experiments, we resize the images from 2048×1024 to 256×256. For our model, we order instances front to back which ensures that an object's occluder is always present in conditioning for that object.

4.5.6 Qualitative results

Novel layout synthesis. We provide qualitative results in Figure 4.6 on both datasets by synthesizing novel layouts for each image. We use the ground truth label count and order to anchor the generation of novel placements. At each step, we generate 25 random samples from the model and compute pairwise IoU between predicted boxes. We use this to rank each box. By summing the pairwise IoUs of each box and normalizing this sum, we pick the box corresponding to the largest normalized sum.

We see that our model generates plausible layouts diverse from the ground truth layout and is able to localize plausible locations for "items" like sunglasses (being near the head) and sports balls (being near the hand). Furthermore, our model avoids placing objects on top of one another. We believe this is straightforward for our model to accomplish since, at any given step, the conditioning can convey exactly which pixels are already occupied. On Cityscapes and CBCL, we see that our model is better able to follow the natural path of a road when laying out cars when compared to LayoutVAE, which appears to produce more predictions that do not realistically appear *on* the corresponding surface. Additionally, the predicted scale of objects appears more plausible in our layouts. LayoutVAE can produce extremely small predictions despite belonging to a large object in the foreground. Since our model operates in the image space, it is possible that it can capture the effects of foreshortening and wide field of views more readily.

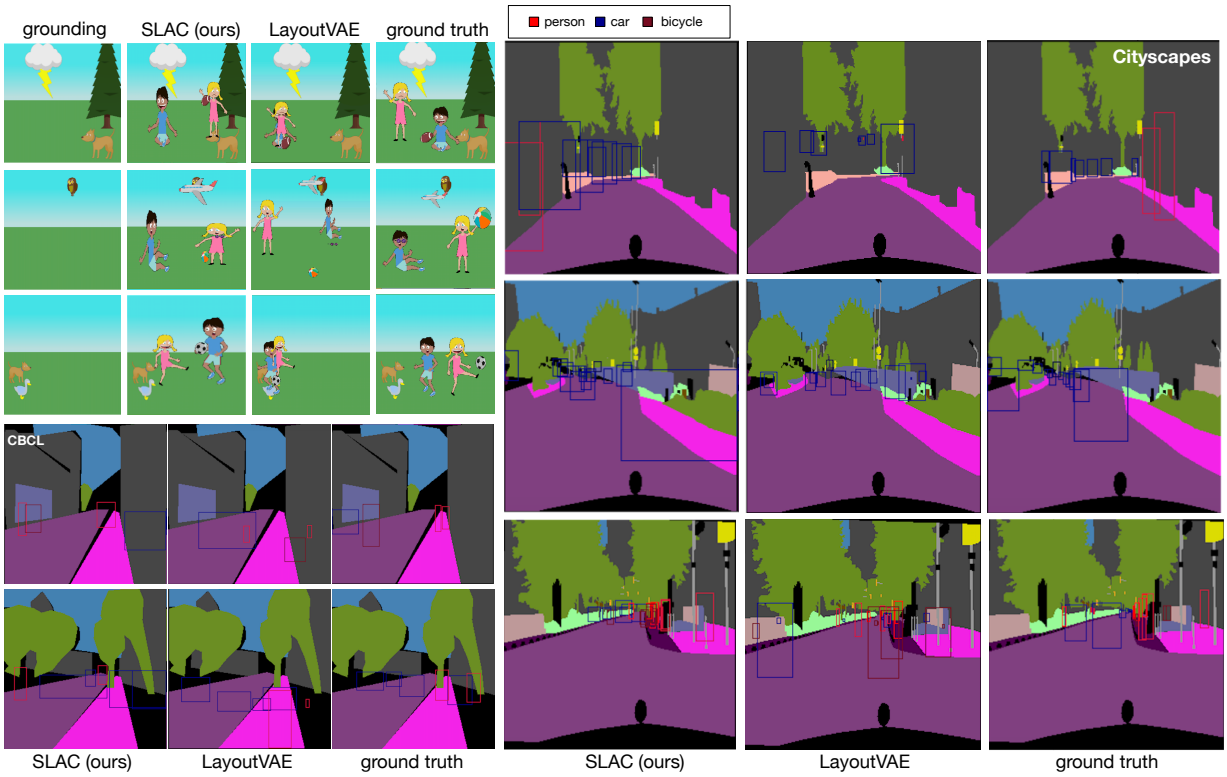


Figure 4.6: Novel syntheses of layouts using our method across Abstract Scenes (top left), CBCL (bottom left), and Cityscapes (right) with their amodal grounding. Best viewed in color.

4.5.7 Quantitative results

Sequence evaluation. Traditional sequence evaluation considers the ability for our model to accurately predict the ground truth bounding box conditional on all preceding bounding boxes (teacher forcing). As in [JDH+19], we adopt intersection over union (IoU) to measure the similarity of two bounding boxes. We compare to the baseline models in Table 4.1. In each dataset, we observe appreciable gains in accuracy from SLAC over the baselines. The dense, amodal context that conditions SLAC provides for more accurate localization of the stuff. This localization further aids the localization of the predicted things.

Table 4.1: **Likelihood evaluation of validation set sequences using IoU**

Method	Abstract Scenes	Cityscapes	COCO Amodal	StreetScenes
LayoutVAE	0.088	0.020	0.078	0.096
SLAC (ours)	**0.108**	**0.027**	**0.105**	**0.136**

Ablation study. Context plays an important role in our encoding of an object. We choose a subset of feature maps from which we interpolate at an object's center position to compute this local context. We ablate this choice of feature maps in Table 4.2 on the Cityscapes dataset. When we include deeper feature maps, we induce a larger receptive field and higher level features into an object's encoding context. We achieve the best performance when we encode the deepest three feature maps from the conditioning network, however, even the shallowest context still outperforms the LayoutVAE baseline.

Table 4.2: **Ablation study on context within the Cityscapes dataset (IoU)**

Depth	Box IoU
3	0.024
3, 4	0.026
3, 4, 5	0.027

Knowing where to look. While traditional sequence evaluation can be useful, we believe that the multimodal nature of layout can be better evaluated in the spirit of object detectors. Modern object detection systems often use a predefined set of (anchor) boxes across a set of

locations and scales as the starting point for deciding whether an object exists within an image. For two-stage detectors e.g. R-CNN [RHGS15], this means calculating the objectness score of the anchor box, and then deciding the class of object actually present there once hypothesized by the first stage. We note that there is no learned class-conditional prior concerning the positions and scales for where the object detector *should look* — the hypothesis space defined by these anchor boxes is entirely enumerated.

While this is a diligent approach for maximizing recall, we propose a less stringent setting where we investigate a combination of perception and generative modeling to ask, *where should we look?* We do not aim to build a detector but to hypothesize the true configuration of a scene when only given the amodal extent of the stuff within it.

We define our quantitative measure of performance by the object recall of each method given a fixed number of guesses. We use a range of IoU thresholds to determine whether a hypothesis should be matched to a ground truth object. We evaluate the performance of a uniform baseline [HBDS15], LayoutVAE [JDH$^+$19], LayoutVAE + Dense, and SLAC on the Cityscapes validation dataset in Table 4.3. Due to the density/complexity of Cityscapes scenes (often having dozens of instances), we believe this is best suited for this type of evaluation.

In order to produce proposals, we sample the specified number of guesses for each class and condition *only on stuff*. This conditioning amounts to the sequence of stuff bounding boxes (and stuff masks for the dense extension) for LayoutVAE and the amodal stuff segmentation for SLAC. We note that this process *does not* sequentially process "things" one after the other. Each guess is independent of all other guesses, and no instances are ever present in the conditioning for either model.

Table 4.3: **Average Recall (%) at IoU of 0.5, 0.6, and 0.7 for Cityscapes**

Method	person	rider	car	truck	bus	bicycle
50 guesses						
Uniform	0.1	0.5	0.2	0.1	0.0	0.0
LayoutVAE	4.3	7.5	6.1	5.6	7.4	2.4
LayoutVAE + Dense	3.5	6.2	7.3	**12.3**	9.5	2.8
SLAC (ours)	**4.9**	**9.4**	**11.1**	11.8	**11.6**	**5.2**

Method	person	rider	car	truck	bus	bicycle
100 guesses						
Uniform	0.1	1.9	0.2	0.1	0.0	0.0
LayoutVAE	7.2	9.1	9.5	5.6	7.4	5.7
LayoutVAE + Dense	7.5	10.1	11.4	16.9	16.4	6.3
SLAC (ours)	**9.4**	**12.7**	**17.3**	**20.5**	**22.8**	**7.8**

Method	person	rider	car	truck	bus	bicycle
200 guesses						
Uniform	0.2	3.6	0.3	0.1	0.0	0.0
LayoutVAE	12.7	19.2	13.5	7.7	9.5	8.4
LayoutVAE + Dense	13.1	17.4	17.6	**28.2**	19.6	10.0
SLAC (ours)	**16.2**	**21.6**	**26.4**	27.2	**41.3**	**13.2**

Method	person	rider	car	truck	bus	bicycle
500 guesses						
Uniform	0.5	6.6	0.5	0.5	0.0	0.1
LayoutVAE	22.4	30.8	18.0	11.8	12.7	9.4
LayoutVAE + Dense	24.8	33.3	25.7	41.5	28.0	14.6
SLAC (ours)	**27.1**	**35.3**	**38.6**	**50.3**	**60.3**	**21.8**

We include the changes in performance as the total number of guesses made varies from 50 to 500 and as the IoU levels (to determine a correct guess) vary between 0.5, 0.6, and 0.7. For reference, the hypothesis space of a typical setting for Faster R-CNN [RHGS15] will consist of about 4000 boxes within the region proposal network on a 256×256 image. From the results, we see that our method outperforms LayoutVAE [JDH$^+$19] across every category. As expected, when LayoutVAE is extended to include segmentation masks as conditioning (matching the information provided to SLAC), performance improves over the coarse model. However, SLAC is still able to outperform it with one exception. We believe the gains from SLAC follow for a variety of reasons. SLAC *only* observes conditioning information at the scene level i.e. each conditioning segmentation is a valid scene in itself. SLAC is free from having to aggregate (and thus bottleneck) information within memory units. At the same time, SLAC provides the decoder with learned skip connections from the conditioning network so that spatial information is readily retained in order to more precisely *ground* the prediction at inference time. Furthermore, since SLAC has no memory, it does not need to be directly trained as a sequence model. We speculate that this allows for a smoother learning process by not having to aggregate gradients over long layout sequences. While a background surface like a sidewalk might be observed early in a layout ordering, a "person" whose context is dependent on this sidewalk might only occur much later within the sequence of objects.

4.6 Conclusion

We have introduced a new model of layout where the layout of "things" is conditioned on a dense segmentation of the "stuff" in the image. In order to do this, we introduce the idea of conditioning of an amodal segmentation of the "stuff" in an image. We show that this is plausible even within a complex dataset as a result of human annotation bias. We apply this learned model to hypothesize the locations of objects within images where only the "stuff" in the image is seen. We show impressive recall that increases as the number of hypotheses is increased. We hope that we can extend and integrate this knowledge of layout into more scene understanding tasks in the future.

This chapter is based on material submitted for publication of material by Justin Lazarow and Zhuowen Tu. This dissertation author was the primary investigator and author of this paper.

Chapter 5

End-to-end Differentiable Instance Segmentation through Boundary Prediction

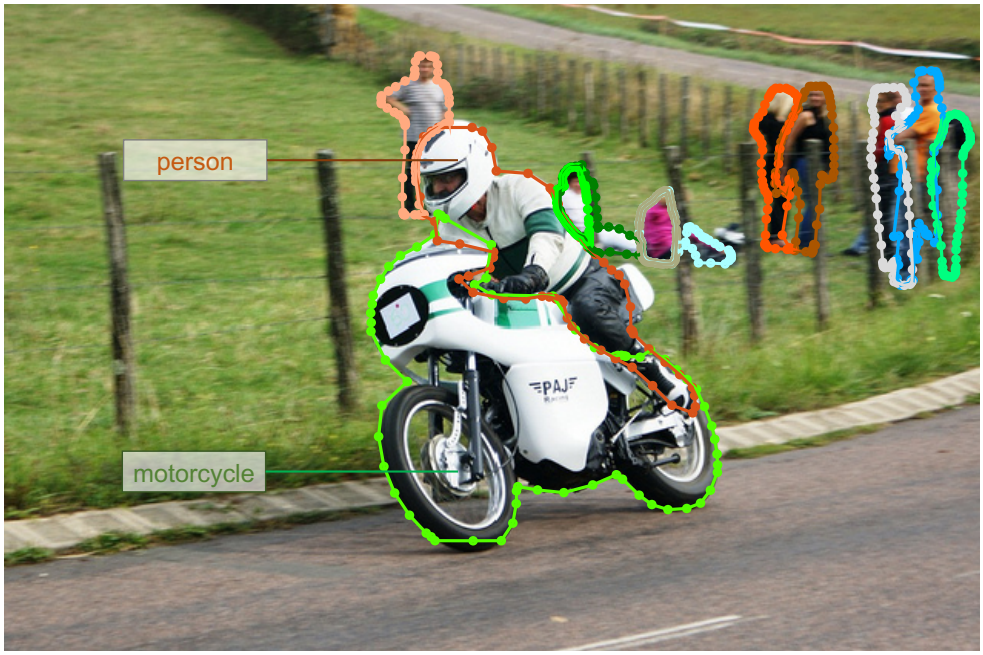


Figure 5.1: We present a model of instance segmentation which predicts the *boundaries* of each object in the form of a polygon. This treats instance segmentation as a regression problem and allows for end-to-end differentiability of predictions. At the same time, our model requires no sacrifices in terms of resulting segmentation quality nor introduces additional supervision requirements compared algorithms which directly predict masks.

5.1 Introduction

Image segmentation [MFTM01] and scene labeling [SWRC06, Tu08] are amongst the most studied topics in computer vision. In addition to pixel-wise masks, representing segments/objects using vectorized boundary representations has applications and significance in many downstream tasks such as shape recognition [BMP02], tracking [BI12], image understanding [TCYZ05], medical imaging [PXP00], and 3D reconstruction [CLP10].

The recently emerged instance segmentation task (objects or areas of interest) [HGDG17] greatly propels the practical significance for object segmentation. Unlike detection, instance segmentation predicts the detailed extent of an object rather than a coarse bounding box. However, while a bounding box can be represented by only two coordinate pairs and is therefore easily turned into a regression problem, there are difficulties when deciding how to best represent and

predict the segmentation of an object. These difficulties combined with a historical preference for convolutional operations has led the computer vision community to become *mask-centric*. This entails almost all segmentation models relying on a spatially dense function which outputs a binary confidence to determine whether each pixel belongs to a particular object. This is in contrast to a *boundary-centric* notion where a sparse set of structured points are predicted to denote the boundary of the object in question. Polygons are one natural choice for this structure. However, because of some inherent difficulties along with mask-centric biases, polygons have large hurdles to overcome. First, there is not an immediately obvious metric (and thus loss) for polygons. Second, the training regimes of mask-based models often relies on operations and augmentations that are *inherently difficult* to robustly and efficiently implement on polygons directly. This includes even basic operations like cropping and intersection. Finally, *evaluation* of segmentation quality is performed with respect to masks and not contours which leads to a possible mismatch in training and testing objectives when predicting polygons.

Despite these obstacles, past efforts to predict polygons within instance segmentation have been made. Nevertheless, none have been able to *achieve parity* with baseline mask-based segmentation models across commonly used benchmarks. By *parity*, we mean a few things:
1). **Parity in supervision:** The method should require no additional sources of supervision than its mask-based counterpart. In particular, the method *should not* rely on polygons directly as a source of supervision. This is crucial since polygons are not always available and deriving polygons from a mask-based ground truth can introduce suboptimal performance or uncertainty – see Table 5.4.
2). **Parity in evaluation:** While polygonal boundary annotations do exist in some instance segmentation datasets, *e.g.*[LMB+14], further technical barriers await when using polygon predictions for evaluation against the ground-truth directly. This is due to the vectorized polygon representation not being unique and there existing a one-to-many representation from masks to polygons. In other words, two similar masks may have polygons that have large differences in the

control points, creating a mismatch between training loss and evaluation.

Finally, we also want to deal with 3). **Parity in access**. While the predictive model itself might deviate in terms of architecture, the model should be generally considered to be a "drop-in" replacement for a mask-based segmentation head. This includes working within one and two-stage architectures, *e.g.*, with respect to either full image or RoI features.

While certain works have touched upon aspects of our model [GSW19, LHM$^+$20], none has yet to provide a fully polygon based solution that is accessible to a myriad of architectures and matches performance of mask-based architectures on standard datasets. We believe providing a clearer picture into the capabilities of polygons in providing a performant and end-to-end differentiable segmentation pipeline could be helpful to further development within the field. We outline our contributions below:

1. In this paper, we present a new instance segmentation method, BoundaryFormer, a Transformer based approach for predicting **an object's boundary as a polygon directly**. Our model outperforms a strong baseline Mask R-CNN [HGDG17] on the MS-COCO and Cityscapes [LMB$^+$14, COR$^+$16b] datasets (both from scratch and when transferring from a COCO-based initialization). To the best of our knowledge, this is the first time a method with polygonal outputs has matched or exceeded Mask R-CNN on the MS-COCO dataset. Furthermore, BoundaryFormer does so without compromising its ability to be trained end-to-end.

2. BoundaryFormer uses pixel-wise masks as ground-truth for supervision and evaluation by utilizing a novel **differentiable rasterization** module. Therefore, BoundaryFormer *adds no new supervision* requirements over Mask R-CNN [HGDG17].

3. By only relying on masks as a source of supervision, our model can be placed as a **drop-in** replacement for the mask-based segmentation head of R-CNN [HGDG17] as either a full image-based component or an RoI-based component. Furthermore, it can be adopted in

other common architectures including FCOS [TSCH19].

5.2 Related Work

We highlight past work which has predicted polygonal (or contour-based) segmentations and make special note of those which have adopted rasterization as a form of supervision for their models.

5.2.1 Point-based Losses

We first discuss contour-based segmentation algorithms which rely primarily on a matching between predicted points and ground-truth points sampled from a known polygon. Therefore, these methods *require* access to some polygonal ground-truth. As one of the first works built on modern architectures, DeepSnake [PJP+20] considers an initial octagonal polygon derived from four extreme points, after which an iterative process of refinement is carried out using circular convolutions. The resulting refinements are supervised using an L_1 loss against a uniform ground-truth polygon. On the other hand, PolarMask [XSS+20] models polygons in a polar representation along with an approximation to IoU as supervisory signal.

Applying a direct distance loss in a Transformer-based architecture for line segmentation detection [XXCT21] and pose recognition [LWZ+21] exits, but they have the direct ground-truth supervision on the points.

PolyTransform [LHM+20] uses an off the shelf mask-based segmentation pipeline to predict an initial binary mask of an object from which highly accurate initial polygons(s) can be derived using an non-differentiable border following algorithm. These polygons are then deformed using Transformers [VSP+17] after which a point-based loss is used for supervisory signal.

Building off of DeepSnake, DANCE [LLCF21] considers an FCOS [TSCH19] detector

to produce an initial box proposal. Points are uniformly sampled along this box to produce an initial polygon which can be deformed using an attention-like process. This initial box contour allows for a novel ground-truth matching which can be more easily optimized. In addition, the attentive process plus predicted edge maps (which relies on access to panoptic annotations) improve performance.

5.2.2 Mask-based Losses

We next consider contour-based models which entirely or in-part rely on rasterized forms of predicted polygons as supervision. ACDRNet [GSW19] provides a system which takes crops of objects of interest as input and iteratively deforms an initial contour by predicting a dense heatmap of offsets in feature-space. A 3D neural renderer [KUH18] is applied to a triangulation of the predictions against the ground-truth masks using an MSE loss. In addition, ACDRNet relies on two additional losses: a balloon like loss to force expansion of the contour and a curvature term. While an interesting proof of concept, the authors do not consider integration into an actual detection pipeline and performance on the standard MS-COCO [LMB$^+$14] is not considered.

CurveGCN [LGK$^+$19] predicts polygonal boundaries through a graph convolutional neural network from an initial contour proposal. For the bulk of training, they use an ordinary Chamfer loss, however, they do note that fine-tuning their model with respect to a differentiable accuracy metric, i.e. triangulating the polygon into a mesh and supervising with respect to masks using a differentiable renderer [LB14] leads to improved results.

Lastly, some recent work[CWHL20] considers using boundary information as an additional supervision for instance segmentation, however, they still predict masks directly and not polygons.

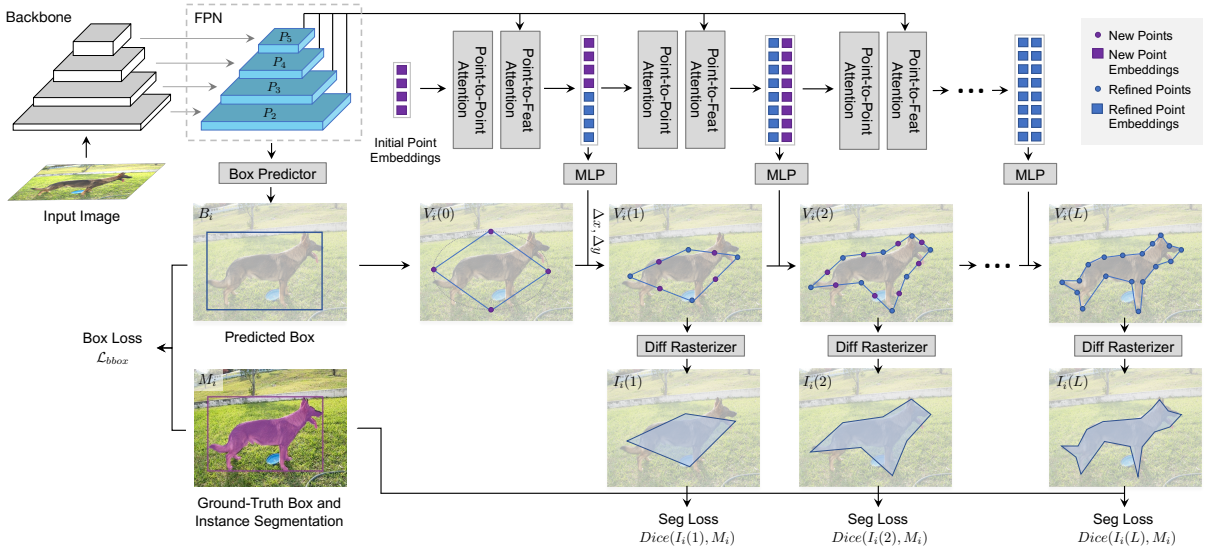


Figure 5.2: **Illustration of BoundaryFormer:** Given an image, multi-scale features are collected and boxes for each object are proposed by the underlying detector. We initialize a proposed polygon for each box using a simple ellipse. At each layer, attention between points within the polygon as well as attention from points to the image features is performed in order to refine the polygon. New points are inserted between existing ones in order to efficiently produce higher resolution polygons as the refinement progresses. After each refinement, the predicted polygon is rasterized in a differentiable manner and compared to the ground-truth mask using a mask-based loss.

5.3 Method

5.3.1 Setting

Instance segmentation considers an input image $I \in \mathbb{R}^{H \times W \times 3}$ and is tasked with producing N ordered instances O_i, $1 \leq i \leq N$. Most modern benchmarks consider instances as a tuple (M_i, c_i) where M_i is a per-pixel binary mask denoting membership by the object and where $c_i \in \{1, \ldots, C\}$ is the predicted class of the object.

Since M_i is a binary mask, most segmentation models find it natural to predict some downsampled version of M_i. M_i is predicted as a discrete grid of continuous confidence values $M_i(x,y)$ with $x \in \{1, \ldots, X'\}$, $y \in \{1, \ldots, Y'\}$ for a mask size of $X' \times Y'$. This phrases mask prediction as a *classification problem*. At inference time, M_i is transformed into a binary mask with an ordinary decision rule (usually $M_i(x,y) > 0.5$). Therefore, without approximations or relaxations, M_i is *not differentiable* for downstream components relying on $M_i(x,y)$.

In this work, we phrase the prediction of M_i rather as a *regression problem*. This decomposes the prediction of M_i into two parts. First, we predict K vertices, denoted as $V_i = \{(x_0^i, y_0^i), \ldots, (x_{K-1}^i, y_{K-1}^i\}$ which define the boundary of a polygon in 2D under a fixed ordering. Then, we produce M_i through rasterization to the desired mask size $X' \times Y'$. M_i must be produced because instance segmentation relies on masks for evaluation. This leads to the following choices: how should V_i be predicted and more importantly, how should V_i be supervised with respect to the ground truth. We present our approach to this problem as **BoundaryFormer**.

5.3.2 Instance Segmentation with BoundaryFormer

We design BoundaryFormer as a component that can be added on to existing detection-based frameworks. While we believe BoundaryFormer is generally applicable, we consider a more concrete setup for the sake of presentation. In particular, we assume a detector built upon a standard FPN architecture (this includes FCOS [TSCH19] or R-CNN [HGDG17]) which

produces feature maps $F = \{P_2, \ldots, P_5\}$ in decreasing resolution from the image I. From these features, the detector proposes N objects in the form of boxes $B_i = (l_i, t_i, w_i, h_i)$ which denote the left edge, top edge, width, and height of the box respectively. Each B_i corresponds to some ground-truth mask M_i where M_i is expected to be clipped to B_i. We use B_i as a means to initialize an ellipsoidal polygon $V_i(0)$ inscribed in B_i, similar to other contour-based methods [PJP+20, LHM+20, LLCF21]. From $V_i(0)$, our model iteratively refines this shape L times by $V_i(j+1) = g_j(F, V_i(j))$ to produce a final prediction $V_i(L)$.

We visualize a concrete implementation in Figure 5.2. Since we are dealing with point sets, we implement g with Transformers [VSP+17] using two kinds of attention. Each vertex within the polygon V_i corresponds to some point embedding P_i^k where $1 \leq k \leq K$ and includes a "point encoding" modeled off of the usual Transformer sine positional encoding. The first kind of attention allows each P_i^k to attend to all other point embeddings $P_i^{k'}$ within the same object. The second allows each P_i^k to attend to the image features F. While the first kind of attention is implemented using ordinary (quadratic) self-attention, we implement the point to image feature attention using Deformable Attention [ZSL+20] which significantly reduces the computational cost. Furthermore, this allows multi-scale *across* levels P_2 through P_5. Finally, for each embedding P_i^k, g_j predicts a 2D offset $(\Delta x, \Delta y)$ to refine the vertex using an MLP.

5.3.3 Mask-based Supervision

Given the predictive model, we expect a final output polygon V_i consisting of K points for each proposed box B_i from the underlying detector. However, in order to train such a model, we must provide a means of supervision. Most previous work has relied upon point-based matching losses where some scheme to assign a predicted point to a point in the *polygonal ground-truth* is devised. These might include: the Chamfer distance [LHM+20], permutation-based matching [LGK+19], and assignment rules that depend on the initial contour [LLCF21]. However, we believe these approaches are undesirable and unnecessary. First, they require access to ground-

truth polygons which are not always available and attempting to derive polygonal contours from masks is a noisy process (Table 5.4). These ground-truth polygons might need to be sampled (up or down) against heuristics in order to satisfy the requirements of point-based losses. Second, we *evaluate* instance segmentation quality with respect to *masks* (i.e. COCO mask AP [LMB+14]), and therefore it's not guaranteed that these metrics are optimizing the desired metric. Lastly, many essential operations *are nontrivial* to implement on polygons directly. This includes: clipping to a box (required for RoI-like operations), intersection, and even union. However, the mask-based counterparts are simple, highly optimized, and differentiable in existing frameworks.

Therefore, we require our model to only require mask-based supervision and furthermore, to require it in the exact same sense as an ordinary mask-based segmentation model. We transparently handle this transformation from polygon space to mask space by the usage of a differentiable 2D rasterizer. Because of the dynamic nature of rasterization, we are afforded more flexibility than a purely mask-based model. We now describe specific details of the rasterization process.

A polygon-specific rasterizer

While previous work [GSW19, LGK+19] has used 3D renderers (leaving the depth component constant) to produce masks from polygons in a differentiable manner, these methods require the additional step of triangulation. We find this to be unnecessary, especially accounting for the time necessary for triangulation and additional choice of triangulation method required. Rather, we use [LLCL19] as inspiration to design a rasterizer that operates on polygons directly rather than triangulated meshes.

The pipeline of the proposed differentiable rasterizer is shown in Figure 5.3. Consider a polygon with vertices V and a desired rasterization pixel size of $X' \times Y'$. For each pixel (x,y) where $0 \leq x < X', 0 \leq y < Y'$, we use a PnP algorithm [wrf] to determine which whether the pixel at position (x,y) lies within the polygon as $C(V,x,y)$, where $C = 1$ if it lies within and -1 if it

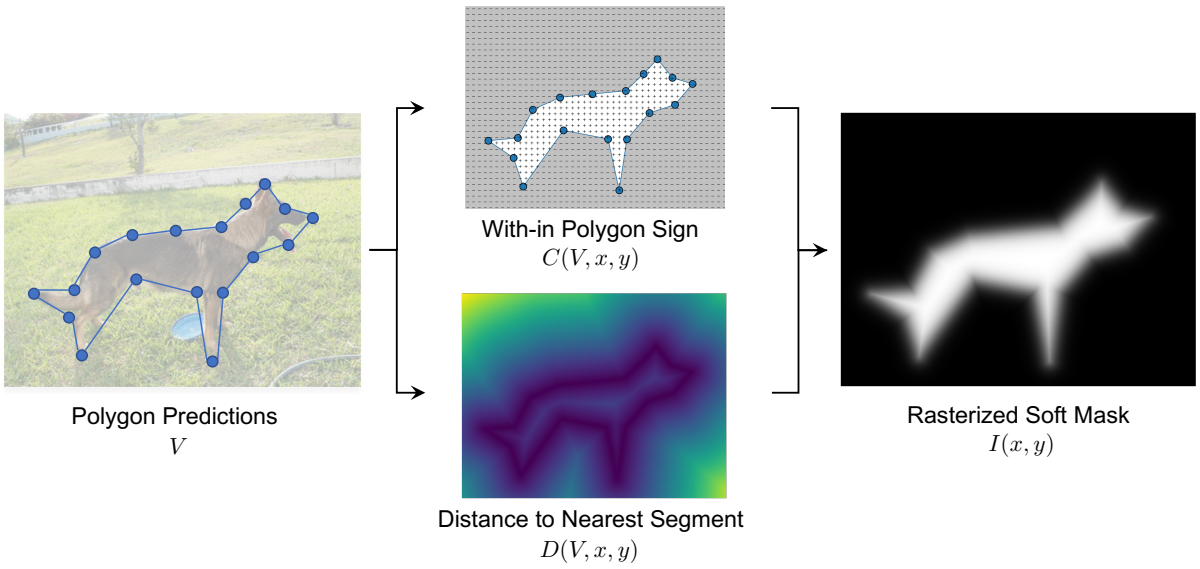


Figure 5.3: **Illustration of differentiable rasterizer:** We illustrate the transformation from polygonal point predictions to a differentiably rasterized mask: pixels are given signs based on being within the polygon or not, projections onto the nearest segment are computed, and Equation 5.1 determines the final rasterized value.

does not. Then, each pixel (x,y) is projected onto the closest segment on the polygon's boundary and this distance is recorded as $D(V,x,y)$. Finally, following [LLCL19], we model each pixel's contribution to the rasterized image as a sigmoidal function (with some sharpness τ) according to the associated distance:

$$I(x,y) = \sigma\left(\frac{C(V,x,y) * D(V,x,y)}{\tau}\right) \tag{5.1}$$

Therefore, this rasterizer provides signal solely from the boundary without the need to consider derived mesh points.

The rasterizer and backwards pass are implemented entirely in CUDA. This affords us efficiency to train solely with the differentiable rasterizer for the entirety of training. We find the Dice [MNA16] loss to be critical to the success of training with rasterized masks, although we find similar losses, *e.g.*, Lovasz-Softmax [BTB18] to perform equally well.

Alignment

Alignment with ground-truth: We emphasize that we use the exact same ground-truth masks (although possibly at differing resolutions) as an ordinary R-CNN pipeline. Thus, it is imperative that we ensure our differentiable rasterization is "aligned" to the method of rasterization built into the COCO API [LMB+14]. Due to differences in what is considered a "pixel", we ensure that we subtract a half pixel on all coordinates before passing them to our rasterizer. This significantly improves alignment and thus performance, *e.g.*, mask AP drops from 36.1 AP to 35.3 AP when not aligned.

Alignment with architecture: When building off of existing architectures, *e.g.*, FPN [LDG+17a], operations often have a bias towards alignment with convolutions. We find that when we attempt to use BoundaryFormer in an RoI-less setting, *i.e.*, directly attending to the full image-based features, it is essential to perform a global pooling-like operation to significantly increase performance — without this, mask AP drops from 36.1 AP to 34.2 AP. We hypothesize this might be due to aliasing in the deconvolutional process of the FPN.

5.3.4 Additional details

We describe specific implementation details that we find essential to satisfactory performance and speed.

Coarse to fine upsampling

By having each point dedicated to a single point query embedding, we afford a great deal of flexibility to our model. However, since often we might have a large number O of objects or object proposals and K control points for each object, this can require computation on the order of $O * K * L$ if we include L layers. At the same time, it's unlikely that we truly need K vertices until the prediction is itself more accurate. Therefore, we consider a base number of control points B (usually 8) and upsample the points by $2\times$ at each layer. Specifically, given $V_i(l)$ consisting

of $K(l)$ points, we first apply $g_l(G, V_i(l))$ to get refinements to produce $V_i(l+1)$ still consisting of $K(l)$ points. Then, for between each point (x_j, y_j) and (x_{j+1}, y_{j+1}), we insert a new point at the midpoint $\left(\frac{x_j+x_{j+1}}{2}, \frac{y_j+y_{j+1}}{2}\right)$. We do not average the corresponding point embeddings P_j and P_{j+1} to initialize the new point and instead insert the corresponding "learned" point embedding . Overall, this speeds up training and memory consumption by about $1.5\times$ versus using the same number of points at each layer.

Deep supervision

Like many other Transformer-based models [CMS+20], we find that deep supervision [LXG+15, XT15] of layer-wise outputs of the Transformer essential. Combined with a coarse to fine approach, this implies that each layer is supervised with respect to the same ground-truth mask (without loss of detail). However, each layer, by virtue of having more and more points can model higher resolution features. This is in contrast to point-based approaches which must heuristically downsample the ground-truth polygons in order to apply a point-based matching.

Loss

We finally present the loss of our polygon-based boundary prediction model trained jointly with a box-based detector. Suppose the detector is trained against $\mathcal{L}_{bbox}$. We denote the subset of foreground-matched boxes as $\{B_i \mid 1 \leq i \leq R\}$ and associate ground-truth mask M_i of resolution $X' \times Y'$ to each. If our predictive model of polygons consists of L layers, then we have dense outputs corresponding to $V_i(l)$ with $1 \leq l \leq L$. Then, using $I_i(l)$ to denote the (differentiably) rasterized version of $V_i(l)$, we define a total loss:

$$\mathcal{L} = \mathcal{L}_{bbox} + \sum_{l}^{L} \sum_{i=1}^{R} Dice\left(I_i(l), M_i\right) \tag{5.2}$$

This loss provides *parity in supervision* since like a standard instance segmentation model, it relies only on access to ground-truth masks M_i.

5.4 Experiments

We evaluate the *parity in performance* of BoundaryFormer across the commonly used MS-COCO dataset [LMB+14] and Cityscapes dataset [COR+16b]. We additionally provide experiments for transferring models from MS-COCO to Cityscapes to illustrate possible differences based on the underlying representation learned. We focus on comparisons with Mask R-CNN [HGDG17] as the standard baseline for instance segmentation, although, we calibrate our results with other contour-based and masked-based models. Furthermore, we consider inclusion of BoundaryFormer in both single stage, FCOS [TSCH19], and two stage approaches, Faster R-CNN [RHGS15].

5.4.1 Training Details

All models are trained in an end-to-end fashion with respect to the entire network. We aim to keep underlying backbones and architectures as close as possible to Mask R-CNN [HGDG17] to compare as fairly as possible. Therefore, unless specified, we use an ResNet50-FPN backbone for all models which follows the exact same settings as Mask-RCNN in Detectron2 [WKM+19] with only the "mask head" of Mask R-CNN replaced with the architecture outlined in Figure 5.2. While Mask R-CNN *requires* RoI pooling to predict masks, this feature is optional in our approach and frees us from constraints that such a grid-based approach imposes. Rather, we present results *without* the need of RoI pooling such that the point to image features attention is with respect to the entirety of $\{P_2, \ldots, P_5\}$. One notable exception to standard training is that instead of SGD, we train all models with the Adam [KB14] optimizer according to the settings generally outlined in Swin [LLC+21] due to our use of Transformers. We find it is necessary increase weight decay on larger models (*e.g.*, ResNet-101), longer training schedules, and smaller datasets (*e.g.*, Cityscapes) in order to combat overfitting (Swin instead uses an increased dropout rate).

For details specific to BoundaryFormer, we train all models with coarse to fine upsampling

over 4 layers of Transformers [VSP+17] to produce a final output of 64 points for experiments on COCO and 128 points on Cityscapes. All other parameters with respect to deformable attention follow the same settings of Deformable DETR's [ZSL+20] decoder layer. The MLP that produces refinements for each point is applied independently and consists of 3 layers.

For rasterization, we find that $\tau = 0.005$ (denoting rasterization smoothness) generally performs well. Additionally, we rasterize polygons during training to a fixed $X' \times Y' = 64 \times 64$ resolution. When performing rasterization, we *only rasterize the predicted polygon within its box*, not with respect to the entire image. This emulates the behavior of RoI pooling. Like Mask R-CNN [HGDG17], the ground-truth mask is clipped to this box. All results are evaluated with respect to the standard COCO mask AP [LMB+14].

Inference

At inference time, we follow the same inference procedures as the underlying detector. We rely on rasterization from the COCO API directly [LMB+14] rather than our own rasterizer since differentiability is not required.

5.4.2 COCO

COCO [LMB+14] is a large-scale dataset containing 118K training images of natural scenes with 80 annotated foreground classes. It has historically been a relatively difficult dataset for contour or boundary-based models to achieve parity in performance with. As far as we can tell, our model *is the first to achieve parity in mask quality to Mask R-CNN on COCO*.

We detail these comparisons in Tables 5.1 and 5.2. We note that BoundaryFormer achieves a slight edge in mask quality over Mask R-CNN and is competitive with the boundary-preserving variant [CWHL20] while significantly outperforming the best contour-based method [LLCF21]. At the same time, we observe that both models attain the same box performance — indicating that both tasks (mask or polygon prediction) provide multi-task training benefits to the underlying

Table 5.1: **Results on MS-COCO val:** We compare BoundaryFormer to the state of the art in contour/boundary prediction [LLCF21] as well as mask-based counterparts. * indicates re-trained with Adam [KB14].

Method	Backbone	Detector	AP	AP_{50}	AP_{bbox}
Mask R-CNN [HGDG17]	R50-FPN	R-CNN	35.2	56.3	38.6
Mask R-CNN* [HGDG17]	R50-FPN	R-CNN	35.8	56.8	38.8
BMask R-CNN [CWHL20]	R50-FPN	R-CNN	36.6	56.7	39.4
BMask R-CNN* [CWHL20]	R50-FPN	R-CNN	36.4	56.3	37.8
DANCE [LLCF21]	R50-FPN	FCOS	34.5	55.3	40.2
BoundaryFormer (ours)	R50-FPN	FCOS	35.8	55.7	40.2
BoundaryFormer (ours)	R50-FPN	R-CNN	36.1	56.7	38.8

Table 5.2: **Results on MS-COCO test-dev:** We evaluate and compare BoundaryFormer on the MS-COCO test-dev set with larger backbones and longer training schedules, showing that its performance scales as expected while staying competitive with strong mask-based baselines.

Method	Backbone	Detector	Schedule	Output	Supervision	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Mask R-CNN [HGDG17]	R50-FPN	R-CNN	1×	masks	masks	36.1	57.3	38.7	19.7	38.0	47.1
Mask R-CNN [HGDG17]	R101-FPN	R-CNN	3×	masks	masks	39.2	61.1	42.3	22.4	41.4	50.7
BMask R-CNN [CWHL20]	R50-FPN	R-CNN	1×	masks	masks	36.6	57.0	39.6	19.4	38.7	48.0
BMask R-CNN [CWHL20]	R101-FPN	R-CNN	1×	masks	masks	38.3	59.2	41.3	20.3	40.8	50.2
DeepSnake [PJP+20]	R50-FPN	CenterNet	-	polygons	polygons	30.5	-	-	-	-	-
PolarMask [PJP+20]	R101-FPN	FCOS	2×	polygons	polygons	32.1	53.7	33.1	14.7	33.8	45.3
DANCE [LLCF21]	R50-FPN	FCOS	1×	polygons	polygons + panoptic	34.6	55.9	36.4	19.3	37.2	43.9
DANCE [LLCF21]	R101-FPN	FCOS	3×	polygons	polygons + panoptic	38.1	60.2	40.5	21.5	40.7	48.8
BoundaryFormer (ours)	R50-FPN	R-CNN	1×	polygons	masks	36.4	57.2	39.0	19.6	38.6	47.9
BoundaryFormer (ours)	R101-FPN	R-CNN	1×	polygons	masks	37.7	58.8	40.5	20.4	40.2	49.0
BoundaryFormer (ours)	R101-FPN	R-CNN	3×	polygons	masks	39.4	60.9	42.6	22.1	42.0	51.2

detector when trained end-to-end. While we find the best results using R-CNN as an underlying architecture, BoundaryFormer can still match Mask R-CNN when using FCOS. Performance appears to be slightly worse despite performing better with respect to box AP. We hypothesize this might be due to FCOS using only P3-P7 within its FPN whereas R-CNN includes P2 and thus might learn enhanced features relevant to boundary prediction.

5.4.3 Cityscapes

Cityscapes [COR+16b] consists of a diverse set of street scenes across European cities. The dataset is relatively small compared to COCO — consisting of only 2975 training images, however, each image is of a high resolution and annotation quality. Furthermore, it is notable for its large amount of occlusion [HGDG17] which causes a significant number of instances to be

Table 5.3: **Results on Cityscapes val:** We establish that BoundaryFormer can maintain parity with Mask R-CNN even on the difficult Cityscapes dataset. Additionally, when using a model initialized from COCO, BoundaryFormer shows improved transfer ability over Mask R-CNN and is competitive with PolyTransform, which requires UPSNet [XLZ+19] and the use of a mask head to initialize its contours. Lastly, BoundaryFormer is the only model that both relies solely on ground-truth masks for supervision and is end-to-end differentiable. * indicates re-trained with Adam [KB14].

Method	model init	poly init	e2e	sup	AP	AP_{50}
Mask R-CNN* [HGDG17]	ImageNet	N/A	-	masks	34.2	60.7
Mask R-CNN* [HGDG17]	COCO	N/A	-	masks	36.5	62.0
DeepSnake [PJP+20]	ImageNet	extreme pts	✓	poly	28.2	-
UPSNet [LHM+20]	COCO	pred masks	-	masks	37.8	-
UPSNet + PolyTransform [LHM+20]	COCO	pred masks	-	both	40.2	-
BoundaryFormer (64 points)	ImageNet	ellipse	✓	masks	34.7	60.8
BoundaryFormer (64 points)	COCO	ellipse	✓	masks	37.6	62.8
BoundaryFormer (128 points)	COCO	ellipse	✓	masks	38.3	62.9

fragmented into multiple simple polygons. This creates problems for boundary prediction which usually only consists of predicting and matching to a single ground-truth polygon. While we compare to other methods which include mitigations (*e.g.*, initialization from masks [LHM+20]), we emphasize that BoundaryFormer *uses no special handling* and does not deviate from the standard training process.

We highlight the performance of our method with respect to both ImageNet and COCO initializations. COCO initializations are made using a model trained under the standard 1x training schedule, i.e., the models in Table 5.1. Mask R-CNN [HGDG17] and BoundaryFormer results are the average of three runs to account for well-known instabilities in Cityscapes training.

We observe that BoundaryFormer is still able to achieve parity with Mask R-CNN despite being at a significant disadvantage by predicting only a single polygon for fragmented objects. Additionally, when initializing BoundaryFormer from COCO, it significantly exceeds Mask R-CNN in final performance which might indicate a polygonal representation transfers well.

5.4.4 Ablation studies

We provide ablations for the hyperparameters of both the supervision (the rasterization process) and some model-specific hyperparameters/justifications.

Polygonal ground-truth from masks? While we promote the usage of masks directly as supervision of our model for both performance and flexibility, contour models that require access to polygonal ground-truth could resort to generating contours from masks themselves when masks are the only available annotation source. Therefore, we re-generate the COCO training dataset (where polygonal ground-truth *is* available) and replace the existing segmentations with those generated using a border following algorithm in order to retrain DANCE [LLCF21]. Details are discussed in the supplementary materials. In Table 5.4, we observe sharp decreases in resulting performance as measured by the unmodified MS-COCO *val* set — indicating that generating polygons from masks can introduce errors. While it's plausible this could be improved, we believe it acts as an unnecessary barrier in producing an accurate, certain model.

Table 5.4: **Comparison in training a point-based supervised model using verbatim polygons from annotators or those generated using standard algorithms from masks.**

DANCE [LLCF21]	Polygons (verbatim)	Polygons (from masks)
Mask AP	34.5	23.1

Rasterization smoothness (τ): The rasterization smoothness dictates the (inverse) steepness of the sigmoidal function in Equation 5.1. While it must be sufficiently large to allow good gradient flow, it should also be small enough to accurately reflect what will be predicted at inference time, *i.e.*, the *hard* rasterization of the predicted polygons. We find that lower values of τ are required (i.e. sharper), however, within that lower range, the performance is generally robust with values around $\tau = 0.005$ to be sufficient. Larger values, *e.g.*, $\tau = 0.15$ lack sufficient sharpness for optimal results.

τ	0.15	0.05	0.005	0.0005
Mask AP	35.6	35.7	36.1	36.1

Figure 5.4: **BoundaryFormer is able to predict high quality instance segmentations over the MS-COCO dataset:** Predicted boundary points are shown in black. The resulting rasterized mask is overlayed onto the instance in color.

Rasterization resolution ($X' \times Y'$): Like Mask R-CNN, each instance is supervised at a fixed resolution with respect to some mask loss. We investigate the impact of the choice of resolution on resulting performance. In our experiments, we find that 64×64 performs well with performance saturating at larger resolutions. However, this might be a consequence of the lower quality annotations in COCO and these characteristics might change on higher quality annotations, *e.g.*, LVIS [GDG19].

$X' \times Y'$	48×48	64×64	80×80
Mask AP	35.8	36.1	36.1

Performance across layers (L): We find that marginal performance increases appears to diminish over the course of four refinement layers, consistent with the need for iterative refinement.

	Layer 1	Layer 2	Layer 3	Layer 4
Mask AP	29.0	33.6	35.4	36.1

Since adding more layers in our coarse-to-fine setting *doubles* the total number of points, we find

four layers to be a good tradeoff in performance and computational needs.

Number of control points (K): Given our coarse-to-fine setting, each layer doubles the number of points. We consider three initial point settings (4, 8, and 16) and models consisting of $L = 4$ layers, predicting final polygons with 32, 64, and 128 points, respectively. We find that a larger number of points is critical, suffering close to a 1 point drop in Mask AP when using less than 64 points. While performance mostly saturates at 64 points, we note that larger gains can be observed on Cityscapes (see Table 5.3). Lastly, we analyze whether the coarse-to-fine strategy sacrifices any performance and find it is equally performant to using 64 points at each layer.

$K_1/K_2/K_3/K_4$	4/8/16/32	8/16/32/64	16/32/64/128	64/64/64/64
Mask AP	35.2	36.1	36.2	36.1

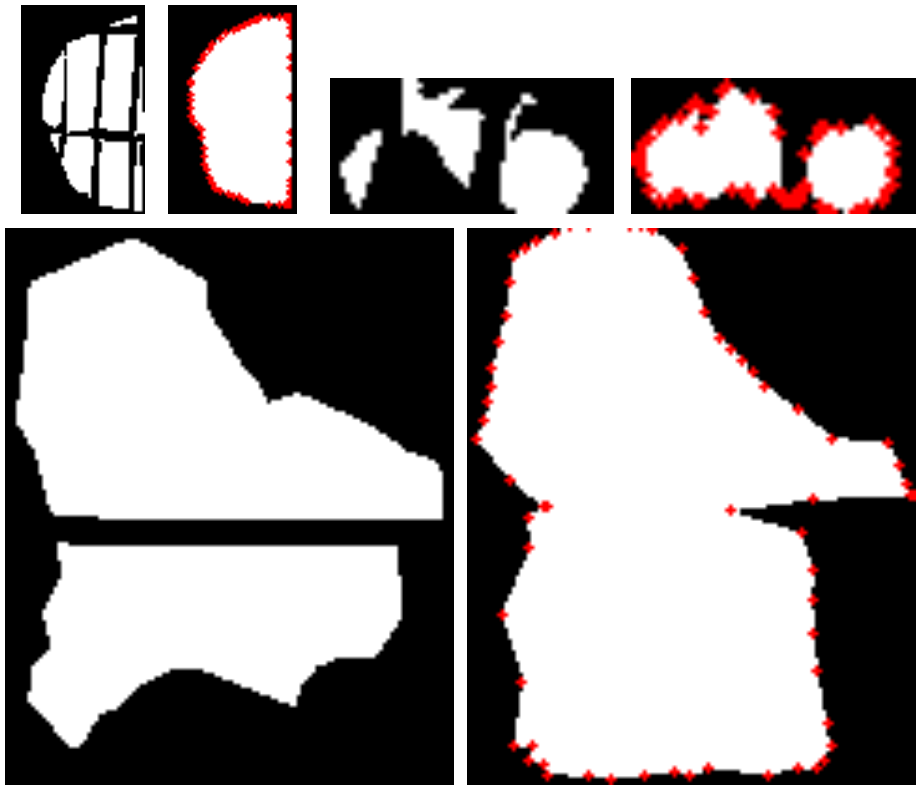
5.5 Qualitative Analysis

We briefly investigate the general qualitative quality of predicted polygons in Table 5.4. Furthermore, we observe some robustness to fragmeneted objects in Figure 5.5b, which we hypothesize can be implicitly learned due to supervision only in the form of masks. Finally, we exhibit the approximations our coarse-to-fine model learns in Figure 5.5a and improvements in quality it makes as more and more points become available.

5.6 Conclusion

We presented BoundaryFormer, a simple baseline for regressing instance segmentations as polygonal boundaries rather than predicting dense masks. Despite regressing polygons, this model relies on supervision *only through masks*. Combined with a strong point-based architecture and supervision in pixel space, BoundaryFormer can match and outperform mask-based counterparts across a variety of datasets. As a result, we believe that many tasks that have historically been

(a)



(b)

Figure 5.5: **Progressive refinement and fragment robustness (a)** illustrates the progressive improvement in boundary quality over layers. **(b)** shows some implicitly learned robustness to fragmented objects.

mask-based, *e.g.*, semantic and panoptic segmentation can now be revisited. Furthermore, we hope downstream tasks that have relied on non-differentiable mask predictions can now consider using the sparse representation of a polygon and the end-to-end differentiability it provides. Finally, we believe further research can alleviate *some limitation*s: predicting fragmented objects faithfully, incorporating additional mask-based advancements, and finding more efficient architectures. How best to *scale* BoundaryFormer to higher fidelities remains an open question.

This chapter is based on material submitted for publication of material by Justin Lazarow, Weijian Xu, Zhuowen Tu. This dissertation author was the primary investigator and author of this paper.

Bibliography

[ACB17] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[AT17] Anurag Arnab and Philip HS Torr. Pixelwise instance segmentation with a dynamically instantiated network. In *CVPR*, pages 441–450, 2017.

[Bal12] Pierre Baldi. Autoencoders, unsupervised learning, and deep architectures. *ICML unsupervised and transfer learning*, 27, 2012.

[BDS18] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

[BI12] Andrew Blake and Michael Isard. *Active contours: the application of techniques from graphics, vision, control theory and statistics to visual tracking of shapes in motion*. Springer Science & Business Media, 2012.

[Bie87] Irving Biederman. Recognition-by-components: a theory of human image understanding. *Psychological review*, 94(2):115, 1987.

[Bil06] Stanley M Bileschi. Streetscenes: Towards scene understanding in still images. Technical report, MASSACHUSETTS INST OF TECH CAMBRIDGE, 2006.

[BL07] Yoshua Bengio and Yann LeCun. Scaling learning algorithms towards ai. *Large-scale kernel machines*, 34(5), 2007.

[BLRW16] Andrew Brock, Theodore Lim, JM Ritchie, and Nick Weston. Neural photo editing with introspective adversarial networks. *arXiv preprint arXiv:1609.07093*, 2016.

[BMP02] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *IEEE transactions on pattern analysis and machine intelligence*, 24(4):509–522, 2002.

[BNJ03] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *J. of machine Learning research*, 3:993–1022, 2003.

[Bre01] Leo Breiman. Random Forests. *Machine Learning*, 2001.

[BTB18] Maxim Berman, Amal Rannen Triki, and Matthew B Blaschko. The lovász-softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4413–4421, 2018.

[CDXH13] Guang Chen, Yuanyuan Ding, Jing Xiao, and Tony X Han. Detection evolution with multi-order contextual co-occurrence. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1798–1805, 2013.

[CFG14] Tianqi Chen, Emily B Fox, and Carlos Guestrin. Stochastic gradient hamiltonian monte carlo. In *ICML*, 2014.

[CLP10] Anne-Laure Chauve, Patrick Labatut, and Jean-Philippe Pons. Robust piecewise-planar 3d reconstruction and completion from large-scale unstructured point data. pages 1261–1268, 2010.

[CLY15] Yi-Ting Chen, Xiaokai Liu, and Ming-Hsuan Yang. Multi-instance object segmentation with occlusion handling. In *CVPR*, 2015.

[CMS+20] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*, pages 213–229. Springer, 2020.

[COR+16a] 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 *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[COR+16b] 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.

[CPK+18] 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, 2018.

[CSK21] 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, 2021.

[CV19] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: High quality object detection and instance segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[CWHL20] Tianheng Cheng, Xinggang Wang, Lichao Huang, and Wenyu Liu. Boundary-preserving mask r-cnn. pages 660–676, 2020.

[dGMD18] Daan de Geus, Panagiotis Meletis, and Gijs Dubbelman. Panoptic segmentation with a joint semantic and instance segmentation network. *arXiv preprint arXiv:1809.02110*, 2018.

[DHS00] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. 2nd edition, 2000.

[DHS16] Jifeng Dai, Kaiming He, and Jian Sun. Instance-aware semantic segmentation via multi-task network cascades. In *CVPR*, 2016.

[DKB14] Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

[DM19] Yilun Du and Igor Mordatch. Implicit generation and generalization in energy-based models. *arXiv preprint arXiv:1903.08689*, 2019.

[DN21] Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis. *arXiv preprint arXiv:2105.05233*, 2021.

[DPDPL97] Stephen Della Pietra, Vincent Della Pietra, and John Lafferty. Inducing features of random fields. *IEEE PAMI*, 19(4):380–393, 1997.

[DQX+17] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *ICCV*, 2017.

[DRF11] Chaitanya Desai, Deva Ramanan, and Charless C Fowlkes. Discriminative models for multi-class object layout. *International journal of computer vision*, 95(1):1–12, 2011.

[EESG10] Markus Enzweiler, Angela Eigenstetter, Bernt Schiele, and Dariu M Gavrila. Multi-cue pedestrian classification with partial occlusion handling. In *CVPR*, 2010.

[EL99] Alexei A Efros and Thomas K Leung. Texture synthesis by non-parametric sampling. In *ICCV*, 1999.

[EVGW+10] 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(2):303–338, 2010.

[FHT01] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics, 2001.

[FS97] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. of Comp. and Sys. Sci.*, 55(1), 1997.

[GDG19] Agrim Gupta, Piotr Dollar, and Ross Girshick. Lvis: A dataset for large vocabulary instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5356–5364, 2019.

[GEB15] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. A neural algorithm of artistic style. In *NIPS*, 2015.

[GH10] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *AISTATS*, volume 1, page 6, 2010.

[GH12] Ruiqi Guo and Derek Hoiem. Beyond the line of sight: labeling the underlying surfaces. In *European Conference on Computer Vision*, pages 761–774. Springer, 2012.

[Gir15] Ross Girshick. Fast R-CNN. In *ICCV*, 2015.

[GLA+21] Kamal Gupta, Justin Lazarow, Alessandro Achille, Larry S Davis, Vijay Mahadevan, and Abhinav Shrivastava. Layouttransformer: Layout generation and completion with self-attention. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1004–1014, 2021.

[GPAM+14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014.

[Gre93] Ulf Grenander. *General pattern theory-A mathematical study of regular structures*. Clarendon Press, 1993.

[GSW19] Shir Gur, Tal Shaharabany, and Lior Wolf. End to end trainable active contours via differentiable rendering. *arXiv preprint arXiv:1912.00367*, 2019.

[GWJ+19] Will Grathwohl, Kuan-Chieh Wang, Jörn-Henrik Jacobsen, David Duvenaud, Mohammad Norouzi, and Kevin Swersky. Your classifier is secretly an energy based model and you should treat it like one. *arXiv preprint arXiv:1912.03263*, 2019.

[HB95] David J Heeger and James R Bergen. Pyramid-based texture analysis/synthesis. In *SIGGRAPH*, pages 229–238, 1995.

[HBDS15] Jan Hosang, Rodrigo Benenson, Piotr Dollár, and Bernt Schiele. What makes for effective detection proposals? *IEEE transactions on pattern analysis and machine intelligence*, 38(4):814–830, 2015.

[HBS17] Jan Hosang, Rodrigo Benenson, and Bernt Schiele. Learning non-maximum suppression. In *CVPR*, 2017.

[HDFN95] Geoffrey E Hinton, Peter Dayan, Brendan J Frey, and Radford M Neal. The" wake-sleep" algorithm for unsupervised neural networks. *Science*, 268(5214):1158, 1995.

[HGDG17] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017.

[HHF] Varsha Hedau, Derek Hoiem, and David Forsyth. Recovering the spatial layout of cluttered rooms. In *ICCV*.

[HHG+19] Zhaojin Huang, Lichao Huang, Yongchao Gong, Chang Huang, and Xinggang Wang. Mask scoring r-cnn. In *CVPR*, 2019.

[HOT06] G. E. Hinton, S. Osindero, and Y. W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18:1527–1554, 2006.

[HOWT06] Geoffrey Hinton, Simon Osindero, Max Welling, and Yee-Whye Teh. Unsupervised discovery of nonlinear structure using contrastive backpropagation. *Cognitive science*, 30(4):725–731, 2006.

[HSEH07] Derek Hoiem, Andrew N Stein, Alexei A Efros, and Martial Hebert. Recovering occlusion boundaries from a single image. In *ICCV*, 2007.

[HYCL18] Seunghoon Hong, Dingdong Yang, Jongwook Choi, and Honglak Lee. Inferring semantic layout for hierarchical text-to-image synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7986–7994, 2018.

[IS15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

[IZZE17] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017.

[JCT16] Long Jin, Zeyu Chen, and Zhuowen Tu. Object detection free instance segmentation with labeling transformations. *arXiv preprint arXiv:1611.08991*, 2016.

[JDH+19] Akash Abdu Jyothi, Thibaut Durand, Jiawei He, Leonid Sigal, and Greg Mori. Layoutvae: Stochastic scene layout generation from a label set. *arXiv preprint arXiv:1907.10719*, 2019.

[Jeb12] Tony Jebara. Machine learning: discriminative and generative, 2012.

[JLT17a] Long Jin, Justin Lazarow, and Zhuowen Tu. Introspective classification with convolutional nets. In *Advances in neural information processing systems*, 2017.

[JLT17b] Long Jin, Justin Lazarow, and Zhuowen Tu. Introspective classifier learning: Empower generatively. In *arXiv preprint arXiv:*, 2017.

[Jol02] Ian Jolliffe. *Principal component analysis*. Wiley Online Library, 2002.

[Kan79] Gaetano Kanizsa. *Organization in Vision: Essays on Gestalt Perception*. Praeger Publishers, 1979.

[KB14] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[KD18] Diederik P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *arXiv preprint arXiv:1807.03039*, 2018.

[KGHD19] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In *CVPR*, 2019.

[KHG+19] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, and Piotr Dollár. Panoptic segmentation. In *CVPR*, 2019.

[Kim16] Taehoon Kim. DCGAN-tensorflow. `https://github.com/carpedm20/DCGAN-tensorflow`, 2016.

[KLA19] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4401–4410, 2019.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *NIPS*, 2012.

[KTCM15] Abhishek Kar, Shubham Tulsiani, Joao Carreira, and Jitendra Malik. Amodal completion and size constancy in natural scenes. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 127–135, 2015.

[KUH18] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3d mesh renderer. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[KW13] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[KWHG20] 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.

[LB14] Matthew M Loper and Michael J Black. Opendr: An approximate differentiable renderer. In *European Conference on Computer Vision*, pages 154–169. Springer, 2014.

[LBD+89a] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R.E. Howard, W. Hubbard, and L.D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1989.

[LBD+89b] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R.E. Howard, W. Hubbard, and L.D. Jackel. Backpropagation applied to handwritten zip code recognition. In *Neural Computation*, 1989.

[LCZ+19] Yanwei Li, Xinze Chen, Zheng Zhu, Lingxi Xie, Guan Huang, Dalong Du, and Xingang Wang. Attention-guided unified network for panoptic segmentation. In *CVPR*, 2019.

[LD18] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 734–750, 2018.

[LDG+17a] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[LDG+17b] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017.

[LGG+17] 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.

[LGK+19] Huan Ling, Jun Gao, Amlan Kar, Wenzheng Chen, and Sanja Fidler. Fast interactive object annotation with curve-gcn. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5257–5266, 2019.

[LHM+20] Justin Liang, Namdar Homayounfar, Wei-Chiu Ma, Yuwen Xiong, Rui Hu, and Raquel Urtasun. Polytransform: Deep polygon transformer for instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9131–9140, 2020.

[LJ08] Percy Liang and Michael I Jordan. An asymptotic analysis of generative, discriminative, and pseudolikelihood estimators. In *ICML*, 2008.

[LJFU17] Shu Liu, Jiaya Jia, Sanja Fidler, and Raquel Urtasun. Sgn: Sequential grouping networks for instance segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3496–3504, 2017.

[LJT17] Justin Lazarow*, Long Jin*, and Zhuowen Tu. Introspective neural networks for generative modeling. In *ICCV*, 2017.

[LLC+21] 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. *arXiv preprint arXiv:2103.14030*, 2021.

[LLCF21] Zichen Liu, Jun Hao Liew, Xiangyu Chen, and Jiashi Feng. Dance: A deep attentive contour model for efficient instance segmentation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 345–354, January 2021.

[LLCL19] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2019.

[LLG+18] Donghoon Lee, Sifei Liu, Jinwei Gu, Ming-Yu Liu, Ming-Hsuan Yang, and Jan Kautz. Context-aware synthesis and placement of object instances. In *Advances in Neural Information Processing Systems*, pages 10393–10403, 2018.

[LLM+18] Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. An intriguing failing of convolutional neural networks and the coordconv solution. In *Advances in Neural Information Processing Systems*, pages 9605–9616, 2018.

[LLT19] Justin Lazarow, Kwonjoon Lee, and Zhuowen Tu. Learning instance occlusion for panoptic segmentation. *arXiv preprint arXiv:1906.05896*, 2019.

[LLW+18] Xiaodan Liang, Liang Lin, Yunchao Wei, Xiaohui Shen, Jianchao Yang, and Shuicheng Yan. Proposal-free network for instance-level object segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2978–2991, 2018.

[LLWT15] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *ICCV*, 2015.

[LM16] Ke Li and Jitendra Malik. Amodal instance segmentation. In *European Conference on Computer Vision*, pages 677–693. Springer, 2016.

[LMB+14] 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 *ECCV*, 2014.

[LPY+19] Huanyu Liu, Chao Peng, Changqian Yu, Jingbo Wang, Xu Liu, Gang Yu, and Wei Jiang. An end-to-end network for panoptic segmentation. In *CVPR*, 2019.

[LRB+18] Jie Li, Allan Raventos, Arjun Bhargava, Takaaki Tagawa, and Adrien Gaidon. Learning to fuse things and stuff. *arXiv preprint arXiv:1812.01192*, 2018.

[LSD15] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *CVPR*, 2015.

[LWZ+21] Ke Li, Shijie Wang, Xiang Zhang, Yifan Xu, Weijian Xu, and Zhuowen Tu. Pose recognition with cascade transformers. pages 1944–1953, 2021.

[LXFT18] Kwonjoon Lee, Weijian Xu, Fan Fan, and Zhuowen Tu. Wasserstein introspective neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3702–3711, 2018.

[LXG+15] Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-Supervised Nets. In *AISTATS*, 2015.

[LYH+19] Jianan Li, Jimei Yang, Aaron Hertzmann, Jianming Zhang, and Tingfa Xu. Layoutgan: Generating graphic layouts with wireframe discriminators. *arXiv preprint arXiv:1901.06767*, 2019.

[LZQ+21] Yanwei Li, Hengshuang Zhao, Xiaojuan Qi, Liwei Wang, Zeming Li, Jian Sun, and Jiaya Jia. Fully convolutional networks for panoptic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 214–223, 2021.

[Mar82] David Marr. Vision: A computational investigation into the human representation and processing of visual information, henry holt and co. *Inc., New York, NY*, 2(4.2), 1982.

[MFTM01] David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. volume 2, pages 416–423, 2001.

[MG18] Francisco Massa and Ross Girshick. maskrcnn-benchmark: Fast, modular reference implementation of Instance Segmentation and Object Detection algorithms in PyTorch. `https://github.com/facebookresearch/maskrcnn-benchmark`, 2018. Accessed: January 5, 2019.

[MHB17] Stephan Mandt, Matthew D Hoffman, and David M Blei. Stochastic gradient descent as approximate bayesian inference. *arXiv preprint arXiv:1704.04289*, 2017.

[MLX+17] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2794–2802, 2017.

[MNA16] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *2016 fourth international conference on 3D vision (3DV)*, pages 565–571. IEEE, 2016.

[MOT15] Alexander Mordvintsev, Christopher Olah, and Mike Tyka. Deepdream - a code example for visualizing neural networks. *Google Research*, 2015.

[NM90] Mark Nitzberg and David Bryant Mumford. *The 2.1-D sketch*. IEEE Computer Society Press, 1990.

[NWC+11] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading Digits in Natural Images with Unsupervised Feature Learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

[OT07] Aude Oliva and Antonio Torralba. The role of context in object recognition. *Trends in cognitive sciences*, 11(12):520–527, 2007.

[PCD15] Pedro O Pinheiro, Ronan Collobert, and Piotr Dollar. Learning to segment object candidates. In *NIPS*, 2015.

[PJP+20] Sida Peng, Wen Jiang, Huaijin Pi, Xiuli Li, Hujun Bao, and Xiaowei Zhou. Deep snake for real-time instance segmentation. In *CVPR*, 2020.

[PLCD16] Pedro O. Pinheiro, Tsung-Yi Lin, Ronan Collobert, and Piotr Dollár. Learning to refine object segments. In *ECCV*, 2016.

[PLWZ19] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[PS00] Javier Portilla and Eero P Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *Int'l j. of computer vision*, 40(1):49–70, 2000.

[PXP00] Dzung L Pham, Chenyang Xu, and Jerry L Prince. Current methods in medical image segmentation. *Annual review of biomedical engineering*, 2(1):315–337, 2000.

[QCY21] Siyuan Qiao, Liang-Chieh Chen, and Alan Yuille. Detectors: Detecting objects with recursive feature pyramid and switchable atrous convolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10213–10224, 2021.

[QJL+19] Lu Qi, Li Jiang, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Amodal instance segmentation with kins dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3014–3023, 2019.

[RAM+16] Scott E Reed, Zeynep Akata, Santosh Mohan, Samuel Tenka, Bernt Schiele, and Honglak Lee. Learning what and where to draw. In *Advances in Neural Information Processing Systems*, pages 217–225, 2016.

[RAY+16] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. *arXiv preprint arXiv:1605.05396*, 2016.

[RB05] Stefan Roth and Michael J Black. Fields of experts: A framework for learning image priors. In *CVPR*, 2005.

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[RHGS15] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[Ris78] Jorma Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465–471, 1978.

[RM15] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR, 2015.

[RMC15] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv:1511.06434*, 2015.

[RMW14] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pages 1278–1286. PMLR, 2014.

[RSD+12] Hayko Riemenschneider, Sabine Sternig, Michael Donoser, Peter M Roth, and Horst Bischof. Hough regions for joining instance localization and segmentation. In *ECCV*. 2012.

[RVG+07] Andrew Rabinovich, Andrea Vedaldi, Carolina Galleguillos, Eric Wiewiora, and Serge Belongie. Objects in context. In *Computer vision, 2007. ICCV 2007. IEEE 11th international conference on*, pages 1–8. IEEE, 2007.

[SBK19] Konstantin Sofiiuk, Olga Barinova, and Anton Konushin. Adaptis: Adaptive instance selection network. In *ICCV*, 2019.

[SE19] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *arXiv preprint arXiv:1907.05600*, 2019.

[SGZ+16] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *arXiv preprint arXiv:1606.03498*, 2016.

[SLKS05] Jian Sun, Yin Li, Sing Bing Kang, and Heung-Yeung Shum. Symmetric stereo matching for occlusion handling. In *CVPR*, 2005.

[SLY15] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *Advances in neural information processing systems*, pages 3483–3491, 2015.

[SM00] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *PAMI*, 22(8):888–905, 2000.

[SWRC06] Jamie Shotton, John Winn, Carsten Rother, and Antonio Criminisi. Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *Computer Vision–ECCV 2006*, pages 1–15. Springer, 2006.

[TCYZ05] Zhuowen Tu, Xiangrong Chen, Alan L Yuille, and Song-Chun Zhu. Image parsing: Unifying segmentation, detection, and recognition. *IJCV*, 63(2):113–140, 2005.

[TGB+17] Ilya Tolstikhin, Sylvain Gelly, Olivier Bousquet, Carl-Johann Simon-Gabriel, and Bernhard Schölkopf. Adagan: Boosting generative models. *arXiv:1701.02386*, 2017.

[TND+08] Zhuowen Tu, Katherine L Narr, Piotr Dollár, Ivo Dinov, Paul M Thompson, and Arthur W Toga. Brain anatomical structure segmentation by hybrid discriminative/generative models. *IEEE Tran. on Medical Imag.*, 2008.

[TNL14] Joseph Tighe, Marc Niethammer, and Svetlana Lazebnik. Scene parsing with object instances and occlusion ordering. In *CVPR*, 2014.

[TSC20] Zhi Tian, Chunhua Shen, and Hao Chen. Conditional convolutions for instance segmentation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*, pages 282–298. Springer, 2020.

[TSCH19] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9627–9636, 2019.

[Tu07] Zhuowen Tu. Learning generative models via discriminative approaches. In *CVPR*, 2007.

[Tu08] Zhuowen Tu. Auto-context and its application to high-level vision tasks. In *CVPR*, 2008.

[ULVL16] Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, and Victor Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *ICML*, 2016.

[Vap95] Vladimir N Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., 1995.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[WHY09] Xiaoyu Wang, Tony X Han, and Shuicheng Yan. An hog-lbp human detector with partial occlusion handling. In *ICCV*, 2009.

[WKM+19] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019.

[wrf] Pnpoly - point inclusion in polygon test. `https://wrf.ecse.rpi.edu/Research/Short_Notes/pnpoly.html`. Accessed: 2021-10-30.

[WT11] Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *ICML*, 2011.

[WXJ+18] Xinlong Wang, Tete Xiao, Yuning Jiang, Shuai Shao, Jian Sun, and Chunhua Shen. Repulsion loss: Detecting pedestrians in a crowd. In *CVPR*, 2018.

[WZH02] Max Welling, Richard S Zemel, and Geoffrey E Hinton. Self supervised boosting. In *NIPS*, 2002.

[XLZ+19] Yuwen Xiong, Renjie Liao, Hengshuang Zhao, Rui Hu, Min Bai, Ersin Yumer, and Raquel Urtasun. Upsnet: A unified panoptic segmentation network. In *CVPR*, 2019.

[XLZW16a] Jianwen Xie, Yang Lu, Song-Chun Zhu, and Ying Nian Wu. Cooperative training of descriptor and generator networks. *arXiv:1609.09408*, 2016.

[XLZW16b] Jianwen Xie, Yang Lu, Song-Chun Zhu, and Ying Nian Wu. A theory of generative convnet. In *ICML*, 2016.

[XSS+20] Enze Xie, Peize Sun, Xiaoge Song, Wenhai Wang, Xuebo Liu, Ding Liang, Chunhua Shen, and Ping Luo. Polarmask: Single shot instance segmentation with polar representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12193–12202, 2020.

[XT15] Saining Xie and Zhuowen Tu. Holistically-nested edge detection. 2015.

[XWY+21] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. *arXiv preprint arXiv:2105.15203*, 2021.

[XXCT21] Yifan Xu, Weijian Xu, David Cheung, and Zhuowen Tu. Line segment detection using transformers without edges. pages 4257–4266, 2021.

[YHC92] Alan L Yuille, Peter W Hallinan, and David S Cohen. Feature extraction from faces using deformable templates. *International journal of computer vision*, 8(2):99–111, 1992.

[YK06] Alan L Yuille and Daniel Kersten. Vision as bayesian inference: analysis by synthesis? *Trends in cognitive sciences*, 10(7):301–308, 2006.

[YK16] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. In *ICLR*, 2016.

[ZFU16] Ziyu Zhang, Sanja Fidler, and Raquel Urtasun. Instance-level segmentation for autonomous driving with deep densely connected mrfs. In *CVPR*, 2016.

[ZJRP+15] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip Torr. Conditional random fields as recurrent neural networks. *arXiv preprint arXiv:1502.03240*, 2015.

[ZML16] Junbo Zhao, Michael Mathieu, and Yann LeCun. Energy-based generative adversarial network. *arXiv:1609.03126*, 2016.

[ZP13] C Lawrence Zitnick and Devi Parikh. Bringing semantics into focus using visual abstraction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3009–3016, 2013.

[ZSL+20] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020.

[ZSQ+17] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, 2017.

[ZTMD17] Yan Zhu, Yuandong Tian, Dimitris Metaxas, and Piotr Dollár. Semantic amodal segmentation. In *CVPR*, 2017.

[ZWM97] Song Chun Zhu, Ying Nian Wu, and David Mumford. Minimax entropy principle and its application to texture modeling. *Neural Computation*, 9(8):1627–1660, 1997.

[ZWM98] Song Chun Zhu, Yingnian Wu, and David Mumford. Filters, random fields and maximum entropy (frame): Towards a unified theory for texture modeling. *International Journal of Computer Vision*, 27(2):107–126, 1998.