Modern Object and Visual Relationship Detection in Images from a Critical, Cognitive and Data Perspective

by

David Abou Chacra

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Systems Design Engineering

Waterloo, Ontario, Canada, 2023

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner:	Qiang Qiu Assistant Professor, Dept. of Electrical and Computer Engineering, Purdue University
Supervisor:	John Zelek Associate Professor, Systems Design Engineering Dept., University of Waterloo
Internal Member:	Alexander Wong Professor, Systems Design Engineering Dept., University of Waterloo
Internal Member:	Bryan Tripp Associate Professor, Systems Design Engineering Dept., University of Waterloo
Internal-External Member:	Oleg Michailovich Associate Professor, Dept. of Electrical and Computer Engineering, University of Waterloo

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

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

Statement of Contributions

This thesis consists in part of four main contributions written for publication. I performed the research, experimentation and writing pertaining to these publications. These publications include my academic supervisor John Zelek as a second author who reviewed them and provided feedback. I hereby declare that no other contributions to my work were made by anyone else.

At time of writing these publications and planned publications are as follows:

(Published) Abou Chacra, David, and John Zelek, "The Topology and Language of Relationships in the Visual Genome Dataset," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, 2022

This publication mainly draws from the work presented in Chapter 4.

(Accepted) Abou Chacra, David, and John Zelek. "Naive Scene Graphs: How Visual is Modern Visual Relationship Detection" 20th Conference on Computer and Robot Vision (CRV), 2023.

This publication mainly draws from the work presented in Chapter 5.

(Under Review) Abou Chacra, David, and John Zelek. "You're Seeing Things: The Role of Shape, Content, and Context in Modern Instance Segmentation".

This publication mainly draws from the work presented in Chapter 3.

(In Preparation) Abou Chacra, David, and John Zelek. "Topological Relationship Fields for Visual Relationship Detection".

This publication mainly draws from the work presented in Chapter 5.

Abstract

Deep learning has dominated the landscape of computer vision for the past decade. Deep learning networks are the top performers on a slew of computer vision challenges (e.g., object detection or image segmentation) and on the most popular datasets. They outperform other approaches by a large margin, each armed with their own tricks to improve upon their predecessors. However recent research highlights several short-comings of deep learning approaches, from poor generalization performance to the difficulty in understanding the rationale behind the decisions they make. More nuanced and human-like tasks such as visual relationship detection still prove difficult for deep learning networks as well.

In this thesis we tackle the problem of scene graph generation: the task of generating a directed graph that describes the relationships between detected objects in an image. We empirically identify, highlight and discuss the shortcomings of modern deep learning approaches to this task along with the reasoning behind these failures. Scene graph generation relies on both object detection and visual relationship detection. Our experiments first tackle object detection (through its more advanced task of instance segmentation) in isolation, then explore visual relationship detection starting with its data and moving on to its deep learning based approaches. Finally we propose and implement Topological Relationship Fields, a novel approach that allows for representing and grounding relationships purely visually. We utilize this representation for a scene graph generation approach that builds upon our findings and tackles the problem radically differently than the current standard approaches.

First, we isolate, evaluate, and quantify the effect of various image and object-level signals on performance of deep learning-based instance segmentation approaches. This is done via specifically crafted augmentations of the COCO 2017 object detection dataset. We also explore how (or even whether) the effect of shape, content and context changes with different training schedules, backbone architectures, and in non ROI-based detectors. We find underlying biases to object mask shapes that plague Instance Segmentation and object detection networks. We explore the bias of object shape versus content (its internal textures and contours) and find this bias to be object-dependent, where certain objects are more readily detected and even hallucinated based on shape (e.g., airplanes), whereas others are more reliant on their content (e.g., zebras), and some on both (e.g., people). An object's content still plays the largest role in its classification, however networks start relying more on shape and context when the object pixels are masked. We also find object context and background play a limited, but not insignificant role in its detection and classification. These assessments are critical in pinpointing certain unknown or unexpected behaviours

in commonly used deep networks, and in shedding light on potential failure modes in out of distribution data.

Then, we explore the shortcomings of modern approaches to Scene Graph Generation from a data, evaluation and an algorithmic standpoint. We begin by exploring the human induced labelling bias in the Visual Genome Dataset (the de facto standard dataset used in Scene Graph generation). It contains a large collection of images with corresponding object and relationship labels. We explore the lingual aspect of the relationship predicates and find that very few symmetric/inverse relationships are represented in the dataset (for example, 'above' and 'under'). We believe this is linked to human spatial cognition, and posit that labelling bias stemming from human representations of relationships creates asymmetric relationship labels that span the whole dataset. We also perform a 2D topological analysis of the bounding boxes linked by different relationship predicates. This analysis sheds light on certain classes and their ambiguity wherein more frequent classes are semantically overloaded and therefore quite confusing. We also show that when reduced to more lingually and topologically well defined spatial relationships Scene Graph Generation performance improves tremendously, but Scene Graph Generators remain far from perfect even with better data. We show this by examining the 'visual-ness' of visual relationship detection with current deep learning approaches. We describe and implement a new Naive Bayes-based 'statistical baseline' for scene graph generation and demonstrate that a classical machine learning approach, one as simple as a categorical Naive Bayes classifier, can perform relationship detection in a manner that achieves competitive performance to that of modern scene graph generators. Most notably, this basic classifier does not utilize the image pixels, but relies on the properties of the bounding boxes (class labels, topological configuration, ... etc.) to predict the relationship labels. This is an alarming finding regarding scene graph generation that implies that visual data in images is often not being utilized in modern visual relationship detection past the point of object detection.

Finally, we propose, implement and experiment with Topological Relationship Fields, a novel paradigm for visual relationship detection and scene graph generation. Noting the lack of reliance on the visual data in images for visual relationship detection, we specifically propose a relationship representation and detector, that, by design, can only utilize the visual, pixel-level, data in images and cannot explicitly make use of class statistics. While this approach does not perform at the level of existing scene graph generation methods, it shows promise as a first step towards a radically different paradigm to this task. It also brings with it several benefits which include explainability, significantly reduced network size, and a generalizable model capable of performing well in a zero shot setting.

Acknowledgements

I acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC).

This work was also supported by Mitacs through the Mitacs Accelerate Entrepreneur program.

I also extend my deepest appreciation to my supervisor Prof. John Zelek for his guidance and tireless support throughout my PhD.

Dedication

I dedicate this thesis to my loving parents Sindy and Bahij, for their tremendous love, their unyielding support and the innumerable sacrifices they've made. To the two lights of my life Issam and Daniel, I cannot wait to see what futures your bright minds will create. To my all-star sister Salma and brother-in-law Anis, for always making sure I take care of myself, I believe in myself, and I remember just how smart I am. To my wonderful brother Walid, for being my best friend and always knowing exactly what to say (and always taking my calls). To my awesome cousin Rebecca, for showing me what true strength is and never failing to amaze me. To my dear aunt Hitaf and cousin Jamal for all of our wonderful conversations.

I also dedicate this thesis to Aunt Julie, Uncle Kamil, Uncle Nabil, Uncle Wajdi, Uncle Issam, Taunte May, and Hanan all of whom I sorely miss. May they all rest in peace.

I finally dedicate this thesis to all my family, friends, mentors and colleagues who've supported me every step of the way.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

CNN Convolutional Neural Network

DL Deep Learning

FCN Fully Convolutional Network

FPN Feature Pyramid Network

GRU Gated Recurrent Unit

LSTM Long Short-Term Memory

mR Mean Recall

NB Naive Bayes

PredCl Predicate Classification

SGCl Scene Graph Classification

SGDet Scene Graph Detection

SGG Scene Graph Generator

SOTA State of the Art

TDE Total Direct Effect

TRF Topological Relationship Field

VAE Variational Autoencoder

VG Visual Genome

VRD Visual Relationship Detection

zsR Zero Shot Recall

Chapter 1

Introduction

Our minds are marvelous.

Over the span of hundreds of thousands of years, our minds have learned to harness fire, first for warmth, then for cooking, and eventually to power our world. Fire is just one example of human ingenuity, and how our brains continue to learn and improve on their knowledge. Humans consistently innovate and build on what they know to propel us forward, each generation harnessing the knowledge of the ones preceding it. The rise has been exponential, with the last few decades bringing with them technologies that would seem almost alien to someone from the 1800's. This is especially apparent when looking at the field of artificial intelligence, where the improvement in hardware, coupled with relatively recent research has created an almost synergistic effect, allowing machines to be 'smarter' than ever.

The field of deep learning (DL) specifically has seen a meteoric rise in popularity since the beginning of the 2010's, with Alexnet [65] motivating researchers to revisit artificial neural networks and utilize them in a variety of different applications. Often times, these deep networks have outperformed classical machine learning approaches, and they appear to solve previously unsolvable problems. Deep learning has effectively revolutionized computer vision including fields such as image segmentation [18, 95, 165, 153], 3D understanding and novel view synthesis [103, 137], and video understanding [43] among many others. Notably, approaches in DALL-E [105] and DALL-E 2 [104] appear to have even learned how to be 'creative' and seem to be capable of generating images that are nothing short of works of art.

1.1 Understanding The Limits of Current Deep Learning Approaches

Despite the immense progress achieved with deep learning, computer vision still isn't a fully solved problem. As they currently stand, deep architectures are usually able to solve problems when provided with sufficient training data, as well as limited data variability during inference time, however, these guarantees do not extend to those same architectures when training data is scarce or when the inference domain is sufficiently varied [89, 106, 94]. Even the impressive results seen in DALL-E 2 lose some of their shine upon closer inspection [91].

Furthermore, performance on 'standard' or benchmark datasets and challenges still has not saturated: these include the COCO [77] and LVIS [41] datasets for more general object detection and instance segmentation, and even more specific use datasets such as the Cityscapes dataset [21] which is restricted to street imagery (from the view of a car). State of the art performance appears to be even worse on benchmark datasets and challenges aimed at higher level tasks, for example the Visual Genome dataset [64] which allows for more human-like tasks such as visual question answering and visual relationship detection (detecting relationships between objects in an image). Also, network performance **outside** these benchmark datasets (which they are trained on) is quite often a large step below their performance within these datasets [147, 94, 106].

A growing body of research on generalization performance and network robustness aims to garner insight into the unpredictable behaviours networks exhibit [144, 6, 111, 116], what some of the driving factors are [131, 23, 52], and how to mitigate them [102, 37]. This type of validation research, and digging into what is driving networks to do well, and where (and why) networks fail is critical to progressing the field of computer vision as it often gives us the best insight into how to proceed with fixing the issues that are holding these networks back. As such, we dedicate a large portion of this thesis to answering these questions in the domains of instance segmentation (Chapter 3) and visual relationship detection (Chapters 4 and 5) where this type of validation research has not yet been thoroughly explored. The performance disparity, and the underlying deep learning artefacts and biases causing it, are a very likely culprit in why applications of computer vision ranging from medical image analysis to self driving cars (and many applications in between) are not trivial to automate with deep learning just yet. For example, in the case of autonomous vehicles, while performance is getting better when measured on the sunny highways of San Francisco, there is a lot of research still geared towards solving some very fundamental and realistic problems [40], such as engineering around image deformations

resulting from raindrops on the camera or radar [155]. We pinpoint these unpredictable artefacts and their likely causes and effects in very commonly used network paradigms (for example in MaskRCNN [44] which is effectively used in literature as an off the shelf object detector/instance segmentation tool). We attempt to understand how Marcus's 5th, 7th, 8th and 9th limits are manifesting in networks performing instance segmentation and Visual Relationship Detection (VRD).

Despite these limitations, DL has proven itself to be a very handy tool, and through a mix of exploration and exploitation has become the state of the art for a significant portion of image-related tasks. We do not believe Marcus's aforementioned limits can be addressed in one fell swoop, nor do we believe that these limits diminish the work and progress that DL researchers have achieved thus far. Instead, we see the limits as important principles to keep in mind when critically evaluating existing architectures, as well as guiding principles on how to build DL-based architectures that can move the field forward.

1.2 Relationship Detection: A Step Towards Human-like Cognition

Human cognition, in a nutshell, is what allows us to reason about the world around us, we take in outside stimuli (through our senses), and analyze them through our cognitive framework to get to an understanding of our surroundings [47]. Through our human cognitive abilities, we tend to build a knowledge of real world concepts that aids us tremendously in understanding the world around us [124]. A conceptual understanding of an object could mean (but is not limited to) understanding it in many different ways: from its form (i.e., how it looks) to its function, and from its sub-parts to its interaction with other objects around it [123]. Take the concept of a school bus, for example: we usually imagine a large yellow and black vehicle (its form), transporting children to and from school (its function), made up of four wheels, a chassis, windows, doors (...etc.) all arranged in a certain manner (its sub-parts) and usually carrying kids, a bus driver and seen driving on roads (its interaction with the world around it). These understandings are often even themselves intertwined [9, 12], adding to the complexity of designing machines that can explicitly understand concepts and making our human ability to understand them seem quite impressive in comparison.

Human's conceptual knowledge of a school bus (and in general) encodes with it a certain robustness, seemingly in direct opposition to the deep learning limits described by Marcus [89]. Whether we see a cartoon school bus, a differently colored school bus, a toy school bus,

or another variant we can still identify it by utilizing our conceptual knowledge of it. This robustness does not necessarily apply to all deep networks trained to detect school buses, in fact, most would struggle with one or more of those changes as they pose radical domain shifts. This is not due to an error in the networks themselves, but is more likely because they are missing crucial elements that allow them to truly understand what they have been trained on. Networks are great at classifying the statistical correlations between pixels, but a significant portion of deep learning networks lack the ability to gain a conceptual understanding of the world simply because their architectures were not designed with that knowledge in mind [89, 69]. Marcus even emphasizes a cognitive, human-like, approach to deep learning as the best path forward for building better AI in [90].

The origin, evolution, and nature of human cognition are still debated to this day [47]. Human cognition is a fascinating research field in an of itself and we opt to take some influence from cognitive psychology, specifically human relationship detection, and progress relationship detection in computer vision in an attempt to build towards better DL architectures. Relationship detection is one of the earlier cognitive abilities we gain as humans [122], with these relationships pertaining to object interactions [122], and our ability to detect and categorize relationships grows and remains a crucial part of our spatial awareness and cognition [70, 88]. Within cognitive psychology, 'image schemas'[1] [88, 42] are used to describe early cognitive notions of spatial object relationship representations in the human brain. Since relationship detection appears to happen rather early on in humans, and human's multiple cognitive systems appear interlinked [124], learning relationships in computer vision may be a good step towards more 'cognitive' AI models [90], and all the benefits this may provide.

However, even a human's own spatial cognition is not free of biases. These biases stem from which objects we, as humans, choose to focus on, and how they effect both our language and how we describe what we see [70]. We, as humans, can innately correct for some of these biases and understand the full scenario we're looking at or reading or hearing about, however if DL networks were trained with data that includes these biases, they will struggle to move past them. In fact these human biases do trickle down into the Visual Genome dataset [64], and we demonstrate this phenomenon and its effect on visual relationship detection networks in Chapter 4.

In computer vision, relationships between different objects are represented using scene graphs (first defined in [57]), which can be *somewhat* viewed as a computational analogue to image schemas as re-defined in [88]. An image's scene graph is simply a directed graph

[1]Note that the term is borrowed from cognitive psychology and the 'image' portion of the term is not a computer vision 'image'-a collection of a pixels on a grid.

with nodes representing the objects seen in the image, grounded via bounding boxes, and edges representing a directed relationship between the detected objects[2]. Current scene graph generation approaches sequentially perform object detection and relationship detection, opting to generate object proposals first, then finding (and ranking) the most likely relationships between those detected objects [10]. This approach is not without merit and captures a portion of how humans reason about relationships. Almost all scene graph approaches follow this approach, however scene graph generation performance, and therefore relationship detection, remains quite poor (as we discuss in Chapter 5). Inspired by human spatial cognition, we propose a novel approach to scene graph generation and relationship detection that is grounded in image pixels, which allows for these visual relationships to be more grounded visually in the objects they relate, and captures another aspect of how humans reason about relationships.

1.3 Contributions

The contributions of this thesis are as follows:

- **In Chapter 3, we determine the negative and unpredictable network behaviour stemming from how object shape, its content and its context affect instance segmentation.** We systematically quantify the roles of these cues in current deep learning approaches and underscore some of the common pitfalls exhibited across different architectures.
  - We show empirical evidence of shape detection biases in the tested networks, that likely extend to other networks. The representation of object shape is brittle, where networks can achieve a maskAP of over 20% on images with the objects completely removed, but only if their outer contours remain rigidly similar to those in the ground truth. They can also achieve around a 10% maskAP simply using plain object silhouettes on a plain background.
  - We also show that non-ROI based instance segmentation networks appear more biased towards shape than their ROI based counterparts.
  - We isolate which of the studied cues seem to be the most critical, and find that it is actually object dependent with different object subsets relying on different

[2]Scene graphs can also include object 'attributes' (e.g., color), however these attributes are often not explicitly detected or used in most scene graph approaches.

cues. However, on average content appears to bias networks more than shape and context.

  - We demonstrate that blurring the content of objects, but maintaining their sharp outer contours causes very little change in network performance.
  - We show that object context plays a smaller role than the other two cues, but plays a non negligible role in aiding performance.

- **In Chapter 4 we demonstrate the effect of human labeller's spatial cognitive biases on the language with which relationships in the Visual Genome (VG) dataset are defined.** We define the 2D topology of these visual relationships and quantify how different topological and lingual combinations affect the learned scene graph generation models.

  - We show that the human labelling bias stemming from human representations of relationships creates asymmetric relationship labels that span the whole VG dataset.
  - We show that the lack of inverse relationships in the VG dataset leads to a lack of it in the learned models.
  - We utilize 2D topology to both shed light on the lingual vagueness of certain classes and show that some of these classes can be understood differently based on their topological configurations.
  - We show that when reduced to more lingually and topologically well defined spatial relationships scene graph generation algorithm performance improves tremendously, but scene graph generators are still far from perfect.

- **In Chapter 5, we present a novel statistical approach to scene graph generation that performs competitively without utilizing image pixel data, then describe and implement Topological Relationship Fields,** an alternative approach to scene graph generation by utilizing topological configurations of bounding boxes and **explicitly constraining relationships to be visually grounded in image pixels.**

  - We describe and implement a novel but simple categorical Naive Bayes classifier for scene graph generation and demonstrate that this basic classical machine learning approach, can perform relationship detection in a manner that achieves comparable performance to that of modern scene graph generators.
  - We discuss the how modern approaches to scene graph generation are being held back from a data, evaluation and algorithmic standpoint.

– We describe, implement and evaluate Topological Relationship Fields a novel representation for scene graph generation grounded in the image pixels themselves.
– We qualitatively demonstrate the visual nature of the learned relationships and how they pertain to specific pixel regions.

Chapter 2

Literature Review

2.1 Introduction

Relationship detection is a key tool humans use for reasoning about the world around them [70, 88]. In this section we begin by discussing relevant literature that pertains to a precursor to relationship detection - object detection. In this thesis (Chapter 3) we explore object detection through the lens of instance segmentation and discuss some of our findings on troubling failure modes we uncovered empirically. We discuss the relevant literature in Section 2.2, pertaining to image segmentation, and Section 2.3 which discusses failure modes in computer vision more generally. While we view relationship detection, and scene graph generation, as a stepping stone to empowering deep learning with conceptual knowledge, we also review some other approaches to this in Section 2.4. Since a large portion of our thesis is dedicated to relationship detection and some of its current shortcomings (Chapters 4 and 5), we provide a more detailed overview of modern scene graph generation in Section 2.5.

2.2 Image Segmentation

Image segmentation is a crucial and well explored topic in computer vision that lends itself to many different tasks. Semantic segmentation is a core part of the image understanding problem where, given an input image, the aim is to output a corresponding mask where every pixel from the input image is labelled according the category it belongs to. Instance

segmentation goes one step further and distinguishes between different occurrences of the same category of object within a single image.

2.2.1 Semantic Segmentation

After the seminal work on image classification with deep networks by Krizhevsky et al. [65], researchers started unlocking the potential of deep learning for other computer vision tasks, including image segmentation. Specifically, convolutional neural networks (CNNs) lent themselves very well to image segmentation. In [84], Fully Convolutional Networks (FCN) modified well performing classification architectures to the image segmentation task by removing their fully connected (i.e., classifying) layers and replacing them with convolution layers capable of outputting spatial maps. Their work beat the PASCAL VOC state of the art proving how powerful deep learning can be for image segmentation. The choice of using convolutional networks when dealing with images was not arbitrary, however, as they were actually inspired by a model of the early human visual cortex [29]. CNNs are a good example of how prior information can be built-in to deep learning architectures, as researchers valued their properties of spatial invariance and connection sparsity.

Building on FCNs, Segnet [4] and U-net [110] are two of the first encoder-decoder architectures for image segmentation which utilize an 'hourglass'-like structure of consecutive convolutions and down-sampling layers (the encoder), followed by another set of convolutions and some up-sampling variant (the decoder) to return to a dense output map similar in shape to the input image. These architectures do differ in their choices of kernel size, pooling type (for the encoder), but their main differences are often seen in the decoder where the choice of how to up-sample the low-level features to achieve a final segmentation proves critical. In the case of Segnet [4] , the authors choose a memory efficient option of utilizing only the pooling indices of the corresponding encoder layer (followed by convolution layers to 'densify' the sparse output) to reconstruct an output segmentation that is faithful to the input image, in essence utilizing information about where a certain feature originated from to produce their output. U-net [110], see figure 2.1, takes a different approach, concatenating the entire feature map produced by the convolutions of the down-sampling layers with those up-sampled in the decoder, utilizing more of the available information at the cost of being more memory intensive. Several other methods use a variant of the encoder-decoder architecture. Both these architectures also incorporated a prior information cue of their own as both Segnet and U-net (and to a lesser extent FCN) used a form of skip connections to maintain fine details in images, these skip connections carry information about the fine details that is lost after every down-sampling stage.

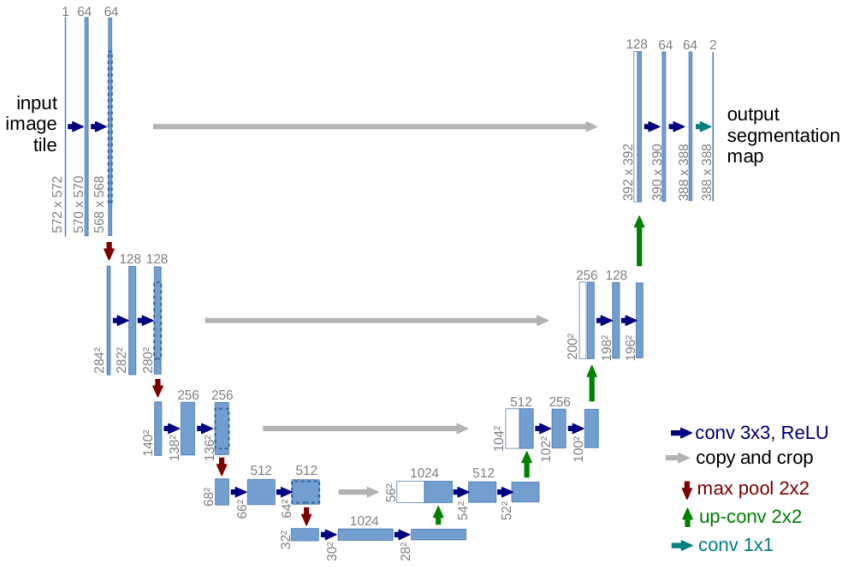


Figure 2.1: A schematic of a U-Net architecture taken from [110]. The name is inspired from its visualization as a 'U' shape since it modifies the traditional hourglass structure to include skip connections that allow feature maps from the downsampling path to be utilized when making decisions in the upsampling path. The intuition is that the middle layer builds a useful compact representation that gets refined as it goes through the various upsampling steps.

Down-sampling was also not added into networks for naught, and it serves a key purpose: modelling long range dependencies in an image. Traditional convolution kernels only look at pixels in a close vicinity, so looking at pixels in a 10x10 vicinity required a 10x10 convolution, which is computationally prohibitive. This is where downsampling operations, such as pooling, or to a lesser extent strided convolutions serve a dual function, they solve the long range dependency problem allowing a convolution operation to have an extended receptive field, while also keeping the computation costs tractable. Though, as mentioned previously, they bring with them the problem of having to sacrifice finer details in an image to serve this function. The Deeplab family of networks [13, 14, 15], see figure 2.2, address this issue head on by utilizing dilated convolutions. Dilated, or atrous, convolutions still grow their receptive field but don't utilize down-sampling, instead kernel are larger but sparser by design, this allows a 3x3 kernel to sample a 5x5 grid in the input by only sampling ever other pixel (a dilation rate of 2). Cascading dilated convolutions with various rates can allow sampling different and larger receptive fields at no extra computational cost. Dilated convolutions, Atrous Spatial Pyramid Pooling (ASPP) [13], and Conditional Random Fields are all used in the Deeplab family, and implement the initial target of downsampling and strided convolutions which was to encode another important piece of conceptual knowledge: contextual information. We know that pixels in natural images almost always, barring noise, belong to objects that the image is capturing and as such context is an important cue that any deep learning network must be able to utilize.

There have been numerous innovations in semantic segmentation apart from the above mentioned ones. The surveys in [30] and [95] were a key resource in classifying these advances and understanding their utility in image segmentation.

2.2.2 Instance Segmentation

In certain cases, instance segmentation is a more desirable output to semantic segmentation. The chief difference being that in instance segmentation every instance of an object is labelled separately as opposed to all instances of the same object being indistinguishable in the output. The Regional Convolutional Neural Network (RCNN) family of networks, and specifically the MaskRCNN [44] family tackle this problem. The original implementation builds upon the Faster R-CNN [108] object detector, which outputs classified bounding boxes around objects only, and adds a branch for mask prediction that enables instance segmentation. Mask RCNN operates by first using a region proposal network trained to generate bounding boxes for the regions in the image most likely to contain an "object", this is followed by a region classifier which outputs the likely class for the object detected in the bounding box, as well as an object mask generator which generates a segmentation

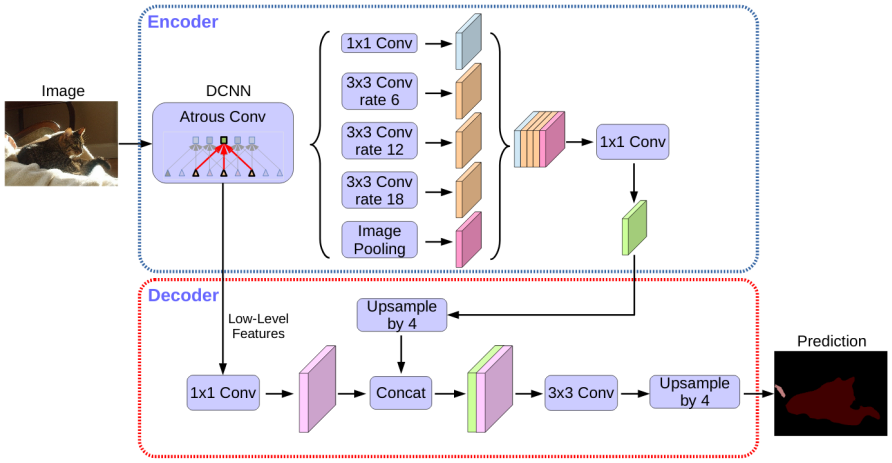


Figure 2.2: A schematic of the Deeplab V3+ [15]. The architecture employs dilated convolutions, and atrous spatial pyramid pooling as well as a variable dilation rate in its encoder. Skip connections allow the decoder to not lose information from small details as well, factoring into its state of the art performance.

mask for this object. Since every object is contained within its own bounding box, the differing object masks are in fact instance segmentation masks. Mask R-CNN thus operates in two stages: a first stage which outputs region proposals (good candidate bounding boxes around regions of interest), and a bounding box classification/regression and mask prediction stage which gives the desired final outputs. Figure 2.3 shows a schematic of the network architecture.

Any 'Mask R-CNN'-type architecture has a feature extractor network, referred to as the 'backbone', as well as networks for bounding box/mask prediction, referred to as the 'heads'. Overall, a deep residual network (Resnet) backbone [45] augmented with a Feature Pyramid Network (FPN) [75] appeared to be one of the better performing overall backbones on the task of instance segmentation in the original implementation [44]. However, [44] gave no restrictions on what the backbone or head architectures need to be, and even proposed a variety of different backbones and measured their performances.

The shifted window (Swin) transformer networks proposed in [83] prove to be a high performing backbone architecture within a Mask R-CNN framework. In a nutshell Swin transformers build upon the work done in [25] for vision transformers (ViT), they start with smaller pixel patches that grow in a hierarchical manner in deeper layers (by merging non overlapping nearby windows), and also utilize local attention making sure patches

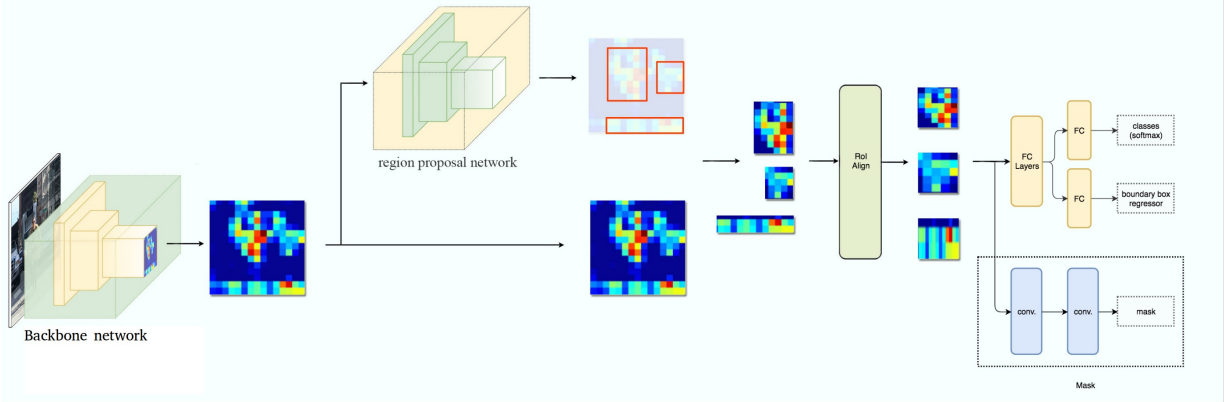


Figure 2.3: A schematic of a MaskRCNN [44] type architecture. Effectively MaskRCNN is identical to its predecessor Faster R-CNN with the exception of adding the 'mask' head that operates on the ROI aligned and cropped feature maps to predict the object mask. The feature extracting 'backbone network' can be modified, and often is in subsequent works that improve upon the original MaskRCNN implementation. Photo was adapted from [50].

don't interact with other patches that are too far from them, with a key trick being that attention is limited by windows that are shifted in consecutive layers (the titular shifted windows). Overall Swin transformers demonstrate top performing results when used as backbones for different image level tasks, with instance segmentation performance being the task of interest for us. There are varying sizes of Swin based architectures, for example the Swin-T backbone which is comparable to a Resnet-50 in its complexity.

Other approaches to instance segmentation [134, 141, 162, 107, 17] opt to not use the region proposals seen in Mask R-CNN and instead output their object detections and masks in a 'single stage'. We note that the term 'single stage' is somewhat overloaded since the 'single stage' approaches aren't always simply one fully convolutional network. The term is used to imply that these networks don't have a ROI Align and cropping operation that ends up generating multiple sub regions (the 'region proposals') that are processed almost as individual 'new images' by the rest of the network.

Tensormask [17] employs a sliding window approach, i.e., a dense approach to instance segmentation, which carries with it some benefits. This approach is no longer limited to using a consistent and class agnostic representation that loses geometric information which is utilized in the R-CNN based methods, and instead can represent their knowledge with structured and geometrically relevant 'sub-tensors' or'tensor-masks'. The authors

show that they're able to achieve comparable results to state of the art R-CNN methods with a dense approach. Much like Faster R-CNN, served as the precursor to Mask R-CNN, Conditional Convolutions for Instance Segmentation (**CondInst**) [134] was inspired by and built upon the Fully Convolutional One-Stage Object Detection (FCOS) approach [135]. CondInst opts to perform instance segmentation in a fully convolutional manner, having the weights of its mask heads be dynamically generated based on the input (the titular conditional convolutions). A schematic of the CondInst architecture is presented in figure 2.4. We also explore the Mask Encoding for Single Shot Instance Segmentation (**MEInst**) approach of [162], who, noting the general redundancy in mask shapes, reparametrize and encode object masks in a fixed size single-dimensional vector of significantly smaller size than the initial 2-dimensional masks and instead train to predict that vector (decoding it for the final output). In [141], the authors propose Segmenting Objects by Locations version 2 (**SOLOv2**), building upon the previous version of the similarly named approach, they utilize dynamically generated convolution heads at a specified grid of locations, having each element in the grid be 'responsible' for predicting the instance whose center it contains. SOLOv2 doesn't rely on predefined 'anchor boxes', which are a set of engineered region shapes that define the acceptable shape or aspect ratio of the detectable regions of interest. These are often gathered from the shapes of boxes in the ground truth, and are critical to MaskRCNN.

Other single stage object detectors do exist as well, the survey in [82] further delves into details and distinguishing characteristics of generic object detectors, and was a key resource for us as well.

2.3 Common Trappings

While the advances in deep learning have allowed for impressive results and performance on image segmentation among other computer vision tasks, a significant portion of these networks still share some key disadvantages [30, 38].

2.3.1 Unpredictability

Several works [144, 6, 111, 116] explore unpredictable behaviours in deep learning based computer vision approaches, these works often tackle one task (e.g., image classification) and describe the behaviour of networks trained on that task. The work done in [144] quantifies the sometimes catastrophic failure modes resulting from over reliance of deep

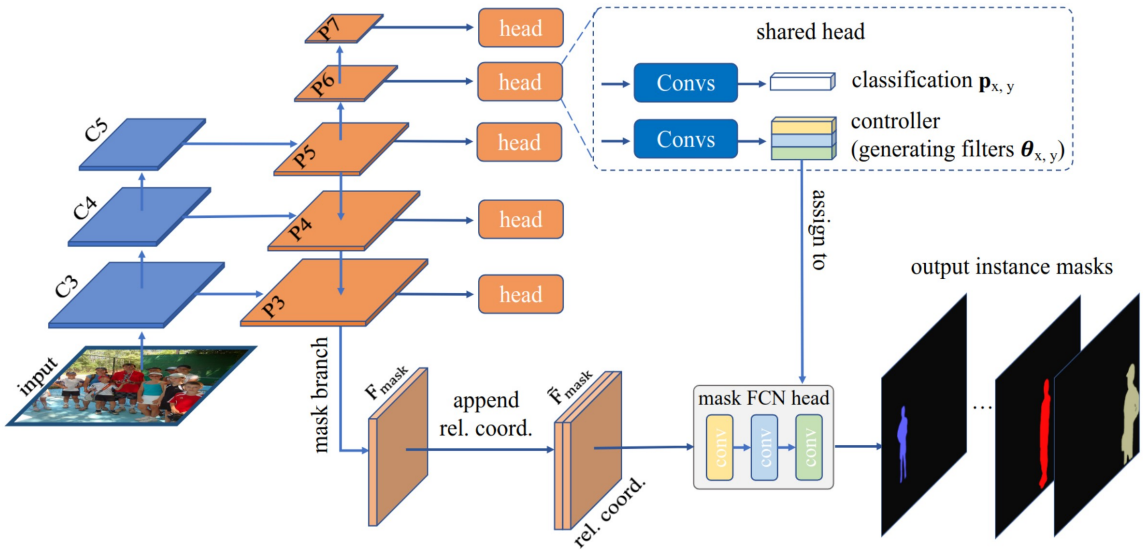


Figure 2.4: A schematic of CondInst [134]. Similarly to other one and two stage instance segmentation approaches, CondInst uses an interchangeable 'backbone' network, however as with other single stage approaches no ROI pooling/cropping is employed. In the case of CondInst (as with its object detection base Fully Convolutional One-Stage Object Detection), the features extracted at multiple stages of the Feature Pyramid Network of the backbone network are fed into prediction heads to predict object class (along with box parameters for the instance). The key to CondInst is that the shared head also predicts the parameters for a small sized Fully Convolutional Network (the mask FCN head) conditioned on the instance. This allows it to predict masks associated with the classes.

learning-based image classifiers on image background. In image classification, networks are trained with image level labels, and do not have pixel-level annotations detailing where exactly the 'foreground' is. As a result, if an object occurs with a specific type of background very frequently (e.g., a cow 'foreground' often standing in a grassy field with a blue sky), networks may conflate the background with the foreground class. The authors of [144] find that image classification networks required the background to correctly predict the foreground class of over a third of images. Furthermore, in images where the foreground was removed, networks were shown to have the ability to predict the image class at around 50% of their original performance, implying their strong reliance on signals that are occurring in the background even when the foreground is completely absent. The authors also demonstrate that adversarially chosen backgrounds can cause for an 88% misclassification rate, where the networks end up choosing their class based on the background.

In [116], Shetty et al. also demonstrate an over-reliance on object context in the case of both image-level classification as well as pixel-level semantic segmentation networks. Specifically, semantic segmentation networks are shown to rely on context to hallucinate objects that have been explicitly removed. They also show that semantic segmentation performance can actually be improved on edge cases by incorporating augmented data with removed objects into their training data. In [111], Rosenfeld et al. discover that object detectors fail to correctly classify or even detect previously correctly classified and detected objects when they are 'transplanted' into another image. A notable example, and the namesake of their title ('The elephant in the room'), is that when pixels belonging to a foreground of 'elephant' are overlaid onto another image, they can become completely undetected, misclassified (seemingly in a manner that matches the general image context), or even affect the detection of other objects in the scene changing their detection and classifications. Another finding in [111] is that the specific location of where the new object is overlaid also seems to change what type of effect it will have on the object detector. In some locations the transplanted object is classified correctly (at the cost of the original image objects having their labels corrupted), and in others it is completely invisible. Rosenfeld et al. believe this may due to 'feature interference', where pixels from outside of the 'region of interest' are effecting network performance inside the region.

In [46], Hermann and Lampinen aim to understand which input features ultimately get used by trained models and more importantly how and why these features get represented. The authors utilize specially crafted synthetic datasets to get some insight into what networks are learning. They find that networks tend to 'fixate' on the easiest feature to use for their task, usually this is a feature that is most readily decoded from the network when it's still untrained. This 'easy feature' is not always the most predictive, and this is specifically designed for in their synthetic dataset, however the authors find that networks still focus

on this easy to decode feature, and in some cases ignore more predictive features that are more difficult to learn.

2.3.2 Data Hungriness and Generalization Performance

A common theme among deep networks is data hungriness, and in the case of image segmentation that data is quite difficult to obtain. Since a network performing image segmentation is meant to be outputting a pixel by pixel classification of the input image, most approaches with full supervision require pixel-level annotations on the ground truth used to train them. This level of supervision can be quite prohibitive, making networks that can train with less ground truth data more lucrative [110]. As expected there is a trade-off that occurs between how much a network can learn, and how much data it needs, and unless the architectures are modified fundamentally to account for less pixel-labelled data, usually the networks training with less data have less network capacity. We look to explore what methods can curb this need for data, and whether conceptual knowledge, in the form of prior information can serve to reduce the amount of data needed for training while maintaining performance.

The other side of the 'data-hungriness' coin is that the generalization performance of the networks is often never reported on or addressed, and network performance on unseen datasets is often not up to par with its performance on the datasets it has been trained with [38, 89]. A good measure of intelligence for an approach is in how well it can adapt to new domains, as it reflects on how robustly it built its knowledge from the training data. In some applications, the data at inference time is identical to that at training time, for a camera positioned in a stationary position over an assembly line, where the lighting is constant and there are very few outliers, DL is a powerhouse. These sorts of problems are a good example of the kind of problems where designing and implementing deep learning architectures is relatively straightforward. However on a relevant problem such as designing a vision system for a self-driving car, neither assuming a stationary data domain (of a midday drive in San Francisco) or training on every possible scenario (From snow, to unmarked roads ..etc.) is an efficient approach.

Data augmentation is presented as an approach to mitigate both 'data-hungriness' and generalization issues. When used in [65], the aim was to make the most of the available training data. Data augmentation links to prior knowledge as well, we understand that a mirror image of a cat, is still a cat, but a purely data driven approach such as deep learning would have to come to that conclusion on its own by observing it in the data. Augmentation techniques range from simple crops and flips to more advanced techniques

such as adversarial example training (where specific detrimental examples are used to mitigate the brittleness of the network to some kinds of noise) [118]. Generative networks have also been used for data augmentation [1] allowing the network to create more data as well as training data with more complex augmentations. Even neural style transfer has been used for data augmentation [118], utilizing the Generative Adversarial Network's (GAN) abilities to learn powerful transformations that can turn a summer scene into a winter one for example. Data augmentation proves that with more data, architectures have the ability to improve their performance, and to a certain extent it can be argued that they make architectures more robust and are generalizable.

2.3.3 Other Drawbacks

In addition to the above, segmentation deep learning networks also aren't commonly built to take advantage of temporal consistencies when dealing with video data, and often opt for a frame by frame approach [30]. Failure metrics are not very well defined, and it is not possible to know what kinds of variances in data or tasks will make a certain architecture fail [38]. Convolutional neural networks have also been shown to be overly reliant on texture and local information for their decision making [31].

Deep networks are limited by their design decisions, from network architectures, inter-network relationships, loss functions among others. All of these factors must be improved upon and approached creatively. We next look to uniquely designed networks that have shown promise in mediating some of the common trappings of image segmentation with deep learning.

2.4 Conceptual Understanding (in Images)

In humans, vision is a complex task that relies on several different cognitive functions [47, 123, 124]. In [9], Cavanagh puts forth the idea that our brains create a "compressed or annotated" version of what our eyes see to send to other centers in our brain. These centers then go on to perform their own vision-related tasks not using measurements of reflected light, but instead using a conceptual abstraction that was provided to them. Our brain has summarized the detected light's 'sensory data' into a form that permits every center of our brain that needs it to use it as effectively as possible. Understanding the world around us requires a conceptual understanding of it. This understanding is gained over time as we grow from children who can identify their parent's faces to adults who utilize vision to perform a wide range of tasks like driving and cooking.

While utilizing deep learning for computer vision, however, this conceptual understanding of the world is often left to the networks to figure out on their own. Some innovations do target one part or another of this "conceptual space", yet algorithms struggle to learn richer concepts and connections beyond those which they are given. For example, on the concept of recursion, humans have a much easier time predicting how a tree will grow and understanding the recursive concept of a branch and the other branches and leaves that grow from it than machines do [67].

In [69] Lake et al. propose several "ingredients" for imbuing machines with more human-like intelligence. These ingredients include concepts such as causality and compositionality under a larger umbrella of "model building", along with intuitive concepts of physics and psychology that humans have evolved to use. Notably, the authors also place a significant bit of importance on the interaction between the "ingredients". This is supported by research in psychology [124], where 'ingredients' similar to the ones cited by Lake et al. are also attributed to human intelligence. While there is some debate as to how or when we learn these concepts, these 'core systems' [124] allow us to function and think and are likely what allows humans to be intelligent.

Furthermore, in [123], Spelke makes the argument we as humans start up with a framework that enables us to survive much like animals do during early life, and one explanation of the 'intelligence' we show is due to our language capabilities. Language encodes important information, it allows the dissemination of ideas from one generation to the next and from one person to another. One barrier to understanding another person is not understanding their language. This extends to machine learning architectures, where we cannot hope to understand their results and demystify their inner workings without a 'common language'.

2.4.1 Human Tools for Conceptual Understanding

Marcus's critiques of Deep Learning [89] are the symptom of deep learning approaches that have not yet capitalized on a lot of what research shows works in humans. In a way, the core of the critiques for deep learning is in their more human-related shortcomings. Humans have a cognitive framework that allows them to go beyond pattern recognition and into 'intelligence' that deep learning methods just can't seem to do. What does psychology, biology and cognitive science tell us humans do different than what deep learning does?

2.4.1.1 Compositionality

Humans have an innate sense of compositionality: sub-parts come together in different ways to form an object that is greater than the sum of its parts. One way of expressing this compositionality is as a hierarchy of objects and their subparts, and even the sub-parts that form those subparts ...etc.

For example, we understand that a certain composition of wood, screws and glue can form a table, and another composition of the same subparts can form a chair, and both a table and a chair are conceptually different to both each other and their building blocks. While it was believed that deep learning learns this compositionality innately by virtue of its many cascading layers (where the shallow layers would learn the smaller parts, and the deeper layers a composition of those parts), [31] shows that networks still rely very heavily on local, textural patterns to make their decisions and may not be learning this compositional hierarchy the way we thought they were.

In [69], Lake et. al. cite the importance of compositionality in a machine learning setting, stemming from its importance and use in our cognitive understanding of the world. The work in [68] showcases the power that learning data as a combination of subparts rather than as individual examples can hold: Bayesian Program Learning is used to classify handwritten letters and relies on commonly reoccurring gestures (the subparts) to classify a character based on detecting what subparts were used to generate it. While their work is limited to handwritten character generation, it serves as a simple, but powerful example of how compositionality can be applied in the realm of computer vision.

Furthermore, humans are not only compositional learners, as we also learn relational information within certain hierarchies [126]. As [126] explains, humans rely heavily on spatial representations, so much so that non-spatial concepts are likely encoded in a spatial, number-line-like, low level representation in our brains. The authors give the example of the words 'insider' and 'outsider', denoting the concepts of whether or not someone belongs to a group, as one of many common words we use that show how deeply rooted our internal representations are in spatial relationships.

This kind of spatial representation does not extend to deep networks. In [6], the authors show how their architecture 'Bagnet', which by design only relies on local features (namely, small image patches) ignoring where they may fall spatially, is able to perform comparably to VGG and Alexnet on the Imagenet challenge. They further show that there is no evidence to suggest that deep learning models are learning or utilizing higher order cues in images, aside from local textural cues and a 'bag of words' type approach.This is further backed up by the experiments in [31] which also show that networks still rely very heavily

on local textural patterns to make their decisions and may not be learning a compositional hierarchy as initially expected.

2.4.1.2 Language and Cross Domain Knowledge Transfer

Much like Cavanaugh cites the importance of an internal representation within our brain that is a more succinct and useful summary of our sensory input [9], Spelke [123] argues the importance of language (a sort of external representation) as part of what makes us intelligent. Language allows us to relay highly distilled knowledge and arguably makes human progress possible because of that. Through language we are able to learn from thousands of years of human progress and build upon that progress from one generation to the next generation. Yet, language plays almost no part in contemporary computer vision that utilizes deep learning.

As an example, think back to when you had your first driving test. You were 16 and had seen others driving almost all your life, however it is unlikely you knew all the rules of the road only through perception. Between ambiguities on when to make a left turn and what the zebra crossings really mean, the knowledge you gained through reading the driver's handbook and even asking your mother questions as she drove gave you a strong base knowledge that you used to justify your perceptive system. You were able to dismiss any ambiguities from your knowledge and give yourself a good head start on driving. Perhaps simply by being a silent observer of others driving you would have gained enough knowledge through pattern recognition, however, your knowledge of the underlying rules of the road that were given to you through language made your driving knowledge much more robust. In addition to your 'quick start', now if you happen to visit the United Kingdom and go for a drive, you no longer have to relearn how to drive from scratch, you can simply tweak a few rules in your mind and you're able to drive there too.

Perception and language are indeed entangled in our minds [123], they do not happen independently and do have the power to influence one another. Language is the tool we use to express our ideas and transfer knowledge. In a sense language can be seen as the tool that allows us to transfer the information that exists in the 'latent space' of our mind, in a way that can be used by others and influences their own mind.

2.4.1.3 Causation and Systems Knowledge

Causality is a concept that greatly affects our understanding of the world. Physical system causality, such as our understanding of the concept of gravity, gives us strong priors as to

how things can and should behave and where we expect them to go. Think of watching a toddler explore this concept for themselves, they pick something up and let it go in mid-air expecting it to stay where they left it, only to be surprised by the object dropping; as adults, we have a pretty good idea of where and how this object will go, and whether it may or may not shatter based on a strong causal understanding of the world. We have the ability to answer Judea Pearl's [99] counter-factual of "what would have happened had the child not dropped the pen" or "what would have happened had they opted to drop the plate of Fine China instead".

Causality is a deep field in and of itself, and it has started making its way into deep learning [99], however specifically in image understanding, it is still only basically explored. The term causality itself is somewhat overloaded, at its core it does revolve around being able to answer the counterfactual proposed by Pearl, however in practice it has been extended to different meanings. When referring to physical system causality, for example, we would be talking about the physics causing what we're observing to be true: a white light passes through a prism and comes out as a 'rainbow', has the underlying physical system causation of the prism bending the different wavelengths of visible light at different angles.

In [85] it was shown that a causal signal does indeed exist in image data, and their work focuses on causal signals between pixels of different object categories. Another causal concept is related to 'objectness' and how our minds understand the concept of objects [124]. Objectness refers to how we understand that objects inherently have certain properties defined in [62] as: cohesion, continuity and contact. Cohesion refers to us understanding objects as a consistent organized set of parts that move together and have a boundary. For example, we wouldn't expect a car's tires to suddenly switch to the middle or the top during its drive, we know that the car is somewhat rigid in its composition of parts. Continuity refers to our inherent knowledge that objects cannot phase in and out of existence, and hence they must move on a predictable path unless modified by something else. Finally, contact is rooted in our physical system knowledge that in order for objects to influence each-other's motions, they must touch. These properties don't hold in all domains, for example, magnets influence metals and other magnets without touching, however as presented in [62], they are usually the properties we learn first as children (hence the fascination with peekaboo) and could serve as a good first step for intelligent machines.

2.4.2 Computer Vision Tools for Conceptual Understanding

In this section, we will go over literature describing deep learning tools that either have been designed specifically in a human-like fashion, or we find can be useful to apply one or more of the three areas of cognition (compositionality, language and causality) mentioned in the Section 2.4.1. We find scene graphs [55] as a candidate that potentially can bring all three of these cues into computer vision, and explore them in depth in section 2.5. However we spend a significant portion of this thesis discussing some of the shortcomings with current scene graph approaches that must be mitigated before they can be fully used as a tool for conceptual understanding (Chapters 4 and 5).

In this section, however, we briefly discuss other approaches that also directly or indirectly utilize the conceptual cues humans use in computer vision. First we discuss generative models that pertain most closely to applying compositionality, and specifically a useful internal representation of objects, but could also be internally modelling causality. We also discuss uniquely supervised approaches that also can allow for utilizing different cognitive cues, depending on the approach utilized. We note that even in cognition, these areas are not easily disentangled, and while we try to evidence our thought process for why each approach may be the best suited to incorporate a certain cue, we cannot be certain of this until we explore these tools experimentally.

2.4.2.1 Architecture Design Decisions

There has been a vast number of recent works that identify the aforementioned drawbacks of deep learning for image understanding, among other drawbacks and aims to rectify them using design decisions in the networks themselves.

In [76], the disparity between performance of bounding box classification approaches utilizing dense sampling versus those utilizing two stage approaches (ROI detection then classification, such as RCNNs) was remedied by the use of the proposed focal loss. This modified loss function is specifically designed to favor modifying weights based on more discriminitive examples of negatives, where the network is failing, over commonly seen ones that the network already can easily classify. This novel loss puts more value on salient and helpful examples, allowing the network to implicitly modify how it learns to suit the data it's looking at without relying on training data to do that.

Object context can relay a lot of information, and some approaches are designed around utilizing this idea. An early example of this is Pyramid Scene Parsing Network [164] which utilizes dilated convolutions and augments CNN extracted features with ones at multiple

scales showing an improvement in performance. The authors of PSPnet argue that while this receptive field was theoretically being reached in later layers of other networks, the networks were still unable to capitalize on it, and their method explicitly allows networks to do just that. The authors of [160] argue that in reality these types of networks attempt to capture contextual information, but still don't do it explicitly. Instead they propose an explicitly defined context encoding module that finds the global context of the image (for example: indoor bedroom) and scales relevant classes according to the detected scene. This context encoding module is added towards the end of inference and right before the final prediction. Their approach showed an improvement in segmentation performance, notably improving segmentation of smaller objects in the scene. Finally, object contextual representation, proposed in [154], attempts utilize the context of a pixel within the object it belongs in as a more robust representation of that pixel. The method augments the feature map of every pixel by utilizing aggregated features from the region of this pixel found using a soft-segmentation, this theoretically improves a pixel's representation because it now shows it in the context of the object it is supposed to belong to, and this improvement is shown experimentally. Context is an important visual cue for humans, and it's one we rely upon to make our decisions, empowering networks to reason about the context of the images they are seeing promises to improve their performance.

Since supplying pixel-level supervision can be prohibitive, the idea of 'weak supervision' gained traction with deep learning researchers. In [98], the authors show that by utilizing an expectation maximization approach, pixel level outputs can be deduced from image-level labels. In a sense, we, as humans perform a similar action, as when shown multiple pictures where the same unknown object exists, we simply find the one object by deducing which object occurs repeatedly in all the images. In learning to segment every thing [49], see figure 2.5, Hu et al. take the weak supervision idea even further. The authors propose an extension to Mask R-CNN [44] that goes beyond the classes that have mask-annotation training data. They propose training a weight transfer function that is designed to transfer weights used in detection to weights to be used for segmentation, allowing them to train with partially supervised data (out of 3000 classes, only 80 have mask annotations, and all have bounding box annotations). The work implicitly recycles visual cues used for detection to be used in segmentation, however in a class-agnostic fashion. They show that this yields a qualitative and quantitative benefit, where they show that their model outperforms one trained with grab-cut, and importantly does not compromise performance on the 80 classes with full instance mask annotations. Essentially, they utilize as much as they can from their bounding box data by allowing a network to reason about how it should be transferring its weights between the two tasks of detection and segmentation.

Tensormask [17] also moved the field of instance segmentation forward, yet tackled the

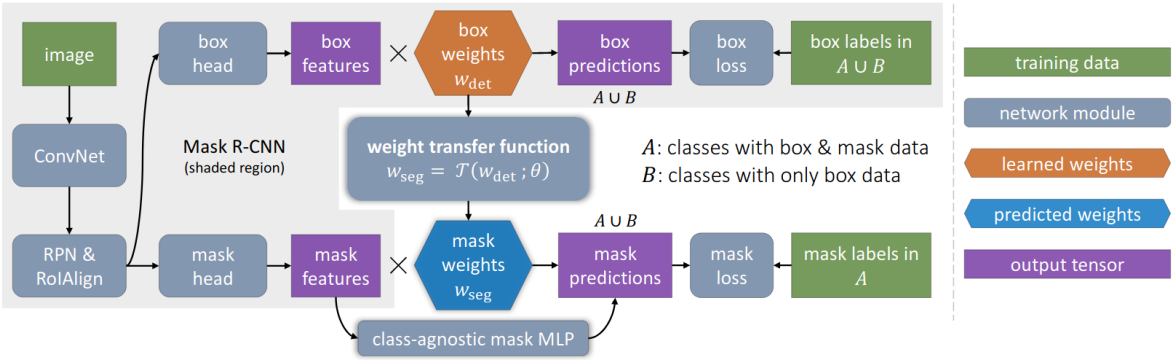


Figure 2.5: A schematic of the Mask$^{\text{X}}$ R-CNN architecture proposed in [49]. The key to its success is is the weight transfer function and class-agnostic mask MLP additions that allow for instance segmentations on classes that only have bounding box annotations.

problem in a different way than R-CNN based methods. Where R-CNN based methods employ a two step process for their instance segmentation, consisting of first performing object and bounding box proposal generation, followed by a segmentation inside a refined subset of the proposals, Tensormask offers a single stage alternative. On one hand the R-CNN based methods allow for a simpler representation that may be easier to engineer with, however Tensormask's geometrically relevant formulation could be more conceptually similar to humans and could be the favourable approach moving forward.

There is a growing body of work improving on image segmentation using novel and promising techniques beyond those we list in this section. We will discuss the utility of generative networks, which have also shown promise when applied to image segmentation in a later section. Specifically, we discuss networks such as PizzaGAN [97] and MONet [7] that approach the image segmentation problem by utilizing the representations learned within their architectures as opposed to generating new training data. These approaches, along with the ones discussed in this section, are a clear indicator that research in deep learning is moving towards making more robust and realistic (in terms of data and processing requirements) architectures. One of the best places to take inspiration from is the way humans approach this problem, and how we utilize our conceptual understanding of the world to perform tasks such as segmentation.

2.4.2.2 Coupled Problems

We view scene graphs as a well developed tool that could aid in incorporating some of the ideas of cognitive science into deep learning for computer vision, as well as our first proposed approach for incorporating conceptual information into deep learning. We are also interested in another promising approach: multitask learning. A pop culture inspired explanation of multitask learning is given in [112] and references The Karate Kid. In the 1984 film 'The Karate Kid' the titular hero is mentored by his sensei Mr. Miyagi, however he is, to his chagrin, given several seemingly unrelated tasks at first (e.g., waxing a car), only to later realize that they all in some way aided him in mastering Karate. It's theorized in cognitive science that humans could be employing multitask learning strategies, recently shown experimentally in the case of a human reinforcement learning task in [136], and in the space of machine learning coupling the tasks to be performed may prove beneficial. Multi-task learning in deep learning is thought to improve performance on each individual task since the many tasks act as network regularizers, prompting network representations to gravitate towards more generalizable and useful representations, and away from overfitting on one task.

2.4.2.2.1 Generative Models

In [68], the authors show how a generative model can outperform other deep learning models on the task of character classification. The authors believe this is more closely related to how we, as humans, perform classification (at least on similar tasks). It is clear that a generative approach, when compared to one that is simply modeling pixel-level statistics would be significantly more generalizable, and that is proven experimentally in [68]. Performance on the task of handwritten character classification does not necessarily indicate that a similar method would be able to perform similarly well on a task with natural images, however this method is an indicator that generative methods could be a good tool to explore.

In [97], the authors present PizzaGAN a Generative Adversarial Network that generates images of custom pizzas based on a set of human-supplied commands, it also seems to learn powerful representations that allow it to perform a variety of sub tasks (including segmentation). PizzaGAN solves a problem on a small scale task, but demonstrates that by tapping into generative networks, useful representations can be extracted. For example, PizzaGAN is trained on images of pizza as inputs and a list of toppings as labels, and its task is modifying an image of a pizza using a set of 'add' or 'remove topping' instructions. While the network is mainly a slightly modified CycleGAN [167], the authors frame the

network's tasks in a manner that allows for some useful sub tasks to be learned 'for free', with very limited labelled data. Regarding segmentation, since the network must be able to add or remove toppings, its 'remove topping' mask generator is in fact a segmentation network learned with weak supervision. Another interesting take-away from PizzaGANs is that prior knowledge of the problem in the form of the ordering of the toppings improved the performance of the segmentation as well. While this work is developed on the small and specific task of generating pizzas, it indicates that the generative learning signal, which could even come from weak labels, could have the ability to indirectly train networks to perform on a different, more relevant task.

The symbol-concept association network (SCAN) [48] showcases the power of a generative representation utilizing Variational Autoencoders (VAE). In their work, the authors propose a coupled detection and generation system, where the detection system is trained on synthetic images that belong a closed space of concepts, and SCAN is the image generating VAE that is designed to have a similar latent space to the detection Autoencoder. In brief the authors are able to demonstrate the utility of a specific class of VAEs (beta VAEs) in learning a disentangled representation of the space it is trained on. This allows the authors to show that they are able to impose logical operators (union, intersection, exclusion) to generate images with concepts they haven't seen before all as a result of using this disentangled space. This provides insight into even more deep learning architectures that can be used to model a hierarchical structure or used to incorporate prior information.

VAEs also are heavily utilized in MONet [7], which takes the idea of disentangling individual elements of an image and moves it even further, see figure 2.6. MONet's architecture is designed with the idea that a network will likely perform better if it looks at individual objects separately, and their experiments back this up. The authors present a two part network that is applied recursively throughout the image. The two parts of the recursively applied sub-network are an attention network that generates a mask based on an input image, and sends forward this mask to a VAE that is tasked with reconstructing the image knowing that its loss is only calculated inside the mask. This process is done recurrently, where every subsequent attention network only suggests object masks in areas it has not previously suggested, and the VAEs reconstruct those image subparts. The end result is a disentangled representation of the scene with every VAE outputting a single portion of the original image, which experimentally converges to outputting individual objects. This work offers several key insights and impressive results: the experimental verification that a VAE will perform better and gravitate towards reconstructing single objects at a time, a unique method of applying a recurrent attention process in an image, the ability to perform segmentation and even inpainting all while utilizing the self-supervisory signal of a VAE. These results showcase a strong and class agnostic prior that was built into the network

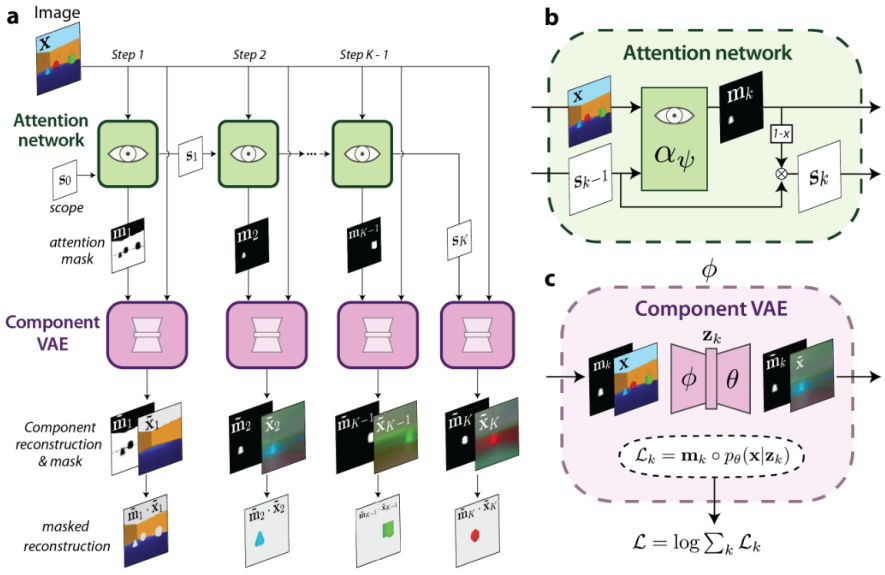


Figure 2.6: The MONet architecture proposed in [7]. Every attention network takes decides on a subsection of the 'scope' to utilize and the VAE is tasked with encoding the image yet is not asked penalized for anything outside its own scope. The summation of all the scope masks is the whole image, hence the combination of all the VAE outputs masked with their scopes is a recreation of the image from its subparts.

(that of disentangling image components and dealing with one at a time).

2.4.2.2.2 Creative Supervision Approaches

In the generative approaches above, the choice of supervisory signal is critical. We would not be able to train such models if those methods required full supervision. In the case of PizzaGANs and SCAN the signal is a weak supervision signal, similar to a classification, yet those networks learn to do so much more than just what their supervisory signal offers. This is especially interesting in MONet, where the whole process is self-supervised because of its creative use of a VAE. There are several non-generative approaches that also make use of unique supervision approaches and bypass the need for full supervision.

One of the most straightforward examples of approaching supervision for the task of multi-task learning is shown in [151]. They show that a single backbone network can generate a feature map representation that only requires a small amount of tweaking to

perform all of the three tasks of image-object classification, bounding box detection and semantic segmentation. The interesting part of the work is that most of the network is trained jointly, i.e., image labels, bounding box labels and pixel level labels all contribute to teaching the backbone network, and during inference only few additional steps need to be taken after the forward pass through the backbone to address any or all of the 3 sub tasks. Since image object labels are cheaper to obtain than bounding box labels which in turn are cheaper to obtain than full pixel level labels, the authors demonstrate that their method can approach the performance of a similar backbone network trained only using pixel-level data with a smaller ratio of pixel labeled images.

We previously discussed the work titled 'learning to segment everything' [49] where the authors used a promising type of supervision. The weight transfer function (a learned network) is used to transfer the filter weights of layers used for object classification into filter weights for segmentation based on a subset of the object classes that have pixel level annotations. The power of the weight-transfer network is in the fact that it learns to understand what 'worked' when transferring these filters in a data agnostic fashion. This unique approach to using the multiple available labels within a single network performing multi-task learning is certainly an out of the box approach that paid off.

Self-supervision is also an interesting approach in image understanding and methods utilizing self-supervision are usually crafted because of a certain prior knowledge and take advantage of their unlabeled data in creative ways [54]. One example utilizing colorization as a proxy task to pre-train networks is presented in [71]. In their work the authors show how a self-supervisory signal of asking the network to predict the colors for a given image (which was converted to gray) can prime the network's filters in a surprisingly effective way comparable to pre-training on labelled data. Generative networks often employ self supervision as well, and it is one of the aspects that makes them so interesting and useful to work with. There have been several works that utilize the self-supervisory signal available in images to solve 'jigsaw puzzle' like tasks, where a network sees a chopped up image and must reason as to where each piece came from [54], and in the same vein, this type of supervision would be a free signal when priming a network to understand the often rigid spatial composition of common objects (a car or a face for example).

2.5 Scene Graphs

Image object detection and segmentation allow us to parse the the visual content of images and automatically determine what exists in those images, scene graphs allow us to gain an additional level of knowledge by classifying the relationships between the detected

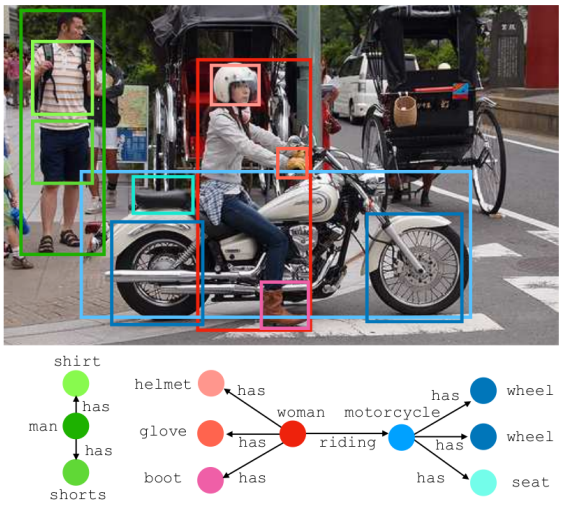


Figure 2.7: An example of a typical scene graph and its corresponding image, taken from [158]. Generating these scene graphs automatically can be quite a challenging task which requires reasoning on which of the detected objects share a relationship, and what relationship that may be.

objects. Scene graphs abstract information in an image into a directed graph consisting of the objects in the image, which serve as the 'nodes' in the scene graph, as well as the relationships between these objects, which serve as the directed edges. The scene graph encodes structural and relational information between image sub-parts that allows for a richer representation and reasoning about the image parts at a higher level, see Figure 2.7 for an example. A scene graph representation of an image is a powerful representation in an of itself, but can also be utilized in a variety of higher level computer vision applications: image captioning [149], visual question answering [117], image retrieval [57], and image generation [36, 55] among others. However before being able to utilize scene graphs and capitalize on the benefits they offer, especially in the context of the conceptual tools they can be useful for, we opted to dig into their current state and understand their shortcomings to be able to build on this knowledge for future works.

2.5.1 Inception and Fundamental Approaches

Scene graphs were initially formalized in [57] where they were manually generated and utilized as a tool for image retrieval. Scene graphs were presented as better suited to describe a 'scene' as opposed to a paragraph of text. Johnson et al. [57] were concerned

with utilizing scene graphs (rather than generating them), and utilized CRFs to connect a queried scene graph with images to find the best matching image. However, the problem of scene graph **generation** from images soon became the focus of extensive research in the computer vision community.

Scene graph generation by iterative message passing (IMP) [146] was aimed specifically at generating scene graphs from images in an end to end manner, where given only an input image they output a scene graph for that image. They first extract a subset of bounding boxes proposed by a region proposal network, and generate visual features for the objects contained within these boxes to be passed to a node Gated Recurrent Unit (GRU), and the visual features for a union box of every two objects to be passed to an edge GRU. Their method utilizes "message passing" between dedicated GRUs for nodes and edges to ensure that the edge GRUs and node GRUs are aware of their contexts. The message for every node GRU is a combination of the hidden state of the previous node GRU along with the hidden states of the GRUs for its incoming and outgoing edges, while the message for every edge GRU is the hidden state of the previous GRU for this edge along with the hidden states of the nodes this edge connects. The authors demonstrate that this message passing improves scene graph generation performance, showing how contextual information can lead to a more robust explanation of the image relationships.

Research on human cognition also indicated that detecting the relationship between two objects and detecting the objects themselves is a 'reciprocal' process [11, 39]. For example, when humans detect *person* and *clothes* we are biased to predict a relationship of *wearing*, and when humans know the relationship is *wearing*, they can also whittle down the space of subject and objects that can share this relationship. VTransE [161] takes influence from this notion, along with work in translation (TransE [5]) to create a lower dimensional 'relation space' where a relationship predicate is a 'translating vector' between the features of its subject and object whatever they may be. This translation vector in the relationship space is consistent for the same relationship predicate and is independent of what the objects and the subjects may be, effectively the nearest neighbor after 'translating' by the relationship vector from the subject feature should be the intended object feature. Visual motif networks (motifnet) [158] are also aimed at generating scene graphs from images in an end-to-end manner. They first generate the object labels only (without generating the relationships), utilizing a bidirectional LSTM to propagate information between the different object proposal stages, once the object/node labels are finalized the edge labels are then generated. The main novelty in this approach lies in how the edge labels are predicted: the authors first query the training set for frequently recurring sub-graphs (motifs), which correspond to object relationships that frequently occur together in the training set (for example, an elephant has a head, a trunk, a leg and an ear). These motifs are incorporated

into the output of the relationship predictor LSTM networks.

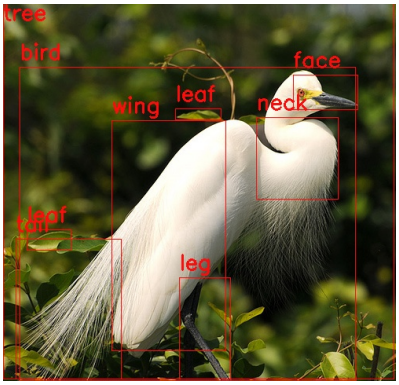
In [130], the authors propose VCTree which, by design, makes use of the parallel and hierarchical nature of relationships. VCTree proposes a dynamically composed and hierarchical tree structure (utilizing TreeLSTM [127]) which allows for a hierarchy of which objects and relationships should be more biased by which context. The tree representation presented in [130] aims to amend the the shortcomings of modelling visual relationships in the manner of motifnets [158], which are too biased by co-occurrence, as well as IMP [146], where the representation doesn't make use of any hierarchical information.

2.5.2 Datasets

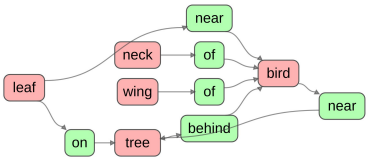
2.5.2.1 The Visual Genome Dataset

The Visual Genome (VG) dataset [64] is a collection of over 100,000 human annotated images that has been used extensively in computer vision research. A main motivation for creating the dataset was to allow for more cognitive-based computer vision research that is focused on image understanding and reasoning, rather than solely image perception tasks such as object detection or image segmentation. VG enables research that incorporates this sort of reasoning such as Scene Graph Generation [158, 129, 130, 24, 59], Visual Question Answering [157, 130], Image Captioning [35], among others [142, 10]. The full VG dataset is composed of a collection of 108K images, along with human generated annotations in the form of class-labelled bounding boxes around the objects in the images, attributes describing those objects, relationships between those objects, as well as question-answer pairs about the images. A sample of some of the kinds of data found in the VG dataset is shown in Figure 2.8.

The images comprising the VG dataset were taken from the YFCC100M [132] and COCO datasets [77] and then annotated rigorously using human annotators crowdsourced though an online platform. In short, labellers were tasked with creating text descriptions of regions in the image, these text descriptions are then grounded into the specific parts they're describing using bounding boxes to ground the objects being described and relationships and attributes being connected to and between the object bounding boxes. The final dataset is comprised of over 3.8 million total bounding boxes classified into 33,877 object categories, these bounding boxes are connected by over 2 million total relationships (classified into 42K distinct relationship predicates), in addition to over 2.5 million attributes describing the classified objects in the bounding boxes (with 68K distinct attributes). On average, one image is expected to contain 35 object bounding boxes, 26 attributes and 21 relationships.



(a) The sample image with labelled bounding boxes.



(b) The labelled relationships between the objects (objects displayed in red, while relationships are in green).

Figure 2.8: A sample of a data point in the VG dataset, the object relations are displayed in the form of a scene graph. Object attributes are not shown for clarity.

The Visual Genome Dataset therefore lends itself very well to the task of scene graph generation [158, 129, 130, 24]. In this task, the vast amount of objects and relationships found in the VG dataset can be a drawback due to severe class imbalance across object categories and relationship predicates. It is common practice across scene graph literature to instead opt for using a subset of the VG dataset, the VG150 dataset [129], containing the 150 most frequently occurring objects along with their 50 most frequent relationships. The final object count in the VG150 dataset is 1,145,398 objects, i.e. the top 150 object classes (out of the 33K classes) accounted for approximately a third of the total bounding boxes. The total preserved relationships in the VG150, which are spread across 50 predicates is 622,705 relationships (these are out of the original 42K predicates that described the 2M original relationships). Overall, this serves to lessen the severity of the inherent class imbalance across objects and relationships, without severely altering the intention original Visual Genome Dataset.

2.5.2.2 Alternative Datasets

The Visual Genome dataset is commonly used across the scene graph community for benchmarking, however there are several other scene graph generation datasets available as well [86, 66, 148]. The Visual Relationship Detection (VRD) [86] dataset is one of the earlier visual relationship detection datasets and predates the Visual Genome datasets, it was commonly used by earlier scene graph literature for training and benchmarking. VRD

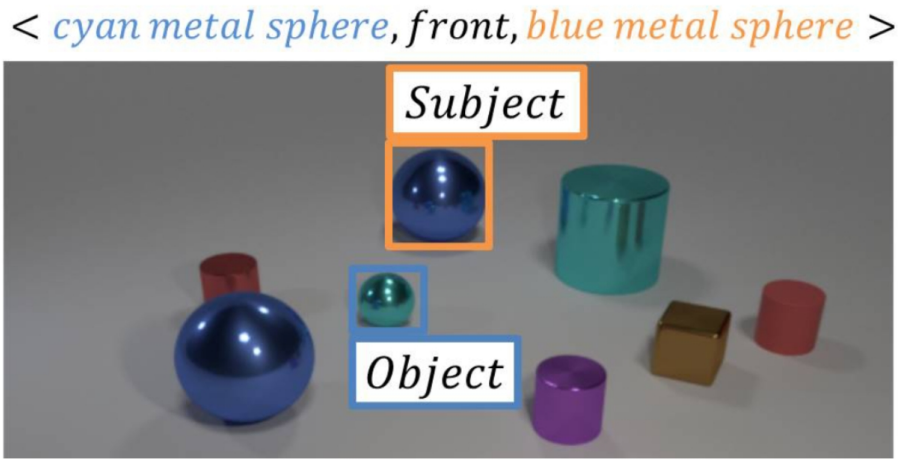


Figure 2.9: A sample image from the CLEVR dataset [56]. An almost infinite number of images and configurations can be synthetically produced using 3D rendering software.

was labelled by computer vision researchers (as opposed to the crowdsourced workers who labelled the VG dataset), however only contains 5000 images. The spatial sense dataset [148] dataset contains around 12000 images with the relationship labels focusing on spatial relationship detection. The dataset specifically aims to provide data with confusing and difficult to infer spatial relationships that can only be predicted by algorithms that have a deeper understanding of images and relationships.

The Open Images dataset [66] is a larger dataset containing annotations for multiple image related tasks (including relationships between objects labelled via bounding boxes), relationship labels in this dataset are more sparse than those found in the VG dataset, and it hasn't been too widely used for benchmarking scene graph generation or relationship detection yet. The CLEVR dataset [56] is a synthetic dataset with rendered 3d images of basic 3d shapes of differing shapes, materials, colors and spatial configurations. Scene graph representations of the images are automatically generated for every image as well, see Figure 2.9. The CLEVR dataset is sometimes used to train and benchmark scene graph approaches [101, 20], however the natural (non-synthetically generated) images in the Visual Genome dataset make for a more challenging and representative dataset and the VG dataset remains more commonly used in scene graph literature.

The previously mentioned datasets all provide a 2D image with 2D bounding box annotations of the included objects along with relationships between those objects. Other datasets focus on relationship detection in videos [114, 113], along with relationship detection with 3D data [2, 61]. These datasets often must be tackled using domain specific

approaches different than the approaches used for the VG dataset.

2.5.3 Evaluation Metrics

Given an input image, scene graph generators are evaluated under 3 different settings:

- **Predicate Classification (PredCl)**: Where the object bounding boxes and the object class labels associated with the bounding boxes are given, hence the scene graph generator must only find the **relationships** between the given objects.
- **Scene Graph Classification (SGCl)**: Where the object bounding boxes are given, but the class labels associated with the bounding boxes are not, so the scene graph generator must infer both the **class labels** as well as the **relationships** between the bounding boxes.
- **Scene Graph Detection/Generation (SGDet/SGGen)**: Where the input image is given without any other labels or information, and the scene graph generator must uncover the relevant **objects** in the image, their **bounding boxes** as well as the **relationships** between them. This is the most challenging setting for evaluation.

A relationship is defined by the subject of the relationship, the object of the relationship, and the relationship between them and is formalized as a relationship triplet in the form of <subject, predicate, object>in scene graph literature. For example, one such triplet observed in Figure 2.8 is <leaf, on , tree>, where 'leaf' is the subject, 'on' is the predicate and 'tree' is the object of the relationship.

Scene graphs are evaluated based on their recall, as opposed to based on their precision. Recall is calculated as: $\text{Recall} = \frac{\text{True Positives}}{\text{True Positives+False Negatives}}$. Whereas precision is evaluated as: $\text{Precision} = \frac{\text{True Positives}}{\text{True Positives+False Positives}}$.

In the case of scene graph generation, 'True Positives' are relationship triplets in the ground truth that were correctly detected, 'False Negatives' are relationship triplets that exist in the ground truth but were not detected, and 'False Positives' are relationship triplets that do not exist in the ground truth but that were 'falsely' detected.

Earlier scene graph literature [146, 161, 130, 158, 130] relied on recall as the metric of choice due to the nature and formulation of the scene graph generation problem, and this continued to be standard practice in the literature (until [129]). Given an input image, the scene graph generator is evaluated on how many of the ground truth relationships it

was able to uncover (its recall) since this aligns more closely with the nature of the task, and also because there may very well be unlabelled relationships in the ground truth that a network could be detecting. Since the datasets used don't exhaustively label all possible relationships, using precision as the metric of choice would effectively be penalizing an algorithm for 'false' detections that are not certainly false.

The relationship predicates in the Visual Genome dataset [64] or the VG150 [129] subset are also not distributed equally. In the case of the VG150 subset, the top 3 most frequent predicates (on, has, wearing) make up more than 55% of the total predicate instances in the dataset. The large predicate class imbalance that exists in the VG and VG150 dataset results in the recall score being quite biased to these predicates and networks trained on the VG150 dataset being much more likely to predict them. As a result, [129] proposes using the mean recall as the metric for evaluating scene graph generation approaches, and has since become the standard comparison metric. The mean recall averages the recall score across every predicate class individually instead of every predicted relationship instance. So the average recall is calculated for every predicate separately first, and then averaged again to get the mean recall which ensures under-represented predicate classes are not being ignored in the evaluation. The defining metric in scene graph literature is the mean recall@K metric. The mean recall@K metric is the mean recall score when the top K scene graph predictions are used for evaluation, so a mean recall@20 would mean the scene graph generator was allowed to predict up to 20 relationship triplets to compare to the ground truth.

Another paradigm for evaluation scene graph performance is whether the evaluation is done with or without graph constraints. The default evaluation mode in the literature is with graph constraints which means only one relationship prediction between two objects in each direction is allowed to be made. This is done so that algorithms are not allowed or incentivised to predict multiple relationships between two objects which may be contradictory (for example, simultaneously predicting 'above' and 'next to'). This forces algorithms to only predict the relationship they are most sure of between two objects (one in each direction). Some literature opts to evaluate and report on network performance with and without these constraints.

Generalizability of the algorithms is also evaluated via the zero shot recall score [86]. Zero shot recall is the recall (and not mean recall) @K specifically for relationship triplets in the test set that have not been observed in the training set. This gives an idea of how well relationships themselves are being 'understood' by the algorithms, since the triplet configurations they are part of are completely novel to scene graph generators trained on the training set.

2.5.4 Progressing Scene Graph Generation

Given the relative recency of scene graph generation in computer vision the fundamental approaches mentioned in section 2.5.1 are still relevant today, and get expanded on by consequent methods. The work in [129] focuses on tackling scene graph generation by addressing the tendency networks have to predict the more common relationship predicate classes due to the imbalance of relationships in the VG dataset. They propose Total Direct Effect, a method to debias how existing scene graph generator predict relationship predicates and is based on counterfactual causality [99]. [129] first finds the relationship predicates predicted for an object pair in an image, then finds the relationships predicted when the visual features of the objects are removed and only the context features remain (the 'counterfactual interference' portion) and compares the two to ensure that the final relationship predicate is more affected by the visual stimuli between the objects rather than the general context. The authors of [129] do acknowledge that general context can be useful in narrowing down relationship predicates, but ultimately causes networks to default to the more dominant relationship classes. This approach is shown to increase performance of existing scene graph generation approaches, especially on the introduced mean recall metric.

Prior information was shown to be successful in improving scene graph generation performance. Implicitly in [146] where the authors formulate the scene graph generation problem specifically to incorporate contextual information (via message passing). The authors of [158] also incorporate their prior knowledge that certain objects' interactions occur frequently enough to warrant biasing their scene graph outputs to reflect these frequently occurring groups of relationships. The authors of [36] take this idea one step further directly incorporating a knowledge database (specifically 'Concept Net' [121]) to resolve ambiguities in the scene graph generation problem by utilizing conceptual and external knowledge, allowing them to improve over existing methods, see figure 2.10. GBNet [156] utilizes a fully connected graph to represent scene graphs, similarly to IMP [146], and refines this graph with iterative message passing. Knowledge bases such as Conceptnet and Wordnet are utilized to add in commonsense knowledge as well, and the message passing algorithm utilizes this external knowledge to refine its predictions. These knowledge bases can be generated in a variety of different ways and offer a powerful cue. However, all scene graph methods (even those relying on external knowledge) still learn an internal representation different than that in the external knowledge base. This memorized representation is not simple to disentangle.

Scene graphs are usually first generated from images, then any following reasoning method is trained and applied on the generated graphs, which is a two step process the

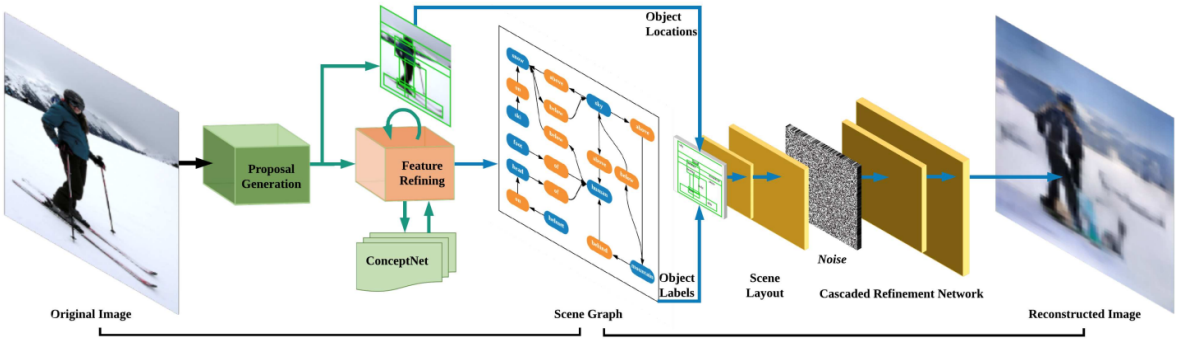


Figure 2.10: A schematic of the scene graph architecture presented in [36] that allows for use of a knowledge bank in to inform scene graph generation. The whole process is regularized by using a generative portion based on predicted locations and labels.

authors of [101] hoped to replace. The authors propose an alternative to the previously seen scene graph formulation, where instead of training a scene graph and following up with a reasoning method, the scene graph generator itself is trained with supervision from the downstream task allowing for a more relevant 'scene graph'-like representation. In a sense, given that scene graphs themselves tend to be intermediate steps, the authors propose a modification where they become trainable with the end-goal in mind instead of being trained independently. Effectively these 'differentiable graphs' are learned and their generated features describe the traditional nodes and the edges in other scene graph formulations, see figure 2.11.

Effectively every scene graph generation approach utilizing the visual genome dataset is forced to represent its objects (and relationships) using bounding box annotations since they are the only annotations provided. In [59], the authors aim to improve scene graph generation by grounding object proposals with 'segmentation masks' as opposed to bounding boxes. Segmentation masks only include the image pixels pertaining to the object itself as opposed to bounding boxes where non-object pixels could be anywhere within the bounding box and in some cases (for strangely shaped objects) may comprise the majority of the area within the box. While segmentation masks are not readily available for the visual genome dataset, the authors of [59] posed scene graph generation as a multi-task learning problem in an attempt to give scene graph generators the ability to utilize the more precise features from a segmentation mask for their object proposals (specifically Motifnets [158] were experimented with). The authors utilize the same object detection backbone (an R-CNN based backbone [44]) as the backbone for object proposals and fea-

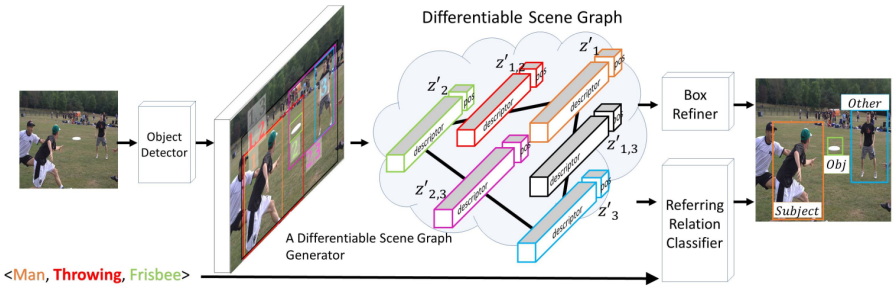


Figure 2.11: A schematic of the differentiable scene graph architecture presented in [101]. The intermediate representation allows for reasoning about the bounding boxes similarly to other scene graphs approaches, however the differentiable formulation allows these types of models to be trained on the downstream task in an end-to-end manner.

ture extraction for scene graph generation on images from the visual genome dataset as well as the backbone for image segmentation on images from the COCO dataset [77] (which includes segmentation mask labels) with different 'heads' performing either of the two tasks depending on the source of the image. Since object labels between the VG150 dataset and the COCO dataset are not overlapping, segmentation masks for the visual genome images were generated by a zero shot segmentation approach that used the lingual similarity (Via GLOVE [100] embeddings) between the VG150 objects with bounding box proposals and the COCO objects' proposed segmentation masks predicted using the object segmentation head trained with the COCO [77] dataset. Effectively the proposed masks for the VG objects are 'transferred' from mask proposals of lingually similar proposed object masks (whose labels come from the COCO dataset). Then, for scene graph prediction on the visual genome images, instead of utilizing bounding boxes, the scene graph prediction networks receive more precise features coming from subregions within the bounding boxes that likely belong to the actual object. This approach [59] was shown to provide a small improvement over the baseline networks that were trained on a single of the two tasks only, improving network performance of both tasks on their respective datasets.

Almost analogously to how 'single stage' approaches differ from RCNN based approaches in object detection, the authors of [81] propose the first 'single stage' scene graph generator opting to utilize a fully convolutional network (FCSGG) approach to generate scene graphs from images. Taking influence from 'part affinity fields' [8] (initially used in 2D pose estimation), the authors encode relationships at the pixel level as directed vectors between the object centers in the image. Additionally, scene graph generation in FCSGG [81] is done simultaneously with object detection (as opposed to sequentially), making the

predicted vector fields that encode relationships independent of the bounding box predictions. Prediction is done using an additional head on a Centernet [166] object detection network. When generating the final scene graph prediction, the densely predicted object bounding box locations (predicted by centernet) and the densely predicted 'relationships' are compared at non-max suppressed pixel locations to generate the final scene graph. While the recall performance of this approach is significantly under the state of the art, it is the first step in a new paradigm for scene graph generation and our approach in Chapter 5 builds upon it.

[24] tackles the bias of scene graph generators proposing illogical relationships (e.g. every person in an image predicted as wearing every item of clothing in the image) that maximize performance but effective produce poor results. EBM [125] tackles some of the issues that arise from using a cross entropy loss for training scene graph generators. The authors note that using cross entropy loss during training of previous approaches translates to networks treating objects and relationships in an image independently as opposed to utilizing some of the structure that exists in the scene graph (i.e., a person cannot be both standing on the beach and surfing in the water in the same image). They propose an energy-based approach that directly incorporates the scene graph structures into its learning framework and show an improvement in performance for the same frameworks when compared to using cross entropy loss. GPSnet [78] use a focal loss to modify the 'importance' of each edge prediction, while also proposing a frequency softening technique to mitigate the long tailed distribution of predicates. They also propose utilizing the visual contexts of the 2 objects in the triplet to further debias the relationship prediction.

In NICE [73] the authors target the Visual Genome data itself, focusing on 'denoising' the ground truth data first before training. They utilize a combination of out of distribution detectors and clustering approaches to refine the VG150 dataset. The refinement is done by finding the unlabelled 'negative' training samples that may actually represent positive relationships, while also modifying the positive labels (and the newfound false negative labelled triplets) by reassigning samples from more vague positive relationships into more the more precise but lesser represented classes. NARE [32] also attempts to modify the input data, and does this by differentiating between 'explicit' labels (such as spatial prepositions e.g., 'on') and 'implicit' labels (such as action verbs e.g. 'riding') in the VG dataset. They then utilize different loss functions depending on which of the two types labels is being predicted. During training whenever an implicit label is seen, a regular cross entropy loss is used, however whenever an explicit label is seen they force the network to predict a smoothed combination of this 'explicit' label with another 'implicit' one and utilize a KL divergence loss to allow for a smoother 2 class prediction on these samples. This approach appears to nudge their networks into predicting the less frequently occur-

ring 'explicit' classes. The authors of HLNet [79] propose and utilize a heterophily-aware message passing algorithm by distinguishing between object homophily, the tendency of objects to have the same labels as their neighbors, and heterophily, the tendency of objects to have different labels than their neighbors, in scene graphs. They utilize a message passing approach, iteratively refining their scene graphs based on the learned homophily and heterophily of both relationships and objects. Many other approaches to scene graph generation were proposed and surveyed in [10] and which we do not exhaustively list.

2.6 Conclusions

As discussed in this chapter, understanding the failure modes of the computer vision tools we use is of paramount importance. Our thesis tackles relationship detection by breaking it up into 3 major portions and aiming to understand the inner workings and failure modes of each of those portions. First, we explore the 'object detection' portion of scene graph generation. We do this in Chapter 3 by observing the behaviours and artefacts present in instance segmentation networks. Instance segmentation offers a dataset which we are able to augment more readily for our experiments, and it is often done using networks build on top of object detectors. Understanding the limitations in instance segmentation networks can likely illuminate the limitations that we could see when object detector backbones are used for relationship detection instead of instance segmentation. In Chapter 4, we experiment with the 'data' portion of scene graph generation and tackle the Visual Genome dataset directly. We leverage the insights we reviewed in this chapter in understanding how human labellers could indirectly be affecting relationship detection done by machines via the training datasets. Finally, in Chapter 5 we tackle scene graph generation from an algorithmic perspective and utilize all our findings to craft a novel representation for use in this, very human-centric, computer vision task.

Chapter 3

The Role of Shape, Content, and Context in Modern Instance Segmentation

3.1 Chapter Summary

The work presented in this chapter focuses on understanding the roles of object shape, its content and its context in object detection and instance segmentation. We systematically quantify the roles of these cues in current deep learning approaches and underscore some of the common pitfalls exhibited across different architectures. The chapter's contributions are as follows:

- We show empirical evidence of shape detection biases in the tested networks, that likely extend to other networks. The representation of object shape is brittle, where networks can achieve a maskAP of over 20% on images with the objects completely removed, but only if their outer contours remain rigidly similar to those in the ground truth. They can also achieve around a 10% maskAP simply using plain object silhouettes on a plain background.
- We also show that non-ROI based instance segmentation networks appear more biased towards shape than their ROI based counterparts.
- We isolate which of the studied cues seem to be the most critical, and find that it is actually object dependent with different object subsets relying on different cues. However, on average content appears to bias networks more than shape and context.

- We demonstrate that blurring the content of objects, but maintaining their sharp outer contours causes very little change in network performance.
- We show that object context plays a smaller role than the other two cues, but plays a non negligible role in aiding performance.

3.2 Introduction

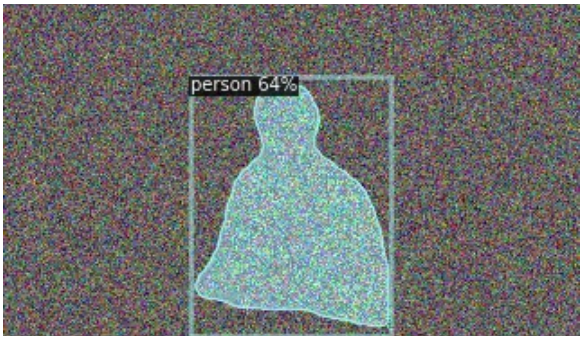
Deep learning networks seem to excel performance-wise on a variety of tasks (e.g., generic object detection [82]), but a significant portion of deep learning networks lack the ability to gain a conceptual understanding of the world simply because their architectures were not designed with that knowledge in mind [69]. By design, and in a nutshell, deep learning models attempt to optimize their feature extractors and classifiers to best suit the task at hand. If certain correlations and conclusions at the pixel level (or other abstraction level) allow for better decision making, these correlations, no matter how brittle they are, may make their way into the networks. This is exacerbated in deeper models with more capacity to learn, and eventually memorize, swathes of correlations that humans are likely incapable of even noticing exist [51]. The optimal feature extractor and discriminator learned on one task and, more importantly, on one dataset is susceptible to bias. Furthermore, performance metrics measured on validation and even test sets are not necessarily the best measure of generalizibility and robustness.

Isolating these biases gives a small peak into the inner workings of these models and can allow for better engineering with them. Until we're able to better understand the inner workings of networks and, more importantly, predict their failure modes before they happen, experiments like the ones we present, along with a growing body of literature [144, 111, 6, 31, 168] remain crucial avenues for insight into these 'black boxes'. Figure 3.1 shows a sample of predictor behaviours that we delve into in this work.

3.3 Related Works

To our knowledge we are the first work to conduct this type of rigorous experimentation on instance segmentation networks, as image-level classification is experimented on in [144, 6, 116], object detection (with a bounding box) is explored in [111], and semantic segmentation is explored in [116]. We further elaborate on some of the relevant literature in Sections 2.3 and 2.4.1.1. Each problem domain differs in its labels and the methodologies

(a) Raw Image



(b) Person predicted due to shape

(c) Zebra predicted due to content despite shape

(d) Person predicted due to context despite content

Figure 3.1: Sample instance predictions on an augmented input image. In (b) shape allows for detection even when foreground and background are replaced with noise (each of different variance). Despite the object shape being of a person, once the content is replaced with that of a distinctive class (c), the predictor favours the content class. Adding in context of an office in (d), then re-biases the predictor.

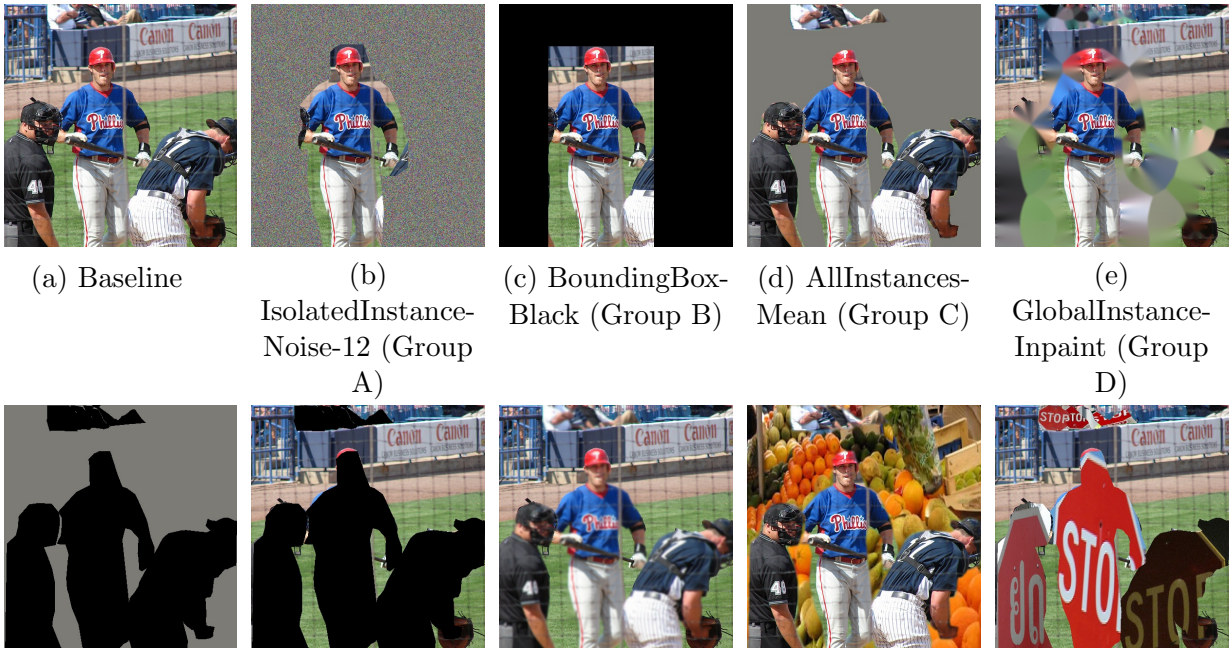
of the deep networks used within it. Consequently, the deep networks all exhibit some task-specific behaviours (seen in the differences in the conclusions of the aforementioned works as well as our own), but also share some commonalities in their behaviours across tasks. This may well be due to these networks 'fixating' on the easiest feature given their own tasks and labelled data. The task of instance segmentation brings with it a specific type of label, the object instance mask, that allows for a variety of novel experiments that we describe in section 3.4, along with a variety of task-specific architectures, and unique-and quite interesting-behaviours which we delve into in section 3.5.

3.4 Experiment Methodology

Our primary experiment focus is on understanding the roles of shape, content and context on instance segmentation networks, and our secondary focus is understanding how differences in network architecture and training paradigm serve to accentuate or de-emphasize these cues. We choose eight different 'networks', all pre-trained on the COCO 2017 object detection train set [77], and evaluate their performance on our specifically designed augmentations of the COCO 2017 object detection validation set. Our choice of architectures and dataset was based on both their extensive use and adoption in the literature [82], though our provided implementation allows for easily extending these experiments to alternative methods and datasets. In order to isolate the different cues, we design a series of experiments which, when taken in comparison to each other, can highlight the effect of a specific cue. Each 'experiment' takes the form of maskAP performance evaluation of a specific network on a different specifically crafted augmentation of the COCO validation dataset.

For every experiment, we evaluate the performance of a network on an augmented version of the COCO validation set. We generate these augmentations on the fly at evaluation time, with randomness (if it's being used in the experiment) being seeded in a way that allows every network to see the same augmented dataset. We perform our augmentations on a class by class basis, then calculate the mean maskAP across all classes to generate the final reported maskAP, this is keeping in line with how evaluation is done on the COCO dataset in general. Note that maskAP is defined as the average area under the precision-recall curves for detection IOU thresholds between 0.5-0.95 (in intervals of 0.05), which is in turn averaged across all 80 object classes in the COCO dataset.

For every experiment we follow the same evaluation procedure described in the official COCO object detection challenge [77]. This ensures that all our results are fair comparisons to the baseline, to each other and to networks that may be tested in the future using our



(a) Baseline

(b) IsolatedInstance-Noise-12 (Group A)

(c) BoundingBox-Black (Group B)

(d) AllInstances-Mean (Group C)

(e) GlobalInstance-Inpaint (Group D)

(f) Silhouette-Black-Mean (Group E)

(g) GlobalHallucination-Black-0 (Group F)

(h) BlurredGlobal-Sharp (Group G)

(i) BackgroundSwap-Soft (Group H)

(j) ContentSwap-Original (Supplemental)

Figure 3.2: A representative subset of augmented images produced by our experiment modes. The experiment mode name is listed first, then the values after the '-' represent mode details. Black/Mean/Noise are the color of the pixels used to replace any removed pixels. Numbers represent how many maximum pixels are added during the dilation operations (when applicable). Other adjectives describe experiment specific modes (e.g., sharp or soft for edge blurring). The group name in parenthesis refers to the relevant results in Table 3.1.

code base. Notably, we pay attention to the number of maximum detections considered during evaluation, and keep it at its default value of 100. In certain experiments (e.g., the Isolated Instance experiments), one input image is augmented multiple times within the same experiment mode (usually to isolate several instances of the same class within an image). We take this into consideration when performing evaluation by concatenating all output proposals on one image during evaluation to make sure that the total number of detections being considered per image is still less than 100. This ensures we don't induce any undue bias in these experiments that may stem from certain images being allowed to have more than the maximum number of detections.

3.4.1 Architecture Specifics

We begin by looking at a Mask R-CNN [44] with a Resnet-50 and FPN backbone trained with various training paradigms. We first consider a fully trained Mask R-CNN with a Resnet-50 and FPN backbone (a 3x learning schedule or ∼37 COCO epochs) which we refer to as **MRCNN50-3x**. For us to explore the effects of training time we also consider an 'undertrained' model (which is trained with a 1x learning schedule or ∼12 COCO epochs) which we refer to as **MRCNN50-1x**, as well as a model trained with a much longer training schedule (400 COCO epochs) and using large scale data jittering (which resizes the image and performs horizontal flips) which we refer to as **MRCNN50-Jitter**. In order to consider the effect of backbone network depth, we also use a Mask R-CNN with a Resnet-101 and FPN backbone that is fully trained (a 3x learning schedule) which we refer to as **MRCNN101-3x**. We measure the effect of a different backbone by also running our experiments on Mask R-CNN with a **Swin-T** transformer backbone [83] which is comparable in complexity to a Resnet-50.

We measure the effect of utilizing non-ROI based detectors by looking at three separate networks. We experiment with Conditional Convolutions for Instance Segmentation (which we refer to as **CondInst**) [134] with a Resnet-50 backbone and a bidirectional FPN, Mask Encoding for Single Shot Instance Segmentation (which we refer to as **MEInst**) [162] with a Resnet-50 backbone, and Segmenting Objects by Locations (which we refer to as **SOLO**) [141] with a Resnet-50 backbone. We choose the checkpoints for all these networks to be the ones trained with a 3x learning schedule on the COCO train set.

3.4.2 Experimental Details

We display a representative subset of the image data produced in each experiment mode in Figure 3.2. We elaborate on the inter-experiment cues being tested in Section 3.5. We experiment with various dilation iterations (which is when a mask of interest is expanded in all directions by a certain number of pixels - denoted by the number in the experiment name). We also run our experiments with different colors (Black/Mean/Noise) used to replace any 'off' pixels. For fairness, we do not fine tune any of the networks on any additional data, and we utilize the author-provided checkpoints for each. They are all trained according to their described training schedule and on the COCO train set only.

3.4.2.1 Experiment Types:

Our experiment modes are as follows, note that we elaborate on the inter-experiment cues being tested in the Results and Observations Section:

- **Baseline:** We baseline every network by evaluating its maskAP on the unmodified image set. See Figure 3.1 for a simple example or refer to Appendix A Figure A.1 for more detailed examples.
- **Isolated Instance:** We take every instance in total isolation by only keeping the image content within the single instance's ground truth mask, and perform inference on that set of images. We test with different background color variants (Black, Mean, Noise) that define the new color of the removed pixels. We test with multiple dilation sizes that increase the area of the object instance masks. These experiments are performed in an instance by instance manner. See Figure 3.1 for a simple example or refer to Appendix A Figure A.1 for more examples.
- **Bounding Box:** We take a bounding box around every instance in isolation. Every bounding box is dilated by a maximum of 8 pixels on every side to add more local context. We test with different background color variants (Black, Mean, Noise) that define the replacement color of the removed pixels. These experiments are performed in an instance by instance manner. See Figure 3.1 for a simple example or refer to Appendix A Figure A.2 for more examples.
- **All Instance:** We keep all object instances of the same class in one image at a time, and remove everything else. We test with different background color variants (Black, Mean, Noise) that define the new color of the removed pixels. These experiments

are performed in an image by image manner. See Figure 3.1 for a simple example or refer to Appendix A Figure A.2 for more examples.

- **Global Instance Inpaint:** For every individual instance we remove all other instances of the same class from its image by inpainting, we also remove and inpaint its own local context defined by the non-instance pixels within its bounding box (dilated by 8 pixels). This experiment is performed in an instance by instance manner. See Figure 3.1 for a simple example or refer to Appendix A Figure A.2 for more examples.

- **Hallucination:** We remove all pixels inside all instances of a specific class (and measure performance on that class). We keep the entire rest of the image in **GlobalHallucination** modes, and keep only the pixels that exist in a bounding box around the instance, looking at one instance at a time, in **LocalHallucination** modes. We test with different color variants (Black, Mean, Noise) that define the replacement color of the removed pixels. We test with and without dilation of increasing magnitude that increases the removed area around the object mask (zero extra pixels means no dilations). **LocalHallucination** experiments are performed in an instance by instance manner, **GlobalHallucination** experiments are performed in an image by image manner. See Figure 3.1 for a simple example or refer to Appendix A Figure A.3 for more examples.

- **Silhouette:** We replace the pixels of all instances of a specific class with one color, and replace the remaining image pixels with another color. Referred to as **Silhouette-*ForegroundColor*-*BackgroundColor***. These experiments are performed in an image by image manner. See Figure 3.1 for a simple example or refer to Appendix A Figure A.2 for more examples.

- **Blurred Objects:** We blur all instances of one class in an image simultaneously (and measure performance on that class). We remove all other pixels and test in isolation in the **BlurredIsolation** modes (using different colors for the replaced pixels). We keep local non-class pixels unblurred one instance at a time in **BlurredLocal**. We keep all the non-class pixels unblurred and in the image in the **BlurredGlobal** modes, with **BlurredGlobal-Soft** blurring the outer boundaries of the object within the background (creating less sharp object contours). **BlurredLocal** is performed in an instance by instance manner, all other modes in this experiment are performed in an image by image manner. See Figure 3.1 for a simple example or refer to Appendix A Figure A.4 for more examples.

- **Modified Background:** For every object class independently we keep all instances of the objects intact and blur all other pixels in **BlurredBackground** mode, or

replace the image background with an alternate image from the COCO val set that doesn't contain instances of that object in the **BackgroundSwap** modes. The difference between the background swap modes is whether or not we blur around the object boundaries in the new images to soften the edge artefacts. These experiments are performed in an image by image manner. See Figure 3.1 for a simple example or refer to Appendix A Figure A.4 for more examples.

- **Content Swap Experiments:** For every object class independently we keep all external contours of the instances of the objects intact but replace their pixels with those belonging to a different class. We measure performance in isolation on a mean colored background in **ContentSwapIsolation**, and within the image context of the original shape instance in **ContentSwapOriginal**. We expand on this experiment mode in section 3.4.2.4. See Figure 3.1 for a simple example or refer to Appendix A Figure A.4 for more examples.

3.4.2.2 Color Modes:

In some of our experiments, we completely mask (or remove) pixels from the original image. Replacing original pixels with black pixels imparts a bias into the output (since black pixels are not 'ignored' by the network but seen like everything else). We 'mask' pixels using four different 'color' modes to ensure that we aren't biasing our results by the color choice:

- **Mean:** For every input image, we take the mean of its pixel values across the 3 RGB channels and then utilize that 'mean value' to replace the pixels we aim to mask. We empirically found this to be the best approach at 'hiding' an object/background. We opted to calculate the mean per input image in the validation set and not use the pre-computed global pixel mean across all images in the COCO dataset.

- **Black:** We replace any pixels we're aiming to mask with black pixels. The reason this doesn't work to 'hide' objects effectively is because networks operate on normalized images which includes a step where the image mean is removed (making black pixels more obvious than 'mean' pixels.)

- **Noise:** We replace the areas/pixels we're aiming to mask with Gaussian noise centered about the image mean.

- **Inpaint:** We use image inpainting to replace the areas we're looking to mask, instead of replacing them with a solid color.

3.4.2.3 Dilation and Blurring Modes

In some experiments we perform a '**dilation**' on the foreground. This entails the using the morphological 'dilation' operation on the masks of the objects belonging to the class we're currently evaluating and using those dilated masks in the experiment as opposed to the ground truth masks found in the COCO validation set. Often this means that more pixels around the object are being added to the foreground. We utilize a 5x5 dilation kernel, so every one dilation operation adds (at most) 2 pixels to the masks. To avoid any confusion, when listing our experiments and results the number used represents the maximum number of pixels being added via the dilation operations (this number equates to $2 * number\_of\_dilation\_operations$).

We denote some of our experiments as **-Sharp** or **-Soft**. These adjectives refer to whether the edges of the object instances are kept 'sharp' (i.e., jagged like the original COCO ground truth masks), or whether they are softened to reduce observed shape bias. We soften edges by blurring gradually, which mostly affects a 2 pixel region outside the object boundary. Furthermore, whenever we blur the foreground or background, we utilize Gaussian blurring with a fixed 9∗9 kernel size. As mentioned in the main body of the paper, increasing or decreasing this kernel is likely to have effects of performance but measuring its effects was outside of the scope of this paper. This measurement can easily be achieved using our code base.

3.4.2.4 Content Swap Specifics

In the Content Swap Experiments our aim is measuring which cue (whether shape or content) the networks seem to rely on more. In every image fed to the network, we keep the object's external contour (its *shape*) intact and replace all its *content* with that of another class. In **ContentSwapIsolation**, we remove all non-object pixels from the background and replace them with the image mean, whereas in **ContentSwapOriginal** we keep the background pixels intact. Since the original images are those which contained the objects of the *shape* class, the background (and context) is that of the *shape* classes not the *content* classes.

3.4.2.4.1 Aspect Ratio

We replace the *shape* object's content in two different ways:

1. Morphing the *content* object's aspect ratio (with experiments referred to as **ContentSwapIsolation -Morph** or **ContentSwapOriginal-Morph**): In this mode, for every *shape* and *content* pair, we resize the *content* object so that its bounding box is exactly the same shape and size as that of the *shape* object. The content object's aspect ratio is not preserved. We then only take the *content* pixels that fall into the *shape* object's mask and remove the rest of the pixels in the resized *content* bounding box. Note these are the main ContentSwap experiments we refer to and rely on in our discussions.

2. Preserving the *content* object's aspect ratio (with experiments referred to as **ContentSwapIsolation -Aspect** or **ContentSwapOriginal-Aspect**): In this mode, we **scale** the content class object while preserving its aspect ratio until its bounding box fully overlaps with the shape object's bounding box. After scaling we take a random crop of the content image that is vertically always taken from the **top** of the content image but is horizontally randomly chosen[1]. We then only take the *content* pixels that fall into the *shape* object's mask and remove the rest of the pixels in the resized *content* bounding box.

Note that in both these replacement methods we only select replacement objects that are of comparable mask size. Therefore a 'small' *content* object is never used to replace a large *shape* object. This limits the amount of scaling and deforming that needs to be done in either of the two content replacement modes and ensures that *content* objects aren't being deformed too severely.

3.4.2.4.2 Evaluation Modes

We run the content swap experiments across all different *shape-content* combinations. For every *shape* class, we augment the same images with every other *content* class in the COCO dataset. We measure and report two separate maskAP performances: performance on the *shape* class and performance on the *content* class:

[1]Our reasoning for choosing not to take a completely random crop is somewhat experimental (finding that completely random crops often don't show the same information even within the same class). We opted to only take top crops that are random horizontally, and this is certainly a big assumption that may have effected the output results in this mode. However this can be easily modified in the source code to measure for **where** content crops are most effective through multiple different experiments (top, bottom, random,... etc.) that require rerunning the same experiment with these different crop locations and were out of our scope.

1. Shape class maskAP (denoted by a '**-ShapeAP**'): For every *shape* class we end up with 79 different maskAPs (80 classes minus the shape class), one for every *shape-content* combination. In this evaluation mode, the maskAP is a measurement of how often and accurately the networks predict the *shape* class as the class of the modified object. For every object class, we calculate the shape maskAP as the mean of the maskAPs across all images where this class's *shape* was used.

2. Content class maskAP (denoted by a '**-ContentAP**'): Similarly to shape, for every *content* class we end up with 79 different maskAPs (80 classes minus the shape class), one for every *shape-content* combination. However, in this evaluation mode, the maskAP is a measurement of how often and accurately the networks predict the ***content*** class as the class of the modified object. For every object class, we calculate the content maskAP as the mean of the maskAPs across all images where this class's *content* was used to fill in other object shapes.

We report on both ShapeAP and ContentAP for *ContentSwapIsolation* modes, whereas we only report on ShapeAP for *ContentSwapOriginal* modes. While we do measure ContentAP in the ContentSwapOriginal modes, we omitted these results from our work. The reason for this omission is that that we calculate maskAPs for the whole image (by utilizing the COCO API) and this means that there is a possibility for the content class to exist in the image background and skew the results of ContentAP. We were able to find a workaround for this in the ContentSwapIsolation and were able to successfully calculate ContentAP, however were unable to do so in the ContentSwapOriginal experiments.

3.4.2.4.3 Random Sampling and Computation Considerations

In total, we run 4 separate Content Swap experiments per one *shape-content* combination (*ContentSwapIsolation-Morph*, *ContentSwapOriginal-Morph*, *ContentSwapIsolation-Aspect* and *ContentSwapOriginal-Aspect*). With 80 different classes, this leads to 6320 different possible shape-content combinations (which is $80 * 80 = 6400$ minus the 80 combinations where shape and content are the same[2]). We found running the four ContentSwap experiments on all the *shape-content* combinations in the COCO validation set to be quite computationally intensive and therefore opted to run them on 33 images per *shape* class. Since images may (and often do) contain more than one object instance, we believe this

[2]We do not run a ContentSwap Experiment where the shape class and the target class are the same, but the content is that of a different instance. Although this would be interesting to explore in future work.

number to be a good trade-off between computationally feasible experiments and statistically relevant experiments.

The randomness associated with choosing the shape and content source images is fixed across networks. Additionally, the *shape* source images are the same across all different content experiments, and the same final augmented images are being shown to every network.

Our aim was to cast a wide net and test across different networks so we opted for this reduction in number of images, these experiments can be run on the full dataset in a specific mode and on a specific network for an estimate of these biases with less uncertainty due to randomness.

3.5 Results and Observations

We present a summary of our results in Table 3.1. Note that due to the large scope of experiments we provide the full range of experiment results in Appendix A as well as the result breakdown by mask size and class. We also provide the results of the ContentSwap experiments which must be tabulated on a class-by-class basis in in Appendix A as well. There are many observations that can be made by looking at the maskAP performance, we list the observations that stood out to us the most.

3.5.1 Objects in Isolation

The isolation modes (Groups A1, A2, A6 in Table 3.1) are designed to baseline the potential best case scenario for a detector. Since the only active pixels in the image are those of one instance of a single object with all others pixels being a plain color (or noise), effectively the 'detection' is handed to them on a silver platter. However, the isolation modes are devoid of any context, and this could potentially impact performance negatively in case networks are using context to aid in detection and classification.

All but one network (MRCNN50-Jitter) had the highest maskAP with the mean color mode, and this is the highest maskAP they achieve across all experiment modes including their baselines. SOLO actually outperforms every other network and achieves the highest maskAP even across all other experiment modes for all other networks. This verifies the claim of [141] regarding SOLO's ability to generate finer masks and is a result of their location-conditioned convolutions. Interestingly MRCNN50-Jitter actually performs best

Table 3.1: Selected maskAP results for all eight architectures. The experiment mode name is listed first, with values after the '-' representing mode details. Black/Mean/Noise are the color of the pixels used to replace any removed pixels. Numbers represent how many maximum pixels are added during the dilation operations. Other adjectives describe experiment specific modes. Group is used for referring to experiments in the main body. Please refer to Figure 3.2 for a more visual description.

		ROI Based					Non-ROI based		
Group	**Experiment Mode**	**MRCNN50-3x**	**MRCNN50-Jitter**	**MRCNN50-1x**	**MRCNN101-3x**	**Swin-T**	**CondInst**	**MEInst**	**SOLO**
	Baseline	37.2%	42.6%	35.2%	38.6%	41.8%	39.4%	34.5%	37.6%
A1	**IsolatedInstance-Black-0**	42.8%	47.0%	43.3%	42.7%	44.7%	46.8%	43.1%	44.7%
A2	**IsolatedInstance-Mean-0**	44.0%	45.2%	44.7%	45.9%	46.1%	49.3%	47.3%	51.7%
A3	**IsolatedInstance-Mean-6**	32.0%	30.7%	30.2%	31.4%	32.4%	29.8%	29.1%	31.2%
A4	**IsolatedInstance-Mean-12**	31.7%	33.3%	29.8%	32.5%	34.8%	30.7%	27.9%	30.1%
A5	**IsolatedInstance-Mean-20**	33.7%	37.4%	31.8%	34.7%	37.0%	33.5%	29.9%	32.6%
A6	**IsolatedInstance-Noise-0**	42.9%	48.3%	41.8%	43.7%	45.6%	48.4%	43.7%	45.3%
B1	**BoundingBox-Mean**	33.5%	36.3%	32.0%	34.7%	37.0%	33.6%	30.0%	33.7%
C1	**AllInstances-Mean**	41.1%	43.7%	40.9%	43.3%	44.2%	48.0%	43.5%	48.1%
D1	**GlobalInstance-Inpaint**	26.9%	28.6%	26.2%	27.2%	29.0%	29.0%	25.6%	28.4%
E1	**Silhouette-Black-Mean**	8.5%	9.8%	9.3%	10.6%	9.9%	12.7%	18.4%	13.2%
E2	**Silhouette-Noise-Noise**	5.3%	9.6%	2.5%	3.4%	2.9%	11.9%	9.0%	4.1%
F1	**GlobalHallucination-Inpaint-0**	2.8%	4.9%	2.7%	3.2%	3.6%	3.8%	2.9%	2.9%
F2	**LocalHallucination-Black**	16.1%	20.6%	15.4%	17.5%	19.3%	19.5%	16.8%	15.0%
F3	**GlobalHallucination-Black-0**	22.7%	27.8%	21.3%	24.6%	26.5%	27.9%	21.2%	23.2%
F4	**GlobalHallucination-Black-2**	10.4%	12.9%	10.1%	11.9%	12.3%	13.9%	10.5%	11.3%
F5	**GlobalHallucination-Black-4**	5.9%	7.6%	5.9%	6.9%	7.3%	8.3%	6.1%	6.4%
F6	**GlobalHallucination-Black-6**	4.0%	5.2%	4.0%	4.6%	5.0%	5.8%	4.2%	4.2%
G1	**BlurredGlobal-Sharp**	36.8%	42.2%	34.8%	38.3%	41.0%	39.9%	34.1%	38.0%
G2	**BlurredGlobal-Soft**	29.0%	33.7%	27.0%	30.4%	33.0%	31.9%	27.5%	30.2%
H1	**BlurredBackground-Soft**	35.8%	40.0%	34.4%	37.1%	40.0%	39.1%	34.2%	38.0%
H2	**BackgroundSwap-Sharp**	32.8%	36.5%	32.5%	33.5%	34.3%	33.3%	29.5%	32.6%
H3	**BackgroundSwap-Soft**	25.8%	28.7%	25.5%	26.3%	27.0%	25.5%	23.6%	25.6%

with a noisy background, and itself and Swin-T (the best performing methods on the baseline) have the smallest increase in performance in the isolation modes. It is possible that both of these networks **likely already incorporate a similar representation in their baseline performance**, hence their improvement is not too great.

Another observation is that **the lack of context doesn't seem to impact performance negatively** in fact without any context at all the networks seemingly performed better on average. However, there are exceptions to this: classes such as *tie, frisbee, surfboard, baseball bat, baseball glove, cellphone, remote*, and *mouse* among others took a performance hit in isolation (see Table 2 in the supplementary). All these classes are usually smaller in area, but they also commonly appear being held or interacted with by people (the most common class in the COCO dataset). This could be an indication of the networks utilizing certain inter-class dependencies for their predictions.

3.5.2 Shape Memorization

In all the IsolatedInstance experiments we inadvertently introduced an additional signal that may have been the cause of the surge in performance: shape. The object instance is inserted using the COCO ground truth provided masks, which are analogous to what the networks were trained with. In the case of COCO, mask annotations are given as a series of connected segments around the object and can be somewhat 'jagged'. By using the mask annotations as-is in the isolation experiments **these sharp boundary edges caused networks to detect the underlying objects more accurately**.

To demonstrate this memorization of object shape, we experiment with detection on the same objects but with a small additional region from their original surroundings added around them (Groups A3, A4, and A5). This only adds a small local region to the objects (e.g., a maximum of 6 pixels in the case of dilation-6), but at the same time modifies the sharp edge shapes into ones that the networks haven't seen before. This additional bit of boundary information drops network performance significantly, with networks performing worse in this experiment than even their baselines. Overall, the larger, longer trained and better performing networks take the most significant relative performance hits (MRCNN50-Jitter, Swin-T and MRCNN101). While every network performs significantly more poorly in this data paradigm, it is noteworthy that the lesser trained and smaller MRCNN50-1x takes the smallest relative hit, even outperforming the larger and more trained MRCNN101-3x network. This could possibly point to network capacity being able to memorize these shapes to a certain extent (in the case of MRCNN101), or the longer training schedules inadvertently leading networks to rely heavily on this feature. This brittle representation

of object shape (stemming from the instance masks) is possibly the most 'easy' [46] feature for the models to learn, leading them to rely on it disproportionately.

3.5.3 Now You're Seeing Things

In order to quantify how much of an influence shape and context have we look at the Silhouette and GlobalHallucination modes (Groups E and F). In both these modes the object itself is masked, i.e., no object pixels are present in the input. Instead every pixel belonging to the object class in question is replaced with either black colored pixels, mean colored pixels or noisy pixels (and in one case, in-painting).

The global hallucination modes serve to illuminate the tendency of networks to predict objects that, effectively, aren't really there. **When object classes are masked and the objects are now texture-less shape silhouettes, networks still consistently predict the object's presence** with non-negligible maskAP. In the case of GlobalHallucination-Black-0 (Group F3), the performance is reasonably high at over 20% maskAP across every network. Masking using mean or noise pixels serves to camouflage the object better, but maskAP is still not negligible. The three different color experiments show a large variation in performance, with black pixels being seen as a bigger indicator of 'objectness' and being less ignored compared to the mean and noise paradigms. This is to be expected as all the networks normalize the input image during train and inference time which involves centering it around a predefined pixel mean.

The question remains, how much of this detection is due to the rigid memorized shape we saw in the IsolatedInstance and AllInstances experiments (Groups A and C), and how much of it is caused by context informing the prediction? In the Dilated GlobalHallucination experiments (Groups F4-F6), we dilate the object masks (and therefore erode their original region in the image) by a varying degree and notice that even in the smallest form of dilation (a mere 2 pixels) maskAP performance drops significantly in all 3 color modes. This drop in performance continues as we dilate further and a 6 pixel dilation is the last time we see significant detection maskAPs. Another interesting finding is that if we replace coloring in the object pixels (which accentuates the sharp mask edges) and instead we in-paint the object locations (GlobalHallucination-Inpaint), maskAP drops substantially. In the in-painted images, object edges are effectively eliminated, however they are replaced with imperfect 'in-painted' replacements that do seem to trigger object detection, though to a very low degree and a very low but not nill maskAP. The main cue informing object detection in this data paradigm is most likely context since both object edges and content are gone.

From the experiment outputs MRCNN50-Jitter is the most susceptible to the hallucination experiments. This could be due to it being the most well performing network in the selected networks to start with, however looking at its closest baseline performer (Swin-T) and the fact that Swin-T seems to be slightly less susceptible to some of the attacks may point to the fact that the MRCNN50-Jitter could have memorized the cues that are causing this result.

Finally, the Silhouette experiments (Group E) further demonstrate the role of shape, even in the absence of all other cues. Most notable is maskAP in the Black-Mean silhouettes when the object is replaced with black pixels, and everything else is replaced with mean pixels. Even the Mean-Black silhouettes have a non-trivial AP, but also interesting is the Noise-Noise silhouettes where the foreground is replaced with noise from a Gaussian distribution and the background is replaced with noise from a Gaussian distribution with a different variance. Even though the silhouettes in the noise-noise images are quite hard to make out for humans (or at least our human colleagues), certain networks are still able to detect them with some accuracy such as MRCNN50-Jitter, CondInst, and MEInst (but peculiarly not SOLO). Figure 3.1 shows an example of this prediction.

3.5.4 Local versus Global Context

Sections 3.5.1 and 3.5.2 discussed the odd behaviour of networks devoid of any context in the IsolatedInstance modes. Network performance seemingly took a hit when a small area of local context was added in IsolatedInstance-6, but that is likely due to the rigid shape boundaries that are 'memorized' by the networks. However as more and more local context gets added in (in IsolatedInstance-12 and 20), this context begins positively influencing prediction performance. This is likely due to local context now playing more of a part in the object detection. MaskAP performance begins tending towards baseline performance, though never exceeding it (unlike in the IsolatedInstance-0 experiments).

The BoundingBox experiments (Group B) supply objects with even more of their local context and performance also improves but still doesn't exceed the baseline. It is worth noting that shape could be a factor here as well, with shape being inferred from the more pronounced difference between the object and the now larger background in its vicinity. The effect of global context can also be seen through the difference between the maskAP performance in the BoundingBox experiments versus that of the baseline. While **networks seem to improve in performance when supplied with local and global context**, their reliance on a rigid representation of shape is still problematic.

Since some images contain multiple instances of the same objects, we designed the

AllInstances experiments (Group C) to see if isolation performance changes when all object instances in an image are shown at the same time. In this mode for every image we keep all the instances of one class of object (as opposed to keeping one instance at a time). Our aim is to measure whether having other instances of the same class would bias the networks into predicting that class, however experimentally that didn't seem to be the case. Network performance on the AllInstances task was comparable to (but consistently lower than) the IsolatedInstance-0 performance with the same background color. **This serves to show that having more of the same object in an image doesn't seem to cause networks to predict it any more readily.**

Local hallucination experiments (Group F2) also highlight the effect of global context on the predictions. The only difference between the local and global hallucination experiments is that in the former pixels outside the area of the object bounding box are also masked (leaving only object shape and local context). Looking at the difference in maskAP between the two experiment modes (Group F2 and F3) shows a consistent positive effect of global information across all networks. Though it still may be that this 'global information' is being utilized by the networks in relationship to the object shape.

It is also worth noting that a similar drop in maskAP is seen between the baseline and the BoundingBox experiments (Group B, where a bounding box is taken around a non masked object). Again in these two modes the removal of global context (in the form of pixels that are outside the object bounding box) affects maskAP negatively across networks. The small difference in results between the ROI-based and non ROI-based detectors also indicates that the non ROI-based detectors are not utilizing context too much more effectively their ROI-based counterparts.

In the GlobalInstance-inpaint experiment (Group D), shape (the sharp object edges) along with the object pixels themselves are kept, and only local context is removed (non object pixels in the object's bounding box). Object instances are viewed one at a time, and other same class instances are inpainted, which, as discussed in 3.5.3, effectively hides them from the detector. Unexpectedly, performance takes a hit over the baseline, even though the shape and content are still aiding the detection. We are unsure what causes the performance hit in this mode, but it is possible that too many new image artefacts are produced after the inpainting and the bounding box removals, that the detectors struggle to still make sense of the instances.

3.5.5 Content Remains Important

As demonstrated by the IsolatedInstance experiments (Group A), taking the rigidly memorized shape out of the equation by dilating the object in isolation illuminates the role of object content only. We define object 'content' as the set of pixels that form the object itself, this set of pixels encodes object texture, color, internal contours, ... etc. After dilating the isolated instances, the shape edges likely aren't too well defined and the network will struggle to utilize them, and the object content remains as the biggest signal in those experiments. Based on the maskAP results, object content is, unsurprisingly, an extremely powerful cue in prediction, as even in complete isolation networks perform relatively close their baselines when only given the content.

The hallucination experiments also serve to reiterate the important role of content. The GlobalHallucination experiments (Group F3-F6) compared to the baseline, as well as the LocalHallucination experiments (Group F2) compared to the bounding box experiments (Group B) are two sets of experiments where the same area of pixels is active in both modes, but one with and one without the object itself. Both sets of experiments show that, again unsurprisingly, **object content is likely the most substantially used signal in prediction**.

The blurring experiments (Group G) also offer interesting insights into what small perturbations on image content can result in. In BlurredGlobal-Sharp we blur the objects in question but keep their outer contour in the image sharp in comparison to BlurredGlobal-Soft where the outer contour is softened by a decreasing Gaussian blur (to remove the previously mentioned shape effect). Our aim is to isolate for the effect of blurring the image content, so we explicitly wanted to make sure shape was removed from the equation. The maskAP in both experiments remains relatively high, notably when the outer contour remains intact network performance is almost exactly similar to the baseline indicating that **a well defined shape plus a weaker representation of content is enough to approximate baseline performance**. This likely says more about shape bias rather than content, however performance drops below the baseline by a non-trivial margin when the sharp edges are blurred and all that the networks can rely on is the blurred content. It may be worth exploring the effect of more severe blurring kernels on performance in future work.

3.5.6 Not All Objects are Seen in the Same Way

Given the biases we've seen so far in our experiments, we aim to measure which bias is 'stronger', and whether a network is biased to different cues that occur in different

Table 3.2: Observed maskAP performance in the content swap experiments for a subset of classes and networks. We present the observed maskAP for the class when only its shape is used (and its content is filled with that of other classes), both in its original image and in isolation. We also present the average maskAP of the object class when its content is used to fill in other shapes in isolation.

	MRCNN 101			MEINST		
Class	*Shape in Original Context*	*Shape in Isolation*	*Content in Isolation*	*Shape in Original Context*	*Shape in Isolation*	*Content in Isolation*
person	18.6%	11.4%	9.4%	15.4%	11.0%	14.5%
bicycle	0.8%	0.2%	6.1%	1.2%	1.0%	8.2%
car	7.3%	3.4%	18.5%	9.2%	8.4%	20.9%
airplane	24.5%	20.3%	6.9%	14.3%	12.8%	11.7%
truck	7.3%	5.1%	19.9%	7.8%	9.5%	21.6%
traffic light	19.7%	12.3%	20.0%	19.2%	35.5%	28.2%
fire hydrant	37.5%	9.0%	10.8%	35.4%	26.5%	15.7%
stop sign	28.8%	5.5%	31.1%	26.7%	31.4%	37.9%
dog	10.8%	2.9%	8.1%	10.8%	7.5%	12.1%
umbrella	25.8%	19.9%	11.3%	20.5%	24.1%	20.5%
kite	32.4%	34.5%	19.8%	23.4%	30.0%	22.0%
baseball bat	33.8%	4.1%	0.4%	20.9%	27.5%	2.6%
surfboard	31.4%	4.2%	5.9%	20.1%	9.0%	10.2%
pizza	3.5%	1.4%	15.6%	4.4%	6.0%	21.1%

objects. In our ContentSwap experiments, we explore how shape, content and context all intertwine and serve to aid (or hinder) networks in confusing examples. We look at 2 sets of ContentSwap experiments and compare their maskAPs. Within each set the data (or evaluation) differs by one cue and the results offer some interesting insight into network behaviour. Table 3.2 presents a subset of the results of the ContentSwap experiments, with the full results found in Appendix A.

3.5.6.1 Set 1 : Shape vs Content Bias

First, we tackle the ContentSwap experiments done in isolation. In these experiments the competing cues are object shape (defined by its external contour and the network bias to those contours) versus object content (the contours/textures internal to the object). Across all networks **a certain set of object shapes seem to bias the network more heavily than others** (e.g., *Airplanes, Kites, Umbrellas*) where on average, despite the fact that another class texture makes up their 'content', the networks still detect these objects as the shape class with a significant maskAP. There are even some shapes that seem network specific in their bias, for example the non-ROI based networks all seem to also favor the shape originating from 'baseball bat' much more than the ROI based networks that don't display the same behaviour. Looking at classes whose content seems to bias detection, we find that land-based transport classes (*bus, car, truck, train*, but notably not *bicycle*)

along with food classes (*pizza, cake, donut*, but peculiarly not *carrots*) and even *stop signs* are among the most biasing 'content' classes. Some objects (e.g., *person* and *traffic light*) seem to even have the ability to bias all networks via their shape and their content. Even more peculiar, street level categories (specifically *stop sign, fire hydrant* and the aforementioned *traffic light*) have this double shape-content bias much more pronounced in non-ROI networks. It is critical to note that on average, across all networks and all classes, content appears to bias networks more than shape.

3.5.6.2 Set 2: Adding In Context

As discussed in section 3.5.1 a specific set of objects seem to depend on context for correct detection. Looking at these same classes (e.g., *baseball bat* and *surfboard*), the same trend emerges when we include and remove context from the ContentSwap experiments. We notice this subset of classes, which had a low maskAP in the isolation modes (both with their original content and their modified content), now display a high maskAP once context is factored in. Effectively **these objects could be more reliant on context than on content for their predictions**. Furthermore most other classes get better detected based on shape after context is added in. The tableware classes (e.g., cup, fork, knife, spoon, bowl) whose content has been modified seem to get significantly better detected within context than without. It's worth noting that the context added in is of the shape-class object (since the instances that have their context replaced exist in those locations in the images).

3.5.7 Background Signals Don't Change Too Much

Taking influence from [111], we attempt to quantify the effect of background signals on the overall quality of detection and classification. This is done in the two background swap experiments (Group H), the only difference between them being whether the object instance's outer contour is blurred in the new background image (to mitigate the shape bias effects). The results don't show an over-reliance on object background, with performance remaining relatively steady in the BackgroundSwap-Sharp mode, but taking a hit in the BackgroundSwap-Soft paradigm. We note that the reason why the maskAP appears to be significantly lower in the blurred edges paradigm is due to us opting for a fixed size blurring across all classes and instances. This lowered the detection accuracy of small area instances significantly, with medium areas getting somewhat effected as well, and large area instances holding steady.

As expected, not all objects behave similarly when background changes. The same set of objects that networks struggle with due to lack of context in section 3.5.1 (e.g., *tie*, *frisbee*), they also struggle with due to uninformative context in the background swap experiments. This trend is seen in all tested networks regardless of architecture. **Overall an adversarial background appears to play as much of a part in damaging detection performance as the lack of context** seen in section 3.5.4. These findings are somewhat different to those in [144], where in their case background and context played a much more important role in detections. The likely reason for this difference in network behaviour is that their image level classifiers were trained with image level categories that were not as severely biased to object shape as instance segmentation networks since they hadn't seen instance masks.

3.6 Conclusions

We quantify the effect that object shape, content and context play in instance segmentation networks of various architectures and training paradigms. Our experiments allow us to disentangle the effects of shape, content and context from each other. Overall, we did not observe too many unexpected performance differences between differing architectures. The non ROI-based approaches seemingly had a stronger bias towards shape which aided them in some of our experiments that included the sharper shape contour, however this may be an undesirable behaviour. The larger Resnet-101 backbone as well as the Swin-T backbone both behaved predictably relative to the baseline Resnet-50 backbone with no major performance shifts across experiments. Swin-T does appear to have a more refined and possibly better understanding of objects as demonstrated by its better relative robustness to certain shape attacks where it maintained its good performance when compared to the jitter-augmented Mask RCNN. The jitter-trained Mask RCNN with a Resnet-50 backbone appeared to have 'memorized' a lot more of the data and behaves accordingly (e.g., by having the highest performance in the noise-noise silhouette experiments), this behaviour likely does cause it to exhibit good maskAP on the validation and even test set (where it always evaluated using the same rigid external contours), however this performance improvement may be a result of memorizing these biases and may not translate well to other datasets.

It's unsurprising, even comforting, that networks more or less perform the same across our different adversarial experiments. They usually all fall into the same traps, and excel at the same classes on the same tasks. They all eventually converged to similarly performing final architectures, and since all the cues and signals we explored came from the data, the

networks all zeroed in on the same overall cues and optimized their performance based on them.

Chapter 4

The Topology and Language of Relationships in the Visual Genome Dataset

4.1 Chapter Summary

This chapter focuses on exploring the topology and language of visual relationships in the Visual Genome (VG) dataset, the most ubiquitous scene graph dataset, and how these affect the learned scene graph generation models. The contributions are reiterated:

- We show that the human labelling bias stemming from human representations of relationships creates asymmetric relationship labels that span the whole VG dataset
- We show that the lack of inverse relationships in the VG dataset leads to a lack of it in the learned models
- We utilize 2D topology to both shed light on the lingual vagueness of certain classes and show that some of these classes can be understood differently based on their topological configurations.
- We show that when reduced to more lingually and topologically well defined spatial relationships scene graph generation algorithm performance improves tremendously, but scene graph generators are still far from perfect.

4.2 Language and Inverse Relationships

In language, spatial prepositions often have inverses, which can serve as a dual (but opposite) representation of the same physical phenomenon being observed. For example, if a table is 'to the right of' a person, it is immediately understood that the person is 'to the left of' the table. Several of the 50 predicate classes that exist in the VG150 data set have linguistically 'inverse' relationship predicates within the set as well. For example, the predicates 'behind' and 'in front of' are both in the VG150 predicate set. It follows that if a subject-predicate-object triplet of subject A-'behind'-object B exists, we would expect to see the inverse triplet of subject B-'in front of'-object A. One very commonly occurring predicate, 'near', could even function as its own inverse.

Figure 4.1 shows a heat map of how often two relationship predicates share an inverse relationship in the VG dataset. An inverse relationship exists if the same two bounding boxes (containing the same specific objects) share two relationships, with one in each direction. In other words, one inverse relationship exists between predicates 'above' and 'under' if for a specific pair of objects the objects are linked by the triplet subject A-under-object B as well as subject B-above-object A in the dataset. Note that two objects may have multiple relationships connecting them. The full heatmap of inverse relationships between all 50 predicates can be found in Appendix B.

We notice that inverse relationships do not form a significant portion of the relationships observed in the VG dataset. In fact, even the expected inverse relationships between linguistically inverse predicates are not at all frequent. Predicates 'under' and 'above' (or 'under' and 'over' whose result can be seen in Appendix B) don't share much of an encoded inverse relationship, in fact 'under' seems to share a stronger inverse relationship with 'on', however that is likely also due to how over-represented the predicate 'on' is in the VG dataset. See Figure 4.2 for an example.

The work done by Landau and Jackendoff [70] on human spatial cognition touches on a relevant issue. They describe the 'asymmetry' in the way humans form spatial representations, where these asymmetries come from many factors, including that certain objects are more likely to be the 'reference point' based on size or relevance or saliency. Even the more apparently 'symmetrical' **spatial** predicate relations tend to become asymmetric in our reasoning by virtue of how humans form their own spatial reasoning. Given that the VG labels are generated by human annotators, there is an asymmetric skew that will inevitably exist in the resulting labels which is the root cause for why these inverse relationships do not exist. For example, of the over 243,000 relationship triplets that include humans in the VG150 dataset, humans are the subjects in approximately 84% of those relationships, while they are objects in only 19%.

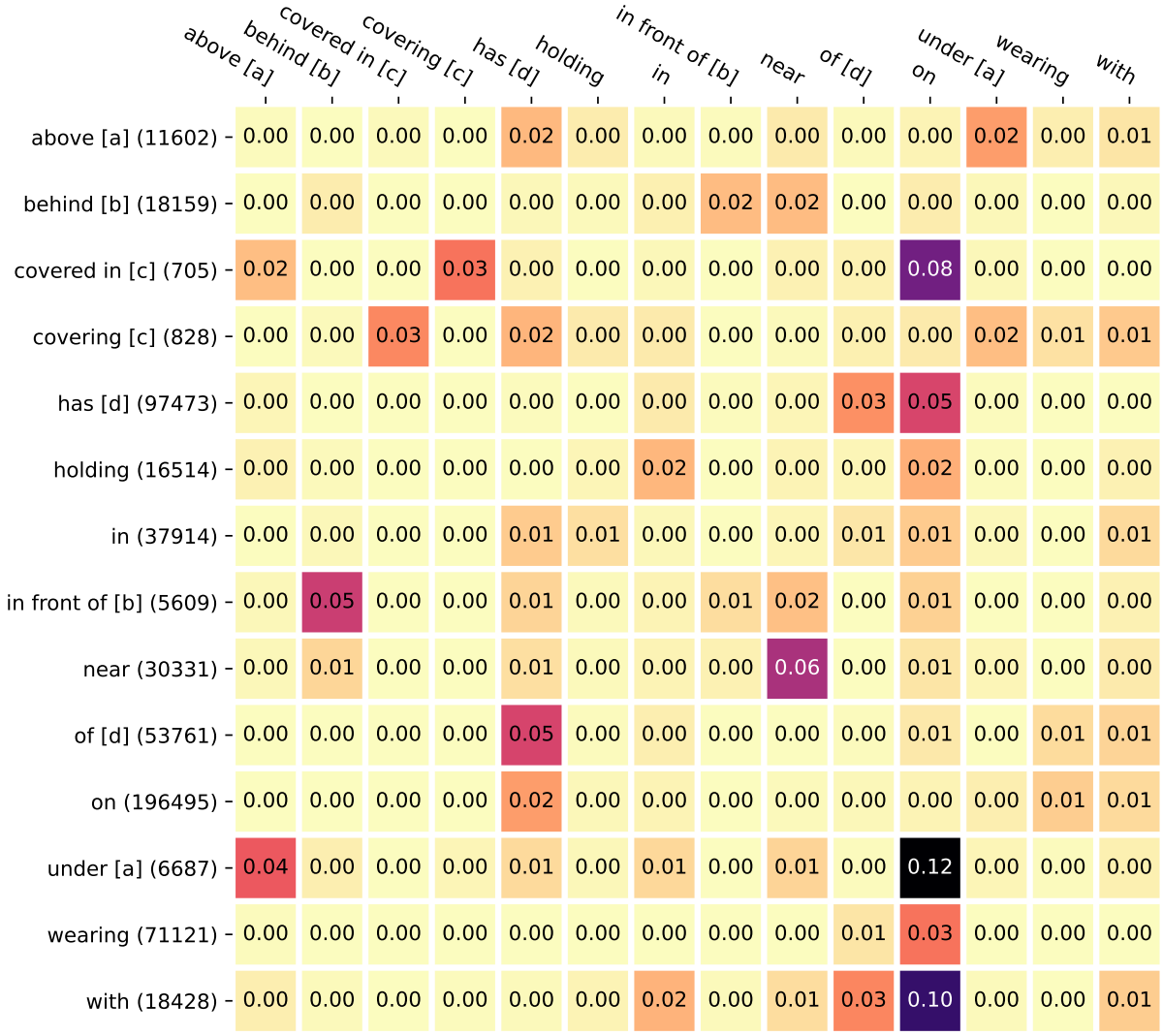
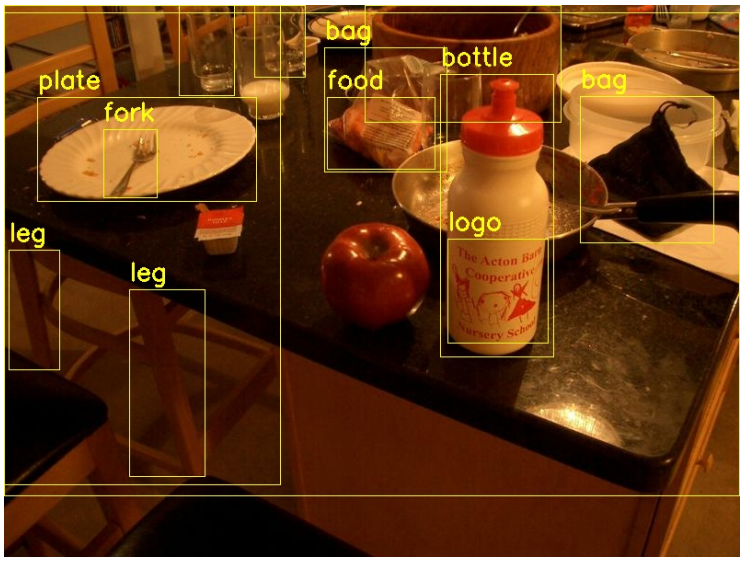


Figure 4.1: A heatmap of the occurrence of inverse relationships for specific predicates. The letters in square brackets indicate which predicates we expect to be inverse pairs, the numbers in parentheses are the total occurrences for this predicate in the dataset. For every row the value in the heat map reflects the ratio of: (inverse relationship occurrences of the row predicate with the predicate in the column) to (total occurrences of the predicate in the row).



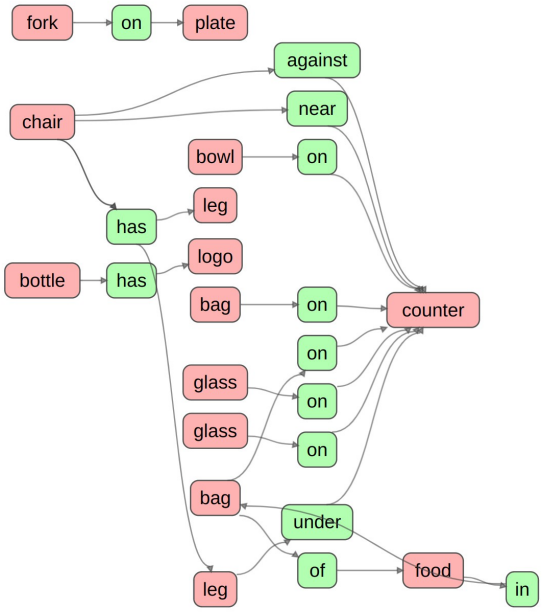


Figure 4.2: Inverse relationships are rarely seen in the VG dataset and relationships usually occur in one direction. For example, while the bottle *has* a logo, the logo is not labelled as being *on* the bottle. This is among other missing inverse relationships in this image alone.

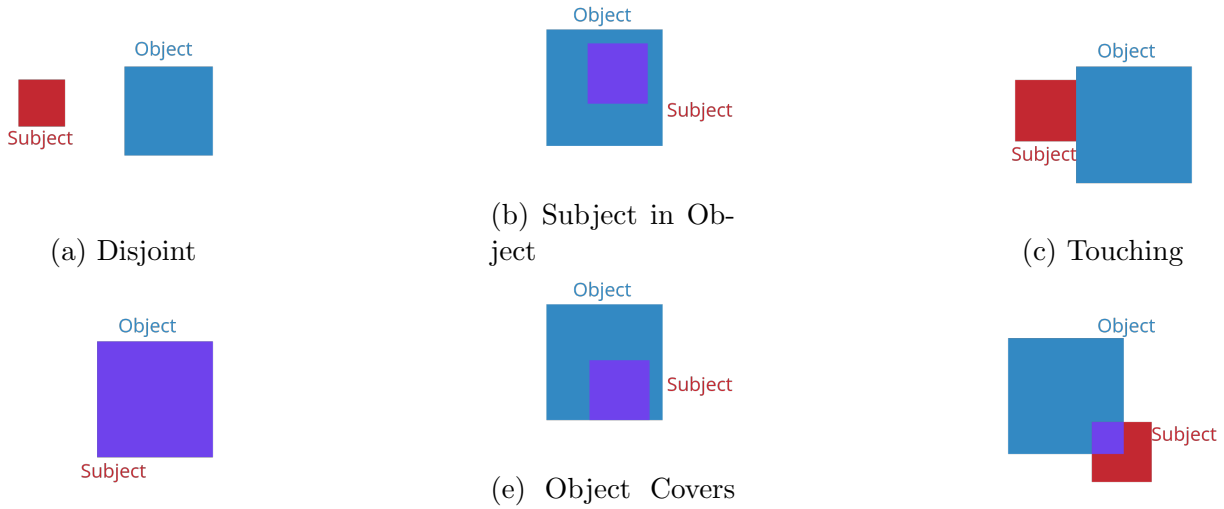
This asymmetric skew is to be expected in human conversation and description, mainly because human reasoning can understand the inverse relationship immediately and it does not need to be explicitly stated. However, this will not be the case for learning algorithms who don't have an existing knowledge of the world, or of the verbal semantics of the relationship predicates they are predicting. A small percentage of inverse relationships existing in the VG150 dataset, even for the predicate classes where we expect them to exist, could likely hinder the ability of learning models to understand these relationships. A potential solution to this could be in the form of data augmentation (for a data-driven solution) or even prior knowledge about these inverse relationships being given to the learning algorithms utilizing this data. Alternatively, inverse relationships could be used as a metric for measuring generalization performance of learned models, especially if certain models were shown to be able to piece together these inverse relationships without explicitly being told about them, or incentivised to learn them.

It is worth noting we also measured co-occurring 'forward' relationships between the same two objects i.e. two objects related in the same subject-object configuration but with different predicates. This measurement yielded no results of interest, as these relationships turned out to be very rare.

4.3 Topological Relationships

The language that creates the relationship triplets may be biased by how humans view and reason about the world, which makes the bounding boxes that also define these triplets worth exploring as well. These bounding boxes are the smallest 2D image axis-aligned rectangles that can hold the object they border and they lend themselves well to a 2D topological analysis. Topological relationships [26, 19] can be determined between two 2D areas, and the topological relationship can be classified depending on the configuration between the borders and the interiors of these areas. Figure 4.3 describes the possible topological relationships between two 2D areas.

A topological analysis of the bounding boxes found in the VG dataset sheds light on the relationships in the dataset. Where our language and how we describe a relationship can be influenced by our cognitive biases, observing the topological relationships between the bounding boxes can give us an understanding of what a certain relationship is prioritizing. They can inform us on whether the subject or the object is the more 'dominant' for a given predicate class as well as validate whether the downstream task of scene graph prediction that utilizes the features in these bounding boxes is being built on valid data. Since several scene graph generation approaches [10] operate by taking the union or intersection



(a) Disjoint

(b) Subject in Object

(c) Touching

(d) Equal

(e) Object Covers Subject

(f) Overlap

Figure 4.3: Topological relationships [26, 19] visualized. Note that two additional relationships exist Object in Subject (similar to (b), but with object and subject reversed), as well as Subject Covers Object (similar to (e) but with the object and subject reversed).

of the detected object bounding boxes to predict the relationship predicate, a topological perspective on how these bounding boxes are related in the VG dataset is quite relevant.

Furthermore, we analyse the dominant directions in which these relationships are occurring. These directions are found by analysing the location of the object relative to the subject when they are linked by a specific relationship predicate. For triplets with predicates describing spatial prepositions, such as [subject, 'above', object], we expect to see the object always being towards the south of the subject. This analysis also sheds light on whether more frequent and more vague predicates (such as 'on' or 'has') are exhibiting any regular directional relationships.

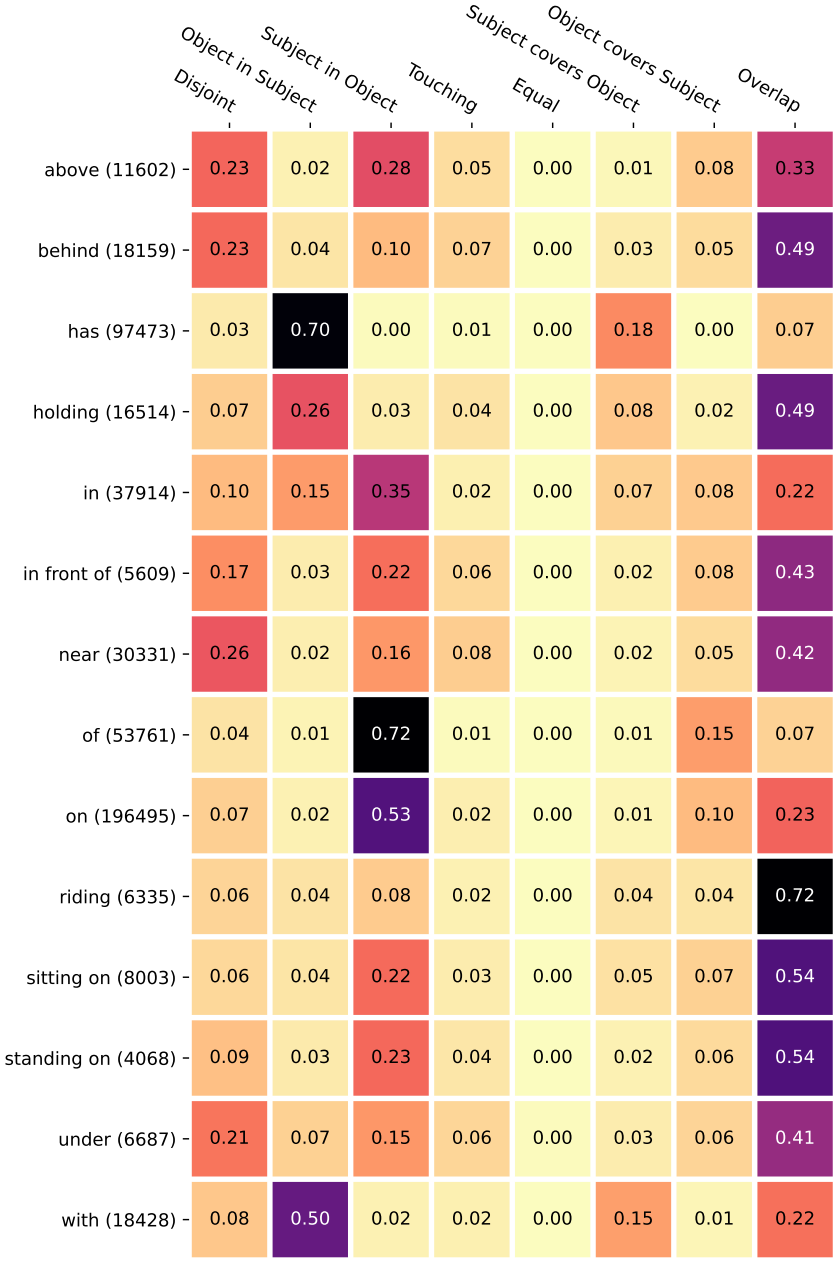


Figure 4.4: A heatmap of the occurrence of topological relationships between bounding boxes related by specific predicates. The values shown in the heatmap are the portions of the total occurrences of the row predicate that exhibit the specific topological configuration in the column. The values in parenthesis next to the predicate names are the total occurrences of that predicate.

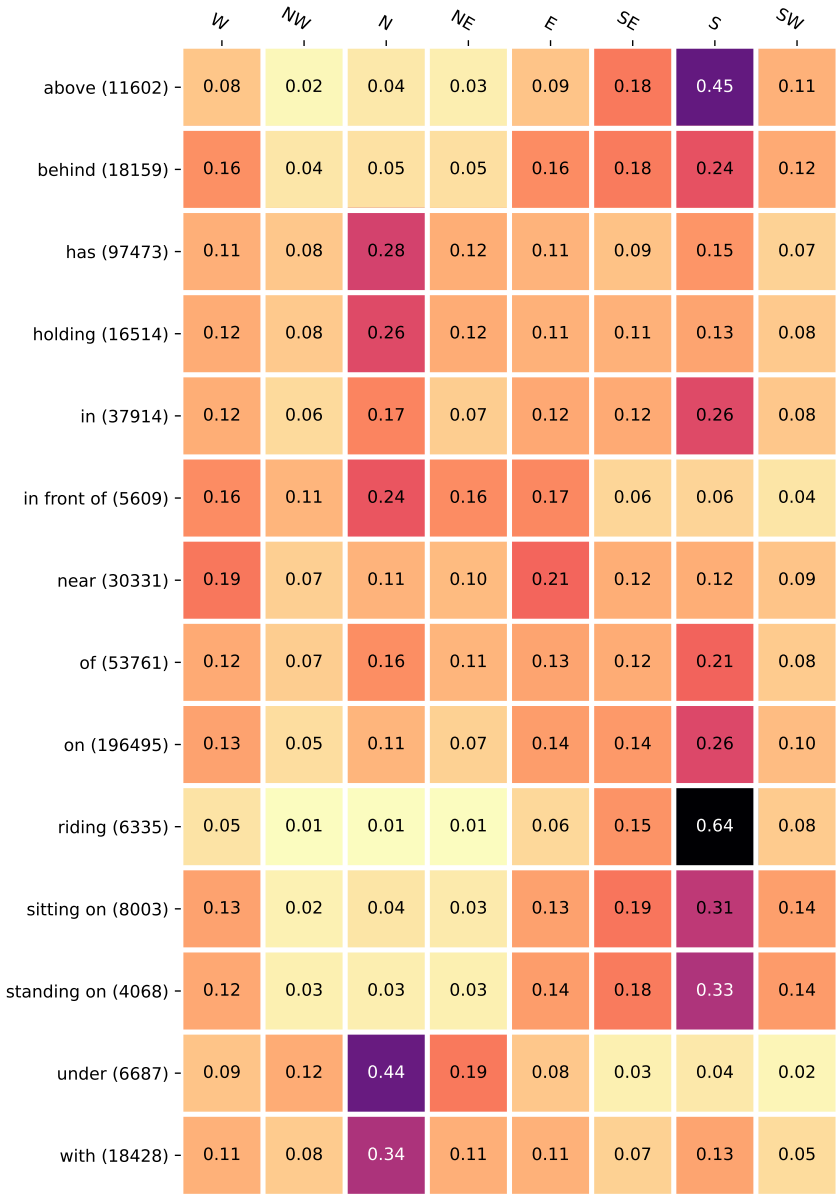


Figure 4.5: A heatmap of the angles between subject and object for selected relationship predicates. The values shown in the heatmap are the portions of the total occurrences of the row predicate that exhibit the specific directional configuration in the column. The values in parenthesis next to the predicate names are the total occurrences of that predicate.

We visualize some of the results of the topological analysis in Figure 4.4 and the directional analysis in Figure 4.5. The full heat maps for all the relationship predicates can be found in Appendix B. Note that while the 'equal' topological relationship doesn't occur in this subset of predicates, mainly due to its more specific and rare configuration, it does show up in the full set. Also noteworthy is our evaluation of the 'covers' versus 'in' topological relationships. While [26] describes these relationships rigidly (as shown in Figure 4.3), we loosened them very slightly (in the order of 5% of the smaller of the two bounding boxes under analysis) to account for human error in labelling the bounding boxes. For example, if a subject A lies completely within object B for a given case, however it is proximal enough to the boundary of object B (though not exactly touching it as shown in Figure 4.3e) we could still bin the topological relationship as a 'subject covers object' relationship depending on how close subject A is to the boundary (of B) relative to its own size. Our directional calculations are binned into the 8 cardinal directions of a compass, and measured based on the relative centers of gravity of the bounding boxes. For example if for a certain relationship triplet [**subject, predicate, object**] the center of gravity of the **object** bounding box is **south-east** of the center of gravity of the **subject** bounding box, it is binned as 'SE'. We also calculate the spread of the directions for the 8 different topological configurations within each predicate, and isolate some predicate-topological pairs of interest where directions exhibit a noteworthy spread, this is shown in Figure 4.6.

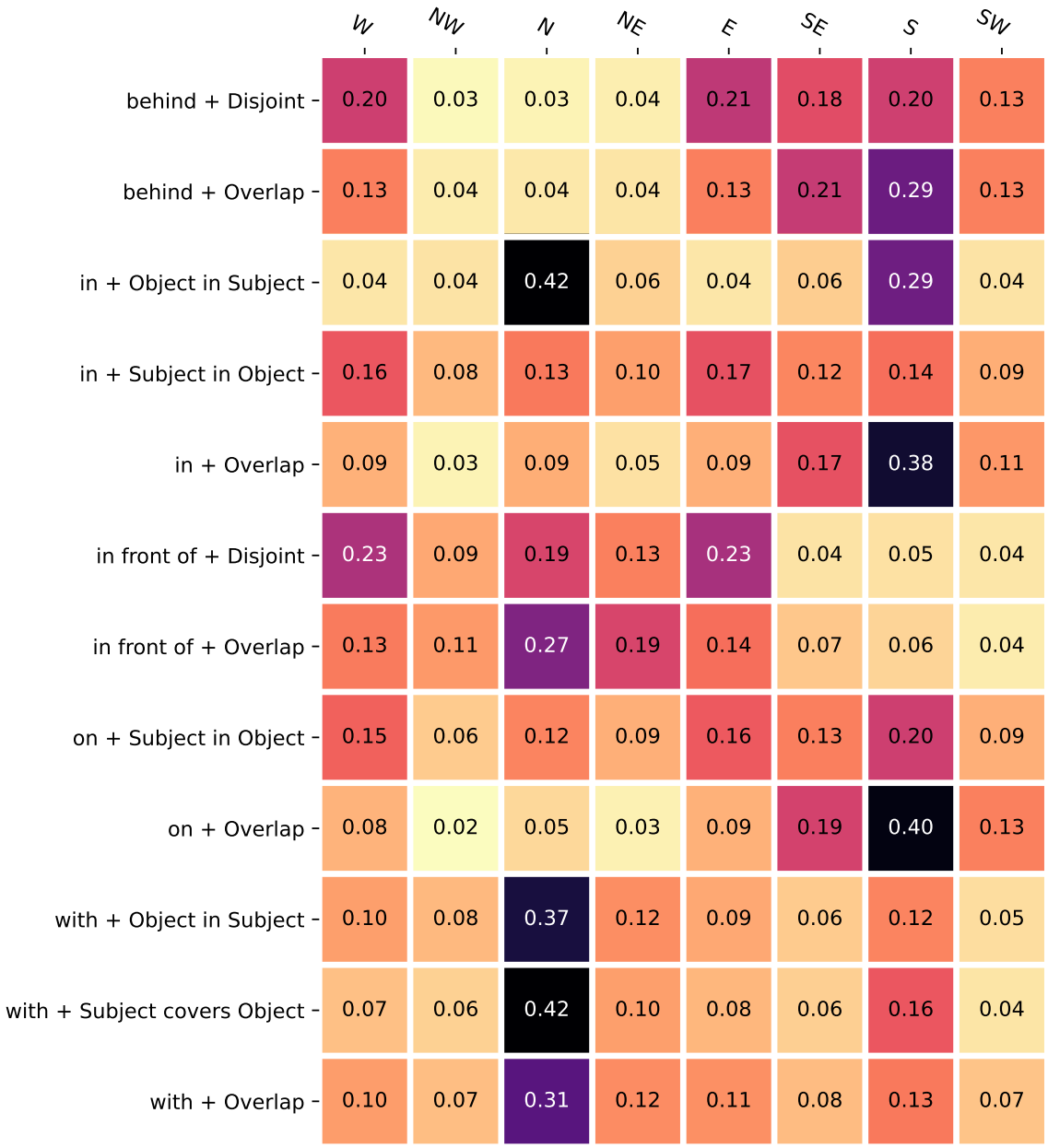
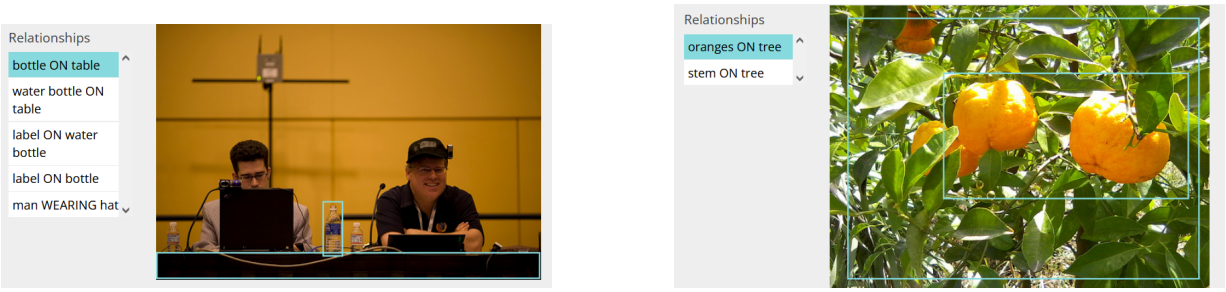


Figure 4.6: A heatmap of the angles between subject and object for selected relationships and specific bounding box topologies for bounding boxes that are connected by those relationships. The values shown in the heatmap are the portions of the total occurrences of the row predicate+topological configuration that exhibit the specific directional configuration in the column.

The topological relationships tend to reveal the more 'dominant' of the subject and object pair linked by a certain predicate. In some cases, such as the predicate 'has' (e.g. building has window), the expected topological configuration is dominant: the object, window, is fully contained in the subject. This follows from how we expect the lingual relationship to occur [70]. Notably, the topological spread of the predicate 'in' is not as would be expected, and further highlights the vagueness of this relationship predicate. While the expected dominant topological configuration (subject in object - it is literally in the name) is the most frequently occurring, it is **not** extremely dominant. We would expect a 'subject in object' topological configuration for the relationship triplet [person in car], but, for example, the triplet [bottle in hand] (where the bounding box of the bottle is actually larger than that of the hand produces an 'object in subject' topological configuration, and an example triplet [plant in pot] counter-intuitively produced a disjoint topological configuration due to how the bounding boxes are labelled.

The directional evaluation produced more expected results. Predicates 'above' and 'under' mostly exhibited directional configurations that are true to their descriptions. In fact, their symmetric relationship is highlighted well by how they exhibit similar topological configuration distributions, while having inverse directional configurations. A similar, but less pronounced, symmetric relationship is also seen in the predicates 'behind' and 'in front of'. Vague predicate classes, however, such as 'on' or 'in' still showed a big variety of directional configurations, likely due to them encoding several different lingual interpretations of 'on' and 'in'. The predicate 'near' interestingly seemed to imply the subject and object were side by side (with the slightly higher chances for a 'W' and 'E' configuration). The results shown in Figure 4.6 help shed some light on the combination of topological and directional configurations and serve to disambiguate some predicate classes. For instance, the predicate 'on' exhibited more predictable directional qualities when the topological relationship was 'overlap'. In this predicate + topological combination, 'on' usually meant the subject was on top of the object e.g.[person on sidewalk], as opposed to 'on' with the configuration 'Subject in Object' (e.g. [fruit on tree]) where the subject is potentially anywhere within the bounding box of the object.

Similarly to how inverse relationships can be used to augment the dataset, it is possible to modify the more vague relationship classes based on their topological configurations. Spatial predicates that are linguistically similar and exhibit similar topological and directional configurations could possibly be merged into broader classes without losing too much of their meaning. For example, predicates such as 'laying on', 'lying on', 'parked on' which all occur in the VG150 dataset, and all seem to be describing a similar spatial configuration (further proven by their topological configurations) can be merged into a



(a) (b)

Figure 4.7: An example of the overloaded relationship predicate 'on'. Different topological configurations of the same predicate can disambiguate between different conceptual and lingual interpretations of the predicate. Both figures showcase the same relationship predicate 'on', however the 'touching' configuration in figure (a) coupled with the relative position of each object implies a support relationship, whereas the 'subject in object' configuration of (b) implies more of a containment relationship.

super set[1]. While on the other hand the larger and vaguer predicate classes 'on' or 'in' can possibly be broken down. A simple example is presented in Figure 4.7.

We would also like to note that a topological analysis of the bounding boxes in VG may be subject to certain biases and shortcomings as well. We live in a 3D world, and it may be difficult for any computer vision system to infer the 3D concepts from 2D images in the Visual Genome dataset annotated with 2D bounding boxes. Concepts like 'behind' and 'in front of' may be extremely difficult for a vision system that has only seen 2D images to reason about, especially if it is not designed with the 3D world in mind. A topological analysis of the VG dataset is likely better suited for the relationship labels that are not overtly 3D in nature. Relationships like 'above' or 'under' are more two dimensional than 'in front of' or 'behind', for example, which may be why the symmetric relationship between the more 2D pair (above-under) was more easily distinguishable in the topological and directional analysis than that of of the more 3D pair (behind-in front of). With that in mind, we still see value in this analysis and the properties that it was able to reveal in the underlying data.

[1]This could provide an alternative to the synset embeddings that are extracted from Wordnet [93] IDs which are already supplied in the VG dataset. Note we did not evaluate the topological configurations while utilizing those IDs.

4.4 Algorithmic Use in Scene Graphs

While we believe the topological and directional configurations along with the augmentations defined by spatial language discussed previously could possibly be incorporated into a novel algorithm for scene graph generation, it is outside of our scope of discussion for this chapter. Instead we aim to experiment with different data configurations based on what our exploration has yielded. The topological and lingual analysis enabled us to better understand the ambiguities of the labels and restructure relationships in a manner that is lingually and topologically sound. We created 2 alternate subsets of the VG150 relationship predicates and measured the performance of the same baseline model when trained with these new labels.

In this section we conclude with 3 simple scene graph generation experiments that are driven by modifying the data rather than modifying the underlying algorithm. To reiterate, in scene graph generation [10], we are given an input image and tasked with identifying the objects in that image along with the relationships that exist between those objects much like the graph shown in Figure 2.8. Scene graph generators are evaluated under 3 different settings:

- **Predicate Classification**: Where the object bounding boxes and the object class labels associated with the bounding boxes are given, hence the scene graph generator must only find the **relationships** between the given objects.
- **Scene Graph Classification**: Where the object bounding boxes are given, but the class labels associated with the bounding boxes are not, so the scene graph generator must infer both the **class labels** as well as the **relationships** between the bounding boxes.
- **Scene Graph Generation**: Where the input image is given without any other labels or information, and the scene graph generator must uncover the relevant **objects** in the image, their **bounding boxes** as well as the **relationships** between them. This is the most challenging setting for evaluation.

Scene graphs are also evaluated based on their recall, as opposed to based on their precision. Given an input image, the scene graph generator is evaluated on how many of the ground truth relationships it was able to uncover. The defining metric in scene graph literature is the mean recall@K metric. The mean recall averages the recall score across every predicate class individually instead of every predicted relationship instance. This is mainly due to the large predicate class imbalance that exists in the VG data set. So the

average recall is calculated for every predicate separately first, and then averaged again to get the mean recall which ensures under-represented predicate classes are not being ignored in the evaluation. The mean recall@K metric is the mean recall score when the top K scene graph predictions are used for evaluation, so a mean Recall@20 would mean the scene graph was allowed to predict up to 20 relationship triplets to compare to the ground truth.

The baseline scene graph generator we utilize for the experiments is the Stacked Motif Network (MOTIF) [158]. In brief, the Stacked Motif Network (MOTIF) [158], first generates the object label only then utilizes a bidirectional LSTM to propagate information between the different object proposal and relationship proposal stages, effectively allowing object context to influence its label and its relationship labels. For our experiments we follow the implementation of [59] and exchange the VGG16 [120] detector with a RESNeXt-101-FPN [145] which was shown to improve performance. Proposing a novel scene graph generation model is out of the scope of this chapter and we only aim to see the differences in performance that a strong baseline generator can observe when the data it uses is better structured. Scene graph generation networks usually achieve relatively low recalls (with the Scene Graph Generation mean Recall still being under 10% in state of the art models [10]). The reasoning authors give is usually the vagueness of the predicates and 'long tailed'-ness of the distributions in the VG dataset. Certain relationships dominate the dataset and learning algorithms struggle to capture the true conceptual information contained in the entirety of the dataset, instead focusing on the dominant classes. As shown in our lingual and topological analysis, the VG dataset does indeed show topological and lingual ambiguity, lack of symmetrical relationships and labelling bias, and these are detrimental to learning models.

We train and evaluate the same scene graph model [158, 59] on three different predicate configurations derived from the VG150 set:

- **Original Predicates**: We use the original 150 classes and 50 predicates from the VG150 data set to baseline the model.
- **Relationship Subset 1**: The 'less vague' subset where we remove 14 of the more vague original 50 predicate classes, and merge 4 others to keep 32 unique relationship predicates. The removed classes are either linguistically vague, or did not exhibit topological and directional configurations that matched their descriptions. That being said, we do keep the larger vague classes (such as 'in' or 'on') since they form such a large subset of the dataset.
- **Relationship Subset 2**: The spatial preposition subset where we take a subset of

the 50 classes that correspond only to spatial prepositions, we also merge classes that exhibit similar lingual, topological and directional configurations. We end up with 8 unique predicates that are a combination of 20 of the original predicates.

The exact predicates we use are listed in Table 4.1. The results of the experiments are shown in in Table 4.2. At first glance the recall results when using relationship subset 2 may seem to indicate a significant leap in performance, though we also understand that this leap is very much expected as the class labels are better balanced and much fewer. That being said, it's interesting to see that even an off the shelf scene graph generator can perform quite well as a spatial preposition predictor when given the right data. In our opinion the more interesting result is that of relationship subset 1. This experiment showed some improvement in recall with the reorganized 50 relationships, but that improvement is not as significant as we would have expected. Relationship subset 1 cleaned up edge cases and some of the more 'vague' predicates of the original 50, but the performance improvement seen was relatively small.

We analyze the existence of inverse relationships in the predictions of the scene graph generators and show a subset of the results in Figure 4.8. In this analysis, we tallied what inverse relationships are found by the scene graph generator for each of the correctly recalled ground truth relationships. In other words, if a ground truth relationship is correctly found by the generator in its top K relationships under a specific setting, we find whether an inverse relationship is also being predicted (whether it exists in the ground truth or not). This yielded some interesting findings on what inverse relationships the generator is learning. For example, in the case of the original 50 predicate classes, the predictor seemed to find a strong inverse relationship between predicates 'of' and 'has', as well as 'on' and 'has'. Both of these pairs are likely a result of a symmetric possessive relationship that is getting encoded (e.g. wing of bird/bird has wing or car has wheel/wheel on car). This is likely due to the formulation of stacked motif networks [158] which honed on certain repeated 'subgraphs' in the ground truth. In the case of relationship subset 2 of spatial predicates, some inverse relationships are more prevalent (such as the large class of 'on' having an 'under' inverse relationship 15% of the time), however other incorrect relationships also show up (such as 'behind' being its own inverse).

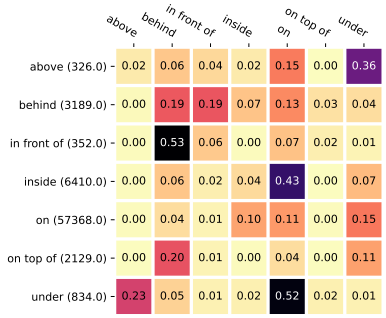
Original Relationships	**Relationship Subset 1**	**Relationship Subset 2**
above	above	above
across	across	-
against	against	-
along	along	-
and	-	-
at	-	-
attached to	attached to	-
behind	behind	behind
belonging to	-	-
between	between	-
carrying	carrying	-
covered in	covered in	-
covering	covering	-
eating	-	-
flying in	*in*	*inside*
for	-	-
from	from	-
growing on	growing on	*on*
hanging from	hanging from	*on*
has	has	-
holding	holding	-
in	in	inside
in front of	in front of	in front of
laying on	laying on	*on top of*
looking at	-	-
lying on	*laying on*	*on top of*
made of	-	-
mounted on	mounted on	*on*
near	-	-
of	-	-
on	on	on
on back of	on back of	*on top of*
over	over	*above*
painted on	painted on	-
parked on	parked on	*on top of*
part of	-	-
playing	*using*	-
riding	riding	*on top of*
says	-	-
sitting on	sitting on	*on top of*
standing on	standing on	*on top of*
to	-	-
under	under	under
using	using	-
walking in	walking in	*inside*
walking on	walking on	*on top of*
watching	-	-
wearing	wearing	-
wears	*wearing*	-
with	-	-

Table 4.1: A breakdown of the relationships used in each of our 3 experiments. A '-' means the relationship from the original set was removed entirely. Italicised text is used to indicate that a relationship has been kept but its label was modified.

Model	Detector	Relationship Set	Predicate Classification			Scene Graph Classification			Scene Graph Generation		
			mR@20	mR@50	mR@100	mR@20	mR@50	mR@100	mR@20	mR@50	mR@100
VCTree [130]	VGG-16 [120]	Original 50 Relationships (Reported in [59])	14.0	17.9	19.4	8.2	10.1	10.8	5.2	6.9	8.0
MOTIF [158]	RESNeXt-101-FPN [145]	Original 50 Relationships (Reported in [59])	14.1	18.0	19.4	8.0	9.9	10.6	5.8	7.7	9.0
MOTIF [158]	RESNeXt-101-FPN [145]	Baseline (Original 50 Relationships)	12.0	15.4	16.7	5.9	7.2	8.9	4.8	6.1	7.2
MOTIF [158]	RESNeXt-101-FPN [145]	Relationship Subset 1 (Less Vague 36 of Original Relationships)	17.4	21.2	22.6	8.5	10.2	10.7	5.8	7.7	8.9
MOTIF [158]	RESNeXt-101-FPN [145]	Relationship Subset 2 (Spatial Prepositions 20 of Original Relationships)	46.1	55.7	59.1	22.2	25.9	27.1	14.4	18.8	21.6

Table 4.2: The results of our three experiments with different VG150 predicate subsets. Our code implementation was adapted by starting with the implementations of [59, 129].

(a) Selected inverse relationships from the original predicate model.



(b) Inverse relationships from the model trained with only spatial prepositions (relationship subset 2).

Figure 4.8: Inverse relationship proportions in the scene graph predictors trained on different data subsets evaluated on the predicate classification task using the top 50 predictions. The numbers in parenthesis indicate the number of correctly recalled instances of each predicate, the numbers in the grid are the portion of those recalled instances that had an inverse relationship with the column predicate class (whether that relationship was in the ground truth or not).

If anything the results of both of our experiments seem to indicate that there is still much to be done in the field of scene graph generation even outside of the dataset domain. Stacked motif networks [158] are an impressive approach to generating scene graphs, that

managed to push the field forward by paying attention to the underlying data. Since then, a few other works have taken interesting approaches as well. For instance, segmentation grounded scene graph generation [59] demonstrates the utility of moving away from solely bounding boxes for scene graph generation and shows that even mask annotations obtained via zero-shot transfer can improve scene graph generation performance. Grounding consistency [24] on the other hand demonstrated how a lack of negative training examples in the data and the reliance on recall alone in the evaluation led most scene graph predictors to learn very biased representations.

4.5 Conclusions

In this chapter we explored the lack of representation of both sides of symmetric relationships in the VG dataset, which likely resulted from the asymmetric spatial representations humans (and thus human labellers) exhibit. However, we noticed even with the 'cleaned' relationship sets, scene graph performance was still relatively subpar. We explore this further in Chapter 5 as we attempt to understand how 'visual' the scene graph generation is. We also utilize the topological configurations discussed in this chapter in two separate approaches to scene graph generation.

Chapter 5

Naive Scene Graphs and Topological Relationship Affinity Fields

5.1 Chapter Summary

This chapter focuses on connecting visual relationship detection to the visual signal in the images themselves (i.e., the pixels). We reiterate the main contributions:

- We describe and implement a new 'statistical baseline' for scene graph generation and demonstrate that a classical machine learning approach, one as simple as a categorical Naive Bayes classifier, can perform relationship detection in a manner that achieves comparable performance to that of state of the art scene graph generators.
- We describe and discuss how modern approaches to scene graph generation are being held back from a data, evaluation and algorithmic standpoint .
- We describe, implement and evaluate topological relationship affinity fields a novel representation for scene graph generation grounded in the image pixels themselves.
- We qualitatively demonstrate the visual nature of the learned relationships and how they pertain to specific pixel regions.

5.2 The 'Visualness' of Visual Relationship Detection

Visual relationship detection in humans is a complex process requiring us to identify the objects we see and understand how they interact with each other. Some complexity comes from the fact that the same two objects don't always share the same relationship, for example during business hours, a chair at a restaurant is likely next to or tucked under a table, whereas after closing time, the chair is possibly above the table (an unusual configuration, but one that we as humans can understand the reasoning behind). The same relationship can also manifest itself differently depending on the subject and the object of it, for example the relationship 'on' manifests itself differently in the statements "fruit *on* a tree" and "cup *on* a table". This vagueness in what a relationship predicate really means is further exacerbated by the overloaded language we use to describe these relationships which we explored in Chapter 4.

Our human cognitive biases, gained via living in and experiencing the world, often can give us a good idea of what relationships will exist between two objects without seeing the objects themselves [119, 34]. If we were told that there is a scene involving a relationship between a 'person' and their 'clothes', it's fair to assume that the person is 'wearing' the clothes without necessarily seeing the scene or image. These biases are not entirely bad, as they often weed out unlikely relationships between two objects (e.g., a person is likely not 'riding' their clothes). Further, when humans look at a scene they are capable of overriding their biases and identifying the correct relationship despite how unusual it may be.

Relationship prediction algorithms appear to be relatively capable of the former form of prediction where, after training with relevant data, they appear to be able to filter out likely and unlikely relationships between objects [158, 129, 130]. The current state of performance on Scene Graph Generation [10] may indicate they aren't doing much past that. While machines appear to be capable of learning the 'bias' that aids them in narrowing down what relationships are most likely, they are less capable of overcoming this bias and reliably predicting the sometimes unseen and unusual visual relationships in a scene and usually default to the broader and more trivial relationships. This behaviour is often attributed to imbalances in the training data (e.g., in the VG dataset [64] where vague relationship predicates such as 'on' or 'has' dominate the dataset), and a 'long tailed distribution' of visual relationships and its effect can be seen in the stark difference between Scene Graph Generator's recall performance and their mean recall performance.

Scene Graph Generation is a relatively recent task in computer vision with the earliest methods for performing it appearing around 2017 [146], and its approaches in the literature always utilize some form of 'deep learning' [10]. Section 2.5 delves into an overview of

modern Scene Graph Generation. The scene graph generators (i.e., the networks) take in an image as input and must output the scene graph that details the relationships they observed in the image. Object detection is its own unsolved task in computer vision (see Chapter 3), and certain modes for scene graph generation also supply the networks with the locations of the object bounding boxes. This allows researchers to measure the performance of networks on relationship prediction alone regardless of how well object detection is being done. One of the critiques cited by Marcus [89] is that the usually complex nature of deep learning approaches makes their decision-making rationale less straightforward to understand. In the case of Scene Graph Generation, we often don't know why a certain relationship predicate was predicted and how much of an influence the object labels play versus the observed visual stimuli themselves.

In this chapter we first explore the extent to which visual relationship detection can be done 'non-visually' and with an extremely simplistic approach: Naive Bayes. At first glance it may seem almost futile to compare the results of a simplistic and basic statistical approach to what a deep network with hundreds of millions of parameters trained for hundreds of thousands of iterations could achieve. However, as we show in Section 5.3, without the use of any pixel data a simple probability based approach can still compete even with more recent approaches to Scene Graph Generation. This is not due to our Naive Bayes approach and features being particularly well suited for the task, but similar to what we showed was the case with object detection in Section 3.5.2 (and as discussed in [46] for toy problems), scene graph generating networks may in fact be 'fixating' on the easiest feature to learn (the statistical biases of the data) and hindering their own performance.

We build on these findings and our findings from Chapter 4, and design an approach to scene graph generation that is, by design, grounded in the visual data. Building on the work in [81], we propose a novel representation for visual relationships where relationships are grounded in image pixels using the topology of their relevant bounding boxes. Our approach does not perform at the level of existing scene graph generation methods, however it shows promise as a step towards a different paradigm for approaching scene graph generation. During training time we leverage the topological configuration of the objects connected by a relationship, however during inference the representation is regressed by networks without the need to first identify the classes or the bounding boxes of the subject and object in the relationship triplet. Effectively, our approach is a 'single stage' scene graph generation approach that regresses both objects and relationships using the same backbone but independently of each other. Our approach is explainable by design, with every inferred relationship being grounded in the image pixels that led to its inference. This also allows our representation to be used to extend scene graph generation past using bounding boxes for objects and into using object instance masks. Furthermore, much

like in [81], our representation can be regressed using less computationally and memory intensive fully convolutional networks without the need for multiple 'stages'.

5.3 Naive Scene Graphs

In effectively all scene graph generation approaches, objects are detected first, then relationships are inferred between pairs of objects in an image. After the objects are detected, visual features from object the object detection backbone are used and further refined by the relationship detection networks/heads. This happens differently in various scene graph generation paradigms (e.g., some methods use the strict areas of the bounding boxes of the objects themselves, others use the visual features in the smallest 'union' box that contains both objects, one method even uses weakly learned 'masks' [59]). The underlying assumption of these approaches is that networks are learning valuable visual features and representations that eventually allow for visual relationship detection in the images.

Using the same ground truth data (e.g., the Visual Genome dataset [64]), scene graph generation networks are trained using 3 different data 'settings' each using the same images, but some also relaying additional data at train and inference time. When training and evaluating under the 'Predicate Classification' (PredCl) setting, networks are given the bounding boxes and class labels of all objects a priori and they must only infer the correct relationships. Whereas when training and evaluating under the 'Scene Graph Classification' (SGCl) data setting, the ground truth bounding box locations are given, but networks must first classify the object labels of the objects within the bounding boxes before predicting relationships. Finally, under the 'Scene Graph Detection' (SGDet) setting networks are only given the image with no additional information. In practice the same network architecture is used in all three settings, noting that in the literature networks are often trained under each data setting individually for optimal performance.

In all three data settings, the object detection network generates the visual features needed for relationship detection as well. The 'relationship detection' portion of the scene graph generation networks' inputs are the visual features from the backbone network as well as the class labels and bounding boxes, and this relationship detection portion must then infer the most likely relationships among all bounding box pairs. In the cases where some a priori information is known, this information overrides the object detection network's predictions for object class label, bounding box location or both, but the relationship detector still receives similar inputs. The strong correlations that exist between the object class combinations and the possible relationships [129, 158, 119, 34] along with a heavily

imbalanced dataset can possibly lead to scene graph generation networks suffering from a similar over-reliance on 'simple' features as previously seen in [46] and Section 3.5.2.

Ideally, scene graph generation networks are utilizing the additional information from the visual features they have access to. However in order to truly understand how well scene graph generators are performing, we aim to baseline how much information can potentially be extracted from the input data given to networks devoid of any visual features. We utilize a simple approach that relies solely on the relatively basic correlations that could be drawn from utilizing the bounding box information (be it given a priori or inferred by the backbone network).

5.3.1 Experimental Approach

In this experiment, we perform scene graph generation in a manner similar to existing approaches [10] (with objects being detected first, followed by relationship detection on those objects), and on all 3 data settings, with the key difference being how we approach relationship detection. In our case, relationship detection intentionally only utilizes the non-pixel data that the bounding box information carries with it. For every image this data always contains the corners of all the detected (or ground truth) bounding boxes in the image as well the detected (or ground truth) classes of the objects in these bounding boxes whether this information is given from the ground truth (fully in the case of PredCl and partially in the case of SGCl) or from the object detection head (SGDet). Our relationship classifier is also not deep learning-based, but is actually a simple naive Bayes classifier conditioned on a small set of hand crafted features generated from this input data. No a priori information on which objects share a relationship is given to our predictor (or any traditional scene graph generation method in the literature). Our predictor is tasked with determining which bounding boxes share relationships, and what those relationships are, to eventually generate a scene graph of the image.

5.3.1.1 Classifier and Feature Selection

For this set of experiments we utilize a Categorical Naive Bayes classifier [150, 96]. This classifier can be seen as a generalization of a Bernoulli Naive Bayes classifier where instead of the input features being strictly [False, True] (or [0,1]), the input features are discrete and have a a known lower and upper bound but can belong to any number of categories ([0,...,n-1] for n categories). Multiple works [158, 129, 130] validate the correlations that exist between object classes and relationship classes between them. In addition, our findings in

Chapter 4 indicated that topological configurations and relative bounding box positions may be discriminative indicators of the type of relationship between two objects. Finally, the areas of the bounding boxes may also help in determining the relationship between the objects they contain. We propose 7 simple categorical features based on these findings that we use in conjunction with our classifier:

1. **Head Class:** The class of the head object in the relationship triplet <head, predicate, tail>. In our experiments this feature can be one of the 150 object categories in the VG150 Dataset.

2. **Tail Class:** The class of the tail object in the relationship triplet <head, predicate, tail>. In our experiments this feature can be one of the 150 object categories in the VG150 Dataset.

3. **Topological Configuration:** The topological configuration of the head and tail of the relationship triplet. This can be one of 8 possible categories as shown in in Figure 4.3.

4. **Angular Configuration:** The location of the object relative to the subject in the relationship triplet binned into the 8 cardinal directions.

5. **Relative Head to Tail Area:** The ratio of the area of the head bounding box to that of the tail bounding box ($\frac{\text{Head Bounding Box Area}}{\text{Tail Bounding Box Area}}$). This feature is logarithmically scaled, rounded, and clipped to convert it to discrete/categorical 'bins' between [-9,...,9]. Where a category of -9 implies that the head object is extremely small compared to the tail object, 0 implies they are roughly equal in area, and 9 implies the head object is extremely large compared to the tail object. The total number of categories is therefore 19.

6. **Head Area Ratio:** The ratio of the area of the head bounding box to the whole image ($\frac{\text{Head Bounding Box Area}}{\text{Image Area}}$). This feature is logarithmically scaled, rounded, clipped and shifted to convert it to discrete/categorical 'bins' between [0,...,9]. Where a category of 0 implies that the head box is roughly extremely smaller than the whole image, and 9 implies the head object is roughly of the same size as the whole image. The total number of categories is 10.

7. **Tail Area Ratio:** The ratio of the area of the tail bounding box to the whole image ($\frac{\text{Tail Bounding Box Area}}{\text{Image Area}}$). This feature is also logarithmically scaled, rounded, clipped and shifted to convert it to discrete/categorical 'bins' between [0,...,9]. Where a category

of 0 implies that the tail box is roughly extremely smaller than the whole image, and 9 implies the tail object is roughly of the same size as the whole image. The total number of categories is 10.

One of the main assumptions made when utilizing a classifier from the Naive Bayes family, the 'naive' assumption, is that the features used are independent. This is not fully the case with our proposed features. As shown in [158], head classes and tail classes can be somewhat predictive of each other, with certain tail classes almost exclusively being associated with certain head classes (e.g., such as tail classes that pertain to clothing being in triplets with human-related head classes). Features such as head class and head area ratio (or tail class and tail area ratio) are also correlated due to the nature of the data in the visual genome dataset [64]. The object classes themselves are also redundant in the VG dataset and the VG150 subset, for example *boy, child, kid* and *person* all exist as separate object classes. Despite this, in practice we find that our Naive Bayes classifier is still well suited to the task of relationship detection while utilizing our selected set of features.

We also note that features 4 through 7 could have been represented as continuous instead of categorical variables, and even utilized with a Naive Bayes classifier. Angular configuration could have been replaced with the exact angle instead of a binned representation, and the area ratios could have been represented as continuous variables with decimal numbers. We opted for a simpler approach with binning these features into categories since the bins still represent the essence of the features, and allow us to pursue a purely categorical classifier.

5.3.1.2 Experiment Details

For our experiments we utilize 3 different subsets of our features which we utilize both in isolation in different combinations together:

1. **Class Feature Subset:** Comprised of features 1-2, this feature set is used to measure the predictability of relationship predicates when the head and tail classes are known. Note that a similar 2 feature set was used in [158] with a different classifier.

2. **Topological Feature Subset:** Comprised of features 3-4, this feature set allows us to validate whether topological and angular configurations carry additional information that is informative to relationship prediction between two object bounding boxes.

3. **Area Feature Subset:** Comprised of features 5-7, this feature set allows us to observe whether the area-related features are also discriminative in a manner different than the topological and angular configurations.

For all proposed feature subsets, and all combinations between them, we evaluate scene graph generation under the three scene graph generation data settings (PredCL, SGCl, SGDet). In the case of PredCl, we extract our proposed features from the ground truth bounding box class and bounding box data (available in this data setting), then train and evaluate our classifier using that data. In the case of SGCl and SGDet, we must first get the missing bounding box information for the image, we utilize a pre-trained Faster R-CNN with a ResNeXt-101-FPN backbone trained in [129], which is the same one used to benchmark several of the scene graph generation approaches we compare against [129, 158]. In our case, this network is used solely for the purpose of generating bounding box information and no visual features from the backbone are used for relationship prediction. For the SGCl data setting, the bounding box corners from the ground truth are utilized (which are available in this setting), and no ground truth information is used in the SGDet data setting. We train a separate Naive Bayes classifier for each feature set and data setting as is common practice in scene graph literature. We evaluate the recall, mean recall and zero-shot recall @ 20, 50 and 100 permitted predictions and perform our evaluation with the graph constraint[1] for a fair comparison with the literature.

Zero shot recall measures how well an approach detects a relationship triplet that has never been seen in the training set. So while the individual objects and the relationship predicates in the <head, predicate, tail>triplet will exist in the training set, the specific combinations that zero shot recall is measured on have not been seen by the detector. Sometimes this means that the head object and the relationship predicate have always had a different tail object (or vice versa), but it could also mean that for a specific relationship predicate either the head object, the tail object or both will have a zero frequency of occurrence in the train set. This will cause a problem for a Naive Bayes type classifier since it would cause certain objects to have a zero posterior probability with certain relationship predicates. We mitigate the zero occurrence problem by using a Laplacian correction on the frequency table which effectively adds 1 to every occurrence, eliminating zero occurrences and giving every object even a slim chance at being in a triplet with any relationship in a zero shot setting.

We train two separate categorical Naive Bayes classifiers using our training data, one for detecting the existence of relationships and another for classifying the relationship between

[1]No two objects are allowed multiple relationship predictions in the same direction.

the 50 possible predicates. We could have alternatively trained one classifier by adding an additional 'background'/'no relationship' class, but we empirically found our approach to be superior. The first of our two classifiers detects the presence of a relationship between two bounding boxes and outputs the probability that the two objects share a relationship. It is trained by examining all possible combinations of two bounding boxes per image in the training set ground-truth, and assigning a positive label (1) to the combinations that share a relationship triplet and a negative label (0) to the combinations that don't share a relationship. The corresponding features are extracted for every combination in the training set, after which we train the 'existence' classifier. The relationship classifier is trained only on the positive relationship samples, and it predicts the most likely relationship predicate between two bounding boxes (out of 50 in the VG150 data set). During inference time, we are allowed 20/50/100 guesses at the relationship triplets (depending on the evaluation mode), and so we usually require a method of ranking the predicted relationships. We experimentally found that utilizing the existence probability to rank the relationships (and choosing the top k based on that ranking) provided the best overall results. We experimented with combining the relationship type probabilities into the ranking and found a very small, but mostly negative effect on the overall results.

5.3.2 Results and Discussions

Our mean recall results are presented and compared with other modern scene graph generation approaches in Table 5.1, our recall results are presented in Table 5.2, and our zero shot recall results are presented in Table 5.3.

First, we compare between our different classifiers defined by the features they utilize. Since our classifier is a simple one that utilizes statistics, comparing between performance of the classifier when utilizing different feature combinations can give us a good idea about which features seem to carry more information based on their prior statistics.

As expected, bounding box classes were the biggest contributors to improving recall and mean recall performance, knowing the head and tail classes alone the classifier still performs reasonably well. Head and tail classes in relationship prediction can be quite biasing to humans [119, 34] and it's no surprise their statistics are informative as well. Topological and area features did not seem to allow classifiers to recall relationships as well as class features did, and even when combined, the classifier utilizing topological and area features together did not measure up to the one utilizing class features only in terms of recall and mean recall. When a classifier was able to use topological and class features, however, recall and mean recall improved significantly further highlighting the utility of these types

Table 5.1: Mean recall rates @ 20/50/100 on the VG150 dataset for our Naive Bayes approach as well as modern approaches to scene graph generation. The approaches denoted by * utilize a ResNeXt-101-FPN backbone. For SGCl and SGDet we utilize this same backbone as the object detector with its weights taken from [129], backbone choice does not affect our PredCl results.

		PredCl			SGCl			SGDet		
	Year	mR@20	mR@50	mR@100	mR@20	mR@50	mR@100	mR@20	mR@50	mR@100
IMP [146, 129]	2017	-	9.8	10.5	-	5.8	6	-	3.8	4.8
Motif * [158, 129]	2018	10.8	14	15.3	6.3	7.7	8.2	4.2	5.7	6.6
VCTree * [130, 129]	2019	14	17.9	19.4	8.2	10.1	10.8	5.2	6.9	8
KERN [16]	2019	-	17.7	19.2	-	9.4	10	-	6.4	7.3
VCTree+TDE * [129]	2020	17.2	23.3	26.6	8.9	11.8	13.4	6.3	8.6	10.3
GPSNet [78]	2020	-	-	22.8	-	-	12.6	-	-	9.8
GB-Net [156]	2020	-	19.3	20.9	-	9.6	10.2	-	6.1	7.3
VCTree + EBM [125]	2021	14.2	18.2	19.7	10.4	12.5	13.4	5.7	7.7	9.1
Segmentation Grounded Motif * [59]	2021	14.5	18.5	20.2	8.9	11.2	12.1	6.4	8.3	9.2
Segmentation Grounded VCTree * [59]	2021	15	19.2	21.1	9.1	11.6	12.3	6.3	8.1	9
RAAL [80]	2021	14.4	18.3	19.9	7.9	6.6	10.3	4.9	6.5	7.4
Schemata [115]	2021	-	19.1	20.7	-	10.1	10.9	-	-	-
FCSGG [81]	2021	4.2	5.7	6.7	2.2	2.9	3.3	1.9	2.7	3.3
VCTree + NICE * [73]	2022	-	29.9	32.3	-	19.9	21.3	-	11.9	14.1
VCTree + EBM + NARE * [32]	2022	21	24.9	26.5	14	16.2	17.1	7.8	10.1	11.8
HL-Net [79]	2022	-	-	22.8	-	-	13.5	-	-	9.2
Naive Bayes With Feature Set:										
Class Only *		8.70	13.38	16.58	5.30	7.59	9.08	4.04	5.75	7.10
Topological Only *		2.44	3.52	3.97	1.45	2.06	2.40	0.98	1.48	1.83
Area Only *		1.27	2.32	3.12	0.81	1.39	1.85	0.33	0.63	1.01
Class + Topological *		12.03	17.43	20.62	6.60	9.43	11.34	4.49	6.93	9.00
Class + Area *		9.49	14.39	17.76	5.43	7.96	9.68	3.53	5.73	7.53
Topological + Area *		2.40	3.92	4.77	1.42	2.27	2.82	0.73	1.34	1.90
Class + Topological + Area *		11.93	17.39	20.81	6.58	9.38	11.47	4.15	6.55	8.78

Table 5.2: Recall rates @ 20/50/100 on the VG150 dataset for our Naive Bayes approach as well as modern approaches to scene graph generation. The approaches denoted by * utilize a ResNeXt-101-FPN backbone. For SGCl and SGDet we utilize this same backbone as the object detector with its weights taken from [129], backbone choice does not affect our PredCl results. Blank rows left in to correspond with Table 5.1.

		PredCl			SGCl			SGDet		
	Year	R@20	R@50	R@100	R@20	R@50	R@100	R@20	R@50	R@100
IMP [146, 129]	2017	54.34	61.05	63.06	34.01	37.48	38.5	18.09	25.94	31.15
Motif * [158, 129]	2018	58.46	65.18	67.01	35.63	38.92	39.77	25.48	32.78	37.16
VCTree * [130, 129]	2019	59.02	65.42	67.18	42.77	46.67	47.64	24.53	31.93	36.21
KERN [16]	2019	-	65.8	67.6	-	36.7	37.4	-	27.1	29.8
VCTree+TDE * [129]	2020	39.1	49.9	54.5	22.8	28.8	31.2	14.3	19.6	23.3
GPSNet [78]	2020	67.6	69.7	69.7	41.8	42.3	42.3	22.3	28.9	33.2
GB-Net [156]	2020	-	59.3	61.3	-	34.6	35.4	-	20.7	27.6
VCTree + EBM [125]	2021	-	-	-	-	-	-	-	-	-
Segmentation Grounded Motif * [59]	2021	-	-	-	-	-	-	-	-	-
Segmentation Grounded VCTree * [59]	2021	-	-	-	-	-	-	-	-	-
RAAL [80]	2021	59.1	66.2	68.4	33.5	36.7	37.6	21.7	27.3	29.9
Schemata [115]	2021	-	66.9	68.4	-	39.1	39.8	-	-	-
FCSGG [81]	2021	28.0	35.8	40.2	13.9	17.7	19.6	11.4	15.7	19.0
VCTree + NICE * [73]	2022	-	55.0	56.9	-	37.8	39.0	-	27	30.8
VCTree + EBM + NARE * [32]	2022	-	-	-	-	-	-	-	-	-
HL-Net [79]	2022	60.7	67	68.9	38.8	42.6	43.5	26	33.7	38.1
Naive Bayes With Feature Set:										
Class Only *		29.98	42.51	50.41	17.86	24.76	29.48	12.01	17.11	20.90
Topological Only *		25.01	37.03	42.44	14.90	21.23	24.94	8.59	13.86	17.89
Area Only *		17.12	27.71	35.09	10.42	16.02	20.33	4.95	8.38	12.08
Class + Topological *		40.40	52.05	57.37	22.62	29.65	33.72	15.36	21.29	25.68
Class + Area *		27.83	41.53	50.10	16.27	23.55	28.62	10.01	15.70	20.34
Topological + Area *		24.46	35.97	41.70	14.09	20.50	24.46	7.51	12.50	16.72
Class + Topological + Area *		37.42	49.6	55.34	21.16	28.16	32.31	13.39	19.33	24.07

Table 5.3: Zero shot recall rates @ 20/50/100 on the VG150 dataset for our Naive Bayes approach as well as modern approaches to scene graph generation. The approaches denoted by * utilize a ResNeXt-101-FPN backbone. Zero shot recall is not always reported on, and we included the approaches from Table 5.1 and 5.2 that do report on this metric.

		PredCl			SGCl			SGDet		
	Year	zsR@20	zsR@50	zsR@100	zsR@20	zsR@50	zsR@100	zsR@20	zsR@50	zsR@100
Motif* [158, 129]	2018	1.9	-	7.2	0.3	-	1.2	0		0.5
VCTree* [130, 129]	2019	1.8	-	7.1	0.4	-	1.2	0.1	-	0.7
VCTree+TDE* [129]	2020	-	14.3	17.6	-	3.2	4	-	2.6	3.2
Segmentation Grounded Motif* [59]	2021	4.1	-	10.5	0.8	-	2.5	0.1	-	1
Segmentation Grounded VCTree* [59]	2021	4.6	-	10.6	0.8	-	2.5	0.3	-	1.5
FCSGG [81]	2021	-	8.2	10.6	-	1.3	1.7	-	0.8	1.1
VCTree + TDE [129] + NARE* [32]	2022	9.11	13.52	-	4.26	6.2	-	2.24	3.25	-
IMP [146] + NARE [32]	2022	7.12	10.5	-	1.57	2.32	-	1.52	2.48	-
Naive Bayes With Feature Set:										
Class Only *		3.44	6.59	9.16	0.79	1.56	2.19	0.2	0.55	0.77
Topological Only *		6.56	10.88	13.35	0.92	1.82	2.72	0.36	0.74	1.37
Area Only *		5.07	8.55	10.89	0.75	1.51	2.18	0.34	0.92	1.27
Class + Topological *		6.94	11.6	14.46	1.26	2.57	3.53	0.29	0.79	1.33
Class + Area *		5.20	8.65	11.19	1.10	1.92	2.63	0.36	0.65	1.10
Topological + Area *		7.23	10.83	12.94	1.10	1.93	2.80	0.37	0.83	1.40
Class + Topological + Area *		7.49	11.9	14.49	1.35	2.55	3.53	0.37	0.82	1.36

of features. The area and class classifier did show an improvement over the class only classifier, however to a lesser degree than what topological features added. Even utilizing all three types of features together, recall and mean recall performance were comparable between the classifier using class and topological features, and the classifier using class, topological and area features.

The trends we observed with recall and mean recall did not extend to the zero shot recall evaluation mode. This mode evaluates the recall performance on never before seen relationship triplets, so object class labels may not be as helpful. This is in fact the case, where the classifier trained with only class features performed significantly worse in this mode than the ones trained with topological or area features alone. Topological and area features of bounding boxes are clearly informative for this mode with the best performing classifier being the one trained on class, topological and area features. Topological features appeared to be the most informative to classifiers in the zero shot recall mode, likely due to the statistics of configurations of bounding boxes being predictive of a relationship between classes that never shared this relationship in the training set.

We present our statistical baselines as a new minimum achievable target for scene graph generation done in the traditional 2-stage approach (where objects are detected first, followed by relationship prediction). This is the case for effectively every scene graph approach

we present and, to our knowledge, every scene graph generation approach in the literature except for FCSGG [81]. Statistics alone contributed to our presented classification results, and no visual signals in the form of image pixels contributed to our final results. We cannot certainly determine which aspects of the compared approaches are causing them to underperform or exceed our presented baselines, however we highlight each compared method's uniqueness to frame and understand what could be causing this method to succeed.

Our Naive Bayes approach effectively models the relationships in the image as a fully connected graph, where every detected object is connected to every other detected object. This is similar to how IMP [146], one of the earliest approaches to scene graph generation, tackles the task. In the case of IMP, relationship prediction between two objects was done via a Gated Recurrent Unit that passed information about the objects and context. In our case the 'existence' and 'type' of relationship connecting the two objects is inferred by our classifier based on prior probabilities. This is a poor assumption to make on our end, on top of an exceptionally oversimplified classification approach. Both Motifs [158] and VCTree [130] design their approaches to not utilize a fully connected graph but instead capitalize on certain aspects of the nature of relationships in the Visual Genome Dataset. These approaches are quite well designed and both approaches are still widely used as a baseline approach for several other methods [129, 59, 73, 32] that augment certain aspects of them to improve their overall performance. Motifnet [158] was explicitly designed with the knowledge that commonly occurring relationship subgraphs (the titular 'motifs') permeate the Visual Genome dataset. While the approach taken was sound, with image context being utilized and refined among relevant bounding boxes for predicting relationships. VCTree [130] moves towards a hierarchical tree structure as opposed to a fully connected graph and also weights its relationship predictions based on the objects that are sharing this relationship. Despite the more logical representations for their data, performance of both these approaches was oddly still below that of our statistical approach.

Several modern approaches opt to build upon the core approaches proposed in VCTree [130] and Motifs [158], such as [125, 59, 129, 152, 73, 32] among many others. 'Unbiasing' scene graph generation using Total Direct Effect (TDE) appears to be one of the most reliable methods for improving performance. It is likely that with TDE the removal of noisy relationships resulting from irrelevant contributions from the context of the image was key to allowing networks to start relying on the relevant pixel level data that is informative of relationship classes. This likely pushed TDE-utilizing approaches past the threshold of our statistical approach. EBM [125] modified the loss function used into an energy-based one that they designed to be aware of the structure of scene graphs, as opposed to a cross entropy loss that had been used previously which treated objects and relationship labels as independent entities. Though EBM did improve upon the baseline VCTree, the overall

performance still did not exceed our baselines. The segmentation grounded approaches [59] also don't appear to improve upon their respective baselines and only hover around the statistical baseline we present. Since no segmentation ground truth data exists for the VG dataset, this approach generates pseudo masks based on the object class similarity to the object classes in an alternate dataset (COCO), and there are no guarantees that the eventually produced object masks for the VG dataset are in fact accurate. We expected this approach to be better grounded in the pixel data and possibly exceed the statistical baselines we present as a result of it being forced to rely on additional visual information, however it's possible that performance of these approaches was hindered by the noisiness of the VG dataset. NARE [32] modifies its loss function during training depending on the type of label being predicted, guiding the network to predict the more 'implicit', verb-based, classes over the abundant 'explicit', spatial preposition-based ones. This targets and improves the mean recall performance directly and allowed NARE's approach to perform well on this metric, since it favours predicting the rarer classes. The more successful approach appeared to be NICE [73] that attempts to train the baseline VCTree model with better data. NICE [73] targets the ground truth data itself and refines it to allow the scene graph generators to perform better. NICE seemed to improve performance most significantly between the approaches we surveyed, and its refined VCTree exceeded our statistical baseline by a large margin.

Looking at the difference between recall and mean recall performance of our statistical classifiers in comparison to the other approaches shows an interesting trend as well. Despite the recall performance of our statistical approaches being significantly lower than a majority of other surveyed approaches, our mean recall performance (the more important metric) is actually on par with these other approaches. This is a somewhat troubling finding which implies that approaches such as GPSNet [78], GB-Net [156], HL-Net [79] along with others such as KERN [16], RAAL [80], and Schemata [115] are still favouring the larger, more common predicates (such as 'on' or 'has') thereby pushing their recall score up, but overall, still don't detect the lesser represented predicates (e.g., 'riding on') well causing their mean recall to remain lower. GPSNet [78] prioritizes certain combinations of nodes and relationships by way of a focal loss and attempts to improve overall performance in that manner, so we are surprised to see this approach may not be improving mean recall. Prior knowledge approaches such as GB-Net [156], which utilizes an augmented message passing approach that is also fed information from knowledge bases (e.g., ConceptNet [121]), appear to have a similar behaviour. Though we do not know the recall performance of EBM [125], the approach of HL-Net [79] also attempts to impose certain structure on the proposed scene graphs (utilizing homophily and heterophily in the case of HL-Net), however on its own the approach still appears to be falling into the trap of predicting the

more common predicates. Notably, the opposite behaviour is observed with TDE [129] and NICE [73] where their recall score is actually lower than our statistical baselines, but their mean recall scores are higher. This implies that these methods appear to truly be performing less biased relationship prediction and certainly indicates that they are gaining a more robust understanding of relationships that goes deeper than statistics.

Overall, the statistical baselines strangely hold up quite well compared to even recent approaches. Note that effectively the number of 'learned' parameters for relationship prediction in our statistical baselines does not exceed 20,000 for the most complex baseline we show, this is in stark difference to the millions, and tens of millions of learnable parameters that can be found in the relationship detection portions of the approaches we compare with. Our relationship prediction models are trained in a matter of seconds, again contrasted with the hours and days required by the more complex approaches. Furthermore, we only train our classifiers on the ground truth bounding box data, and hence train a single model that performs all 3 subtasks (PredCL, SGCl, and SGDet) whereas a common trend in the literature is training with the non gold-standard bounding box data that emerges from the object detector in each of the 3 models leading to even longer training times and hyperparameter tuning. In the case of SGCl and SGDet, we were still able to show meaningful results when we coupled a deep learning based object detector with our simple relationship detectors, even if the classifiers were not trained with the likely noisy boxes that the detector outputs.

A common justification for the performance of scene graph generation approaches is that the long tailed distribution of the data in the VG dataset caused low performance. Approaches such as [129, 125, 32] demonstrate how this can be overcome at an algorithmic level, with [73] even demonstrating how refining the ground truth can yield better results. Our classifiers also show that the VG data, while not exceptionally clean from noise, still can be used in its current form for better or equivalent performance than what had been achieved even in recent years simply by observing its statistics.

The main conclusion we can draw from our experiments is not that scene graph research is performing poorly, it's that the more successful approaches appear to understand the Visual Genome data and its limitations significantly better and account for it in their approaches. In a newly burgeoning field like scene graph generation it is important to explore alternative approaches and identify what strategies appear to be working, and why they do. One observation is that that approaches that tended to also utilize more of the 'visual' signal also perform better. This is done by TDE [129] which modifies its underlying VCTree [130] approach by utilizing causality and comparing the visual signal of the counterfactual with what is being observed in an image in terms of visual features. It is also emphasized in NICE [73] that operates on the ground truth data explicitly. Also

noteworthy is that scene graph generation is done on a 3D world by models that have only observed and understood 2D images. Topological relationships and angular configurations seemed to be a discriminatory feature even in this 2D space, allowing predictions about even unseen triplets. It's possible they will also serve as a powerful feature when looking at the 3D topology of relationships if that data were available. We don't believe it's possible to predict whether the performance improvements of successive approaches to scene graph generation would translate to a cleaner dataset than the Visual Genome. Finally, there are more paradigms to scene graph generation that warrant further exploration as there may be alternative approaches that could yield more robust, explainable and generalizable relationship generation.

5.4 Scene Graph Generation With Topological Relationship Fields

Our results from Section 5.3 highlight just how informative class labels can be for relationship detection. This is not an unexpected result in and of itself, after all even humans exhibit a bias in choosing a statistically likely relationship just based on knowing what objects are sharing that relationship [119, 34]. There is certainly value in predicting the objects first, and the relationship bias networks build is not a completely bad bias since certain relationships are quite unlikely or even impossible between certain objects. The drawback is that scene graph generation networks appear to heavily rely on these class labels for their relationship predictions. Unlike humans who are capable of overcoming their biases if faced with an unusual relationship between objects (e.g., a zebra wearing a hat), networks are unlikely to do so (the zero shot recall metric is a good indicator of this). The correlations between object labels and relationships also likely serve as an 'easy' cue for them to hone in on during training [46] and this could be causing networks to bypass building a more robust understanding of visual relationships. We also noticed a trend where more 'visual' models appeared to perform more predictably, with approaches like TDE [129] being more able to predict less frequently occurring labels a lot better.

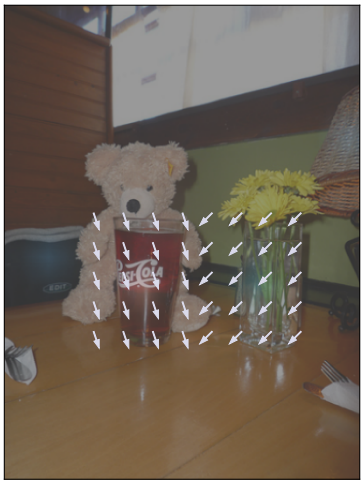
To this end we aim to explore whether it's possible to predict visual relationships solely based on visual data (image pixels) and completely devoid of object labels. We modify the relationship detection portion of scene graph generation to remove the reliance on class labels and allow for a separate representation for objects and relationships. We specifically design a data representation to be used in a deep learning setting in a manner such that the networks cannot rely on the aforementioned bias. In our representation, relationships are converted from triplets that must be predicted, to pixel level annotations that can be

regressed in a per pixel fashion. In a nutshell, we encode a relationship that connects two objects using a pixel-level 'vector field' that acts in the regions that the relationship is occurring. This data is not available in the Visual Genome dataset, and we therefore must approximate the location of the relationships by utilizing the topological information of the bounding boxes connected by it.

During training we convert the existing relationship triplets in the ground truth into our representation, and the networks are trained to regress that representation only. We then compare that representation with the locations of the detected objects after both are generated by the network to create the relationship triplets for scene graph generation. A benefit of this regressed output is its explainablity, where networks output a representation that can allow us to localize and understand where the inferred relationships are occurring. Building on the work in [81], we also utilize a fully convolutional approach to scene graph generation, an approach that is still relatively unexplored. This carries with it the benefit of being significantly less computationally intensive than contemporary scene graph approaches. Our results show promise, however even with our refined representation, the fully convolutional and 'single stage' approach to scene graph generation still lags behind its counterparts.

5.4.1 Topological Relationship Fields

Effectively all approaches to scene graph generation operate in two 'stages', the first stage is concerned with detecting the objects and the second is concerned with finding possible relationships between the objects. Approaches first utilize a 'detector' backbone (e.g., Resnet) to generate feature maps for the image, then detect the object bounding boxes and labels (whether predicted in SGDet mode, or partially or fully given in SGCl and PredCl respectively). The object labels and feature maps (and in some cases the feature maps originating from the union box containing both objects) are then re-used as the 'input' to the relationship detection portion and one of many approaches is utilized to predict the final scene graph (Section 2.5 describes some of these approaches in more detail). In FCSGG [81], the authors propose the first, and to our knowledge only, 'single stage' scene graph generator opting to utilize a fully convolutional network approach to generate scene graphs from images. Their proposed 'detector' is a simple additional prediction head appended to a Centernet [166] type architecture, which generates object predictions in an anchor-free manner.



(a) cup *sitting on* table

(b) children *under* umbrella

(c) sign *hanging on* post

(d) bus *has* window

(e) car *behind* bus

(f) woman *wearing* shirt

Figure 5.1: Examples of our Topological Relationship Fields. The generating relationship predicates are italicised. Areas of action of the TRFs are exaggerated for better viewing. In every relationship the fields point from the subject towards the object and occur within an area that is defined by the topological configuration of both objects.

5.4.1.1 Part Affinity Fields in Relationship Detection

They key to FCSGG's approach is in the manner they represent their data for single stage prediction. In FCSGG, the authors take influence from Part Affinity Fields (PAF) [8], an approach to perform 2D human pose estimation. In PAF, every limb is represented by a vector pointing from the start of the limb to the end of it, this vector is then broken down by pixel into a vector 'field' of pixels along the direction of the limb. This per pixel vector fields representation is used to model the limbs of humans, and as a representation learned by a pose estimation network. In FCSGG, a relationship triplet (<subject, predicate, object>) is modeled as a vector originating from the center of the subject and pointing towards the center of the object. This same vector is broken down into smaller vectors in a per pixel manner to create a vector 'field', referred to as a Relationship Affinity Field (RAF), representing the relationship. For a limb or a relationship in the PAF and the RAF representations respectively, the fields are 2D per pixel unit vectors on the segment connecting their start and end points, and zero vectors outside this segment[2].

The start and end points of a limb are a logical start and end point of the vector field for limb detection, mainly because the 'limbs' detected in PAF [8] are rigid sub-parts of the human body, e.g., from shoulder to elbow. The vector itself is aligned with a human bone or rigid system of bones and it usually lies within the pixels of the limb it's describing. On the other hand, subject and object centers in a relationship don't usually obey the same rigidity. A vector directed from the center of the bounding box containing the subject to the center of the bounding box containing an object may not align with the the acting area of the relationship or even with the visual or pixel data of either object. While the work done in FCSGG [81] showed that the PAF formulation can be utilized for scene graph generation and allowed for a fully convolutional approach to it, their data representation may not be ideal. Not all relationships, even the ones present in the VG dataset or VG150 subset, can be represented in the same manner. Some relationships, such as 'holding', imply a support relationship between one object and another, whereas other relationships, e.g., 'next to', act from a distance, finally, as we show in Chapter 4 some coarse and overloaded relationships such as 'has' can imply several different finer relationships. While the PAF representation is applicable for pixel-wise regression when determining human parts in [8], the same representation shouldn't be directly used when if attempting to perform pixel-wise regression for relationships.

In a 3D representation of a 3D world, the acting locations of these relationships are usually definable, e.g., using real world physics. When looking at a 2D image of the 3D

[2]Functionally, these segments can have a width of 1 or more pixels and are effectively long rectangles with a small width.

world, these same relationship locations may become slightly less intuitive to define and represent on a 2D pixel grid, for example a support relationship may be hidden from view. However if pixel level data about the objects is known (i.e., segmentation masks), the relationship locations may be easier to parse. The VG dataset does not offer any such data, and all that it contains are object bounding boxes in 2D images. Our work in Chapter 4 and Section 5.3 highlighted the utility of 2D bounding box topology in relationship detection. As a result, we propose Topological Relationship Fields (TRF), a novel vector field representation that is grounded in bounding box topology to represent the relationships present in an image in a pixel-wise manner. In TRFs we represent a relationship as a directed vector field between two objects that changes depending on the topological configuration of the bounding boxes of the objects. While this may not be as complete of a representation as the one we can get from perfectly annotated 3D data, it does allow us to capture the structure of visual relationships given the data we have. For a given image with known bounding boxes of the objects in that image and the relationships between those objects, we can generate the Topological Relationship Field representation of those relationships. This TRF representation is an alternative representation of those relationships that we can utilize with a fully convolutional neural network for relationship detection.

5.4.1.2 TRF Definition and Generation

For every image our TRFs, F, are a set of $\mathcal{P}$ two dimensional vector fields, $\{F_p\}$, of the same width and height as the original image, with $\mathcal{P}$ being the number of relationship predicate classes in the dataset (e.g., $\mathcal{P} = 50$ for the VG150 dataset). In the case an image of width w and height h in a dataset with $\mathcal{P}$ relationship predicate classes, our TRFs $F = \{F_p\} \in \mathbb{R}^{\mathcal{P} \times w \times h \times 2}$, where for every predicate class p, we generate a 2D vector field spanning the whole image and represented by a vector's x and y components at every pixel. All of the relationships in one image are represented using one set of TRFs corresponding to that image, with every F_p encoding all relationships that correspond to predicate class p .

For a given relationship triplet <subject, predicate, object>, we observe the bounding boxes B_S and B_O corresponding to its subject and object respectively, and the bounding box centers C_S and C_O representing the geometric centers of subject and object bounding boxes respectively. The unit vector $\widehat{\mathbf{C_S C_O}}$ in the direction of $\overrightarrow{C_S C_O}$ is defined using the pixel coordinates of the subject and object centers, C_S and C_O respectively, as:

$$\widehat{\mathbf{C_S C_O}} = \frac{C_O - C_S}{||C_O - C_S||_2} \tag{5.1}$$

The TRF F_p describing the relationship with predicate p at pixel location $\mathbf{X} = (x, y) \in (w, h)$ depends on the 2D topological configuration of the bounding boxes:

- **Disjoint:** This likely means that the two objects do not touch in the real world and we borrow the RAF [81]/PAF [8] representation to represent the relationship between them for this configuration. A field composed of unit vectors pointing in the direction of $\widehat{\mathbf{C_S C_O}}$ and on the segment $\overline{C_S C_O}$ connecting the centers of the two objects is used to represent the relationship.

$$F_{p,\mathbf{X}} = \begin{cases} \widehat{\mathbf{C_S C_O}}, & \text{if } \mathbf{X} \in \overline{C_S C_O} \\ 0, & \text{otherwise} \end{cases} \tag{5.2}$$

- **Touching:** This implies the two objects may be interacting somewhere around the edge of the bounding boxes where they touch. A vector field composed of unit vectors pointing in the direction of $\widehat{\mathbf{C_S C_O}}$ on an area surrounding the touching edges of the bounding boxes is used.

$$F_{p,\mathbf{X}} = \begin{cases} \widehat{\mathbf{C_S C_O}}, & \text{if } \mathbf{X} \text{ is around } B_S \cap B_O \\ 0, & \text{otherwise} \end{cases} \tag{5.3}$$

- **Overlap:** This implies the two objects may be interacting somewhere within the area of overlap. A vector field composed of unit vectors pointing in the direction of $\widehat{\mathbf{C_S C_O}}$ within the area of overlap is used.

$$F_{p,\mathbf{X}} = \begin{cases} \widehat{\mathbf{C_S C_O}}, & \text{if } \mathbf{X} \in B_S \cap B_O \\ 0, & \text{otherwise} \end{cases} \tag{5.4}$$

- **Subject Fully Contains Object:** This occurs in 2 separate topological configurations [Object In Subject] and [Subject Covers Object], the difference between them is that in the latter configuration, the object touches an internal boundary of the subject bounding box. In both cases, the relationship is likely occurring somewhere within the boundaries of the object bounding box. A vector field composed of unit vectors pointing in the direction of $\widehat{\mathbf{C_S C_O}}$ all throughout the smaller, object, bounding box is used to represent this relationship.

$$F_{p,\mathbf{X}} = \begin{cases} \widehat{\mathbf{C_S C_O}}, & \text{if } \mathbf{X} \in B_O \\ 0, & \text{otherwise} \end{cases} \tag{5.5}$$

- **Object Fully Contains Subject:** This is the inverse of the 'Subject Fully Contains Object' case and occurs in the topological configurations [Subject In Object] and [Object Covers Subject], again the difference between these configurations is that in the latter configuration, the subject touches an internal boundary of the object bounding box. In both topological configurations, the relationship is likely occurring somewhere within the boundaries of the subject bounding box. A vector field composed of unit vectors pointing in the direction of $\widehat{\mathbf{C_S C_O}}$ all throughout the smaller, subject, bounding box is used to represent this relationship.

$$F_{p,\mathbf{X}} = \begin{cases} \widehat{\mathbf{C_S C_O}}, & \text{if } \mathbf{X} \in B_S \\ 0, & \text{otherwise} \end{cases} \tag{5.6}$$

Note that aside from the Disjoint case, the TRFs all occur within the area of intersection between B_S and B_O[3], with this area getting larger and smaller depending on how the two boxes are configured. We opted to define the cases individually for clarity when encoding and decoding the TRFs, and generality in case they are used with non bounding box data as well, e.g., pixel-wise mask object data. Utilizing a vector representation is necessary to distinguish between the object and the subject in a given relationship. Opting to use a scalar value per predicate channel may indicate whether a relationship of that predicate exists at the specific pixel location but will not be able to relay information about which 'direction' this relationship is occurring and which of the bounding boxes correspond to its subject or object.

5.4.1.3 Decoding TRFs

These vector fields are encoded according to the above rules for every relationship triplet in the ground truth to create the training set of TRFs for every image independently. In some cases when generating the ground truth TRFs, the vectors of multiple relationship triplets may overlap at certain pixel locations, when this occurs we take the mean vector at that location. Similarly, we can decode the TRFs representation in an inverse manner, to get back to the triplet representation found in the ground truth dataset. We decode the TRFs by integrating the vector fields in the relevant areas.

Specifically, for two detected objects O_i and O_j in the image, we denote their detected bounding boxes B_i and B_j, with centers c_i and c_j, and predicted object class probabilities d_i and d_j (given by an object detector). We calculate a relationship scores between O_i and

[3] Around the area of intersection in the case of Touching

O_j by summing the projections of the TRF vectors onto the unit vector in the direction of the unit vector $\widehat{\mathbf{c_i c_j}}$ within the relevant topological region τ_{ij} as defined by the topological configuration of B_i and B_j. The relationship scores are found by the pixel-wise summation:

$$K_{ij} = \frac{d_i \cdot d_j}{t_{ij}} \sum_{p \in \mathbb{R}^{\mathcal{P}}} \sum_{\mathbf{X} \in \tau_{ij}} F_{p,\mathbf{X}} \cdot \widehat{\mathbf{c_i c_j}} \tag{5.7}$$

Where t_{ij} is the number of pixels in τ_{ij}. $K_{ij} \in \mathbb{R}^{\mathcal{P}}$ and every element κ_{ij} in K_{ij} is the probability of a relationship κ between O_i and O_j, specifically a triplet $<O_i, \kappa_{ij}, O_j>$. The product $d_i \cdot d_j$ weights the summation by the scores of the bounding box detection. The values in K_{ij} may be negative, implying an inverse relationship, and K_{ji} can simply be calculated as $K_{ji} = -K_{ij}$ to find the triplets with flipped subject and object $<O_j, \kappa_{ji}, O_i>$rather than recomputing the same pixel-wise summation in equation 5.7.

5.4.1.4 Benefits, Drawbacks and Viability

Our proposed TRF representation of relationship data carries with it many benefits. The representation is in fact an interim representation that is explicitly explainable. A network trained with TRFs does not output relationship triplets directly, but is trained to output this interim representation that is then converted to the final scene graph. The TRF pinpoints the exact pixel locations corresponding to exact acting boundaries/areas between objects that led to classifying any relationship it eventually predicts. This visual representation shows exact visual evidence of where predicted relationships are occurring. This may serve to alleviate some of the impossible relationships predicted by two stage approaches, as noted in [24], where, for example, networks have learned that humans tend to wear glasses, and hence predict a 'wearing' relationship between every human and every pair of glasses in an image to maximize their performance. On the other side, TRFs can only represent relationships that are supported with visual, pixel level, data more effectively.

Our TRF representation allows relationships to be represented spatially in closer proximity to the pixels that are likely causing them. Intuitively, with TRFs relationships are localized closer to the acting boundaries of their respective subjects and objects, for example, a plate is not on the entirety of the table, but its relationship is really localized to whether the plate touches the table. While this doesn't happen in a pixel perfect manner given the data constraints and the fact that we don't know where the relationships are really occurring, utilizing the bounding box topology gives a good guess. The representation is flexible enough to be able to incorporate and be improved upon further with object mask data, or even 3D data, where the relationships can be represented with more precision spatially.

The TRF representation can utilize and model inverse relationships. While the data in the VG dataset does not take this into consideration and current approaches to scene graph generation do not utilize this, a small modification can allow the TRF representation to be used to specifically identify when inverse relationships are occurring, e.g., a strong 'negative' integral between two objects in the 'under' predicate implying that an 'above' relationship is the case, with an 'under' relationship occurring in the inverse direction. Instead of predicting inverse relationships as two separate classes, they can be predicted as a a single class that can occurs in two directions.

It compliments a fully convolutional scene graph generation approach, which, as discussed in [81], is a significantly less computationally intensive approach to scene graph generation. Since the TRF representation does not rely on object classes, it can also be included in a single stage scene graph generation approach. Furthermore, training a separate models for each evaluation mode (PredCl, SGCl, SGDet) is not necessary, unlike traditional scene graph generation approaches which utilize a separate model to improve performance in each mode.

On the other hand, the TRF representation has a few drawbacks as well. Out of all the possible 2D topological configurations for bounding boxes, we cannot represent the case of 'Equal'. In this case the two bounding boxes are of the exact same dimension and are exactly overlapping, the unit vector $\widehat{\mathbf{C_S C_O}} = \overrightarrow{\mathbf{0}}$, and our TRF representation will exhibit a vagueness regarding which bounding box corresponds to the subject and which to the object. Note that this also occurs in the RAF/PAF representation. However the case of exactly overlapping bounding boxes and an Equal topological configuration is quite rare in the VG150 dataset, as we show in Chapter 4 and does not affect our performance.

The TRF representation also does not directly utilize object class information. While this information appears to bias other approaches quite significantly, as we demonstrated in Section 5.3, this bias does serve a purpose in relationship detection and accounting for it would certainly improve performance.

Finally, we demonstrate the viability of the TRF representation by showing that it is in fact invertable, and relationship triplets encoded using TRFs can be decoded successfully. We design an oracle experiment, where we encode the relationships in the VG150 ground truth and measure how well the oracle can decode these relationships. For every image in the validation set, our oracle receives the ground truth encoded TRFs directly along with all the ground truth bounding boxes and classes. The oracle does not know which objects are connected by a relationship, or which relationships are present in the given image and must decode this information from the given TRF. Our oracle achieves a R@20/50/100 of 88.0/92.3/93.1 with graph constraints and a R@20/50/100 91.9/97.5/98.8 without graph

constraints, as well as a mR@20/50/100 of 87.2/90.5/91.0 with graph constraints and a mR@20/50/100 of 93.6/98.6/99.4 without graph constraints. Note that a recall or mean recall of 100% is not possible on the VG 150 dataset (with graph constraints) due to the fact that some objects have multiple relationships connecting them in the ground truth and only one of them is allowed to be predicted. The recalls and mean recalls of the oracle are an upper bound on the possible performance of a detector that utilizes TRFs as an intermediate representation. This experiment shows that the original relationship triplets can in fact be decoded from our TRFs.

5.4.2 Generating TRFs with Deep Learning

Our oracle experiment results demonstrate that our proposed Topological Relationship Fields can be inverted back into scene graphs (composed of <subject, predicate, object>triplets). We now aim to generate this intermediate representation from images using deep learning. As discussed in Section 5.4.1, the TRF representation lends itself well to the relatively unexplored single stage paradigm for scene graph generation. Building on the work presented in [81], we opt to utilize a fully convolutional approach for scene graph generation with our new representation.

Our network architecture is based on CenterNet [166, 81], an anchor free and single stage object detector similar to FCOS [135] discussed in Chapter 2. A 2D object detection CenterNet uses an interchangeable backbone network (for example, a Resnet [45]) for feature extraction from the image, and then predicts three separate dense feature maps that correspond to the bounding box characteristics of the detected objects. These feature maps are actually a dense, pixel-wise, prediction of the object's bounding box center, its center offsets (for recovering exact locations in down-sampled images and feature maps), and its size. We simply add an additional head to CenterNet that is tasked with regressing TRFs. A schematic of our proposed modification to CenterNet is presented in Figure 5.2.

Following the formulation of [166], we also model object detection as a dense prediction task. We denote an input image as $I \in \mathbb{R}^{W \times H \times 3}$, where W and H correspond to the image width and height respectively with 3 color channels. The image I is processed by a backbone network to produce a downsampled feature map with the new width and height $w = \lfloor \frac{W}{R} \rfloor$ and $h = \lfloor \frac{H}{R} \rfloor$ where R denotes the backbone output stride. The feature maps generated by the backbone network are then processed by four fully convolutional networks, the prediction 'heads', where each head is responsible for outputting one of the required predictions. Three of the four heads are responsible for object detection, and the fourth head is responsible for regressing the TRFs.

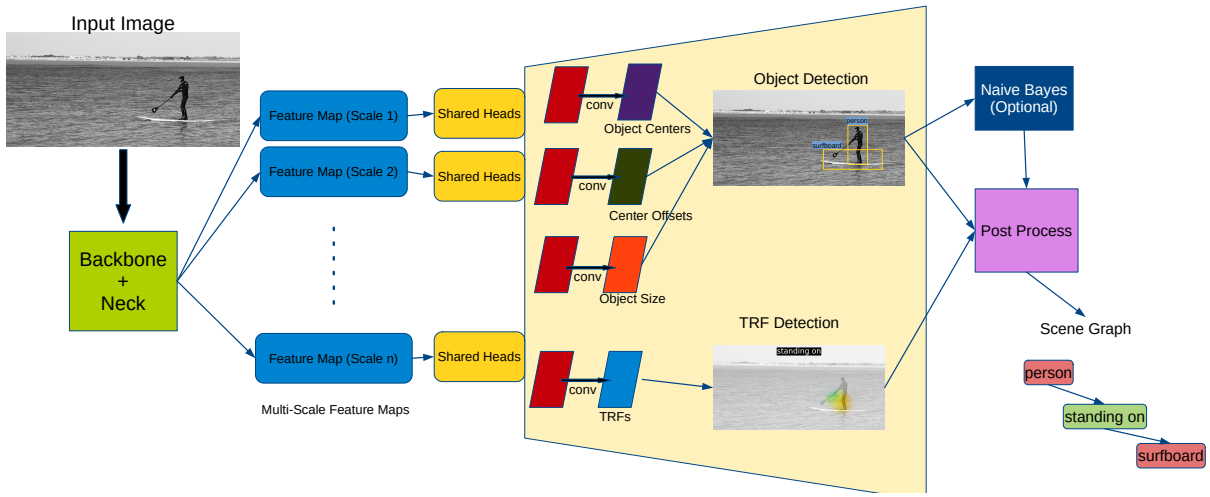


Figure 5.2: A schematic of a modified CenterNet [166] for use with relationship detection. Similarly to [81], we utilize an additional prediction head, on top of the 3 object detection heads of CenterNet, that is specifically tasked with outputting our TRFs for every image. The backbone and neck produce feature maps at multiple scales which the heads process. Prediction heads are shared across all scales. During inference, we combine the object predictions and the TRF predictions in a post processing step to generate the final scene graphs. Our naive Bayes approach can optionally be used to boost performance.

The object detection heads predict the object bounding box center heatmaps $\Phi \in \mathbb{R}^{C \times w \times h}$ where C denotes the number of object classes in the dataset, the object center local offsets $\Delta \in \mathbb{R}^{2 \times w \times h}$ which are used to undo the discretization errors caused by the feature map downsampling, and the object bounding box sizes $S \in \mathbb{R}^{2 \times w \times h}$ which holds the width and the height of the predicted bounding boxes.

Denote a set of ground truth objects in the image as $\mathbf{B} = \{b^i\}$, where for each ground truth object $b^i = (x_1^i, y_1^i, x_2^i, y_2^i, c^i)$, (x_1^i, y_1^i) are the coordinates of the top left corner of the bounding box, and (x_2^i, y_2^i) are the coordinates of the bottom right corner of the bounding box, and c^i is the class of the object.

The center of the bounding box is

$$\phi^i = (\phi_x^i, \phi_y^i) = (\frac{x_1^i + x_2^i}{2}, \frac{y_1^i + y_2^i}{2}) \tag{5.8}$$

The center offset of the bounding box due to using a stride of R is

$$\delta^i = (\delta_x^i, \delta_y^i) = \frac{\phi^i}{R} - \lfloor \frac{\phi^i}{R} \rfloor \tag{5.9}$$

The size of the bounding box is

$$s^i = (x_2^i - x_1^i, y_2^i - y_1^i) \tag{5.10}$$

For each object center with its location modified to match the stride R by $\widetilde{\phi^i} = \lfloor \frac{\phi^i}{R} \rfloor$ the value of $\Phi_{c^i,x,y} = 1$ at this center point. $\Phi_{c^i,x,y}$ is also 'splatted' using a Gaussian kernel to obtain the ground truth heat map for training. Specifically, following Law and Deng [72] and [166, 81], the object center is 'splatted' using a bivariate Gaussian distribution in the x and y axes, where for an object center ϕ^i of class c^i the object center heatmap Φ is modified for the object class in the region of the object center as

$$\Phi_{c^i,x,y} = exp(-\frac{||x - \lfloor \frac{\phi_x^i}{R} \rfloor||_2^2}{2\sigma_x^2} - \frac{||y - \lfloor \frac{\phi_y^i}{R} \rfloor||_2^2}{2\sigma_y^2}) \tag{5.11}$$

Where R is the stride and σ_x and σ_y control the spread of the distribution in the x and y direction. σ_x and σ_y are computed based on the object size s^i.

$$(\sigma_x, \sigma_y) = \frac{1}{3}\lfloor \frac{(\sqrt{2} - 1)s^i}{R} + 1 \rfloor \tag{5.12}$$

Since multiple objects of the same class may exist in close proximity, the Gaussian spreads are found for each object and an element-wise maximum value is taken for every point in the ground truth.

During training time the predicted center heatmaps $\widehat{\Phi} \in [0,1]^{C \times \frac{W}{R} \times \frac{H}{R}}$ are supervised using a pixel-wise regression with focal loss [76] as in [166, 81]. Specifically

$$\mathcal{L}_{cen} = \mathcal{L}_{\Phi_{c,x,y}} = \begin{cases} -(1 - \widehat{\Phi}_{c,x,y})^{\alpha} \log(\widehat{\Phi}_{c,x,y}) & \text{if } \Phi_{c,x,y} = 1 \\ -(\widehat{\Phi}_{c,x,y})^{\alpha}(1 - \Phi_{c,x,y})^{\beta} \log(1 - \widehat{\Phi}_{c,x,y}) & \text{otherwise} \end{cases} \tag{5.13}$$

Where the hyperparameters $\alpha = 2$ and $\beta = 4$ are set following [72, 166, 81].

The loss for the predicted center offsets $\widehat{S}$ and box size $\widehat{\Delta}$ heatmaps are calculated using L1 loss only at the center locations:

$$\mathcal{L}_{off} = \mathcal{L}_{\Delta_{x,y}} = |\widehat{\Delta}_{x,y} - \delta^i| \text{ wherever } \Phi_{c,x,y} = 1 \text{ for any } c \tag{5.14}$$

$$\mathcal{L}_{size} = \mathcal{L}_{S_{x,y}} = |\widehat{S}_{x,y} - \widetilde{s^i}| \text{ wherever } \Phi_{c,x,y} = 1 \text{ for any } c \tag{5.15}$$

Where $\widetilde{s^i} = \lfloor \frac{s^i}{R} \rfloor$ are the sizes scaled by the stride.

The object detection loss is

$$\mathcal{L}_{det} = \frac{1}{N} \sum_{c,x,y} (\mathcal{L}_{cen} + \lambda_{off} \mathcal{L}_{off} + \lambda_{size} \mathcal{L}_{size}) \tag{5.16}$$

Where N is the total number of objects in the image and the weighting parameters $\lambda_{off} = 1$ and $\lambda_{off} = 0.1$ are set empirically following [166, 81].

We utilize a fourth head in parallel to the existing object detection heads specifically to produce the TRF for the input image. Similarly to how the three fully convolutional heads in the original CenterNet [166] receive features extracted using the backbone network and are tasked with outputting object centers, offsets, and sizes, in our case, and in a similar manner to [81], we utilize a fourth head tasked with outputting the image TRF. This additional head receives the same feature map input and is a simple fully convolutional head trained to regress the TRF $\mathbf{F}$.

For a predicted TRF $\widehat{\mathbf{F}}$ we calculate the loss in a per pixel manner, and use an L1 regression loss, though $\mathcal{L}_{reg}$ can be any regression loss (e.g., L2 or smooth L1). This loss is also weighted by a weighting tensor $\mathbf{\Lambda} \in \mathbb{R}^{\mathcal{P} \times w \times h \times 2}$ depending on the ground truth relationships at specific locations. Specifically:

$$\mathcal{L}_{rel} = \mathbf{\Lambda} \mathcal{L}_{reg}(\widehat{\mathbf{F}}, \mathbf{F}) \tag{5.17}$$

Where the weighting tensor $\mathbf{\Lambda}$ scales the loss inversely to the distance or area where the relationship occurs

$$\mathbf{\Lambda} = \begin{cases} \frac{1}{||\widehat{C_S C_O}||_2} & \text{where a disjoint relationship exists} \\ \frac{1}{\text{Area}(B_S \cap B_O)} & \text{where a non disjoint relationship exists} \\ W_{neg} & \text{where no relationship exists} \end{cases} \tag{5.18}$$

and W_{neg} is a hyperparameter that controls the penalty for predicting erroneous relationships, which we empirically set to 1.

The total training loss for the model accounts for both TRF regression and object detection:

$$\mathcal{L}_{tot} = \mathcal{L}_{det} + \lambda_{rel} \mathcal{L}_{rel} \tag{5.19}$$

with λ_{rel} controlling the weight of the relationship loss in the total loss which we keep at 1.

A backbone network takes in the input image and outputs a feature map that is processed by each of the heads separately. This backbone network can be combined with a 'neck' for multiscale predictions. In our case, we experiment with two common backbone networks: a ResNet50 [45] and an HRNet [140] backbone. We utilize a ResNet-50 backbone [45] with a Bidirectional FPN as a neck [75, 128] which we discussed in Chapter 3 and has been successfully used in object detection by [135, 134, 128, 81] and for relationship detection in [81]. We also experiment with utilizing an HRNet [140] (specifically HRNet48) which was also used in [81]. This backbone maintains a **H**igh **R**esolution feature map throughout its structure (where its HR namesake comes from), and may be able to preserve specific image-level details better.

In order to compare with [81], we perform our ResNet predictions at 4 scales using a FPN with strides $R = [8, 16, 32, 64]$ with the prediction heads shared across all 4 scales, and our HRNet predictions at 5 scales with strides $R = [8, 16, 32, 64, 128]$. In multi-scale prediction the ground truth for both object detection and TRFs is modified to include only the objects and relationships that exist at the specific scale. In the case of our ResNet objects and relationships objects and relationships fall into the respective stride scale $R = [8, 16, 32, 64]$ if their longest bounding box edge is within {[0,64],[64,128],[128,256],[256,1024]} pixels respectively in the original image. Whereas in the case of the HRNet objects and relationships fall into the respective stride scale $R = [8, 16, 32, 64, 128]$ if their longest bounding box edge is within {[0,64],[64,128],[128,256],[256,512],[512,1024]} pixels respectively in the original image.

The four prediction heads are each composed of 4 convolution blocks (consisting of: 3x3 conv - Batch Norm - ReLU), with each head followed by a 1x1 convolution to get to the required number of channels: C for $\Phi \in \mathbb{R}^{C \times w \times h}$ where C denotes the number of object classes in the dataset, 2 for $\Delta \in \mathbb{R}^{2 \times w \times h}$ to account for the center offsets in the x and y direction, 2 for $S \in \mathbb{R}^{2 \times w \times h}$ to account for box length and width, and $\mathcal{P} \times 2$ for $F \in \mathbb{R}^{\mathcal{P} \times w \times h \times 2}$ where $\mathcal{P}$ is the number of relationship predicates in the dataset and 2 is to represent the TRF vector in the x and y directions.

Training is done on the VG150 training set until convergence is observed, and the trained model is evaluated on the VG150 test set. We train one model using each backbone and evaluate on all three standard scene graph generation tasks: PredCl, SGCl, and SGDet. During training the target outputs are the TRFs generated from the ground truth scene graphs for the specific image. During inference and evaluation time these TRFs are post-processed and decoded using the pixel-wise summation in equation 5.7 to produce the image scene graph. This summation is done efficiently in code for every pair of detected objects (a maximum of 100 objects for every scale) and does not impede performance.

Noting the utility of 'non-visual' features in Section 5.3, we also choose to measure whether utilizing these statistics can boost the performance of the network for relationship detection. In these experiments we modify the summation in equation 5.7 to add a Naive Bayes MAP probability given the object classes and their topological configuration. Both object class and topological configuration are known at post-processing time since all the network heads (including the object detection heads) have already produced their outputs. Equation 5.7 is modified as follows:

$$K_{ij} = \frac{d_i \cdot d_j}{t_{ij}} \cdot \mathbb{P}(p \in \mathbb{R}^{\mathcal{P}} | g_i, g_j, B_i, B_j) \odot \sum_{p \in \mathbb{R}^{\mathcal{P}}} \sum_{\mathbf{X} \in \tau_{ij}} F_{p,\mathbf{X}} \cdot \widehat{\mathbf{c_i c_j}} \tag{5.20}$$

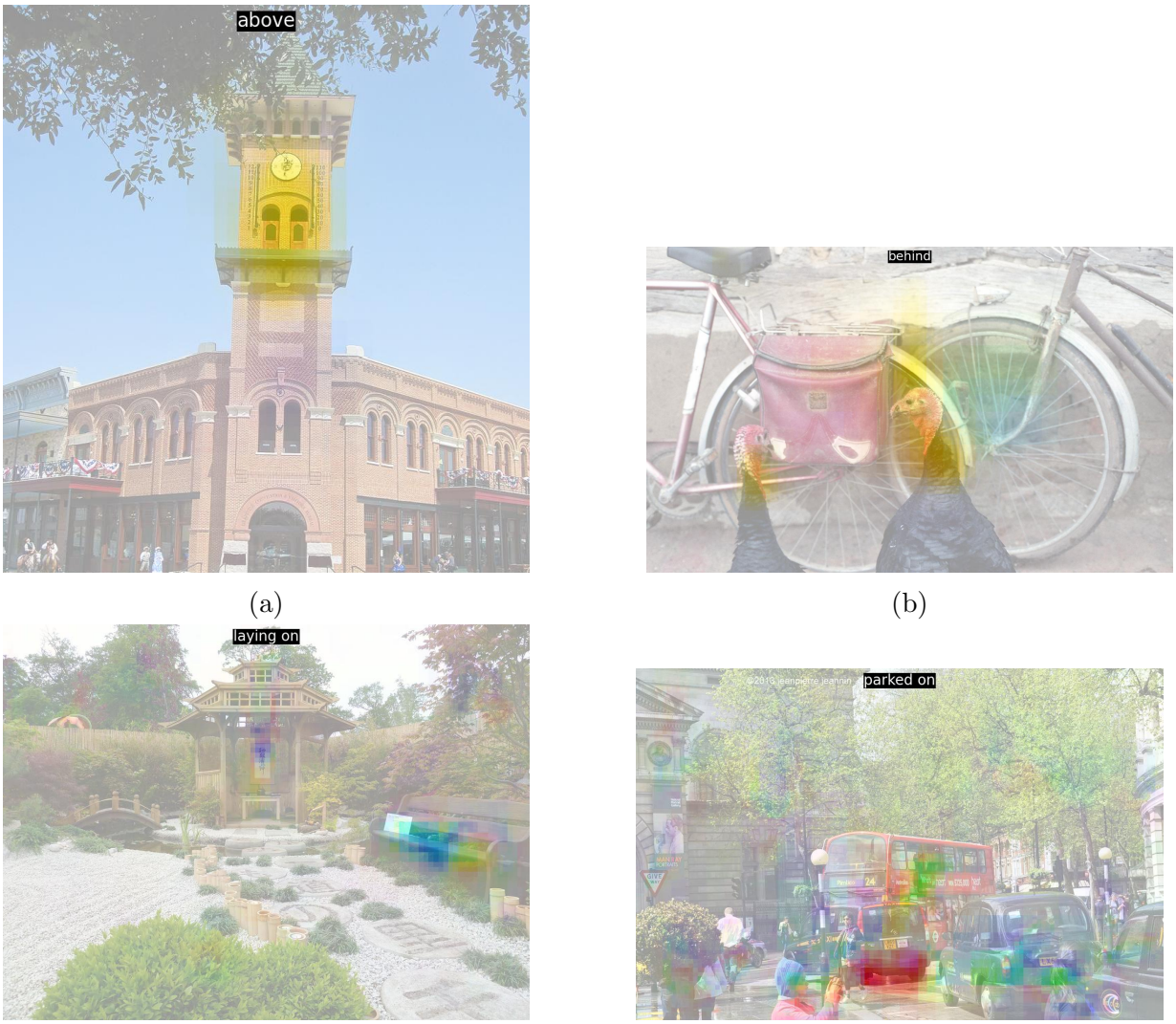
Where the detected bounding boxes are denoted by B_i and B_j, and their predicted classes are g_i and g_j, and $\mathbb{P}(p|g_i, g_j, B_i, B_j) \in \mathbb{R}^{\mathcal{P}}$ is the MAP probability of every predicate given the box classes and bounding box configuration, and $\odot$ is an element-wise multiplication operation. This probability function does not aid in detecting new relationships but serves to reorder and re-prioritize already detected visual relationships using the TRFs.

5.4.3 Results and Discussions

Our mean recall results are presented and compared with other scene graph generation approaches in Table 5.4, our recall results are presented in Table 5.5, and our zero shot recall results are presented in Table 5.6. We also present sample TRF outputs in Figure 5.3.

Overall our networks trained to output TRFs perform relatively consistently regardless of backbone choice. Small improvements can be seen with the more complex backbone (HRNet-48), which become more apparent in the SGCL and SGDet modes where object detection is of more importance. It appears the larger capacity of HRNet-48 in comparison to ResNet50 may be able to learn a representation that is better suited to both object detection and relationship detection. However we argue the ResNet50 is learning a more robust relationship representation as evidenced by its zero shot recall performance where it sometimes outperforms the larger and more complex HRNet-48.

Boosting detected relationships using the prior knowledge Naive Bayes model also almost consistently improves recall performance across all modes. In the boosted modes, the Naive Bayes predictor is treated as a simple 'prior knowledge' model. Unlike in Section 5.3, we do not utilize it for direct prediction instead it simply 'confirms or denies' what our TRF trained networks predicted. This also justifies the discrepancy in performance



(a) (b)

(c) (d)

Figure 5.3: Examples of predicted Topological Relationship Fields. Each figure is the direct, unmodified, TRF output and offers justification for where and why a relationship could be predicted. In figure (a), the clock tower exhibits 'above'-ness (in a specific direction) with respect to its surroundings, whereas the bike in (b) exhibits 'behind'-ness with respect to the turkeys. Even though nobody is laying on the bench in figure (c), it still is primed to predict this relationship but won't do so without another object interacting with it in the specific zone, and the car is figure (d) shows a strong 'parked on' response with the street.

Table 5.4: Mean recall rates @ 20/50/100 on the VG150 dataset for our single stage scene graph generation approach that utilizes TRFs as well as modern approaches to scene graph generation. We isolate the only other single stage scene graph generation approaches and indicate the backbone architectures for both our and their approaches in parenthesis. Our boosted models are indicated by '+NB'.

	PredCl			SGCl			SGDet		
	mR@20	mR@50	mR@100	mR@20	mR@50	mR@100	mR@20	mR@50	mR@100
IMP [146, 129]	-	9.8	10.5	-	5.8	6	-	3.8	4.8
Motif [158, 129]	10.8	14	15.3	6.3	7.7	8.2	4.2	5.7	6.6
VCTree [130, 129]	14	17.9	19.4	8.2	10.1	10.8	5.2	6.9	8
KERN [16]	-	17.7	19.2	-	9.4	10	-	6.4	7.3
VCTree+TDE [129]	17.2	23.3	26.6	8.9	11.8	13.4	6.3	8.6	10.3
GPSNet [78]	-	-	22.8	-	-	12.6	-	-	9.8
GB-Net [156]	-	19.3	20.9	-	9.6	10.2	-	6.1	7.3
VCTree + EBM [125]	14.2	18.2	19.7	10.4	12.5	13.4	5.7	7.7	9.1
Segmentation Grounded Motif [59]	14.5	18.5	20.2	8.9	11.2	12.1	6.4	8.3	9.2
Segmentation Grounded VCTree [59]	15	19.2	21.1	9.1	11.6	12.3	6.3	8.1	9
RAAL [80]	14.4	18.3	19.9	7.9	6.6	10.3	4.9	6.5	7.4
Schemata [115]	-	19.1	20.7	-	10.1	10.9	-	-	-
VCTree + NICE [73]	-	29.9	32.3	-	19.9	21.3	-	11.9	14.1
VCTree + EBM + NARE [32]	21	24.9	26.5	14	16.2	17.1	7.8	10.1	11.8
HL-Net [79]	-	-	22.8	-	-	13.5	-	-	9.2
SOTA Single Stage Approaches:									
RAF (ResNet-50) [81]	4.2	5.7	6.7	2.2	2.9	3.3	1.9	2.7	3.3
RAF (HRNet-48) + freq [81]	4.9	6.3	7.1	2.9	3.7	4.1	2.7	3.6	4.2
Our Single Stage Approaches:									
TRF (ResNet-50)	4.6	6.0	6.9	2.4	3.1	3.4	2.1	2.9	3.3
TRF (HRNet-48)	5.3	6.6	7.5	3.2	3.8	4.1	2.6	3.6	4.0
TRF (ResNet-50) + NB	5.0	6.3	7.1	2.9	3.6	3.9	2.5	3.3	3.8
TRF (HRNet-48) + NB	5.9	7.0	7.9	3.6	4.2	4.5	3.0	3.9	4.3

Table 5.5: Recall rates @ 20/50/100 on the VG150 dataset for our single stage scene graph generation approach that utilizes TRFs as well as modern approaches to scene graph generation. We isolate the only other single stage scene graph generation approaches and indicate the backbone architectures for both our and their approaches in parenthesis. Our boosted models are indicated by '+NB'. Blank rows left in to correspond with Table 5.4.

	PredCl			SGCl			SGDet		
	R@20	R@50	R@100	R@20	R@50	R@100	R@20	R@50	R@100
IMP [146, 129]	54.34	61.05	63.06	34.01	37.48	38.5	18.09	25.94	31.15
Motif [158, 129]	58.46	65.18	67.01	35.63	38.92	39.77	25.48	32.78	37.16
VCTree [130, 129]	59.02	65.42	67.18	42.77	46.67	47.64	24.53	31.93	36.21
KERN [16]	-	65.8	67.6	-	36.7	37.4	-	27.1	29.8
VCTree+TDE [129]	39.1	49.9	54.5	22.8	28.8	31.2	14.3	19.6	23.3
GPSNet [78]	67.6	69.7	69.7	41.8	42.3	42.3	22.3	28.9	33.2
GB-Net [156]	-	59.3	61.3	-	34.6	35.4	-	20.7	27.6
VCTree + EBM [125]	-	-	-	-	-	-	-	-	-
Segmentation Grounded Motif [59]	-	-	-	-	-	-	-	-	-
Segmentation Grounded VCTree [59]	-	-	-	-	-	-	-	-	-
RAAL [80]	59.1	66.2	68.4	33.5	36.7	37.6	21.7	27.3	29.9
Schemata [115]	-	66.9	68.4	-	39.1	39.8	-	-	-
VCTree + NICE [73]	-	55.0	56.9	-	37.8	39.0	-	27	30.8
VCTree + EBM + NARE [32]	-	-	-	-	-	-	-	-	-
HL-Net [79]	60.7	67	68.9	38.8	42.6	43.5	26	33.7	38.1
SOTA Single Stage Approaches:									
RAF (ResNet-50) [81]	28.0	35.8	40.2	13.9	17.7	19.6	11.4	15.7	19.0
RAF (HRNet-48) + freq [81]	33.4	41.0	45.0	19.0	23.5	25.7	16.1	21.3	25.1
Our Single Stage Approaches:									
TRF (ResNet-50)	28.9	37.3	41.3	13.9	18.1	19.9	12.1	16.5	19.6
TRF (HRNet-48)	34.1	42.0	45.6	19.3	24.0	26.0	15.8	21.0	24.6
TRF (ResNet-50) + NB	30.9	38.2	44.3	15.1	20.2	24.9	14.8	18.8	21.5
TRF (HRNet-48) + NB	35.0	44.4	48.1	20.7	26.2	28.1	17.4	23.4	26.0

Table 5.6: Zero shot recall rates @ 20/50/100 on the VG150 dataset for our single stage scene graph generation approach that utilizes TRFs as well as some modern approaches to scene graph generation. We isolate the only other single stage scene graph generation approaches and indicate the backbone architectures for both our and their approaches in parenthesis. Our boosted models are indicated by '+NB'. We include the results for 'no graph constraint' zero shot recall rates @ 20/50/100 (ng-zsR@20/50/100) where multiple relationships between objects are allowed since we discuss this in our results section.

	PredCl		SGCl		SGDet	
	zsR@20/50/100	ng-zsR@20/50/100	zsR@20/50/100	ng-zsR@20/50/100	zsR@20/50/100	ng-zsR@20/50/100
Motif [158, 129]	1.9/-/7.2	-	0.3/-/1.2	-	0/-/0.5	-
VCTree [130, 129]	1.8/-/7.1	-	0.4/-/1.2	-	0.1/-/0.7	-
VCTree+TDE [129]	-/14.3/17.6	-	-/3.2/4	-	-/2.6/3.2	-
Segmentation Grounded Motif [59]	4.1/-/10.5	-	0.8/-/2.5	-	0.1/-/1	-
Segmentation Grounded VCTree [59]	4.6/-/10.6	-	0.8/-/2.5	-	0.3/-/1.5	-
VCTree + TDE [129] + NARE [32]	9.1/13.5/-	-	4.2/6.2/-	-	2.2/3.2/-	-
IMP [146] + NARE [32]	7.1/10.5/-	-	1.6/2.3/-	-	1.5/2.5/-	-
SOTA Single Stage Approaches:						
RAF (ResNet-50) [81]	-/8.2/10.6	-/11.7/18.1	-/1.3/1.7	-/2.4/3.8	-/0.8/1.1	-/1.0/1.7
RAF (HRNet-48) + freq [81]	-/7.8/10.0	-/11.4/17.6	-/1.6/2.0	-/2.8/4.8	-/0.8/1.4	-/1.4/2.3
Our Single Stage Approaches:						
TRF (ResNet-50)	6.0/8.9/11.0	7.1/13.2/19.5	0.5/1.4/2.1	1.1/2.5/4.2	0.6/0.9/1.3	0.7/1.2/2.0
TRF (HRNet-48)	6.1/9.1/11.2	7.3/13.5/19.9	0.8/1.7/2.5	1.4/3.0/4.6	0.8/1.3/1.9	1.2/1.6/2.3
TRF (ResNet-50) + NB	6.5/9.4/11.7	7.7/14.0/22.0	0.6/1.4/2.2	1.3/2.9/4.5	0.7/1.1/1.4	1.0/1.4/2.1
TRF (HRNet-48) + NB	6.4/9.4/11.6	7.6/14.0/21.0	1.0/1.8/2.6	1.6/3.3/5.0	0.9/1.4/1.9	1.4/1.8/2.4

between when it was used in isolation and its inclusion in this experiment as a performance 'booster'.

In comparison to the only other approaches that perform relationship detection in a single stage, TRFs appear to be a better representation. Our TRF trained networks almost consistently outperform the RAF trained networks across all modes. In some cases our smaller ResNet-50 models trained with TRFs are able to outperform the larger HRNet-48 models trained with RAFs, indicating that the TRF representation may be more 'learnable' by networks.

Our aim for these experiments is to demonstrate that the TRF representation is learnable and useable by a deep learning network, and we do not fully optimize performance of our networks by tweaking hyperparameters. While we expect overall performance can be fine tuned slightly, though notably, even with our TRF representation, and even when boosted with a statistical approach, performance of single stage relationship detectors still lags heavily behind their multistage counterparts. With regards to recall and mean recall, our approaches are still not at the level of modern scene graph generation approaches. This may be a result of convolutional networks not being able to contextualize relationships at the same level of message passing approaches, LSTM based approaches or energy-based

approaches [146, 158, 130]. Even in PredCl mode, our representation is unaware of the object classes for which its assigning TRFs which may be to its detriment, since we've shown how powerful of a bias these can be in Section 5.3.

Despite this, our TRF trained networks show a lot of promise on the zero shot recall mode. In this mode, networks and approaches are tested on their generalizability as they're tasked with predicting triplet combinations they have never seen. Our models actually compete with the state of the art in this mode, which is quite impressive given their low overall recall and mean recall performances. This may indicate that while our networks trained with TRFs are still not able to fully model all relationships in the VG150 dataset, the representations they do learn are quite robust and transferable. This is evidenced by the smaller drop in performance our models exhibit between their recall and their zero shot recall rates. Our TRF representation certainly shows its utility in this mode, as the networks trained with it perform better than the RAF representation networks by a larger margin. Examples of this positive behaviour are shown in Figures 5.3 and 5.4.

Some of the key benefits of fully convolutional approaches are their smaller network size, and faster inference and training time. As noted by [81], both our tested backbones have a quarter of the parameters of the other SOTA networks for scene graph generation, and in the case of ResNet50, a tenth. This leads to significantly faster inference which can make our networks useful for deploying in real-time relationship detection, though their recall performance may not be up to par just yet. Furthermore, in our case it's not necessary to retrain the network for every data mode (PredCl, SGCl and SGDet), whereas this is usually common practice in other scene graph generation approaches [130, 129, 32, 10] since they must adapt to the boxes their object detector outputs in each mode. In our case, we could decouple object detection completely and perform it separately, with a dedicated TRF producing approach for relationship detection.

Our TRF intermediate representation is also useful in and of itself. Utilizing the same TRF generated only once, a user can perform relationship detection on the fly between 2 objects of their choice. Effectively, for whichever 2 objects a user chooses, the processing summation of equation 5.7 can produce a relationship prediction. Objects can be chosen at any time even after the TRFs have been generated. TRFs can be easily adapted to other data types as well, for example, 3D data. Since CenterNet [166] can be used in 3D object detection too, we can also utilize it as a network for 3D relationship detection, with 3D topological configurations and 3D TRFs. Furthermore, TRFs can be easily adapted to utilize segmentation masks instead of bounding boxes. With a new scene graph detection dataset that provides segmentation masks for its objects rather than bounding boxes, the TRF regions presented in Section 5.4.1.2 can be easily modified to incorporate more meaningful topological configurations between objects. This can be especially beneficial to

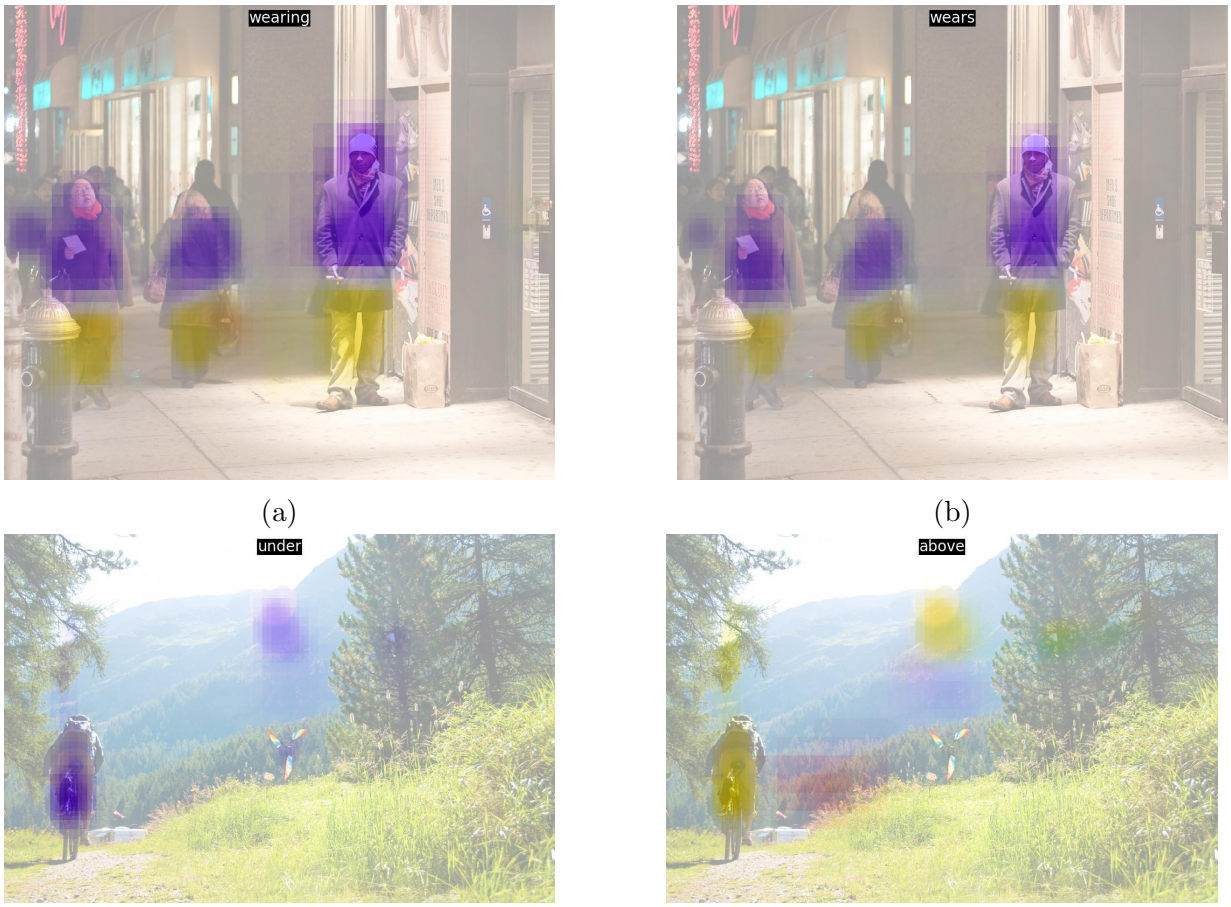
boosting relationship classes that rely on object contact (such as 'riding on', 'in', etc.) and further disambiguating relationship predicates that are overloaded, as the ones discussed in Chapter 4. The examples we show in Figures 5.3 and 5.4 may also point to TRF-based models learning a representation akin to the way humans conceptualize relationships [126]. Recall that [126] cited our brain's tendency towards conceptualizing notions spatially, which extends even to non-spatial concepts (e.g., hot, high temperature vs low, cold temperature). In the case of scene graph generation, TRFs appear to be encoding spatial concepts rather well, and the models predicting them demonstrated good generalization performance which possibly stems from their conceptual representations being robust.

Overall, we believe the TRF representation is a powerful representation worth exploring further for relationship detection. Our oracle experiments in Section 5.4.1 demonstrated that the representation is invertible and can be used to return to the scene graph representation. We demonstrated that it is capable of being learned by a fully convolutional deep learning approach in this section, though recall performance still requires improvement. We also showed the utility of this representation specifically in how networks can utilize it to build more robust representations, and in its inherent explainability and possible utility in other data modes.

5.5 Conclusions

In this chapter we split the relationship detection problem into two halves which we tackled separately. First, in Section 5.3, we explored relationship detection in a purely non-visual approach, Naive Bayes, and demonstrated how powerful biases can be learned and leveraged without ever looking at a single image pixel. We then explored the other end of the spectrum with Topological Relationship Fields, Section 5.4, a purely visual representation that grounds relationships in the image pixels they likely occur in.

Trained fully convolutional networks utilizing TRFs still did not reach the performance achieved by larger, multi-stage, predictors, even when combined with a Naive Bayes-based booster. However, TRF-based approaches showed significant promise on zero shot recall, indicating they may be learning more robust representations. Qualitatively, TRF-based approaches are able to justify their decisions using the specific pixels they used to make those decisions, and they also appear to be learning past the specific training data such as gaining an understanding of inverse relationships that was not in the ground truth.



(a) (b)

(c) (d)

Figure 5.4: Examples of specific behaviours of the predicted Topological Relationship Fields. Figures (a) and (b) show an example of the TRFs learning the same representation for two similar classes in VG150. Both *wears* and *wearing* are used in different predicate triplets in the VG150, however the learned TRF generators appear to be able to connect them to each other and generalize over both. Figures (c) and (d) show an example of inverse relationships *above* and *below*, and how the predicted TRFs also appear to be learning to connect them without explicitly being told to.

Chapter 6

Conclusions and Future Directions

In our approach to research, and specifically our approach to this thesis, our focus was on understanding why DL networks behave the way they do more-so than how to improve their performance. We answer questions related to the failure modes of networks, whether these stem from architecture design, data or otherwise. We also incorporate this type of thinking when designing our novel approaches in Chapter 5, prioritizing explainability over raw performance. We don't aim to exceed the current scene graph generation state of the art, but instead opted to build an approach that is a step towards a more complete understanding of images.

We opted to tackle the problem of visual relationship detection by breaking it down into its sub-parts and identifying the failure modes in each of them. First we tackled the biases and unpredictability found in instance segmentation and object detection alone, then the data bias and poor performance of relationship detection using scene graph generators. Finally, we presented an approach that leverages our findings to allow for a better representation of relationships, grounded in the parts of the image they occur.

6.1 The Role of Object Shape, Content and Context in Instance Segmentation

We quantified the effect that object shape, content and context play in instance segmentation networks of various architectures and under different training and architecture paradigms [44, 83, 134, 162, 141]. Our experiments allowed us to disentangle the effects of shape, content and context from each other in this task. We empirically demonstrated

that much like what research has shown is the case in networks operating on various other tasks across the field of computer vision [144, 6, 111, 116], networks trained and utilized for instance segmentation also suffer from their own task-specific pitfalls and biases. We reveal and empirically show the existence of these biases.

Instance segmentation networks appear to be memorizing a shape bias based on instance label data. While at first glance networks utilizing object shape without being explicitly trained to do so may seem desirable, in our experiments we show that this shape memorization actively works against these networks. DL networks' rigid memorization of shape, which we demonstrated gets fooled by simply adding a few pixels to the border of the object, is not robust enough to be of help to them. Shape could be a useful cue when detecting rigid objects, however the tendency for object hallucination from shape only is more likely to be a failure mode. The instance segmentation networks (and by extension object detection networks) that we experimented on are often used 'off the shelf' by practitioners. This even extends within computer vision research such as in the field of scene graph detection where these kinds of networks are used as-is as part of a more complex task. Special attention needs to be paid when utilizing these networks, as these biases are unpredictable (as seen in the isolation dilation experiments) and could likely work against the networks when given out of distribution data. For example, what happens in low light settings when a vision system for a self driving car mistakes 2 objects of similar shapes? Is it also possible that these shape biases can allow for malicious adversarial attacks?

An interesting question that we would like to explore further is why network bias around certain class groups seems to be behave similarly. For example, why is it that networks seem to be able to correctly predict the content class for land vehicles in the Content Swap experiments (Section 3.5.6)? It's very possible that the underlying feature extractor, which likely is the one imparting these biases, extracts similar features for all these classes hence they fail or succeed together. While this may make sense in the case of land vehicles, which share similar overall features, it starts to make less sense in the case of food classes (which are semantically similar, but visually different) yet seem to exhibit a similar behaviour. It would also be interesting to explore whether utilizing data augmentations similar to these experiments at train time can help networks form a more robust understanding of objects.

6.2 The Topology and Language of Relationships in the Visual Genome Dataset

We also revealed the role human labellers play in biasing relationship detection data-sets, and subsequently relationship detection networks. The Visual Genome dataset [64] is the most ubiquitous natural image dataset that includes relationship data and is used to train and evaluate a large majority scene graph generators presented in the literature [10]. Identifying these biases is the key first step to remedying them, whether these biases stem from how data is collected and labelled, where data is collected and labelled, or even how algorithms are being designed and used [92]. With these dataset biases now known in the case of the VG dataset, both training and evaluation processes with the dataset can be revisited. We presented a basic evaluation of what the result of weeding out less 'reliable' relationship classes in the Visual Genome dataset can achieve (through the 2 modified datasets), however this is only a first step in how to remedy the issues that exist in the VG dataset.

The lack of representation of both sides of symmetric relationships in the VG dataset means that inverse relationships such as 'above' and 'under' are not encoded in the dataset, and could potentially be used as a measure of generalizability of models or to improve performance. We also discussed the topological and directional configurations exhibited by relationships in the Visual Genome Dataset and showed how overloaded predicate classes (such as 'on') exhibit topological vagueness which may confuse trained models. Even scene graph generators predicting the relationship 'on' may not grasp the full meaning of this relationship unless we somehow encode the ability to disambiguate the multiple meanings of this predicate, along with other predicate classes.

An interesting avenue for future exploration is a compositional analysis of the objects and relationships in the VG dataset. It would be quite interesting to see how well hierarchical relationships hold and whether more conceptual understandings can be garnered by utilizing a combination of language, topology and a hierarchy of parts.

6.3 Naive Scene Graphs and Topological Relationship Fields for Scene Graphs

Dataset bias is not the only cause for the poor performance scene graph generators appear to exhibit. We designed and evaluated a simple, statistical (Naive Bayes based) relationship

classifier, this classifier operates on the object bounding box classes and locations only without paying any notice to the image pixels or features extracted from those pixels. This simple classifier being comparable in performance to relatively recent works while being an order of a *million* times less complex than its deep learning based counterparts (compared to the number of weights and biases that are learned by scene graph generation networks) is quite alarming. Effectively, we were able to build a competitive scene graph generator by utilizing an object detector and basic statistics, in some cases performing either better or more consistently (in terms or recall and zero shot recall) than deep learning approaches. This could be an indicator that 'visual relationship detection' may not be relying on what it sees visually in images and instead building its own statistical correlations based on subject and object classes. Unfortunately, we don't really know if this is the case or not due to the 'black box'-y nature of the relationship detectors which is part of their drawbacks.

This motivated us to build towards an alternative scene graph generator that is grounded and localized in the image regions it describes. Our proposed Toplogical Relationship Fields build upon fully convolutional scene graph generation approaches and despite their lack-lustre mean recall and recall performance, they appear to be capable of learning a robust and generalizable representation (as demonstrated by their zero shot recall performance). Furthermore, they offer several benefits including explainability, adaptability to other data types as well as less computationally intensive training and inference.

An future avenue we believe is of value to explore is the combination of our TRFs with segmentation masks. While no scene graph datasets currently exist that would allow for directly incorporating object segmentation into scene graph generation, it's possible that exploring a weakly supervised approach such as ones built on top of [134] may be fruitful. This is in contrast to the approach taken by [59], which poses segmentation as a zero-shot transfer learning problem via language similarity. We are curious to explore how a weak supervision signal for both instance segmentation and pixel level masks for relationships would fare. Furthermore other future avenues could be learning object detection through triplets where the subject and predicate are known. For example, if a TRF-based predictor is capable of predicting an object (e.g., 'person') and a relationship (e.g., '*sitting on*') robustly enough, a user could define an unknown object for object detection (e.g., a scooter) by supplying images with known instances of 'person *sitting on* scooter' and teach networks to detect a 'scooter' in a weakly supervised manner.

6.4 Limitations

Throughout our research, we contended with a few limitations that we did our best to circumvent. However, despite our best effort, these still limited the scope of our research and steered us away from questions we would have liked to answer within this thesis.

Dataset limitations are not uncommon in DL research, and this was no different in our case. In a perfect world, we would have liked to have a clean and unbiased, pixel annotated visual relationship dataset that allowed us to immediately train without needing to resort to topological relationships as an intermediate step. Though, the lack of the dataset for this sort of task forced us to opt for engineering more creative solutions, and embrace the mindset of how best to utilize the data available to us in a logical and relevant way to make progress towards better relationship detection. We also were unable to continue building on TRFs to utilize them in a segmentation coupled scene graph approach as we discussed in Section 6.3.

Vision transformer-based architectures have rapidly gained momentum in computer vision and become one of the most popular topics in DL based computer vision research today [58]. While we were aware of their existence and followed their literature we were wary of fully committing to utilizing transformers for multiple reasons. Up until recently, the field of vision transformers was not mature and developed enough for us to utilize their breakthroughs, nor was it accessible enough to us to reasonably believe we could contribute to it. Between their data hungriness, and longer training schedules they seemed to come with drawbacks we were intending to avoid. They were, relatively successfully, also very recently were applied to relationship detection [74] and we do believe that a common text-vision representation may come in handy in relationship detection. This extends to multi-modal neurons such as the CLIP neurons [33] which were integral to DALLE 2, and we would have liked to explore some of these alternate representations as well. We also hoped to explore further research directions with utilizing generative models [69, 97] and auto-encoders [7, 48] and the representations that they seem to excel at learning in our work as well.

6.5 Concluding Remarks

Relationship detection is only a sliver of what human cognition encompasses. We are very curious as to how other cognitive abilities humans possess such as reasoning about hierarchies and compositions, reasoning about physical systems, gaining prior knowledge

through language among other abilities can progress the field of computer vision. As we reviewed in Chapter 2, there has been a rising number of work taking influence from one or more cognitive functions human possess. We believe that as long as we are still trying to solve human-related problems, human intelligence is a source of inspiration that we cannot ignore. As the field of Artificial Intelligence and Deep Learning grows more and more, we cannot help but wonder whether AI will eventually converge to human-like intelligence, or whether it will form its own optimal intelligence that is vastly different, and maybe superior?!

References

[1] Antreas Antoniou, Amos Storkey, and Harrison Edwards. Data augmentation generative adversarial networks. *arXiv preprint arXiv:1711.04340*, 2017.

[2] Iro Armeni, Zhi-Yang He, JunYoung Gwak, Amir R Zamir, Martin Fischer, Jitendra Malik, and Silvio Savarese. 3d scene graph: A structure for unified semantics, 3d space, and camera. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5664–5673, 2019.

[3] Aharon Azulay and Yair Weiss. Why do deep convolutional networks generalize so poorly to small image transformations? *Journal of Machine Learning Research*, 20(184):1–25, 2019.

[4] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 39(12):2481–2495, 2017.

[5] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. *Advances in Neural Information Processing Systems*, 26, 2013.

[6] Wieland Brendel and Matthias Bethge. Approximating CNNs with bag-of-local-features models works surprisingly well on imagenet. In *International Conference on Learning Representations*, 2019.

[7] Christopher P Burgess, Loic Matthey, Nicholas Watters, Rishabh Kabra, Irina Higgins, Matt Botvinick, and Alexander Lerchner. Monet: Unsupervised scene decomposition and representation. *arXiv preprint arXiv:1901.11390*, 2019.

[8] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7291–7299, 2017.

[9] Patrick Cavanagh. Visual cognition. *Vision Research*, 51(13):1538 – 1551, 2011. Vision Research 50th Anniversary Issue: Part 2.

[10] Xiaojun Chang, Pengzhen Ren, Pengfei Xu, Zhihui Li, Xiaojiang Chen, and Alexander G Hauptmann. A comprehensive survey of scene graphs: Generation and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[11] Linda L Chao and Alex Martin. Representation of manipulable man-made objects in the dorsal stream. *Neuroimage*, 12(4):478–484, 2000.

[12] Anjan Chatterjee. Disembodying cognition. *Language and Cognition*, 2(1):79–116, 2010.

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

[14] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

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

[16] Tianshui Chen, Weihao Yu, Riquan Chen, and Liang Lin. Knowledge-embedded routing network for scene graph generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6163–6171, 2019.

[17] Xinlei Chen, Ross Girshick, Kaiming He, and Piotr Dollár. Tensormask: A foundation for dense object segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2061–2069, 2019.

[18] Bowen Cheng, Omkar Parkhi, and Alexander Kirillov. Pointly-supervised instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2617–2626, 2022.

[19] Eliseo Clementini, Paolino Di Felice, and Peter van Oosterom. A small set of formal topological relationships suitable for end-user interaction. In *International Symposium on Spatial Databases*, pages 277–295. Springer, 1993.

[20] Weilin Cong, William Wang, and Wang-Chien Lee. Scene graph generation via conditional random fields. *arXiv preprint arXiv:1811.08075*, 2018.

[21] 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/CVF Conference on Computer Vision and Pattern Recognition*, pages 3213–3223, 2016.

[22] Grégoire Delétang, Anian Ruoss, Jordi Grau-Moya, Tim Genewein, Li Kevin Wenliang, Elliot Catt, Marcus Hutter, Shane Legg, and Pedro A Ortega. Neural networks and the chomsky hierarchy. *arXiv preprint arXiv:2207.02098*, 2022.

[23] Weijian Deng, Stephen Gould, and Liang Zheng. What does rotation prediction tell us about classifier accuracy under varying testing environments? In *International Conference on Machine Learning*, pages 2579–2589. PMLR, 2021.

[24] Markos Diomataris, Nikolaos Gkanatsios, Vassilis Pitsikalis, and Petros Maragos. Grounding consistency: Distilling spatial common sense for precise visual relationship detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15911–15920, 2021.

[25] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020.

[26] Max J Egenhofer and Robert D Franzosa. Point-set topological spatial relations. *International Journal of Geographical Information System*, 5(2):161–174, 1991.

[27] Markus Eisenbach, Ronny Stricker, Daniel Seichter, Karl Amende, Klaus Debes, Maximilian Sesselmann, Dirk Ebersbach, Ulrike Stoeckert, and Horst-Michael Gross. How to get pavement distress detection ready for deep learning? a systematic approach. In *International Joint Conference on Neural Networks*, pages 2039–2047. IEEE, 2017.

[28] 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.

[29] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202, 1980.

[30] Alberto Garcia-Garcia, Sergio Orts-Escolano, Sergiu Oprea, Victor Villena-Martinez, Pablo Martinez-Gonzalez, and Jose Garcia-Rodriguez. A survey on deep learning techniques for image and video semantic segmentation. *Applied Soft Computing*, 70:41–65, 2018.

[31] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A. Wichmann, and Wieland Brendel. Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In *International Conference on Learning Representations*, 2019.

[32] Arushi Goel, Basura Fernando, Frank Keller, and Hakan Bilen. Not all relations are equal: Mining informative labels for scene graph generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15596–15606, 2022.

[33] Gabriel Goh, Nick Cammarata, Chelsea Voss, Shan Carter, Michael Petrov, Ludwig Schubert, Alec Radford, and Chris Olah. Multimodal neurons in artificial neural networks. *Distill*, 6(3):e30, 2021.

[34] Till Grüne-Yanoff. Bounded rationality. *Philosophy Compass*, 2(3):534–563, 2007.

[35] Jiuxiang Gu, Shafiq Joty, Jianfei Cai, Handong Zhao, Xu Yang, and Gang Wang. Unpaired image captioning via scene graph alignments. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10323–10332, 2019.

[36] Jiuxiang Gu, Handong Zhao, Zhe Lin, Sheng Li, Jianfei Cai, and Mingyang Ling. Scene graph generation with external knowledge and image reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2019.

[37] Devin Guillory, Vaishaal Shankar, Sayna Ebrahimi, Trevor Darrell, and Ludwig Schmidt. Predicting with confidence on unseen distributions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1134–1144, 2021.

[38] Yanming Guo, Yu Liu, Theodoros Georgiou, and Michael S Lew. A review of semantic segmentation using deep neural networks. *International Journal of Multimedia Information Retrieval*, 7(2):87–93, 2018.

[39] Abhinav Gupta, Aniruddha Kembhavi, and Larry S Davis. Observing human-object interactions: Using spatial and functional compatibility for recognition. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 31(10):1775–1789, 2009.

[40] Abhishek Gupta, Alagan Anpalagan, Ling Guan, and Ahmed Shaharyar Khwaja. Deep learning for object detection and scene perception in self-driving cars: Survey, challenges, and open issues. *Array*, 10:100057, 2021.

[41] 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.

[42] Beate Hampe and Joseph E Grady. *From Perception to Meaning: Image Schemas in Cognitive Linguistics*, volume 29. Walter de Gruyter, 2005.

[43] Tengda Han, Weidi Xie, and Andrew Zisserman. Temporal alignment networks for long-term video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2906–2916, 2022.

[44] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2961–2969, 2017.

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

[46] Katherine Hermann and Andrew Lampinen. What shapes feature representations? exploring datasets, architectures, and training. *Advances in Neural Information Processing Systems*, 33:9995–10006, 2020.

[47] Cecilia Heyes. New thinking: the evolution of human cognition, 2012.

[48] I Higgins, N Sonnerat, L Matthey, A Pal, CP Burgess, M Bošnjak, M Shanahan, M Botvinick, D Hassabis, and A Lerchner. Scan: Learning hierarchical compositional visual concepts. In *International Conference on Learning Representations*, 2018.

[49] Ronghang Hu, Piotr Dollár, Kaiming He, Trevor Darrell, and Ross Girshick. Learning to segment every thing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4233–4241, 2018.

[50] Jonathan Hui. Image segmentation with mask r-cnn, Apr 2018.

[51] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. *Advances in Neural Information Processing Systems*, 32, 2019.

[52] Md Amirul Islam, Matthew Kowal, Patrick Esser, Sen Jia, Björn Ommer, Konstantinos G. Derpanis, and Neil Bruce. Shape or texture: Understanding discriminative features in {cnn}s. In *International Conference on Learning Representations*, 2021.

[53] Lei Ji, Yujing Wang, Botian Shi, Dawei Zhang, Zhongyuan Wang, and Jun Yan. Microsoft concept graph: Mining semantic concepts for short text understanding. *Data Intelligence*, 1(3):238–270, 2019.

[54] Longlong Jing and Yingli Tian. Self-supervised visual feature learning with deep neural networks: A survey. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 43(11):4037–4058, 2020.

[55] Justin Johnson, Agrim Gupta, and Li Fei-Fei. Image generation from scene graphs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1219–1228, 2018.

[56] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2901–2910, 2017.

[57] Justin Johnson, Ranjay Krishna, Michael Stark, Li-Jia Li, David Shamma, Michael Bernstein, and Li Fei-Fei. Image retrieval using scene graphs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3668–3678, 2015.

[58] Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. Transformers in vision: A survey. *ACM computing surveys (CSUR)*, 54(10s):1–41, 2022.

[59] Siddhesh Khandelwal, Mohammed Suhail, and Leonid Sigal. Segmentation-grounded scene graph generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15879–15889, 2021.

[60] Markus Kiefer and Friedemann Pulvermüller. Conceptual representations in mind and brain: Theoretical developments, current evidence and future directions. *cortex*, 48(7):805–825, 2012.

[61] Ue-Hwan Kim, Jin-Man Park, Taek-Jin Song, and Jong-Hwan Kim. 3-d scene graph: A sparse and semantic representation of physical environments for intelligent agents. *IEEE Transactions on Cybernetics*, 50(12):4921–4933, 2019.

[62] Katherine D Kinzler and Elizabeth S Spelke. Core systems in human cognition. *Progress in Brain Research*, 164:257–264, 2007.

[63] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, and Piotr Dollar. Panoptic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2019.

[64] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 123(1):32–73, 2017.

[65] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[66] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Alexander Kolesnikov, et al. The open images dataset v4. *International Journal of Computer Vision*, 128(7):1956–1981, 2020.

[67] Brenden M Lake and Steven T Piantadosi. People infer recursive visual concepts from just a few examples. *Computational Brain & Behavior*, 3:54–65, 2020.

[68] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[69] Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. Building machines that learn and think like people. *Behavioral and Brain Sciences*, 40, 2017.

[70] Barbara Landau and Ray Jackendoff. Whence and whither in spatial language and spatial cognition? *Behavioral and Brain Sciences*, 16(2):255–265, 1993.

[71] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Colorization as a proxy task for visual understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6874–6883, 2017.

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

[73] Lin Li, Long Chen, Yifeng Huang, Zhimeng Zhang, Songyang Zhang, and Jun Xiao. The devil is in the labels: Noisy label correction for robust scene graph generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18869–18878, 2022.

[74] Rongjie Li, Songyang Zhang, and Xuming He. Sgtr: End-to-end scene graph generation with transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19486–19496, 2022.

[75] 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/CVF Conference on Computer Vision and Pattern Recognition*, pages 2117–2125, 2017.

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

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

[78] Xin Lin, Changxing Ding, Jinquan Zeng, and Dacheng Tao. Gps-net: Graph property sensing network for scene graph generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3746–3753, 2020.

[79] Xin Lin, Changxing Ding, Yibing Zhan, Zijian Li, and Dacheng Tao. Hl-net: Heterophily learning network for scene graph generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19476–19485, 2022.

[80] An-An Liu, Hongshuo Tian, Ning Xu, Weizhi Nie, Yongdong Zhang, and Mohan Kankanhalli. Toward region-aware attention learning for scene graph generation. *IEEE Transactions on Neural Networks and Learning Systems*, 33(12):7655–7666, 2021.

[81] Hengyue Liu, Ning Yan, Masood Mortazavi, and Bir Bhanu. Fully convolutional scene graph generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11546–11556, 2021.

[82] Li Liu, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen. Deep learning for generic object detection: A survey. *International Journal of Computer Vision*, 128(2):261–318, 2020.

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

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

[85] David Lopez-Paz, Robert Nishihara, Soumith Chintala, Bernhard Scholkopf, and Léon Bottou. Discovering causal signals in images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6979–6987, 2017.

[86] Cewu Lu, Ranjay Krishna, Michael Bernstein, and Li Fei-Fei. Visual relationship detection with language priors. In *Proceedings of the European Conference on Computer Vision*, 2016.

[87] Hiroya Maeda, Yoshihide Sekimoto, Toshikazu Seto, Takehiro Kashiyama, and Hiroshi Omata. Road damage detection using deep neural networks with images captured through a smartphone. *arXiv preprint arXiv:1801.09454*, 2018.

[88] Jean M Mandler and Cristóbal Pagán Cánovas. On defining image schemas. *Language and Cognition*, 6(4):510–532, 2014.

[89] Gary Marcus. Deep learning: A critical appraisal. *arXiv preprint arXiv:1801.00631*, 2018.

[90] Gary Marcus. The next decade in ai: four steps towards robust artificial intelligence. *arXiv preprint arXiv:2002.06177*, 2020.

[91] Gary Marcus, Ernest Davis, and Scott Aaronson. A very preliminary analysis of dall-e 2. *arXiv preprint arXiv:2204.13807*, 2022.

[92] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning. *ACM Computing Surveys (CSUR)*, 54(6):1–35, 2021.

[93] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[94] John Miller, Karl Krauth, Benjamin Recht, and Ludwig Schmidt. The effect of natural distribution shift on question answering models. In *International Conference on Machine Learning*, pages 6905–6916. PMLR, 2020.

[95] Shervin Minaee, Yuri Boykov, Fatih Porikli, Antonio Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. Image segmentation using deep learning: A survey. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 44(7):3523–3542, 2021.

[96] Kazuhiro Omura, Mineichi Kudo, Tomomi Endo, and Tetsuya Murai. Weighted naïve bayes classifier on categorical features. In *2012 12th International Conference on Intelligent Systems Design and Applications (ISDA)*, pages 865–870. IEEE, 2012.

[97] Dim P Papadopoulos, Youssef Tamaazousti, Ferda Ofli, Ingmar Weber, and Antonio Torralba. How to make a pizza: Learning a compositional layer-based gan model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8002–8011, 2019.

[98] George Papandreou, Liang-Chieh Chen, Kevin P. Murphy, and Alan L. Yuille. Weakly- and semi-supervised learning of a deep convolutional network for semantic image segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, December 2015.

[99] Judea Pearl. The seven tools of causal inference, with reflections on machine learning. *Communications of the ACM*, 62(3):54–60, 2019.

[100] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543, 2014.

[101] Moshiko Raboh, Roei Herzig, Jonathan Berant, Gal Chechik, and Amir Globerson. Differentiable scene graphs. In *The IEEE Winter Conference on Applications of Computer Vision*, pages 1488–1497, 2020.

[102] A. Raghunathan, S. M. Xie, F. Yang, J. C. Duchi, and P. Liang. Understanding and mitigating the tradeoff between robustness and accuracy. In *International Conference on Machine Learning*, 2020.

[103] Jathushan Rajasegaran, Georgios Pavlakos, Angjoo Kanazawa, and Jitendra Malik. Tracking people by predicting 3d appearance, location and pose. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2740–2749, 2022.

[104] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

[105] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR, 2021.

[106] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? In *International Conference on Machine Learning*, pages 5389–5400. PMLR, 2019.

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

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

[109] Blake A Richards, Timothy P Lillicrap, Philippe Beaudoin, Yoshua Bengio, Rafal Bogacz, Amelia Christensen, Claudia Clopath, Rui Ponte Costa, Archy de Berker, Surya Ganguli, et al. A deep learning framework for neuroscience. *Nature neuroscience*, 22(11):1761–1770, 2019.

[110] 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.

[111] Amir Rosenfeld, Richard Zemel, and John K Tsotsos. The elephant in the room. *arXiv preprint arXiv:1808.03305*, 2018.

[112] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

[113] Xindi Shang, Donglin Di, Junbin Xiao, Yu Cao, Xun Yang, and Tat-Seng Chua. Annotating objects and relations in user-generated videos. In *Proceedings of the 2019 on International Conference on Multimedia Retrieval*, pages 279–287, 2019.

[114] Xindi Shang, Tongwei Ren, Jingfan Guo, Hanwang Zhang, and Tat-Seng Chua. Video visual relation detection. In *Proceedings of the 25th ACM International Conference on Multimedia*, pages 1300–1308, 2017.

[115] Sahand Sharifzadeh, Sina Moayed Baharlou, and Volker Tresp. Classification by attention: Scene graph classification with prior knowledge. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 5025–5033, 2021.

[116] Rakshith Shetty, Bernt Schiele, and Mario Fritz. Not using the car to see the sidewalk–quantifying and controlling the effects of context in classification and segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8218–8226, 2019.

[117] Jiaxin Shi, Hanwang Zhang, and Juanzi Li. Explainable and explicit visual reasoning over scene graphs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2019.

[118] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):60, 2019.

[119] Herbert A Simon. Bounded rationality. In *Utility and Probability*, pages 15–18. Springer, 1990.

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

[121] Robyn Speer, Joshua Chin, and Catherine Havasi. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[122] Elizabeth S Spelke. Principles of object perception. *Cognitive Science*, 14(1):29–56, 1990.

[123] Elizabeth S Spelke. What makes us smart? core knowledge and natural language. *Language in mind: Advances in the Study of Language and Thought*, pages 277–311, 2003.

[124] Elizabeth S Spelke and Katherine D Kinzler. Core knowledge. *Developmental Science*, 10(1):89–96, 2007.

[125] Mohammed Suhail, Abhay Mittal, Behjat Siddiquie, Chris Broaddus, Jayan Eledath, Gerard Medioni, and Leonid Sigal. Energy-based learning for scene graph generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13936–13945, 2021.

[126] Christopher Summerfield, Fabrice Luyckx, and Hannah Sheahan. Structure learning and the posterior parietal cortex. *Progress in Neurobiology*, page 101717, 2019.

[127] Kai Sheng Tai, Richard Socher, and Christopher D Manning. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566, 2015.

[128] Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10781–10790, 2020.

[129] Kaihua Tang, Yulei Niu, Jianqiang Huang, Jiaxin Shi, and Hanwang Zhang. Unbiased scene graph generation from biased training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3716–3725, 2020.

[130] Kaihua Tang, Hanwang Zhang, Baoyuan Wu, Wenhan Luo, and Wei Liu. Learning to compose dynamic tree structures for visual contexts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6619–6628, 2019.

[131] Rohan Taori, Achal Dave, Vaishaal Shankar, Nicholas Carlini, Benjamin Recht, and Ludwig Schmidt. Measuring robustness to natural distribution shifts in image classification. *Advances in Neural Information Processing Systems*, 33:18583–18599, 2020.

[132] Bart Thomee, David A Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. Yfcc100m: The new data in multimedia research. *Communications of the ACM*, 59(2):64–73, 2016.

[133] Zhi Tian, Hao Chen, Xinlong Wang, Yuliang Liu, and Chunhua Shen. AdelaiDet: A toolbox for instance-level recognition tasks. https://git.io/adelaidet, 2019.

[134] Zhi Tian, Chunhua Shen, and Hao Chen. Conditional convolutions for instance segmentation. In *Proceedings of the European Conference on Computer Vision*, 2020.

[135] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. FCOS: A simple and strong anchor-free object detector. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 2021.

[136] Momchil S. Tomov, Eric Schulz, and Samuel J. Gershman. Multi-task reinforcement learning in humans. *bioRxiv*, 2019.

[137] Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T Barron, and Pratul P Srinivasan. Ref-nerf: Structured view-dependent appearance for neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5481–5490. IEEE, 2022.

[138] Olga Vysotska. *Visual Place Recognition in Changing Environments*. PhD thesis, University of Bonn, 2019.

[139] Guangrun Wang, Keze Wang, and Liang Lin. Adaptively connected neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[140] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, et al. Deep high-resolution representation learning for visual recognition. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 43(10):3349–3364, 2020.

[141] Xinlong Wang, Rufeng Zhang, Tao Kong, Lei Li, and Chunhua Shen. SOLOv2: Dynamic and fast instance segmentation. In *Advances in Neural Information Processing Systems*, 2020.

[142] Chenyun Wu, Zhe Lin, Scott Cohen, Trung Bui, and Subhransu Maji. Phrasecut: Language-based image segmentation in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10216–10225, 2020.

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

[144] Kai Yuanqing Xiao, Logan Engstrom, Andrew Ilyas, and Aleksander Madry. Noise or signal: The role of image backgrounds in object recognition. In *International Conference on Learning Representations*, 2021.

[145] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1492–1500, 2017.

[146] Danfei Xu, Yuke Zhu, Christopher B. Choy, and Li Fei-Fei. Scene graph generation by iterative message passing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, July 2017.

[147] Chhavi Yadav and Léon Bottou. Cold case: The lost mnist digits. *Advances in Neural Information Processing Systems*, 32, 2019.

[148] Kaiyu Yang, Olga Russakovsky, and Jia Deng. Spatialsense: An adversarially crowdsourced benchmark for spatial relation recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2051–2060, 2019.

[149] Xu Yang, Kaihua Tang, Hanwang Zhang, and Jianfei Cai. Auto-encoding scene graphs for image captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10685–10694, 2019.

[150] Ying Yang and Geoffrey I Webb. A comparative study of discretization methods for naive-bayes classifiers. In *Proceedings of the 2002 Pacific Rim Knowledge Acquisition Workshop*, pages 159–173, 2002.

[151] Linwei Ye, Zhi Liu, and Yang Wang. Learning semantic segmentation with diverse supervision. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1461–1469. IEEE, 2018.

[152] Jing Yu, Yuan Chai, Yujing Wang, Yue Hu, and Qi Wu. Cogtree: Cognition tree loss for unbiased scene graph generation. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pages 1274–1280, 2021.

[153] Qihang Yu, Huiyu Wang, Dahun Kim, Siyuan Qiao, Maxwell Collins, Yukun Zhu, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Cmt-deeplab: Clustering mask transformers for panoptic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2560–2570, 2022.

[154] Yuhui Yuan, Xilin Chen, and Jingdong Wang. Object-contextual representations for semantic segmentation. In *Proceedings of the European Conference on Computer Vision*, pages 173–190. Springer, 2020.

[155] Shizhe Zang, Ming Ding, David Smith, Paul Tyler, Thierry Rakotoarivelo, and Mohamed Ali Kaafar. The impact of adverse weather conditions on autonomous vehicles: how rain, snow, fog, and hail affect the performance of a self-driving car. *IEEE Vehicular Technology Magazine*, 14(2):103–111, 2019.

[156] Alireza Zareian, Svebor Karaman, and Shih-Fu Chang. Bridging knowledge graphs to generate scene graphs. In *Proceedings of the European Conference on Computer Vision*, pages 606–623. Springer, 2020.

[157] Rowan Zellers, Yonatan Bisk, Ali Farhadi, and Yejin Choi. From recognition to cognition: Visual commonsense reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6720–6731, 2019.

[158] Rowan Zellers, Mark Yatskar, Sam Thomson, and Yejin Choi. Neural motifs: Scene graph parsing with global context. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5831–5840, 2018.

[159] Chi Zhang, Guosheng Lin, Fayao Liu, Rui Yao, and Chunhua Shen. Canet: Class-agnostic segmentation networks with iterative refinement and attentive few-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5217–5226, 2019.

[160] Hang Zhang, Kristin Dana, Jianping Shi, Zhongyue Zhang, Xiaogang Wang, Ambrish Tyagi, and Amit Agrawal. Context encoding for semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7151–7160, 2018.

[161] Hanwang Zhang, Zawlin Kyaw, Shih-Fu Chang, and Tat-Seng Chua. Visual translation embedding network for visual relation detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5532–5540, 2017.

[162] Rufeng Zhang, Zhi Tian, Chunhua Shen, Mingyu You, and Youliang Yan. Mask encoding for single shot instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[163] Bo Zhao, Jiashi Feng, Xiao Wu, and Shuicheng Yan. A survey on deep learning-based fine-grained object classification and semantic segmentation. *International Journal of Automation and Computing*, 14(2):119–135, 2017.

[164] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, July 2017.

[165] Tianfei Zhou, Wenguan Wang, Ender Konukoglu, and Luc Van Gool. Rethinking semantic segmentation: A prototype view. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2582–2593, 2022.

[166] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arXiv preprint arXiv:1904.07850*, 2019.

[167] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2223–2232, 2017.

[168] Zhuotun Zhu, Lingxi Xie, and Alan Yuille. Object recognition with and without objects. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 3609–3615, 2017.

APPENDICES

Appendix A

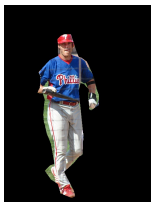
Appendix A : The Role of Shape, Content, and Context in Modern Instance Segmentation

A.1 Full Result Tables

We present pictorial examples of all experiments in Figures A.1, A.2, A.3, A.4. We present the expanded results for our experiments in the following Tables:

- We present the **maskAPs for all experiment modes** (except for the ContentSwap experiments) in Table A.1.
- We present a **per-class maskAP** breakdown of all architectures on the *Baseline* and *IsolatedInstance-Mean-0* modes in Table A.2.
- We present a **per mask size breakdown of the maskAP** results in Tables A.3 and A.4. Mask size ranges are taken from the COCO dataset API.
- We present the **ContentSwap maskAPs** on a per-class basis for every network in Tables A.5, A.6, A.7, and A.8.

(a)
Baseline

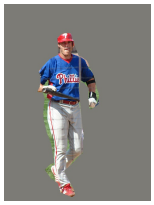


(b)
IsolatedInstance-
Black-0

(c)
IsolatedInstance-
Black-6

(d)
IsolatedInstance-
Black-12

(e)
IsolatedInstance-
Black-20



(f)
IsolatedInstance-
Mean-0

(g)
IsolatedInstance-
Mean-6

(h)
IsolatedInstance-
Mean-12

(i)
IsolatedInstance-
Mean-20

(j)
IsolatedInstance-
Noise-0

(k)
IsolatedInstance-
Noise-6

(l)
IsolatedInstance-
Noise-12

(m)
IsolatedInstance-
Noise-20

Figure A.1: Our **Baseline** and **IsolatedInstance** experiment modes shown on one sample image. The experiment mode name is listed first, then the values after the '-' represent mode details. Black/Mean/Noise are the color of the pixels used to replace any removed pixels. Numbers represent how many maximum pixels are added during the dilation operations (when applicable).

(a) BoundingBox-Black

(b) BoundingBox-Mean

(c) BoundingBox-Noise

(d) AllInstances-Black

(e) AllInstances-Mean

(f) AllInstances-Noise

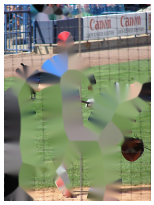
(g) GlobalInstance-Inpaint

(h) Silhouette-Black-Mean

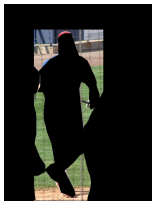
(i) Silhouette-Mean-Black

(j) Silhouette-Noise-Noise

Figure A.2: Our **BoundingBox**, **AllInstances**, **GlobalInstance** and **Silhouette** experiment modes shown on one sample image. The experiment mode name is listed first, then the values after the '-' represent mode details. Black/Mean/Noise/Inpaint are the color of the pixels used to replace any removed pixels.



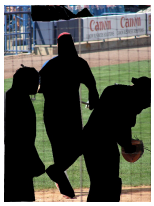
(a) GlobalHallucination-Inpaint-0



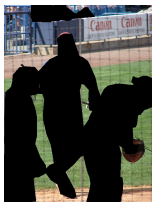
(b) LocalHallucination-Black

(c) LocalHallucination-Mean

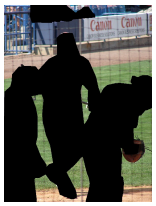
(d) LocalHallucination-Noise



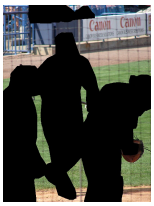
(e) GlobalHallucination-Black-0



(f) GlobalHallucination-Black-2



(g) GlobalHallucination-Black-4



(h) GlobalHallucination-Black-6

(i) GlobalHallucination-Mean-0

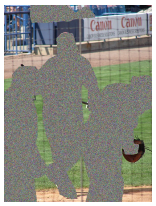
(j) GlobalHallucination-Mean-2

(k) GlobalHallucination-Mean-4

(l) GlobalHallucination-Mean-6

(m) GlobalHallucination-Noise-0

(n) GlobalHallucination-Noise-2



(o) GlobalHallucination-Noise-4

(p) GlobalHallucination-Noise-6

Figure A.3: Our **Hallucination** experiment modes shown on one sample image. The experiment mode name is listed first, then the values after the '-' represent mode details. Black/Mean/Noise/Inpaint are the color of the pixels used to replace any removed pixels. Numbers represent how many maximum pixels are added during the dilation operations (when applicable).

(a) BlurredIsolation-Black

(b) BlurredIsolation-Mean

(c) BlurredIsolation-Noise

(d) Blurred-Local

(e) BlurredGlobal-Sharp

(f) BlurredGlobal-Soft

(g) BlurredBackground-Soft

(h) BackgroundSwap-Sharp

(i) BackgroundSwap-Soft

(j) ContentSwapIsolation-Aspect

(k) ContentSwapIsolation-Morph

(l) ContentSwapOriginal-Aspect

(m) ContentSwapOriginal-Morph

Figure A.4: Our **BlurredIsolation**, **BlurredLocal**, **BlurredGlobal**, **BackgroundSwap** and **ContentSwap** experiment modes shown on one sample image. The experiment mode name is listed first, then the values after the '-' represent mode details. Black/Mean/Noise are the color of the pixels used to replace any removed pixels. Other adjectives describe experiment specific modes (e.g., sharp or soft for edge blurring).

Table A.1: Full maskAP results for all eight architectures. The experiment mode name is listed first, with values after the '-' representing mode details. Black/Mean/Noise are the color of the pixels used to replace any removed pixels. Numbers represent how many maximum pixels are added during the dilation operations. Other adjectives describe experiment specific modes.

		ROI Based					Non-ROI based		
Group	**Experiment Mode**	**MRCNN50-3x**	**MRCNN50-Jitter**	**MRCNN50-1x**	**MRCNN101-3x**	**Swin-T**	**CondInst**	**MEInst**	**SOLO**
	Baseline	37.2%	42.6%	35.2%	38.6%	41.8%	39.4%	34.5%	37.6%
A	IsolatedInstance-Black-0	42.8%	47.0%	43.3%	42.7%	44.7%	46.8%	43.1%	44.7%
A	IsolatedInstance-Black-6	29.1%	30.8%	28.3%	28.0%	28.3%	28.2%	27.3%	27.6%
A	IsolatedInstance-Black-12	29.8%	34.9%	28.5%	29.9%	31.7%	29.2%	27.4%	28.6%
A	IsolatedInstance-Black-20	32.8%	38.4%	31.0%	33.3%	35.1%	32.4%	30.0%	31.8%
A	IsolatedInstance-Mean-0	44.0%	45.2%	44.7%	45.9%	46.1%	49.3%	47.3%	51.7%
A	IsolatedInstance-Mean-6	32.0%	30.7%	30.2%	31.4%	32.4%	29.8%	29.1%	31.2%
A	IsolatedInstance-Mean-12	31.7%	33.3%	29.8%	32.5%	34.8%	30.7%	27.9%	30.1%
A	IsolatedInstance-Mean-20	33.7%	37.4%	31.8%	34.7%	37.0%	33.5%	29.9%	32.6%
A	IsolatedInstance-Noise-0	42.9%	48.3%	41.8%	43.7%	45.6%	48.4%	43.7%	45.3%
A	IsolatedInstance-Noise-6	27.3%	28.8%	26.6%	27.6%	29.3%	26.7%	26.9%	27.3%
A	IsolatedInstance-Noise-12	26.5%	30.5%	25.1%	26.9%	30.4%	26.3%	25.1%	26.1%
A	IsolatedInstance-Noise-20	29.8%	35.6%	28.3%	30.2%	34.1%	30.4%	27.5%	29.6%
B	BoundingBox-Black	32.0%	36.0%	31.3%	32.8%	35.7%	32.4%	29.5%	31.8%
B	BoundingBox-Mean	33.5%	36.3%	32.0%	34.7%	37.0%	33.6%	30.0%	33.7%
B	BoundingBox-Noise	29.0%	33.3%	27.6%	30.2%	33.0%	29.9%	27.4%	29.4%
C	AllInstances-Black	39.6%	43.9%	39.3%	40.7%	42.9%	45.4%	40.8%	42.4%
C	AllInstances-Mean	41.1%	43.7%	40.9%	43.3%	44.2%	48.0%	43.5%	48.1%
C	AllInstances-Noise	40.1%	45.4%	38.8%	41.2%	43.2%	47.0%	40.6%	43.3%
D	GlobalInstance-Inpaint	26.9%	28.6%	26.2%	27.2%	29.0%	29.0%	25.6%	28.4%
E	Silhouette-Mean-Black	1.4%	1.6%	1.4%	2.4%	3.1%	7.4%	4.2%	1.8%
E	Silhouette-Black-Mean	8.5%	9.8%	9.3%	10.6%	9.9%	12.7%	18.4%	13.2%
E	SilhouetteNoiseNoise	5.3%	9.6%	2.5%	3.4%	2.9%	11.9%	9.0%	4.1%
F	GlobalHallucination-Inpaint-0	2.8%	4.9%	2.7%	3.2%	3.6%	3.8%	2.9%	2.9%
F	LocalHallucination-Black	16.1%	20.6%	15.4%	17.5%	19.3%	19.5%	16.8%	15.0%
F	GlobalHallucination-Black-0	22.7%	27.8%	21.3%	24.6%	26.5%	27.9%	21.2%	23.2%
F	GlobalHallucination-Black-2	10.4%	12.9%	10.1%	11.9%	12.3%	13.9%	10.5%	11.3%
F	GlobalHallucination-Black-4	5.9%	7.6%	5.9%	6.9%	7.3%	8.3%	6.1%	6.4%
F	GlobalHallucination-Black-6	4.0%	5.2%	4.0%	4.6%	5.0%	5.8%	4.2%	4.2%
F	LocalHallucination-Mean	6.1%	8.9%	6.1%	7.1%	7.5%	16.4%	8.2%	5.0%
F	GlobalHallucination-Mean-0	10.4%	13.8%	10.1%	11.4%	13.3%	22.7%	11.7%	8.0%
F	GlobalHallucination-Mean-2	4.0%	5.5%	4.2%	4.8%	5.3%	12.4%	5.9%	3.5%
F	GlobalHallucination-Mean-4	1.7%	3.0%	1.9%	2.3%	2.8%	7.5%	3.0%	1.3%
F	GlobalHallucination-Mean-6	0.9%	2.0%	0.9%	1.2%	1.7%	5.2%	2.0%	0.6%
F	LocalHallucination-Noise	9.4%	11.2%	9.1%	8.3%	8.7%	10.4%	11.2%	8.9%
F	GlobalHallucination-Noise-0	15.8%	18.6%	15.4%	15.5%	16.2%	18.9%	15.5%	16.8%
F	GlobalHallucination-Noise-2	8.3%	8.7%	8.1%	8.1%	8.1%	10.1%	8.8%	8.9%
F	GlobalHallucination-Noise-4	5.2%	5.2%	5.1%	5.1%	5.1%	6.5%	5.7%	5.4%
F	GlobalHallucination-Noise-6	3.7%	3.5%	3.8%	3.7%	3.8%	4.7%	4.3%	3.9%
G	BlurredIsolation-Black	26.2%	29.8%	27.1%	28.2%	29.4%	32.8%	28.7%	29.5%
G	BlurredIsolation-Mean	30.8%	32.2%	31.6%	26.0%	33.8%	38.2%	28.7%	30.7%
G	BlurredIsolation-Noise	31.4%	32.9%	30.8%	25.3%	34.4%	38.5%	27.4%	27.7%
G	BlurredLocal	31.1%	35.2%	29.8%	31.8%	33.6%	32.2%	28.7%	30.7%
G	BlurredGlobal-Sharp	36.8%	42.2%	34.8%	38.3%	41.0%	39.9%	34.1%	38.0%
G	BlurredGlobal-Soft	29.0%	33.7%	27.0%	30.4%	33.0%	31.9%	27.5%	30.2%
H	BlurredBackground-Soft	35.8%	40.0%	34.4%	37.1%	40.0%	39.1%	34.2%	38.0%
H	BackgroundSwap-Sharp	32.8%	36.5%	32.5%	33.5%	34.3%	33.3%	29.5%	32.6%
H	BackgroundSwap-Soft	25.8%	28.7%	25.5%	26.3%	27.0%	25.5%	23.6%	25.6%

Table A.2: Per class breakdown of the maskAP in the **Baseline** and **IsolatedInstance-Mean-0** experiment modes for all eight architectures.

	MRCNN50-3x		MRCNN50-Jitter		MRCNN50-1x		MRCNN101-3x		Swin-T		CondInst		MEInst		SOLO	
	Baseline	Isolated	Baseline	Isolated	Baseline	Isolated	Baseline	Isolated	Baseline	Isolated	Baseline	Isolated	Baseline	Isolated	Baseline	Isolated
person	47.7%	67.4%	51.7%	70.3%	46.0%	68.4%	48.6%	69.6%	50.3%	72.1%	48.4%	70.6%	42.7%	62.5%	45.9%	74.6%
bicycle	18.0%	34.4%	25.5%	29.7%	16.3%	32.2%	19.9%	33.6%	21.3%	33.3%	19.9%	35.6%	14.8%	28.6%	18.3%	38.2%
car	41.8%	51.2%	46.4%	45.6%	39.1%	53.1%	42.4%	50.0%	44.3%	47.2%	42.9%	50.9%	40.6%	48.7%	38.2%	49.4%
motorcycle	33.0%	50.8%	39.9%	55.0%	30.8%	49.3%	35.1%	51.2%	37.9%	52.8%	36.5%	55.3%	29.6%	45.7%	33.8%	59.3%
airplane	49.3%	56.6%	56.1%	72.7%	49.0%	61.3%	51.2%	62.9%	55.3%	70.7%	53.7%	70.4%	36.7%	44.5%	56.0%	73.3%
bus	63.7%	63.8%	69.4%	73.8%	61.7%	67.5%	65.1%	68.4%	69.4%	72.9%	67.5%	79.1%	62.6%	73.2%	65.6%	76.7%
train	61.0%	65.1%	68.2%	75.2%	57.6%	63.6%	62.7%	68.8%	66.9%	70.0%	63.3%	79.2%	59.3%	69.8%	65.7%	74.0%
truck	35.1%	58.8%	42.7%	63.2%	31.5%	63.1%	36.3%	61.3%	40.5%	63.2%	39.7%	70.3%	34.4%	62.8%	36.9%	69.2%
boat	23.0%	31.3%	28.4%	34.6%	22.0%	37.4%	25.1%	39.1%	26.8%	39.1%	24.9%	52.5%	20.6%	35.7%	22.6%	51.3%
traffic light	26.8%	46.5%	30.9%	55.2%	26.0%	51.6%	27.9%	51.1%	30.4%	52.6%	27.4%	56.6%	25.7%	63.7%	23.6%	52.8%
fire hydrant	62.4%	68.2%	68.6%	73.7%	63.1%	70.4%	64.9%	70.7%	66.7%	68.0%	63.5%	77.6%	58.4%	68.5%	62.1%	76.8%
stop sign	66.2%	68.4%	66.9%	67.3%	64.3%	69.1%	68.9%	70.6%	67.7%	74.6%	64.6%	70.7%	62.0%	84.4%	61.2%	68.4%
parking meter	45.0%	49.3%	52.1%	45.3%	46.4%	47.9%	48.3%	51.1%	52.3%	58.5%	48.9%	62.1%	47.8%	59.0%	45.1%	55.5%
bench	17.3%	23.7%	22.5%	21.7%	15.6%	23.6%	18.4%	24.1%	20.7%	25.1%	18.4%	33.9%	13.8%	24.3%	18.0%	39.8%
bird	30.3%	47.1%	34.4%	62.8%	29.2%	53.4%	32.3%	52.7%	35.4%	56.5%	29.9%	48.3%	24.9%	44.1%	28.5%	47.1%
cat	66.9%	76.5%	69.8%	82.3%	65.3%	74.7%	67.8%	79.3%	72.6%	81.0%	69.6%	86.1%	60.4%	74.6%	69.8%	85.2%
dog	57.2%	67.1%	62.3%	71.9%	56.3%	65.1%	59.5%	69.4%	63.0%	70.1%	62.1%	76.5%	50.1%	61.7%	59.7%	76.3%
horse	41.6%	46.6%	48.2%	56.7%	39.2%	48.4%	42.0%	49.4%	45.8%	56.5%	44.5%	60.0%	34.1%	43.7%	45.3%	61.3%
sheep	43.7%	52.4%	51.0%	47.1%	41.1%	57.7%	46.2%	58.1%	49.7%	53.7%	48.6%	60.2%	41.9%	53.0%	47.4%	69.6%
cow	46.9%	49.3%	53.5%	62.4%	44.0%	55.3%	48.2%	56.6%	52.5%	57.0%	49.6%	57.1%	45.8%	55.4%	48.6%	58.3%
elephant	55.8%	59.3%	62.7%	59.9%	55.6%	57.6%	56.9%	61.5%	60.6%	64.7%	61.8%	75.9%	54.7%	62.5%	61.8%	69.9%
bear	69.4%	75.2%	70.0%	78.1%	64.8%	77.8%	70.0%	74.7%	73.4%	76.6%	75.1%	89.6%	66.8%	80.1%	72.8%	83.4%
zebra	56.3%	71.9%	60.2%	75.4%	55.1%	73.2%	58.7%	73.7%	59.3%	72.8%	62.3%	77.7%	50.2%	60.2%	62.8%	81.7%
giraffe	51.5%	67.2%	55.4%	72.3%	49.1%	67.3%	51.4%	67.5%	54.9%	70.5%	57.4%	76.8%	41.5%	52.9%	57.3%	78.6%
backpack	16.4%	31.1%	21.2%	12.9%	13.8%	27.9%	18.0%	29.8%	20.9%	19.7%	17.7%	28.5%	15.6%	25.3%	13.8%	27.1%
umbrella	44.7%	50.1%	50.1%	54.8%	42.6%	57.0%	46.8%	57.6%	51.5%	57.3%	49.9%	67.8%	41.4%	60.1%	47.1%	65.9%
handbag	14.9%	17.8%	20.8%	12.6%	12.6%	16.0%	16.6%	17.5%	21.2%	16.3%	17.3%	17.7%	12.9%	22.1%	13.9%	18.9%
tie	31.2%	14.5%	38.8%	18.0%	30.3%	17.7%	35.4%	16.7%	36.1%	14.2%	32.5%	15.1%	26.9%	30.6%	30.1%	21.2%
suitcase	39.4%	38.6%	48.5%	39.0%	36.1%	37.0%	42.3%	39.5%	45.9%	38.3%	46.9%	45.3%	41.3%	56.4%	39.5%	50.5%
frisbee	62.3%	55.5%	66.5%	57.8%	61.6%	60.0%	64.2%	50.9%	66.5%	55.3%	63.5%	64.3%	63.7%	67.1%	61.4%	65.9%
skis	3.2%	4.5%	7.2%	6.4%	2.0%	4.1%	4.3%	7.6%	6.3%	6.8%	5.1%	9.4%	0.7%	4.3%	5.8%	10.4%
snowboard	22.0%	16.6%	32.1%	12.0%	20.3%	13.7%	21.5%	15.6%	26.5%	9.3%	23.3%	16.9%	14.5%	24.0%	22.9%	20.4%
sports ball	46.8%	57.9%	50.0%	61.8%	46.5%	62.9%	48.4%	59.8%	51.2%	68.2%	44.4%	57.6%	45.5%	59.5%	38.1%	56.7%
kite	30.9%	46.8%	35.0%	69.4%	29.3%	51.8%	31.6%	54.1%	32.9%	64.3%	30.6%	52.5%	25.1%	44.9%	29.5%	52.7%
baseball bat	24.7%	28.0%	32.4%	29.6%	20.1%	40.0%	25.3%	31.2%	30.4%	20.2%	25.1%	39.9%	16.8%	44.3%	25.4%	39.3%
baseball glove	38.6%	33.5%	42.8%	23.3%	36.9%	36.7%	41.3%	29.1%	44.5%	31.8%	40.1%	29.5%	38.0%	43.0%	38.2%	38.3%
skateboard	31.5%	28.7%	39.9%	28.9%	27.3%	31.4%	33.6%	26.5%	37.6%	34.3%	35.1%	29.3%	24.4%	34.5%	37.0%	35.3%
surfboard	31.2%	21.6%	39.0%	19.8%	28.9%	19.9%	32.0%	22.1%	37.1%	21.2%	32.3%	34.5%	24.0%	28.8%	30.5%	27.1%
tennis racket	53.7%	56.2%	60.8%	48.8%	52.3%	54.1%	56.3%	54.8%	59.7%	52.8%	52.7%	55.6%	47.6%	57.0%	53.7%	61.9%
bottle	38.2%	54.4%	43.1%	43.1%	36.2%	52.8%	39.8%	52.2%	41.9%	57.1%	37.4%	44.9%	35.8%	58.5%	31.0%	56.3%
wine glass	30.9%	45.5%	38.3%	35.4%	29.1%	42.7%	33.8%	46.6%	36.5%	46.3%	33.8%	41.1%	30.3%	37.5%	29.0%	49.0%
cup	41.3%	52.1%	49.4%	38.9%	40.4%	51.9%	43.8%	53.6%	47.1%	49.6%	43.7%	49.5%	41.4%	43.5%	37.3%	51.8%
fork	15.3%	21.4%	23.9%	30.7%	13.4%	22.4%	18.3%	27.3%	22.7%	21.1%	20.7%	25.0%	9.6%	17.3%	17.9%	26.4%
knife	12.8%	10.3%	19.0%	11.9%	9.0%	11.4%	13.5%	14.5%	18.0%	10.5%	14.4%	12.8%	10.2%	16.8%	12.3%	11.2%
spoon	12.2%	13.4%	19.0%	14.7%	9.1%	12.2%	13.9%	16.3%	16.1%	14.2%	13.9%	12.4%	7.4%	10.7%	11.5%	11.1%
bowl	40.0%	50.1%	43.8%	40.4%	38.2%	48.7%	39.9%	51.2%	41.8%	50.0%	39.3%	51.6%	37.2%	38.9%	36.2%	50.7%
banana	19.4%	45.3%	24.3%	44.8%	17.2%	49.3%	19.4%	46.3%	23.0%	48.5%	24.9%	53.3%	19.0%	47.7%	20.0%	52.4%
apple	20.2%	39.8%	21.7%	44.1%	18.0%	42.3%	20.8%	40.7%	24.2%	45.3%	24.4%	41.4%	19.3%	38.2%	19.5%	44.8%
sandwich	36.7%	48.2%	41.4%	55.9%	32.4%	48.9%	38.0%	51.4%	44.9%	56.5%	37.2%	53.8%	33.7%	54.7%	35.3%	60.4%
orange	30.4%	42.6%	35.2%	37.3%	29.0%	44.2%	31.8%	42.5%	35.2%	44.2%	33.1%	50.5%	29.9%	49.7%	28.7%	59.4%
broccoli	21.6%	47.4%	24.0%	52.0%	22.1%	55.8%	23.0%	43.1%	24.0%	52.4%	24.4%	59.8%	20.7%	60.6%	22.2%	63.0%
carrot	18.6%	24.4%	20.4%	14.3%	18.1%	24.7%	19.4%	21.1%	21.4%	21.1%	23.4%	16.3%	18.6%	21.8%	19.1%	30.4%
hot dog	27.4%	43.6%	33.7%	48.2%	20.6%	49.8%	30.5%	49.3%	31.6%	45.7%	33.2%	51.2%	28.0%	51.5%	29.9%	43.0%
pizza	50.3%	67.5%	53.7%	70.1%	49.3%	64.5%	50.7%	63.8%	52.4%	71.9%	51.9%	73.0%	48.6%	69.2%	48.7%	70.4%
donut	45.3%	52.2%	50.3%	47.9%	43.5%	58.5%	46.4%	55.6%	51.3%	57.3%	50.0%	52.9%	46.9%	75.7%	43.7%	52.6%
cake	35.0%	42.4%	41.9%	47.8%	34.2%	46.8%	37.7%	45.8%	42.6%	46.4%	38.8%	55.4%	35.1%	65.0%	36.6%	57.4%
chair	18.1%	33.7%	24.4%	20.7%	16.4%	29.6%	19.7%	34.7%	23.4%	27.4%	23.7%	30.3%	17.8%	23.2%	20.6%	39.6%
couch	36.1%	38.8%	39.4%	36.3%	32.8%	36.8%	36.1%	41.5%	40.3%	36.8%	42.9%	48.7%	34.2%	39.5%	40.7%	50.0%
potted plant	22.7%	47.1%	27.8%	39.9%	21.6%	48.5%	23.2%	49.4%	27.1%	47.5%	24.0%	54.2%	20.8%	43.4%	22.0%	57.0%
bed	32.0%	38.9%	34.0%	46.3%	30.7%	36.5%	33.8%	43.7%	35.6%	44.4%	35.6%	56.8%	27.8%	47.8%	36.6%	58.9%
dining table	16.1%	26.5%	18.3%	27.9%	14.1%	26.2%	17.1%	29.4%	17.9%	27.6%	17.0%	32.1%	14.3%	26.1%	17.9%	35.8%
toilet	57.4%	66.2%	61.1%	67.2%	55.2%	65.4%	59.6%	65.6%	61.0%	64.2%	62.2%	72.4%	56.2%	68.3%	63.5%	74.7%
tv	57.3%	61.5%	62.1%	70.6%	55.1%	58.3%	60.4%	62.4%	63.9%	66.1%	62.5%	72.7%	56.4%	66.0%	58.1%	69.2%
laptop	59.2%	61.4%	65.5%	60.7%	58.0%	58.4%	60.8%	63.5%	65.6%	57.6%	60.2%	64.4%	54.7%	58.8%	59.9%	66.5%
mouse	64.2%	46.6%	63.6%	53.8%	62.6%	46.0%	63.9%	53.6%	64.4%	34.7%	61.5%	38.9%	60.7%	37.1%	56.8%	41.9%
remote	28.4%	19.6%	36.5%	17.1%	25.2%	21.4%	32.4%	20.6%	36.2%	18.0%	30.8%	15.3%	24.9%	21.3%	27.2%	22.5%
keyboard	50.8%	55.9%	53.6%	60.2%	47.3%	51.0%	50.4%	58.6%	53.7%	56.5%	49.6%	59.6%	44.7%	61.5%	50.7%	66.7%
cell phone	34.0%	25.7%	40.9%	27.2%	32.6%	26.3%	36.7%	29.9%	39.6%	24.5%	34.4%	27.9%	30.1%	29.5%	32.0%	30.0%
microwave	56.0%	54.4%	61.1%	48.9%	55.9%	52.8%	57.0%	51.9%	63.7%	55.0%	59.5%	64.9%	57.7%	69.2%	57.8%	61.9%
oven	31.1%	36.0%	34.9%	37.4%	28.8%	34.4%	32.2%	44.9%	35.3%	43.4%	35.1%	47.9%	30.9%	41.6%	35.1%	51.3%
toaster	43.3%	40.7%	56.7%	48.8%	42.5%	20.5%	40.4%	18.8%	42.3%	37.2%	29.1%	21.7%	45.7%	51.8%	36.7%	22.8%
sink	35.8%	28.1%	40.3%	24.4%	32.7%	26.8%	36.2%	30.4%	39.5%	24.8%	38.9%	35.8%	36.1%	32.1%	35.8%	47.8%
refrigerator	57.0%	59.9%	61.6%	65.4%	53.8%	60.4%	58.8%	63.8%	61.3%	64.7%	59.0%	72.8%	52.0%	65.6%	55.3%	74.9%
book	10.2%	8.3%	13.1%	4.1%	9.0%	7.4%	10.9%	9.2%	13.4%	7.6%	10.0%	9.1%	8.6%	8.3%	6.2%	10.4%
clock	50.3%	53.4%	54.2%	54.8%	50.0%	55.8%	51.7%	52.7%	53.4%	59.4%	52.1%	61.1%	49.7%	63.4%	48.4%	60.5%
vase	36.6%	43.7%	43.8%	42.1%	34.7%	45.1%	38.7%	45.3%	39.4%	48.4%	39.1%	49.5%	35.9%	54.6%	33.7%	58.4%
scissors	20.6%	29.9%	26.9%	36.7%	15.1%	28.2%	22.4%	31.2%	28.2%	34.3%	21.3%	39.4%	14.6%	26.3%	22.0%	43.0%
teddy bear	43.6%	61.7%	48.4%	69.1%	42.5%	64.1%	44.7%	67.2%	48.0%	65.6%	48.3%	76.2%	41.3%	70.8%	46.8%	78.3%
hair drier	0.6%	13.6%	7.5%	0.0%	0.0%	5.0%	2.2%	28.8%	10.7%	10.6%	1.3%	0.0%	4.6%	30.6%	12.4%	24.3%
toothbrush	15.0%	21.3%	23.4%	30.0%	13.6%	27.4%	15.2%	23.7%	22.0%	21.4%	15.1%	12.1%	8.7%	26.4%	16.2%	34.6%

Table A.3: Per mask size breakdown of the maskAP results across the Resnet-based MaskR-CNN architectures. Mask size ranges are taken from the COCO dataset API. **Small** instances are instances with a total mask_area <= 1024 pixels, **Medium** instances are instances with a total mask_area 1024 <mask_area<= 9216 pixels, **Large** Instances are instances with a total mask_area> 9216 pixels.

	MRCNN50-3x				MRCNN50-Jitter				MRCNN50-1x				MRCNN101-3x			
	All Instances	Small Instances	Medium Instances	Large Instances	All Instances	Small Instances	Medium Instances	Large Instances	All Instances	Small Instances	Medium Instances	Large Instances	All Instances	Small Instances	Medium Instances	Large Instances
Baseline	37.2%	18.6%	39.5%	53.3%	42.6%	23.8%	45.0%	60.1%	35.2%	17.2%	37.7%	50.3%	38.6%	19.5%	41.3%	55.3%
IsolatedInstance-Black-0	42.8%	15.9%	48.4%	68.5%	47.0%	20.9%	52.2%	73.2%	43.3%	17.6%	50.1%	67.0%	42.7%	13.6%	47.9%	71.9%
IsolatedInstance-Black-6	29.1%	4.6%	28.2%	59.1%	30.8%	6.5%	29.6%	59.8%	28.3%	4.5%	27.4%	57.1%	28.0%	3.8%	26.4%	58.5%
IsolatedInstance-Black-12	29.8%	8.7%	29.9%	52.2%	34.9%	13.4%	35.5%	56.5%	28.5%	8.8%	27.6%	50.7%	29.9%	9.3%	29.4%	52.2%
IsolatedInstance-Black-20	32.8%	12.7%	33.8%	52.0%	38.4%	17.7%	40.1%	57.8%	31.0%	12.2%	31.8%	49.3%	33.3%	13.4%	34.5%	52.2%
IsolatedInstance-Mean-0	44.0%	18.5%	50.8%	64.9%	45.2%	18.5%	48.1%	72.7%	44.7%	19.5%	51.5%	65.6%	45.9%	19.9%	52.3%	69.8%
IsolatedInstance-Mean-6	32.0%	7.2%	33.6%	59.3%	30.7%	7.4%	29.0%	59.2%	30.2%	5.6%	30.9%	57.8%	31.4%	6.0%	32.3%	59.9%
IsolatedInstance-Mean-12	31.7%	10.2%	32.2%	53.2%	33.3%	13.0%	33.1%	55.0%	29.8%	9.0%	29.4%	51.3%	32.5%	10.9%	33.2%	54.3%
IsolatedInstance-Mean-20	33.7%	13.3%	34.5%	52.8%	37.4%	17.8%	39.1%	56.1%	31.8%	12.9%	32.0%	50.5%	34.7%	14.6%	35.8%	54.2%
IsolatedInstance-Noise-0	42.9%	13.1%	50.9%	71.5%	48.3%	18.5%	56.0%	77.8%	41.8%	12.0%	49.9%	69.1%	43.7%	13.8%	50.4%	73.8%
IsolatedInstance-Noise-6	27.3%	5.2%	25.2%	57.3%	28.8%	5.7%	25.8%	59.0%	26.6%	3.3%	24.7%	56.2%	27.6%	4.0%	25.5%	58.7%
IsolatedInstance-Noise-12	26.5%	7.9%	24.6%	48.2%	30.5%	10.6%	29.5%	52.7%	25.1%	6.7%	23.2%	46.5%	26.9%	8.3%	24.9%	49.5%
IsolatedInstance-Noise-20	29.8%	11.8%	30.2%	47.8%	35.6%	16.3%	36.7%	54.9%	28.3%	11.4%	28.6%	45.6%	30.2%	12.9%	30.3%	48.4%
BoundingBox-Black	32.0%	8.0%	34.3%	55.1%	36.0%	12.1%	38.1%	60.8%	31.3%	8.8%	33.6%	52.7%	32.8%	9.4%	35.1%	57.2%
BoundingBox-Mean	33.5%	10.4%	35.7%	54.8%	36.3%	12.9%	38.1%	60.7%	32.0%	9.5%	33.9%	53.4%	34.7%	11.1%	37.5%	57.7%
BoundingBox-Noise	29.0%	8.0%	29.5%	51.4%	33.3%	9.9%	34.0%	58.4%	27.6%	6.5%	28.1%	49.5%	30.2%	8.5%	31.1%	53.7%
AllInstances-Black	39.6%	14.5%	44.1%	63.8%	43.9%	18.8%	48.7%	69.2%	39.3%	15.4%	44.7%	61.6%	40.7%	14.8%	44.8%	67.3%
AllInstances-Mean	41.1%	18.0%	46.5%	61.0%	43.7%	19.8%	46.6%	68.5%	40.9%	18.2%	46.0%	61.1%	43.3%	20.3%	48.4%	65.9%
AllInstances-Noise	40.1%	13.2%	46.8%	66.6%	45.4%	18.1%	52.2%	73.2%	38.8%	12.0%	45.4%	64.1%	41.2%	14.1%	47.1%	68.6%
GlobalInstance-Inpaint	26.9%	7.4%	26.1%	48.0%	28.6%	9.5%	27.9%	49.9%	26.2%	6.7%	25.8%	46.2%	27.2%	7.2%	26.7%	49.2%
Silhouette-Mean-Black	1.4%	1.9%	1.8%	1.7%	1.6%	0.8%	1.7%	2.7%	1.4%	1.1%	2.0%	1.2%	2.4%	1.7%	3.4%	2.7%
Silhouette-Black-Mean	8.5%	4.7%	10.5%	13.0%	9.8%	5.7%	10.1%	15.8%	9.3%	5.0%	11.4%	12.9%	10.6%	5.0%	11.7%	16.5%
SilhouetteNoiseNoise	5.3%	0.3%	6.0%	11.4%	9.6%	0.8%	11.6%	21.1%	2.5%	0.0%	2.0%	6.0%	3.4%	0.2%	4.5%	6.2%
GlobalHallucination-Inpaint-0	2.8%	1.9%	4.4%	4.2%	4.9%	2.4%	7.3%	8.3%	2.7%	1.5%	4.4%	4.4%	3.2%	1.8%	5.1%	4.6%
LocalHallucination-Black	16.1%	6.4%	18.5%	28.0%	20.6%	10.0%	22.6%	34.9%	15.4%	7.0%	17.5%	25.8%	17.5%	7.9%	19.1%	29.2%
GlobalHallucination-Black-0	22.7%	17.0%	25.4%	28.6%	27.8%	20.4%	30.6%	36.8%	21.3%	16.2%	24.4%	26.0%	24.6%	17.2%	28.1%	30.8%
GlobalHallucination-Black-2	10.4%	3.1%	13.7%	22.6%	12.9%	3.0%	15.7%	28.8%	10.1%	2.8%	13.0%	20.6%	11.9%	3.0%	15.4%	24.5%
GlobalHallucination-Black-4	5.9%	0.3%	6.1%	17.6%	7.6%	0.4%	7.1%	22.1%	5.9%	0.3%	5.9%	16.1%	6.9%	0.3%	6.8%	19.9%
GlobalHallucination-Black-6	4.0%	0.0%	2.5%	13.8%	5.2%	0.0%	3.0%	17.2%	4.0%	0.0%	2.4%	12.7%	4.6%	0.0%	2.9%	15.3%
LocalHallucination-Mean	6.1%	3.2%	8.8%	10.3%	8.9%	4.4%	10.0%	16.7%	6.1%	2.6%	9.4%	10.2%	7.1%	3.1%	10.5%	12.2%
GlobalHallucination-Mean-0	10.4%	7.7%	15.4%	12.8%	13.8%	10.2%	16.3%	19.3%	10.1%	7.6%	15.0%	12.2%	11.4%	7.7%	16.8%	15.3%
GlobalHallucination-Mean-2	4.0%	1.6%	7.3%	8.7%	5.5%	1.1%	7.9%	14.2%	4.2%	1.7%	7.6%	8.1%	4.8%	1.3%	8.5%	11.2%
GlobalHallucination-Mean-4	1.7%	0.1%	2.9%	6.0%	3.0%	0.1%	3.3%	10.8%	1.9%	0.1%	3.2%	6.3%	2.3%	0.1%	3.6%	8.2%
GlobalHallucination-Mean-6	0.9%	0.0%	0.9%	4.4%	2.0%	0.0%	1.4%	8.5%	0.9%	0.0%	1.2%	4.1%	1.2%	0.0%	1.3%	5.9%
LocalHallucination-Noise	9.4%	1.7%	8.1%	19.6%	11.2%	2.7%	11.6%	22.7%	9.1%	1.7%	7.9%	19.2%	8.3%	1.8%	6.9%	17.7%
GlobalHallucination-Noise-0	15.8%	7.8%	17.2%	25.0%	18.6%	11.1%	20.9%	26.7%	15.4%	7.8%	17.0%	24.1%	15.5%	7.8%	16.4%	25.4%
GlobalHallucination-Noise-2	8.3%	1.5%	8.3%	19.3%	8.7%	2.0%	10.2%	20.3%	8.1%	1.4%	8.4%	18.9%	8.1%	1.4%	8.2%	19.8%
GlobalHallucination-Noise-4	5.2%	0.3%	3.7%	15.7%	5.2%	0.2%	4.3%	16.0%	5.1%	0.1%	3.7%	14.8%	5.1%	0.1%	3.8%	15.7%
GlobalHallucination-Noise-6	3.7%	0.0%	1.6%	11.9%	3.5%	0.0%	1.6%	12.4%	3.8%	0.0%	1.6%	11.7%	3.7%	0.0%	1.7%	12.4%
BlurredIsolation-Black	26.2%	3.1%	25.6%	53.7%	29.8%	3.7%	29.9%	60.7%	27.1%	3.9%	28.4%	53.2%	28.2%	3.5%	28.1%	58.4%
BlurredIsolation-Mean	30.8%	5.5%	33.9%	58.2%	32.2%	5.4%	32.6%	64.5%	31.6%	6.4%	35.0%	59.4%	26.0%	3.3%	26.1%	55.0%
BlurredIsolation-Noise	31.4%	5.2%	34.8%	61.1%	32.9%	4.8%	34.4%	67.3%	30.8%	5.4%	34.6%	58.7%	25.3%	4.0%	24.6%	54.4%
BlurredLocal	31.1%	7.7%	31.2%	57.1%	35.2%	9.7%	35.9%	62.8%	29.8%	8.2%	29.8%	54.5%	31.8%	7.8%	31.9%	58.7%
BlurredGlobal-Sharp	36.8%	15.1%	39.1%	57.4%	42.2%	18.8%	45.2%	63.6%	34.8%	13.7%	37.2%	54.7%	38.3%	15.3%	40.8%	60.1%
BlurredGlobal-Soft	29.0%	7.9%	29.8%	51.9%	33.7%	9.7%	35.4%	58.0%	27.0%	6.3%	27.9%	49.1%	30.4%	7.3%	31.4%	54.9%
BlurredBackground-Soft	35.8%	13.1%	38.8%	57.1%	40.0%	15.9%	43.5%	62.7%	34.4%	12.0%	37.6%	55.0%	37.1%	13.9%	39.7%	59.6%
BackgroundSwap-Sharp	32.8%	9.2%	34.8%	55.2%	36.5%	12.5%	38.6%	61.4%	32.5%	9.5%	35.3%	53.7%	33.5%	9.0%	35.5%	58.2%
BackgroundSwap-Soft	25.8%	2.5%	25.4%	51.6%	28.7%	3.1%	27.9%	57.5%	25.5%	2.5%	25.6%	49.6%	26.3%	2.6%	25.6%	52.7%

Table A.4: Per mask size breakdown of the maskAP results across Swin-T and the Non-ROI based architectures. Mask size ranges are taken from the COCO dataset API. **Small** instances are instances with a total mask_area <= 1024 pixels, **Medium** instances are instances with a total mask_area 1024 <mask_area<= 9216 pixels, **Large** Instances are instances with a total mask_area> 9216 pixels.

	Swin-T				CondInst				MEInst				SOLO			
	All Instances	Small Instances	Medium Instances	Large Instances	All Instances	Small Instances	Medium Instances	Large Instances	All Instances	Small Instances	Medium Instances	Large Instances	All Instances	Small Instances	Medium Instances	Large Instances
Baseline	41.8%	23.4%	44.2%	59.9%	39.4%	19.6%	43.0%	56.9%	34.5%	18.4%	37.1%	48.5%	37.6%	15.5%	41.1%	56.9%
IsolatedInstance-Black-0	44.7%	15.7%	49.1%	73.3%	46.8%	15.9%	53.0%	76.8%	43.1%	17.4%	49.3%	64.2%	44.7%	14.9%	50.3%	73.7%
IsolatedInstance-Black-6	28.3%	4.6%	25.4%	58.4%	28.2%	4.1%	26.6%	57.8%	27.3%	4.3%	26.6%	55.4%	27.6%	3.1%	26.0%	57.1%
IsolatedInstance-Black-12	31.7%	11.8%	30.2%	53.1%	29.2%	9.2%	28.5%	50.9%	27.4%	7.9%	26.6%	49.3%	28.6%	7.0%	27.9%	52.4%
IsolatedInstance-Black-20	35.1%	15.7%	35.4%	53.5%	32.4%	14.5%	33.1%	50.6%	30.0%	12.5%	31.1%	46.8%	31.8%	10.7%	32.9%	52.4%
IsolatedInstance-Mean-0	46.1%	19.9%	51.6%	68.9%	49.3%	19.0%	56.9%	77.3%	47.3%	23.8%	53.2%	65.9%	51.7%	23.7%	59.7%	76.9%
IsolatedInstance-Mean-6	32.4%	7.7%	31.9%	61.2%	29.8%	4.8%	29.7%	59.6%	29.1%	5.3%	29.5%	58.5%	31.2%	4.9%	31.4%	61.2%
IsolatedInstance-Mean-12	34.8%	13.3%	34.8%	56.3%	30.7%	9.9%	30.2%	52.8%	27.9%	9.1%	26.5%	50.4%	30.1%	8.1%	29.3%	54.4%
IsolatedInstance-Mean-20	37.0%	17.2%	37.3%	56.5%	33.5%	15.1%	34.0%	52.5%	29.9%	14.0%	29.9%	47.3%	32.6%	11.6%	33.3%	53.3%
IsolatedInstance-Noise-0	45.6%	15.0%	52.6%	75.4%	48.4%	16.3%	56.0%	79.7%	43.7%	15.4%	52.9%	67.4%	45.3%	14.8%	52.0%	75.1%
IsolatedInstance-Noise-6	29.3%	5.7%	26.8%	59.3%	26.7%	3.8%	24.3%	56.8%	26.9%	5.0%	25.7%	55.1%	27.3%	4.4%	24.9%	56.5%
IsolatedInstance-Noise-12	30.4%	11.6%	28.3%	51.5%	26.3%	7.9%	24.0%	48.2%	25.1%	7.6%	23.0%	46.7%	26.1%	6.8%	23.5%	48.9%
IsolatedInstance-Noise-20	34.1%	16.5%	34.1%	51.6%	30.4%	14.1%	30.5%	48.2%	27.5%	12.0%	27.7%	44.4%	29.6%	10.6%	29.2%	49.4%
BoundingBox-Black	35.7%	12.0%	37.6%	59.6%	32.4%	9.2%	34.9%	56.4%	29.5%	8.3%	31.1%	50.5%	31.8%	7.7%	33.9%	57.2%
BoundingBox-Mean	37.0%	13.9%	39.0%	59.8%	33.6%	10.6%	35.8%	57.1%	30.0%	9.6%	31.4%	51.3%	33.7%	9.4%	35.8%	58.9%
BoundingBox-Noise	33.0%	11.4%	33.7%	56.6%	29.9%	8.8%	30.5%	54.0%	27.4%	8.4%	27.8%	48.3%	29.4%	7.1%	29.3%	54.3%
AllInstances-Black	42.9%	17.8%	46.1%	68.9%	45.4%	18.5%	50.9%	72.1%	40.8%	18.4%	45.6%	60.2%	42.4%	15.5%	47.0%	69.4%
AllInstances-Mean	44.2%	21.5%	48.1%	65.7%	48.0%	22.2%	54.0%	72.3%	43.5%	21.8%	48.5%	61.8%	48.1%	22.8%	54.8%	71.7%
AllInstances-Noise	43.2%	16.5%	48.9%	70.6%	47.0%	18.8%	54.3%	74.7%	40.6%	15.3%	48.0%	62.7%	43.3%	14.9%	49.2%	70.1%
GlobalInstance-Inpaint	29.0%	8.5%	28.6%	51.9%	29.0%	9.0%	29.1%	50.4%	25.6%	7.2%	24.9%	45.9%	28.4%	6.5%	27.6%	50.7%
Silhouette-Mean-Black	3.1%	3.5%	3.8%	2.4%	7.4%	2.7%	9.2%	14.0%	4.2%	3.7%	5.7%	3.5%	1.8%	2.0%	2.4%	0.7%
Silhouette-Black-Mean	9.9%	7.3%	11.7%	12.2%	12.7%	7.2%	14.0%	20.7%	18.4%	12.5%	22.9%	24.9%	13.2%	8.2%	17.0%	20.1%
SilhouetteNoiseNoise	2.9%	0.7%	4.5%	4.5%	11.9%	1.4%	12.9%	28.6%	9.0%	1.9%	11.6%	16.3%	4.1%	0.4%	4.7%	8.7%
GlobalHallucination-Inpaint-0	3.6%	1.9%	5.3%	5.6%	3.8%	1.7%	5.6%	7.3%	2.9%	2.4%	4.3%	4.5%	2.9%	1.6%	4.8%	4.6%
LocalHallucination-Black	19.3%	10.5%	20.6%	28.8%	19.5%	7.8%	20.6%	36.2%	16.8%	7.1%	19.0%	28.6%	15.0%	6.5%	18.3%	25.2%
GlobalHallucination-Black-0	26.5%	18.6%	29.2%	33.2%	27.9%	17.8%	31.9%	39.0%	21.2%	15.8%	23.7%	26.9%	23.2%	15.1%	27.6%	30.6%
GlobalHallucination-Black-2	12.3%	2.9%	15.0%	26.0%	13.9%	3.4%	16.2%	30.0%	10.5%	2.8%	13.3%	21.8%	11.3%	3.2%	14.1%	23.4%
GlobalHallucination-Black-4	7.3%	0.4%	6.6%	20.6%	8.3%	0.4%	7.2%	22.7%	6.1%	0.3%	5.9%	17.7%	6.4%	0.4%	6.1%	17.5%
GlobalHallucination-Black-6	5.0%	0.0%	2.8%	15.8%	5.8%	0.0%	3.3%	18.0%	4.2%	0.0%	2.4%	14.5%	4.2%	0.0%	2.6%	13.7%
LocalHallucination-Mean	7.5%	3.7%	8.6%	13.6%	16.4%	4.1%	16.3%	34.8%	8.2%	3.1%	9.1%	18.5%	5.0%	2.2%	8.7%	7.1%
GlobalHallucination-Mean-0	13.3%	8.6%	16.8%	18.5%	22.7%	10.7%	26.1%	37.8%	11.7%	8.0%	14.3%	17.0%	8.0%	6.1%	13.5%	7.1%
GlobalHallucination-Mean-2	5.3%	1.7%	7.4%	12.8%	12.4%	2.6%	13.8%	29.0%	5.9%	1.8%	8.0%	13.8%	3.5%	1.5%	6.8%	5.1%
GlobalHallucination-Mean-4	2.8%	0.1%	3.2%	9.4%	7.5%	0.4%	6.4%	21.9%	3.0%	0.2%	3.4%	10.5%	1.3%	0.2%	2.7%	3.8%
GlobalHallucination-Mean-6	1.7%	0.0%	1.3%	7.3%	5.2%	0.0%	2.7%	17.0%	2.0%	0.0%	1.4%	8.9%	0.6%	0.0%	0.9%	2.9%
LocalHallucination-Noise	8.7%	1.9%	6.6%	19.5%	10.4%	2.4%	8.9%	23.3%	11.2%	1.7%	9.3%	26.2%	8.9%	2.1%	8.5%	18.6%
GlobalHallucination-Noise-0	16.2%	8.2%	16.7%	27.0%	18.9%	8.9%	20.3%	33.4%	15.5%	7.4%	16.1%	26.9%	16.8%	7.3%	19.1%	28.2%
GlobalHallucination-Noise-2	8.1%	1.4%	7.9%	20.5%	10.1%	2.1%	10.4%	25.6%	8.8%	1.4%	9.1%	21.1%	8.9%	1.4%	9.7%	22.1%
GlobalHallucination-Noise-4	5.1%	0.1%	3.6%	15.9%	6.5%	0.2%	4.8%	20.0%	5.7%	0.1%	4.2%	17.6%	5.4%	0.1%	4.2%	17.1%
GlobalHallucination-Noise-6	3.8%	0.0%	1.6%	12.7%	4.7%	0.0%	2.0%	15.7%	4.3%	0.0%	1.8%	14.4%	3.9%	0.0%	1.8%	13.0%
BlurredIsolation-Black	29.4%	3.5%	28.9%	60.3%	32.8%	6.8%	34.8%	61.2%	28.7%	6.4%	31.5%	51.7%	29.5%	4.5%	28.6%	58.4%
BlurredIsolation-Mean	33.8%	7.2%	35.7%	64.3%	38.2%	11.1%	41.6%	68.5%	28.7%	6.7%	29.7%	54.8%	30.7%	6.0%	30.9%	61.5%
BlurredIsolation-Noise	34.4%	7.3%	36.6%	65.8%	38.5%	10.3%	43.1%	68.3%	27.4%	6.9%	29.6%	52.0%	27.7%	5.4%	27.7%	56.8%
BlurredLocal	33.6%	9.3%	32.6%	61.0%	32.2%	8.2%	32.1%	59.5%	28.7%	7.3%	28.9%	51.8%	30.7%	6.0%	30.4%	58.5%
BlurredGlobal-Sharp	41.0%	17.8%	44.0%	62.9%	39.9%	16.1%	43.4%	62.1%	34.1%	14.6%	37.0%	52.3%	38.0%	12.6%	41.2%	61.7%
BlurredGlobal-Soft	33.0%	9.0%	34.3%	57.9%	31.9%	8.9%	33.8%	55.6%	27.5%	7.9%	28.8%	47.5%	30.2%	6.7%	31.2%	54.6%
BlurredBackground-Soft	40.0%	16.8%	43.1%	62.3%	39.1%	15.9%	42.6%	61.6%	34.2%	13.8%	37.4%	52.8%	38.0%	13.2%	41.4%	61.8%
BackgroundSwap-Sharp	34.3%	11.0%	36.2%	56.9%	33.3%	9.3%	35.1%	55.3%	29.5%	9.0%	31.8%	48.0%	32.6%	8.1%	34.4%	55.2%
BackgroundSwap-Soft	27.0%	3.1%	25.7%	53.4%	25.5%	3.1%	24.7%	49.4%	23.6%	2.8%	23.9%	44.8%	25.6%	2.5%	24.3%	50.2%

Table A.5: The per class breakdown of maskAP on **MRCNN50-3x** and **MRCNN50-Jitter** and multiple ContentSwap experiment and evaluation modes. Bolded experiments in the table are the more interesting results.

	MRCNN50-3x						MRCNN50-Jitter					
	ContentSwapOriginal-Morph-ShapeAP	*ContentSwapOriginal-Aspect-ShapeAP*	**ContentSwapIsolation-Morph-ShapeAP**	**ContentSwapIsolation-Morph-ContentAP**	*ContentSwapIsolation-Aspect-ShapeAP*	*ContentSwapIsolation-Aspect-ContentAP*	**ContentSwapOriginal-Morph-ShapeAP**	*ContentSwapOriginal-Aspect-ShapeAP*	**ContentSwapIsolation-Morph-ShapeAP**	**ContentSwapIsolation-Morph-ContentAP**	*ContentSwapIsolation-Aspect-ShapeAP*	*ContentSwapIsolation-Aspect-ContentAP*
person	17.2%	18.7%	10.6%	9.7%	11.0%	9.8%	21.6%	22.9%	9.3%	8.1%	10.0%	8.9%
bicycle	1.1%	0.4%	0.5%	6.2%	0.3%	4.1%	1.0%	0.4%	0.1%	4.8%	0.1%	3.4%
car	8.2%	6.6%	3.7%	19.0%	2.6%	18.0%	9.6%	7.9%	1.6%	15.7%	1.0%	14.7%
motorcycle	3.9%	3.4%	0.5%	10.0%	0.6%	8.5%	5.4%	5.2%	0.6%	9.6%	0.5%	8.3%
airplane	20.2%	16.4%	10.9%	3.9%	8.6%	3.6%	28.4%	24.2%	32.0%	11.6%	28.5%	10.7%
bus	8.2%	7.8%	1.1%	20.0%	1.1%	19.4%	11.2%	9.8%	1.3%	24.0%	1.1%	23.2%
train	16.8%	17.4%	2.3%	9.8%	2.0%	8.0%	20.3%	18.8%	3.0%	11.4%	3.0%	10.5%
truck	8.9%	9.0%	4.4%	19.5%	4.1%	17.6%	11.5%	12.6%	3.9%	20.2%	3.9%	18.9%
boat	17.9%	13.9%	2.8%	9.2%	0.8%	6.6%	25.7%	20.8%	5.0%	11.6%	2.7%	9.7%
traffic light	19.6%	20.9%	11.3%	17.5%	13.4%	16.8%	23.8%	24.1%	13.8%	25.3%	15.5%	25.2%
fire hydrant	36.8%	41.9%	8.4%	10.3%	10.6%	8.9%	41.7%	45.8%	5.5%	9.6%	7.0%	8.3%
stop sign	27.2%	28.5%	6.7%	29.5%	7.8%	26.5%	32.0%	34.2%	9.2%	30.5%	11.5%	27.4%
parking meter	21.2%	24.9%	1.6%	9.3%	2.4%	8.8%	21.9%	26.1%	1.6%	10.1%	1.8%	9.9%
bench	7.9%	8.1%	2.2%	4.8%	1.8%	4.5%	10.8%	11.5%	1.6%	3.5%	1.2%	3.9%
bird	13.0%	12.3%	5.6%	6.2%	4.8%	4.9%	19.8%	19.3%	18.1%	12.6%	18.0%	10.7%
cat	1.9%	2.2%	1.4%	7.1%	1.3%	7.8%	2.9%	3.6%	2.2%	7.2%	4.1%	9.1%
dog	10.6%	11.1%	2.8%	7.2%	2.9%	8.1%	12.1%	12.9%	2.8%	7.9%	3.6%	8.9%
horse	17.0%	15.4%	4.5%	4.2%	4.2%	5.9%	23.5%	21.9%	8.7%	5.0%	8.6%	6.6%
sheep	3.0%	3.4%	1.0%	8.6%	1.0%	9.5%	3.4%	3.9%	1.4%	6.5%	1.5%	8.0%
cow	16.7%	18.6%	6.0%	10.2%	6.2%	11.5%	22.0%	24.2%	9.0%	11.2%	9.8%	12.4%
elephant	9.6%	10.0%	1.2%	9.6%	1.1%	11.3%	10.0%	10.5%	1.3%	9.8%	1.7%	10.7%
bear	5.2%	6.8%	0.5%	10.8%	0.6%	10.6%	5.8%	8.9%	1.6%	11.3%	3.4%	11.7%
zebra	0.7%	0.7%	0.3%	10.2%	0.4%	12.7%	1.8%	1.5%	0.5%	9.3%	0.3%	11.5%
giraffe	14.7%	14.3%	4.2%	1.7%	3.7%	1.5%	20.4%	20.3%	7.5%	1.9%	7.8%	2.0%
backpack	5.1%	4.0%	2.2%	13.8%	1.6%	11.2%	5.5%	4.4%	0.5%	9.5%	0.5%	7.9%
umbrella	23.8%	21.8%	13.7%	9.3%	9.8%	8.9%	27.0%	24.8%	20.8%	11.5%	17.5%	11.5%
handbag	16.0%	13.3%	2.0%	5.8%	1.3%	4.3%	21.9%	18.9%	0.3%	4.3%	0.3%	3.3%
tie	35.3%	34.1%	0.7%	4.0%	0.6%	2.3%	42.6%	42.7%	2.6%	5.5%	2.9%	4.4%
suitcase	7.3%	7.9%	1.1%	11.3%	1.1%	8.7%	8.7%	9.6%	1.0%	10.2%	0.8%	8.1%
frisbee	20.5%	17.3%	2.8%	15.8%	1.6%	11.2%	25.5%	22.1%	2.1%	12.2%	1.0%	7.8%
skis	7.8%	7.0%	0.6%	0.3%	0.6%	0.3%	12.4%	11.3%	1.0%	0.5%	1.1%	0.3%
snowboard	29.9%	30.1%	2.6%	1.4%	3.8%	1.0%	38.6%	40.3%	3.9%	2.1%	5.8%	1.6%
sports ball	17.4%	17.5%	10.4%	14.6%	10.7%	11.8%	20.5%	20.3%	16.5%	12.0%	16.1%	9.7%
kite	31.5%	31.1%	25.6%	13.5%	24.3%	12.3%	36.3%	35.7%	52.5%	33.7%	52.0%	32.7%
baseball bat	32.7%	32.0%	4.2%	0.4%	2.7%	0.2%	40.1%	38.5%	5.6%	0.2%	5.1%	0.2%
baseball glove	30.7%	28.1%	1.5%	9.2%	1.2%	7.4%	39.2%	35.3%	0.7%	11.0%	0.3%	9.0%
skateboard	19.4%	18.2%	0.5%	1.3%	0.3%	1.1%	29.4%	28.4%	0.3%	1.7%	0.3%	1.6%
surfboard	29.4%	25.8%	3.2%	4.5%	1.6%	2.7%	38.3%	34.9%	3.6%	5.9%	2.0%	4.0%
tennis racket	19.4%	15.6%	0.9%	4.6%	0.4%	4.3%	32.7%	28.5%	0.7%	4.0%	0.8%	3.8%
bottle	15.6%	20.4%	8.3%	6.9%	12.7%	4.2%	18.9%	24.5%	4.3%	4.5%	6.8%	2.9%
wine glass	9.8%	10.4%	4.6%	4.6%	5.2%	6.6%	14.2%	15.5%	2.6%	3.7%	2.5%	4.4%
cup	10.3%	10.9%	2.9%	10.6%	2.3%	9.7%	11.8%	12.0%	1.6%	8.0%	1.2%	8.4%
fork	7.4%	7.2%	3.4%	0.6%	4.1%	0.6%	14.1%	13.1%	6.4%	0.4%	8.5%	0.7%
knife	12.0%	12.1%	3.5%	0.5%	3.7%	0.5%	22.4%	24.4%	6.4%	0.8%	6.6%	0.7%
spoon	7.9%	7.8%	0.1%	0.6%	0.1%	0.7%	14.5%	13.9%	1.2%	0.5%	1.7%	0.6%
bowl	11.6%	11.0%	2.4%	11.7%	1.7%	10.3%	13.6%	13.3%	1.5%	11.1%	1.3%	10.7%
banana	0.6%	0.4%	0.4%	11.4%	0.2%	9.5%	0.8%	0.5%	0.5%	9.2%	0.4%	8.5%
apple	0.5%	0.5%	0.7%	13.1%	0.6%	12.7%	0.4%	0.5%	3.9%	13.4%	4.4%	13.5%
sandwich	3.8%	4.1%	1.2%	11.6%	1.4%	11.7%	4.8%	4.6%	1.7%	14.1%	1.8%	13.8%
orange	0.3%	0.3%	0.2%	15.3%	0.2%	14.6%	0.2%	0.2%	0.2%	11.6%	0.3%	11.2%
broccoli	0.3%	0.4%	0.0%	11.0%	0.1%	10.3%	0.9%	0.8%	0.1%	12.0%	0.1%	12.7%
carrot	0.3%	0.3%	0.3%	5.5%	0.3%	5.3%	0.2%	0.2%	0.2%	5.7%	0.3%	5.7%
hot dog	1.7%	1.4%	1.2%	7.3%	1.2%	7.3%	1.5%	1.5%	0.7%	9.2%	0.9%	8.5%
pizza	2.7%	2.5%	1.4%	16.3%	1.0%	18.3%	3.7%	2.6%	1.4%	17.4%	1.5%	20.3%
donut	3.5%	3.6%	2.7%	17.6%	3.2%	16.1%	2.8%	2.6%	2.4%	15.0%	2.4%	13.3%
cake	20.5%	21.2%	5.3%	15.0%	5.2%	13.2%	23.8%	25.3%	8.4%	16.2%	8.1%	15.8%
chair	7.5%	8.0%	2.7%	7.7%	2.6%	7.3%	9.3%	9.9%	2.0%	4.7%	2.0%	4.1%
couch	3.9%	5.7%	1.1%	6.4%	1.6%	7.1%	5.6%	7.5%	0.5%	5.6%	0.7%	6.0%
potted plant	1.7%	1.6%	1.4%	9.6%	1.3%	9.6%	2.1%	2.2%	0.8%	8.1%	0.8%	8.8%
bed	5.4%	4.8%	3.4%	7.2%	2.4%	6.8%	6.2%	6.0%	4.8%	7.5%	4.4%	7.3%
dining table	4.3%	3.7%	2.4%	5.9%	1.4%	5.1%	6.8%	6.8%	2.2%	5.9%	1.4%	5.4%
toilet	4.4%	5.1%	2.1%	11.8%	2.0%	11.8%	6.2%	6.6%	1.8%	9.6%	1.4%	8.8%
tv	14.4%	16.3%	3.2%	16.9%	3.4%	14.8%	15.5%	17.2%	11.4%	18.0%	11.6%	16.9%
laptop	4.1%	4.6%	1.2%	8.2%	1.4%	7.2%	7.2%	8.3%	2.0%	9.0%	1.8%	8.5%
mouse	25.3%	27.1%	0.8%	11.9%	0.8%	8.2%	29.7%	30.4%	1.7%	11.5%	1.5%	8.6%
remote	12.9%	12.9%	1.7%	6.0%	1.0%	6.2%	16.6%	16.7%	0.9%	5.8%	0.9%	5.8%
keyboard	4.7%	4.2%	0.5%	6.5%	0.3%	10.9%	7.6%	7.0%	2.6%	9.2%	1.8%	13.1%
cell phone	16.9%	17.0%	0.5%	8.4%	0.3%	6.7%	18.6%	17.9%	1.2%	11.3%	0.9%	9.0%
microwave	4.6%	6.2%	0.8%	12.7%	1.1%	10.2%	6.9%	8.5%	1.2%	14.0%	1.3%	11.9%
oven	4.0%	3.6%	0.9%	7.3%	1.1%	6.6%	4.2%	4.0%	1.3%	8.4%	1.1%	8.2%
toaster	29.9%	30.3%	0.0%	8.1%	0.0%	5.4%	27.3%	30.9%	0.2%	8.1%	0.4%	7.1%
sink	12.1%	13.8%	0.2%	5.9%	0.2%	5.9%	13.2%	15.6%	0.1%	5.7%	0.2%	5.8%
refrigerator	16.2%	12.5%	3.7%	8.5%	1.6%	9.0%	20.1%	18.3%	4.5%	8.3%	3.0%	9.1%
book	8.1%	6.3%	3.1%	3.1%	1.2%	2.0%	7.8%	6.7%	3.4%	2.2%	2.3%	1.6%
clock	13.6%	11.1%	2.6%	19.4%	1.0%	15.4%	14.9%	12.0%	2.4%	20.7%	1.4%	17.2%
vase	22.0%	18.1%	5.2%	10.1%	5.0%	7.6%	28.9%	26.5%	6.2%	11.3%	5.7%	9.4%
scissors	2.2%	1.7%	0.4%	2.3%	0.3%	1.8%	3.9%	3.2%	2.6%	3.1%	2.4%	2.7%
teddy bear	1.8%	2.0%	5.7%	11.6%	6.6%	14.3%	1.8%	2.0%	4.2%	10.5%	5.3%	13.2%
hair drier	6.9%	7.5%	0.3%	3.1%	0.3%	1.8%	5.6%	5.6%	0.0%	1.0%	0.0%	0.8%
toothbrush	10.9%	9.1%	2.2%	1.1%	2.4%	1.5%	18.5%	16.0%	8.7%	1.2%	8.9%	1.6%

Table A.6: The per class breakdown of maskAP on **MRCNN50-1x** and **MRCNN101-3x** and multiple ContentSwap experiment and evaluation modes. Bolded experiments in the table are the more interesting results.

	MRCNN50-1x						MRCNN101-3x					
	ContentSwapOriginal-Morph-ShapeAP	*ContentSwapOriginal-Aperct-ShapeAP*	**ContentSwapIsolation-Morph-ShapeAP**	**ContentSwapIsolation-Morph-ContentAP**	*ContentSwapIsolation-Aperct-ShapeAP*	*ContentSwapIsolation-Aperct-ContentAP*	**ContentSwapOriginal-Morph-ShapeAP**	*ContentSwapOriginal-Aperct-ShapeAP*	**ContentSwapIsolation-Morph-ShapeAP**	**ContentSwapIsolation-Morph-ContentAP**	*ContentSwapIsolation-Aperct-ShapeAP*	*ContentSwapIsolation-Aperct-ContentAP*
person	16.5%	17.3%	11.5%	11.2%	12.3%	10.9%	18.6%	19.7%	11.4%	9.4%	12.0%	9.7%
bicycle	1.1%	0.6%	0.4%	5.4%	0.1%	3.6%	0.8%	0.3%	0.2%	6.1%	0.1%	4.2%
car	8.3%	6.8%	6.0%	20.8%	4.9%	19.7%	7.3%	5.9%	3.4%	18.5%	2.7%	17.3%
motorcycle	3.9%	3.4%	0.6%	10.3%	0.6%	8.4%	4.3%	4.1%	0.6%	10.3%	0.5%	8.8%
airplane	19.3%	16.1%	19.4%	8.4%	15.2%	7.8%	24.5%	20.5%	20.3%	6.9%	16.6%	6.8%
bus	7.1%	7.2%	2.1%	23.3%	1.6%	22.3%	9.3%	8.5%	1.7%	21.7%	1.2%	20.6%
train	17.6%	19.2%	2.7%	11.6%	3.1%	9.7%	17.1%	17.6%	2.6%	11.4%	2.7%	9.9%
truck	10.2%	11.0%	8.2%	22.0%	7.9%	20.8%	7.3%	7.8%	5.1%	19.9%	4.1%	18.0%
boat	16.9%	12.8%	5.7%	11.6%	2.6%	8.9%	20.7%	15.7%	7.3%	13.4%	2.9%	11.1%
traffic light	19.9%	20.9%	16.9%	22.2%	20.4%	21.4%	19.7%	20.2%	12.3%	20.0%	14.3%	18.9%
fire hydrant	36.9%	41.2%	12.7%	12.3%	17.1%	10.4%	37.5%	43.4%	9.0%	10.8%	12.8%	9.2%
stop sign	29.5%	30.7%	11.0%	31.2%	12.7%	29.1%	28.8%	31.4%	5.5%	31.1%	7.0%	27.8%
parking meter	17.4%	21.1%	2.6%	9.9%	3.3%	9.7%	19.8%	24.2%	1.6%	9.4%	1.9%	9.4%
bench	7.6%	7.5%	3.0%	4.4%	2.1%	4.0%	7.8%	8.3%	2.4%	4.9%	1.7%	4.7%
bird	13.2%	12.6%	11.5%	9.3%	10.7%	8.0%	15.4%	14.7%	8.8%	8.0%	8.1%	6.5%
cat	1.9%	2.3%	0.9%	6.7%	1.0%	7.6%	1.8%	2.3%	1.4%	7.5%	1.5%	8.4%
dog	8.9%	8.8%	2.5%	7.1%	2.8%	7.9%	10.8%	11.7%	2.9%	8.1%	3.4%	8.8%
horse	13.6%	11.6%	6.2%	5.3%	5.5%	7.0%	19.8%	17.9%	5.8%	4.9%	5.0%	6.7%
sheep	3.3%	3.6%	2.4%	10.8%	2.5%	11.6%	3.8%	4.1%	2.0%	9.7%	2.3%	11.1%
cow	15.6%	17.1%	8.3%	12.2%	8.7%	13.2%	18.3%	20.1%	7.0%	11.7%	7.2%	12.9%
elephant	9.0%	9.4%	1.3%	10.6%	1.4%	12.1%	10.7%	11.6%	1.3%	10.2%	1.4%	11.6%
bear	3.7%	5.1%	0.7%	11.2%	1.0%	10.7%	6.0%	8.4%	0.6%	10.4%	1.0%	10.1%
zebra	0.7%	0.6%	0.6%	12.3%	0.6%	15.3%	1.3%	1.2%	0.2%	10.4%	0.2%	13.1%
giraffe	14.3%	13.9%	5.4%	2.2%	5.1%	2.3%	17.4%	17.3%	5.7%	1.6%	5.5%	1.7%
backpack	4.3%	3.1%	1.3%	12.1%	0.8%	9.7%	6.7%	5.7%	2.0%	13.1%	1.3%	11.1%
umbrella	23.1%	21.7%	20.9%	13.9%	17.2%	12.8%	25.8%	23.4%	19.9%	11.3%	17.1%	11.0%
handbag	14.8%	13.0%	1.3%	6.3%	0.7%	4.7%	20.7%	18.2%	1.6%	6.1%	1.4%	4.1%
tie	35.1%	33.5%	1.2%	4.8%	1.8%	3.5%	36.1%	35.0%	0.9%	3.8%	1.1%	3.0%
suitcase	9.7%	10.2%	1.7%	11.1%	1.6%	8.7%	7.5%	8.3%	1.3%	12.5%	1.3%	9.9%
frisbee	25.5%	23.2%	4.9%	16.9%	3.2%	12.2%	18.7%	16.0%	1.6%	15.3%	0.6%	11.5%
skis	4.6%	3.9%	0.4%	0.4%	0.3%	0.4%	6.2%	5.8%	1.1%	0.7%	1.1%	0.5%
snowboard	28.9%	28.2%	2.6%	1.5%	3.6%	1.1%	32.5%	33.5%	3.2%	1.7%	6.1%	1.1%
sports ball	17.1%	17.3%	11.3%	16.0%	11.5%	13.1%	15.5%	16.0%	12.7%	13.6%	13.2%	11.4%
kite	29.4%	28.5%	34.8%	20.1%	33.8%	19.3%	32.4%	32.0%	34.5%	19.8%	33.4%	18.7%
baseball bat	28.4%	27.4%	8.1%	0.4%	7.1%	0.3%	33.8%	33.0%	4.1%	0.4%	2.6%	0.3%
baseball glove	29.1%	26.5%	1.9%	11.4%	1.7%	8.6%	35.6%	32.4%	1.9%	10.3%	1.5%	8.8%
skateboard	17.6%	15.8%	0.7%	1.4%	0.5%	1.3%	23.7%	22.4%	0.1%	1.3%	0.2%	1.2%
surfboard	26.4%	23.0%	3.5%	5.0%	1.2%	2.7%	31.4%	28.6%	4.2%	5.9%	2.6%	3.7%
tennis racket	18.9%	15.5%	1.6%	4.3%	0.4%	4.2%	23.3%	20.2%	0.6%	3.9%	0.3%	3.7%
bottle	14.1%	18.4%	9.3%	6.9%	15.7%	4.3%	17.3%	22.4%	8.4%	6.5%	13.3%	3.9%
wine glass	7.8%	8.6%	4.0%	4.1%	4.7%	5.0%	11.1%	11.9%	3.0%	4.1%	3.2%	5.5%
cup	12.0%	12.4%	2.8%	11.0%	2.4%	10.1%	11.5%	11.8%	2.7%	10.0%	2.1%	9.7%
fork	5.5%	5.1%	5.6%	0.6%	6.0%	0.7%	8.6%	8.4%	4.8%	0.5%	6.0%	0.6%
knife	8.3%	8.3%	3.5%	0.6%	3.3%	0.4%	11.8%	12.5%	4.9%	0.5%	5.3%	0.6%
spoon	8.4%	8.0%	0.1%	1.0%	0.2%	0.8%	12.0%	11.6%	0.8%	0.8%	0.8%	0.7%
bowl	10.2%	9.8%	2.7%	11.9%	1.9%	9.8%	11.0%	10.5%	2.6%	12.0%	2.2%	10.3%
banana	0.5%	0.2%	0.5%	14.8%	0.4%	12.4%	0.9%	0.5%	0.3%	11.8%	0.2%	9.9%
apple	0.3%	0.3%	1.3%	14.1%	1.5%	13.4%	0.4%	0.4%	0.9%	12.9%	1.0%	12.6%
sandwich	4.4%	4.7%	1.6%	12.3%	1.6%	12.5%	5.0%	5.4%	1.6%	12.5%	1.8%	12.2%
orange	0.3%	0.3%	0.5%	15.8%	0.5%	14.9%	0.2%	0.3%	0.3%	13.9%	0.3%	13.2%
broccoli	0.5%	0.5%	0.1%	13.9%	0.1%	12.7%	0.6%	0.7%	0.0%	9.8%	0.1%	8.9%
carrot	0.2%	0.2%	0.3%	6.3%	0.3%	5.8%	0.2%	0.2%	0.3%	5.3%	0.2%	4.4%
hot dog	1.2%	1.0%	1.1%	9.4%	1.1%	8.9%	1.4%	1.2%	1.0%	9.0%	1.1%	8.3%
pizza	3.4%	2.9%	1.0%	15.6%	0.8%	18.1%	3.5%	2.9%	1.4%	15.6%	1.1%	17.6%
donut	4.0%	3.9%	4.5%	19.7%	4.9%	17.9%	3.2%	3.2%	3.0%	18.2%	3.4%	16.7%
cake	21.7%	22.1%	6.6%	16.1%	6.1%	14.4%	22.6%	23.4%	6.4%	16.0%	5.9%	14.6%
chair	5.9%	6.3%	2.5%	7.4%	2.8%	6.5%	7.2%	7.6%	2.7%	8.2%	3.2%	7.6%
couch	4.0%	5.7%	0.7%	5.8%	1.2%	6.4%	3.6%	5.1%	1.2%	6.6%	1.8%	7.4%
potted plant	1.6%	1.5%	1.1%	10.1%	0.9%	10.3%	2.0%	1.9%	1.4%	9.8%	1.3%	9.8%
bed	4.7%	4.5%	3.4%	7.6%	3.0%	7.3%	3.9%	3.3%	4.7%	9.4%	4.3%	8.9%
dining table	3.4%	3.1%	2.1%	5.3%	1.4%	4.9%	4.2%	3.8%	3.1%	7.5%	2.3%	6.7%
toilet	5.0%	5.3%	2.0%	11.5%	2.2%	11.1%	6.4%	7.1%	2.4%	11.5%	2.4%	11.1%
tv	13.1%	14.5%	2.2%	13.9%	2.7%	11.3%	16.6%	18.5%	2.5%	16.4%	3.0%	13.9%
laptop	3.7%	4.3%	1.1%	8.4%	1.1%	7.0%	5.0%	5.9%	1.6%	8.6%	1.5%	8.0%
mouse	23.1%	24.2%	0.7%	11.7%	0.5%	8.3%	23.6%	25.2%	1.5%	13.8%	0.8%	9.9%
remote	12.0%	11.7%	0.9%	5.8%	0.6%	6.0%	13.3%	13.0%	1.2%	6.7%	0.7%	6.3%
keyboard	4.9%	3.8%	0.4%	6.8%	0.4%	11.2%	4.8%	4.8%	1.8%	8.3%	1.2%	13.2%
cell phone	13.9%	13.7%	0.5%	9.3%	0.3%	7.6%	15.3%	15.0%	0.5%	9.3%	0.2%	7.0%
microwave	4.1%	5.7%	0.9%	11.7%	1.2%	9.3%	6.7%	8.8%	0.9%	13.0%	1.1%	10.6%
oven	3.1%	3.0%	1.1%	7.0%	1.2%	6.3%	4.7%	4.4%	1.6%	9.0%	1.6%	8.4%
toaster	25.0%	26.8%	0.0%	4.9%	0.0%	3.0%	33.7%	35.8%	0.0%	6.5%	0.0%	4.3%
sink	8.8%	9.8%	0.1%	5.3%	0.1%	5.2%	10.8%	13.1%	0.2%	7.0%	0.4%	7.1%
refrigerator	14.5%	12.4%	3.5%	8.5%	1.9%	8.9%	13.9%	12.9%	3.6%	8.7%	2.8%	9.3%
book	8.0%	6.2%	4.7%	2.5%	2.6%	1.6%	8.6%	7.2%	3.3%	3.5%	1.7%	2.0%
clock	13.4%	10.6%	2.1%	18.8%	1.2%	15.1%	14.8%	12.5%	2.2%	20.5%	1.0%	16.7%
vase	21.3%	17.8%	6.0%	11.1%	6.5%	8.9%	25.1%	22.8%	6.2%	11.6%	6.3%	8.9%
scissors	0.9%	0.4%	0.9%	2.6%	0.7%	2.0%	2.0%	1.5%	0.7%	2.5%	0.5%	2.2%
teddy bear	2.1%	2.1%	6.4%	11.8%	7.0%	14.1%	1.8%	2.0%	6.5%	12.1%	7.5%	15.7%
hair drier	4.6%	4.7%	0.2%	1.0%	0.0%	0.7%	4.7%	4.9%	0.2%	2.5%	0.4%	1.6%
toothbrush	8.7%	7.1%	5.7%	1.9%	6.4%	1.7%	11.1%	9.7%	0.3%	1.1%	0.2%	1.4%

Table A.7: The per class breakdown of maskAP on **Swin-T** and **CondInst** and multiple ContentSwap experiment and evaluation modes. Bolded experiments in the table are the more interesting results.

	Swin-T						CondInst					
	ContentSwapOriginal-Morph-ShapeAP	*ContentSwapOriginal-Aperct-ShapeAP*	**ContentSwapIsolation-Morph-ShapeAP**	**ContentSwapIsolation-Morph-ContentAP**	*ContentSwapIsolation-Aperct-ShapeAP*	*ContentSwapIsolation-Aperct-ContentAP*	**ContentSwapOriginal-Morph-ShapeAP**	*ContentSwapOriginal-Aperct-ShapeAP*	**ContentSwapIsolation-Morph-ShapeAP**	**ContentSwapIsolation-Morph-ContentAP**	*ContentSwapIsolation-Aperct-ShapeAP*	*ContentSwapIsolation-Aperct-ContentAP*
person	19.2%	21.2%	8.4%	10.5%	9.0%	10.2%	18.4%	19.7%	11.4%	14.9%	11.7%	14.6%
bicycle	0.6%	0.3%	0.1%	5.0%	0.1%	3.8%	0.9%	0.5%	1.2%	11.2%	0.9%	8.7%
car	6.3%	5.0%	2.2%	16.9%	1.3%	14.8%	8.1%	6.6%	5.3%	24.8%	5.0%	22.5%
motorcycle	4.2%	3.4%	0.3%	10.7%	0.4%	9.7%	4.0%	3.3%	2.1%	13.9%	1.8%	12.1%
airplane	26.9%	21.7%	27.9%	10.0%	21.5%	9.4%	28.7%	24.7%	21.4%	10.5%	18.6%	9.5%
bus	8.5%	7.4%	1.3%	24.6%	1.1%	23.8%	10.7%	10.8%	3.4%	30.6%	3.9%	28.7%
train	22.9%	22.3%	2.9%	10.9%	2.2%	10.0%	26.3%	27.8%	8.0%	20.2%	8.5%	17.4%
truck	7.0%	8.0%	4.4%	22.6%	3.6%	19.8%	10.9%	11.2%	8.3%	27.3%	9.4%	24.8%
boat	20.5%	16.1%	5.2%	13.1%	3.1%	10.8%	22.2%	17.4%	13.4%	19.7%	9.8%	16.4%
traffic light	21.4%	21.6%	10.4%	25.1%	9.6%	21.3%	16.4%	17.4%	29.3%	30.9%	31.4%	30.9%
fire hydrant	31.5%	36.9%	6.6%	12.0%	8.0%	9.7%	38.5%	41.3%	17.7%	18.1%	20.9%	15.4%
stop sign	24.8%	27.5%	5.2%	30.1%	7.3%	27.0%	28.8%	31.9%	17.0%	34.7%	18.4%	32.3%
parking meter	20.2%	24.0%	1.0%	13.0%	1.6%	13.5%	20.9%	23.1%	7.4%	17.4%	7.7%	18.3%
bench	7.8%	8.6%	2.0%	4.0%	1.7%	3.8%	9.3%	10.1%	7.1%	8.4%	7.4%	8.7%
bird	16.6%	16.0%	8.6%	9.8%	8.3%	8.2%	8.6%	7.6%	10.6%	12.8%	9.6%	10.8%
cat	1.8%	2.3%	1.1%	8.2%	1.4%	9.1%	7.5%	9.7%	3.6%	12.8%	4.1%	13.6%
dog	8.6%	9.2%	3.0%	8.5%	3.5%	9.4%	10.9%	11.7%	5.3%	14.3%	6.5%	15.5%
horse	17.2%	16.6%	7.9%	5.7%	7.7%	7.9%	16.5%	16.1%	5.5%	9.4%	5.6%	12.2%
sheep	3.2%	3.5%	2.0%	8.9%	2.2%	9.7%	5.5%	5.5%	5.2%	16.0%	5.2%	18.0%
cow	19.2%	20.0%	9.1%	12.5%	9.6%	12.8%	14.4%	15.8%	9.2%	15.3%	9.7%	16.9%
elephant	6.1%	7.3%	1.0%	10.5%	1.0%	12.1%	8.9%	11.1%	2.7%	15.9%	3.2%	19.2%
bear	4.6%	6.6%	1.8%	10.5%	2.1%	10.3%	9.2%	11.5%	7.6%	23.8%	10.2%	25.5%
zebra	1.0%	1.1%	0.3%	10.4%	0.3%	13.3%	2.0%	1.8%	0.6%	13.7%	0.5%	17.1%
giraffe	17.5%	17.8%	8.4%	2.1%	8.3%	2.2%	20.9%	19.6%	10.4%	3.6%	8.9%	3.5%
backpack	4.6%	4.0%	0.5%	9.7%	0.6%	8.1%	5.3%	4.2%	2.6%	16.9%	2.6%	14.9%
umbrella	24.7%	22.2%	21.7%	12.9%	17.1%	12.1%	25.1%	22.0%	23.0%	18.5%	18.0%	17.1%
handbag	14.8%	12.0%	1.0%	5.1%	0.8%	3.3%	16.1%	13.7%	2.4%	9.6%	1.9%	7.4%
tie	36.0%	34.3%	0.6%	4.7%	1.3%	3.2%	32.1%	32.3%	1.4%	7.4%	1.6%	7.5%
suitcase	5.2%	5.8%	0.8%	11.4%	0.7%	8.5%	9.8%	10.2%	3.2%	16.1%	3.5%	14.6%
frisbee	25.9%	22.8%	4.8%	14.8%	3.8%	10.3%	32.4%	30.9%	18.2%	22.8%	17.3%	18.2%
skis	10.3%	9.2%	0.1%	0.5%	0.1%	0.4%	5.6%	4.8%	2.0%	1.8%	2.1%	1.7%
snowboard	33.8%	34.7%	1.8%	1.4%	3.4%	0.9%	30.9%	30.0%	6.7%	3.7%	7.0%	3.6%
sports ball	20.2%	20.8%	11.8%	13.7%	11.7%	11.3%	18.5%	18.7%	17.2%	19.4%	17.8%	17.6%
kite	36.3%	35.8%	46.6%	26.6%	46.7%	25.6%	24.3%	23.7%	30.2%	26.8%	28.8%	25.1%
baseball bat	37.9%	35.5%	1.8%	0.1%	1.0%	0.1%	31.4%	28.5%	10.5%	1.7%	8.4%	1.3%
baseball glove	30.3%	25.6%	2.4%	9.6%	1.4%	7.7%	35.3%	33.4%	2.9%	11.5%	2.1%	10.0%
skateboard	26.3%	24.7%	0.3%	1.9%	0.5%	1.8%	24.0%	21.9%	2.2%	3.3%	1.7%	3.7%
surfboard	35.2%	31.5%	2.6%	7.0%	1.1%	3.5%	29.3%	26.2%	8.4%	13.5%	6.3%	9.8%
tennis racket	26.3%	24.2%	0.5%	4.6%	0.3%	4.1%	27.0%	22.4%	3.0%	8.9%	1.8%	9.4%
bottle	14.8%	18.6%	7.0%	7.1%	10.1%	3.9%	16.8%	19.7%	10.1%	10.5%	12.3%	6.2%
wine glass	7.9%	8.2%	2.7%	4.6%	2.7%	6.3%	8.3%	8.4%	2.5%	7.7%	3.1%	8.8%
cup	9.8%	11.3%	1.7%	10.3%	1.6%	9.2%	11.6%	10.9%	4.2%	14.2%	3.6%	13.1%
fork	10.7%	9.8%	2.4%	0.3%	3.1%	0.5%	9.1%	8.5%	4.5%	1.3%	4.6%	1.4%
knife	16.4%	17.5%	1.3%	0.3%	1.2%	0.4%	15.0%	15.0%	5.3%	1.8%	4.3%	1.4%
spoon	13.1%	13.1%	0.5%	0.6%	1.4%	0.6%	10.4%	9.8%	0.3%	1.6%	0.3%	1.4%
bowl	13.5%	13.0%	2.0%	11.3%	1.2%	9.9%	11.0%	10.6%	7.5%	15.2%	5.4%	13.4%
banana	0.4%	0.2%	0.2%	11.8%	0.1%	9.8%	1.1%	0.9%	1.4%	17.5%	1.1%	14.7%
apple	0.4%	0.4%	1.2%	15.2%	1.3%	15.2%	0.8%	0.9%	3.5%	16.5%	3.7%	15.9%
sandwich	5.4%	5.5%	1.7%	14.7%	1.4%	14.5%	8.2%	9.3%	2.4%	16.5%	2.5%	16.9%
orange	0.4%	0.4%	0.2%	15.4%	0.2%	14.2%	0.9%	0.9%	2.3%	24.2%	2.1%	22.4%
broccoli	0.5%	0.5%	0.0%	14.6%	0.0%	13.1%	1.0%	1.1%	0.2%	18.0%	0.3%	17.1%
carrot	0.3%	0.3%	0.1%	5.1%	0.1%	4.4%	0.3%	0.3%	0.5%	7.9%	0.7%	6.4%
hot dog	1.7%	1.3%	0.6%	8.8%	0.6%	8.7%	3.5%	3.7%	2.1%	12.4%	2.5%	12.7%
pizza	4.1%	3.0%	1.4%	19.2%	1.3%	22.0%	7.6%	8.0%	3.5%	22.9%	3.3%	24.5%
donut	1.7%	1.8%	1.6%	16.8%	1.8%	15.7%	4.0%	4.2%	3.3%	20.1%	4.1%	18.4%
cake	22.5%	23.8%	5.8%	17.1%	5.3%	16.3%	21.7%	22.0%	9.1%	21.7%	8.5%	19.7%
chair	6.2%	7.2%	1.2%	4.2%	1.1%	3.9%	9.0%	9.3%	6.9%	10.0%	6.7%	9.1%
couch	5.8%	8.6%	0.3%	5.9%	0.2%	6.3%	9.0%	12.1%	2.0%	8.4%	2.5%	9.4%
potted plant	1.5%	1.5%	0.9%	9.2%	0.6%	9.9%	2.0%	2.1%	5.3%	14.5%	5.3%	15.1%
bed	4.2%	3.8%	2.7%	7.6%	1.8%	7.3%	7.0%	7.1%	6.3%	13.0%	6.7%	13.0%
dining table	5.5%	5.4%	1.3%	5.9%	0.8%	4.7%	5.5%	5.5%	3.6%	9.3%	3.0%	8.9%
toilet	7.3%	7.9%	1.3%	10.4%	1.3%	9.6%	10.1%	11.2%	3.6%	15.1%	4.1%	14.8%
tv	16.9%	18.9%	2.4%	17.3%	2.8%	14.3%	14.7%	17.6%	9.3%	20.2%	10.8%	17.7%
laptop	4.5%	5.4%	0.7%	7.9%	1.0%	7.1%	6.8%	8.1%	3.7%	11.5%	4.1%	10.0%
mouse	21.6%	22.9%	0.3%	9.0%	0.1%	5.9%	26.3%	26.5%	1.3%	12.4%	1.1%	8.0%
remote	13.0%	12.8%	0.4%	5.8%	0.4%	5.6%	12.0%	11.4%	0.9%	8.8%	0.9%	8.5%
keyboard	4.9%	3.4%	0.4%	7.3%	0.1%	10.0%	9.9%	8.0%	4.0%	10.8%	2.2%	16.0%
cell phone	19.9%	20.3%	0.3%	9.7%	0.2%	7.3%	15.8%	15.1%	1.0%	14.3%	1.0%	11.8%
microwave	7.8%	9.2%	0.9%	13.7%	0.8%	10.9%	6.2%	7.6%	3.5%	18.3%	3.5%	15.1%
oven	4.0%	3.5%	1.0%	8.4%	0.6%	7.5%	5.7%	5.7%	2.7%	10.8%	3.1%	10.6%
toaster	28.8%	30.6%	0.1%	6.2%	0.2%	5.8%	17.8%	20.9%	2.2%	9.8%	2.2%	8.6%
sink	12.0%	13.6%	0.1%	6.4%	0.1%	5.8%	11.2%	11.7%	2.0%	11.7%	2.6%	10.1%
refrigerator	16.2%	14.0%	1.8%	8.3%	1.1%	9.1%	19.5%	18.6%	7.0%	13.7%	7.7%	15.0%
book	6.2%	5.1%	3.4%	2.6%	1.2%	1.7%	6.6%	4.6%	5.7%	5.9%	2.8%	4.3%
clock	12.9%	9.5%	2.9%	23.2%	1.6%	19.2%	15.4%	13.5%	7.0%	25.2%	5.9%	22.1%
vase	22.7%	20.2%	6.8%	13.3%	7.6%	9.9%	19.2%	16.5%	7.5%	16.1%	7.3%	13.3%
scissors	3.6%	2.3%	0.9%	2.6%	0.6%	2.5%	4.8%	4.6%	2.1%	4.4%	1.9%	3.4%
teddy bear	0.9%	1.2%	2.3%	11.7%	3.1%	14.9%	2.3%	2.6%	8.8%	17.4%	9.2%	21.2%
hair drier	6.2%	7.5%	0.0%	0.2%	0.0%	0.2%	15.4%	17.5%	1.6%	3.7%	0.8%	3.3%
toothbrush	14.1%	12.0%	4.0%	0.9%	5.0%	1.2%	14.4%	12.9%	2.9%	2.3%	2.2%	2.8%

Table A.8: The per class breakdown of maskAP on **MEInst** and **SOLO** and multiple ContentSwap experiment and evaluation modes. Bolded experiments in the table are the more interesting results.

	MEInst						SOLO					
	ContentSwapOriginal-Morph-ShapeAP	*ContentSwapOriginal-Aspect-ShapeAP*	**ContentSwapIsolation-Morph-ShapeAP**	**ContentSwapIsolation-Morph-ContentAP**	*ContentSwapIsolation-Aspect-ShapeAP*	*ContentSwapIsolation-Aspect-ContentAP*	**ContentSwapOriginal-Morph-ShapeAP**	*ContentSwapOriginal-Aspect-ShapeAP*	**ContentSwapIsolation-Morph-ShapeAP**	**ContentSwapIsolation-Morph-ContentAP**	*ContentSwapIsolation-Aspect-ShapeAP*	*ContentSwapIsolation-Aspect-ContentAP*
person	15.4%	16.0%	11.0%	14.5%	11.1%	14.2%	16.8%	17.4%	13.0%	15.3%	12.5%	15.9%
bicycle	1.2%	0.7%	1.0%	8.2%	0.9%	6.1%	1.2%	0.6%	1.1%	9.4%	0.6%	7.3%
car	9.2%	8.0%	8.4%	20.9%	7.8%	19.2%	8.8%	7.6%	6.5%	23.1%	5.6%	20.8%
motorcycle	3.3%	2.9%	5.0%	13.0%	4.0%	11.1%	4.2%	3.5%	1.6%	15.3%	1.2%	13.8%
airplane	14.3%	11.5%	12.8%	11.7%	11.2%	10.8%	25.0%	19.8%	23.7%	15.3%	18.5%	14.4%
bus	8.3%	8.1%	5.5%	27.7%	5.8%	25.4%	5.8%	5.5%	2.0%	29.7%	1.6%	28.4%
train	14.0%	15.6%	8.0%	16.5%	8.6%	14.7%	19.7%	19.4%	3.4%	16.6%	2.7%	15.0%
truck	7.8%	8.1%	9.5%	21.6%	9.9%	19.5%	7.7%	8.6%	12.3%	27.5%	11.6%	25.8%
boat	15.4%	12.7%	8.2%	11.8%	6.6%	9.9%	20.3%	15.9%	12.5%	21.0%	7.2%	16.7%
traffic light	19.2%	19.7%	35.5%	28.2%	35.4%	27.5%	14.8%	15.6%	20.4%	30.1%	22.6%	28.2%
fire hydrant	35.4%	38.7%	26.5%	15.7%	27.8%	14.0%	32.6%	34.3%	15.8%	14.7%	18.5%	13.2%
stop sign	26.7%	28.8%	31.4%	37.9%	32.9%	35.5%	20.3%	22.4%	10.0%	36.4%	12.7%	34.5%
parking meter	19.6%	22.2%	10.0%	20.4%	9.0%	19.3%	19.9%	21.7%	3.5%	12.7%	3.8%	13.4%
bench	4.7%	5.1%	3.0%	5.3%	3.1%	5.3%	8.5%	9.4%	6.1%	10.0%	6.3%	9.8%
bird	10.6%	10.2%	11.9%	12.0%	11.6%	9.9%	11.4%	10.6%	10.2%	12.8%	9.4%	11.0%
cat	4.5%	5.0%	3.7%	11.4%	4.4%	11.6%	3.0%	3.4%	2.3%	10.4%	3.0%	11.7%
dog	10.8%	10.8%	7.5%	12.1%	7.6%	13.1%	10.8%	11.5%	4.4%	11.4%	5.2%	12.4%
horse	10.7%	9.6%	6.8%	7.7%	6.1%	9.8%	16.5%	14.3%	6.7%	8.5%	5.7%	11.2%
sheep	5.6%	6.1%	5.6%	14.5%	5.7%	15.7%	6.5%	6.7%	8.1%	20.4%	8.0%	22.0%
cow	16.6%	18.1%	12.1%	17.1%	12.4%	17.4%	16.4%	17.7%	12.3%	16.3%	12.7%	18.0%
elephant	12.3%	12.7%	5.2%	14.5%	5.7%	16.9%	10.9%	11.9%	2.4%	15.3%	2.8%	17.9%
bear	14.8%	17.4%	8.1%	18.5%	8.7%	18.6%	7.8%	10.9%	2.4%	17.3%	3.9%	18.3%
zebra	1.9%	1.6%	1.1%	10.7%	1.1%	13.1%	1.3%	1.0%	1.3%	15.1%	1.1%	18.0%
giraffe	14.7%	14.4%	8.7%	2.8%	7.9%	2.7%	10.7%	10.7%	5.0%	2.6%	4.8%	2.8%
backpack	4.7%	3.8%	3.3%	12.7%	2.5%	11.3%	3.3%	2.7%	1.8%	14.3%	0.9%	12.3%
umbrella	20.5%	18.7%	24.1%	20.5%	20.8%	20.1%	22.0%	19.1%	23.9%	19.5%	19.6%	17.9%
handbag	14.7%	13.0%	7.3%	11.6%	5.9%	9.1%	16.0%	13.1%	2.3%	9.2%	1.8%	7.2%
tie	25.6%	24.8%	11.1%	17.2%	9.4%	13.7%	28.0%	26.8%	2.0%	7.2%	2.3%	6.8%
suitcase	9.4%	10.1%	7.4%	20.3%	7.9%	17.5%	8.4%	8.7%	2.3%	16.7%	2.2%	14.3%
frisbee	33.0%	31.8%	25.6%	25.2%	25.0%	20.5%	30.9%	28.8%	4.5%	17.7%	3.3%	14.0%
skis	1.1%	1.1%	0.5%	2.2%	0.5%	1.8%	5.5%	4.8%	1.9%	2.3%	2.1%	1.7%
snowboard	18.5%	19.0%	15.1%	6.5%	14.1%	5.0%	28.5%	27.8%	4.8%	4.4%	6.6%	3.3%
sports ball	19.9%	20.0%	19.0%	20.1%	18.7%	18.0%	17.2%	17.2%	11.9%	19.2%	11.5%	17.8%
kite	23.4%	22.7%	30.0%	22.0%	29.3%	21.0%	24.5%	23.7%	30.2%	24.8%	29.5%	23.0%
baseball bat	20.9%	21.1%	27.5%	2.6%	27.6%	2.6%	35.3%	32.8%	15.6%	1.6%	16.7%	1.4%
baseball glove	30.5%	27.9%	17.7%	17.3%	15.0%	16.2%	32.5%	29.7%	4.4%	15.2%	3.3%	13.4%
skateboard	15.4%	14.0%	9.2%	7.4%	9.3%	7.0%	24.7%	22.4%	2.7%	3.5%	2.8%	3.1%
surfboard	20.1%	18.1%	9.0%	10.2%	8.2%	8.0%	25.6%	22.1%	6.0%	11.5%	4.3%	7.7%
tennis racket	26.2%	22.2%	6.9%	7.0%	5.4%	7.2%	26.9%	22.4%	3.5%	7.6%	1.4%	7.4%
bottle	15.2%	18.1%	18.2%	9.5%	20.1%	6.9%	13.6%	15.0%	11.2%	11.9%	14.7%	7.7%
wine glass	10.2%	10.3%	5.6%	5.8%	5.2%	7.4%	8.2%	8.0%	3.9%	6.4%	3.8%	8.5%
cup	9.2%	9.5%	5.2%	12.2%	4.9%	11.5%	11.8%	11.3%	5.4%	16.4%	4.2%	15.6%
fork	5.3%	4.9%	5.0%	1.3%	5.7%	1.6%	10.6%	9.3%	6.0%	1.6%	6.5%	1.6%
knife	8.2%	7.4%	8.0%	3.2%	7.3%	2.8%	11.2%	10.9%	4.4%	1.8%	5.1%	1.4%
spoon	6.8%	6.4%	1.9%	2.7%	1.7%	2.2%	8.4%	7.4%	0.3%	1.8%	0.3%	1.3%
bowl	10.1%	9.9%	5.0%	12.4%	4.7%	10.8%	9.3%	8.8%	7.4%	15.6%	5.4%	13.7%
banana	1.0%	0.7%	3.4%	16.6%	3.2%	13.9%	0.9%	0.6%	1.9%	17.8%	1.5%	14.9%
apple	0.4%	0.5%	6.1%	13.1%	5.6%	12.3%	0.5%	0.6%	1.7%	16.8%	1.7%	16.5%
sandwich	5.3%	6.4%	5.3%	17.1%	5.5%	17.1%	5.0%	5.3%	2.6%	18.6%	2.6%	18.4%
orange	0.5%	0.6%	4.2%	20.9%	4.4%	20.0%	0.6%	0.7%	1.1%	22.8%	1.4%	22.1%
broccoli	1.2%	1.2%	0.7%	17.8%	0.8%	16.2%	1.2%	1.2%	0.2%	20.0%	0.3%	18.1%
carrot	0.2%	0.2%	0.7%	8.3%	0.6%	7.8%	0.3%	0.3%	0.5%	10.3%	0.4%	9.4%
hot dog	2.1%	1.9%	2.9%	11.8%	2.8%	12.3%	3.0%	2.6%	1.3%	12.3%	1.2%	11.5%
pizza	4.4%	3.6%	6.0%	21.1%	4.5%	22.5%	4.0%	3.3%	1.9%	18.6%	1.3%	21.3%
donut	3.6%	3.6%	8.5%	24.6%	8.7%	21.8%	4.1%	4.1%	4.2%	23.0%	4.7%	21.5%
cake	19.1%	19.8%	17.6%	26.1%	18.3%	24.3%	21.8%	21.7%	12.8%	23.2%	11.7%	21.7%
chair	6.8%	7.8%	2.9%	7.0%	2.9%	6.8%	8.4%	8.6%	6.3%	11.3%	6.5%	11.2%
couch	5.3%	7.4%	2.0%	7.7%	3.2%	9.5%	7.3%	9.3%	2.3%	10.9%	3.7%	12.4%
potted plant	1.3%	1.4%	3.5%	12.0%	3.6%	12.4%	1.8%	1.7%	3.1%	15.4%	2.5%	15.7%
bed	4.4%	4.4%	5.7%	14.4%	6.2%	14.0%	7.0%	5.8%	7.4%	17.7%	6.0%	17.0%
dining table	4.7%	4.9%	3.7%	8.2%	3.1%	7.0%	5.4%	5.3%	5.0%	10.4%	5.0%	9.4%
toilet	8.1%	8.4%	7.3%	17.0%	7.4%	17.8%	6.7%	7.5%	7.4%	19.0%	6.9%	18.8%
tv	14.3%	15.4%	8.6%	20.0%	8.8%	18.1%	10.9%	11.9%	4.1%	22.5%	4.5%	21.2%
laptop	5.3%	5.5%	2.7%	9.4%	2.8%	8.3%	4.2%	5.2%	2.0%	10.9%	2.4%	9.5%
mouse	23.6%	24.5%	1.7%	13.9%	1.5%	10.0%	25.9%	25.5%	0.9%	14.7%	0.8%	10.6%
remote	12.2%	11.4%	5.2%	12.0%	4.7%	10.9%	11.6%	11.1%	2.4%	12.6%	2.3%	11.1%
keyboard	6.5%	5.3%	5.1%	10.0%	2.6%	13.2%	6.7%	4.0%	1.9%	10.9%	1.7%	15.3%
cell phone	13.7%	13.0%	4.1%	16.6%	3.7%	14.0%	15.3%	14.5%	1.1%	14.4%	1.6%	12.0%
microwave	6.0%	7.9%	8.5%	18.6%	9.4%	15.7%	7.2%	8.7%	2.1%	16.4%	2.5%	13.2%
oven	3.6%	3.7%	5.2%	10.3%	5.0%	9.6%	5.2%	5.2%	3.3%	12.9%	3.8%	12.2%
toaster	30.1%	30.4%	10.8%	12.9%	9.0%	11.8%	28.5%	30.7%	1.2%	4.3%	0.9%	3.0%
sink	8.6%	9.5%	3.9%	9.5%	3.7%	9.3%	13.4%	13.1%	10.3%	14.4%	7.9%	12.9%
refrigerator	12.2%	10.4%	10.2%	13.9%	8.4%	15.2%	16.6%	12.8%	9.8%	13.8%	7.1%	15.1%
book	4.4%	3.1%	3.4%	4.4%	2.7%	2.8%	2.8%	2.0%	2.2%	6.4%	1.7%	4.6%
clock	13.6%	11.6%	16.1%	25.6%	13.4%	21.1%	16.3%	14.7%	8.8%	26.0%	6.9%	22.0%
vase	20.3%	18.5%	21.3%	18.3%	18.8%	16.2%	19.6%	17.3%	13.5%	21.8%	11.5%	18.1%
scissors	2.0%	1.6%	6.7%	8.8%	5.5%	8.0%	2.4%	1.8%	4.9%	7.2%	3.4%	5.7%
teddy bear	2.1%	2.3%	15.1%	20.4%	14.6%	23.7%	2.7%	2.9%	9.5%	20.2%	10.3%	25.2%
hair drier	13.7%	14.2%	15.9%	8.9%	13.8%	8.3%	5.5%	5.7%	1.5%	4.3%	1.8%	3.6%
toothbrush	5.8%	4.9%	9.8%	6.6%	8.6%	7.2%	10.8%	8.6%	11.1%	7.4%	10.4%	7.3%

Appendix B

Appendix B : The Topology and Language of Relationships in the Visual Genome Dataset

B.1 Full Figures

In this appendix we provide the full figure for the inverse relationship analysis on all 50 relationship predicates, see Figure B.1. We also provide the full topological analysis in Figure B.2 and the directional analysis in Figure B.3. These figures are best viewed on a screen where you can zoom into them.

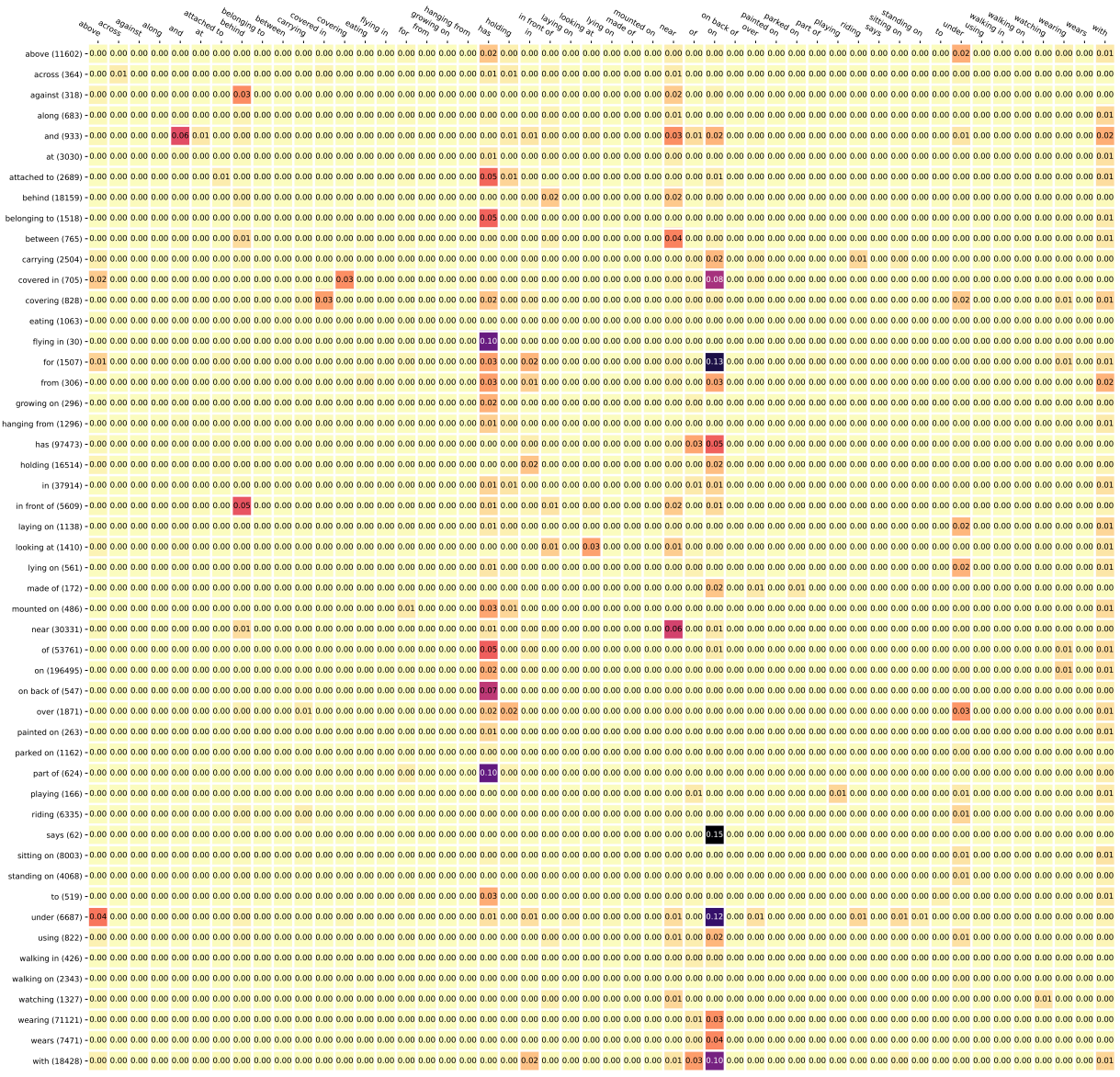


Figure B.1: A heatmap of the occurrence of inverse relationships for all 50 predicates in the VG200 dataset. The numbers in parentheses are the total occurrences for this predicate in the dataset. For every row the value in the heat map reflects the ratio of: (inverse relationship occurrences of the row predicate with the predicate in the column) to (total occurrences of the predicate in the row).

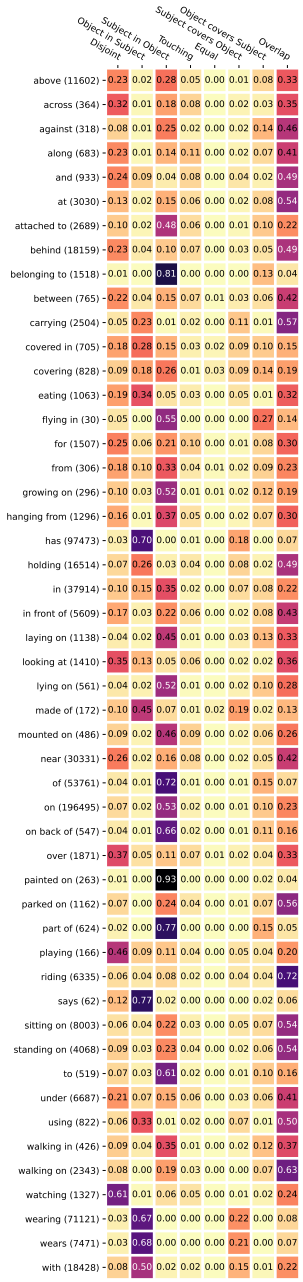


Figure B.2: A heatmap of the occurrence of topological relationships between bounding boxes related by all 50 predicates in the VG200 dataset. The values shown in the heatmap are the portions of the total occurrences of the row predicate that exhibit the specific topological configuration in the column. The values in parenthesis next to the predicate names are the total occurrences of that predicate.

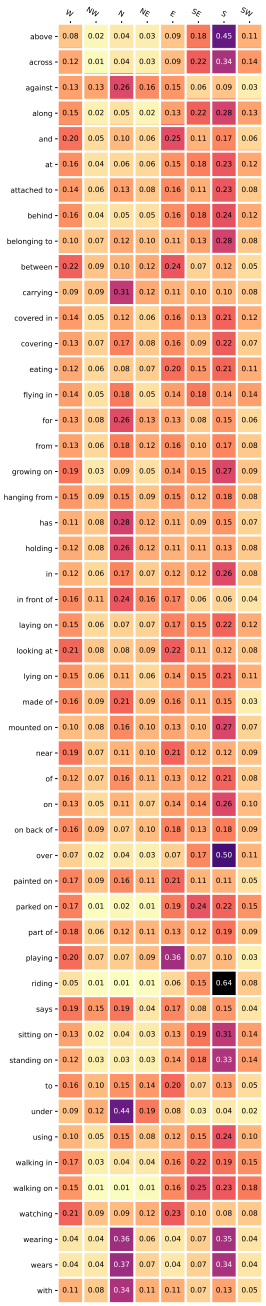


Figure B.3: A heatmap of the angles between subject and object for all 50 relationship predicates in the VG200 dataset. The values shown in the heatmap are the portions of the total occurrences of the row predicate that exhibit the specific directional configuration in the column.