Aristotle University of Thessaloniki
Faculty of Engineering
School of Electrical and Computer Engineering

Scene Graph Generation using Message Passing Neural Networks and Graph Convolutional Networks

Diploma Thesis

Miltiadis Kofinas

SRN: 7458

Thessaloniki, October 2018

Supervisors:

Anastasios Delopoulos, Associate Professor
Christos Diou, Postdoctoral Research Associate

Abstract

Holistic image interpretation constitutes a long-studied problem in computer vision that extends well beyond object detection. Visual relationships comprise a means to that end, capturing a wide variety of interactions between pairs of objects in an image. In this work, we model objects and their relationships using scene graphs, a visually-grounded graphical structure of an image's semantic information. We propose an end-to-end model that generates such scene representations, by incorporating a message passing scheme that propagates contextual information between objects and their relationships to iteratively refine its predictions. We experiment on a variety of message passing propagation architectures, including a modified version of a Graph Convolutional Network. Furthermore, we propose a very simple yet effective relationship pruning network that learns to identify and dismiss unlikely relationships. We report our performance on scene graph generation and other auxiliary evaluation tasks using Visual Genome dataset, outperforming related methods.

Acknowledgements

The completion of this work was a challenge on a personal level, since it required laborious effort and the combination of a wide range of knowledge acquired during my studies. Its completion would not have been successful without the contribution of some people, whom I would like to thank deeply.

First and foremost, I would like to thank my supervisor, associate professor Dr. Anastasios Delopoulos for the opportunity to work with him and for our excellent collaboration. I would also like to thank the postdoctoral research associate Dr. Christos Diou for his valuable contribution to completing this work, through his ideas and observations and the guidance he offered me.

I would like to give special thanks to the supervising professors of P.A.N.D.O.R.A. robotics team, associate professor Dr. Loukas Petrou, associate professor Dr. Andreas Symeonidis and lecturer Dr. Charalampos Dimoulas. This team has been my academic cradle, since I was taught how to experiment and conduct research in a real world setting. Hence, as a tribute, I would like to informally name the product of this work Pyrrha, daughter of the mythological Pandora and lone survivor of the great Deluge along with her husband Deucalion, in hope that this work will mark a new beginning for mankind through artificial intelligence.

Needless to say, none of the above could have been accomplished without the support of my family throughout my studies, for which I am truly grateful.

Last but not least, I would like to thank my friends for their help and for everything they have offered me in my development as a human being.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introductory Notes

Holistic image interpretation constitutes the holy grail of computer vision. Over the past few years, advances in machine learning along with the rapidly evolving computer hardware have been the cornerstone of progress in the fields of computer vision and image understanding. Image recognition [39, 61, 24, 25] and object detection [20, 19, 56] constitute remarkable examples of that progress, reaching extremely high levels of reasoning through the integration of deep learning and convolutional neural networks.

The construction of increasingly richer and thoroughly annotated datasets has been a crucial factor for this progress. These datasets are necessary for deep learning systems to function, since they require great amounts of data to be trained properly and in order to avoid overfitting. PASCAL Visual Object Classes (PASCAL VOC) [15], Microsoft Common Objects in Context (MS COCO) [44] and ImageNet [57] constitute typical examples of such datasets.

However, despite the progress of both deep learning systems and datasets, object detection and instance segmentation remain the main focus of interest in the field of computer vision. These tasks, albeit of great scientific interest fail to interpret the semantic content of an image in a holistic manner and instead focus on detecting objects using bounding boxes or object masks.

In order to fully understand the content of an image, though, one has not only to detect the objects in it, but to detect the relationships between them, as well as the attributes that describe them. This ensemble constitutes a visual scene graph [32], a compact mathematical representation that can describe an image in a holistic view. Such a representation would allow a computer system to reason about the visual world as well as allow for human-computer communication in a cognitive level, since the computer could describe an image using natural language and reason about it, answering questions related to its content. Thus, it can be used in cognitive visual tasks, such as image captioning or visual question answering. Finally, scene graphs can be used to enhance semantic image retrieval systems performance [32].

The lack of datasets that focus on such cognitive level tasks was, until recently, a major obstacle towards this direction. Visual Genome [38] was the first dataset to incorporate cognitive tasks in image understanding. It was followed, among others, by CLEVR [30] and more recently, Google Open Images [37].

Visual scene graphs, a compact mathematical representation of the visual world are of high importance towards semantic image understanding since they summarize cognitive level information. Thus, one the one hand, their generation can be viewed as a main task, for example in the problem of action recognition, while, on the other hand, they can function as an intermediate representation in tasks that involve human interaction through natural language, as are for example the problems of image captioning and visual question answering.

1.2 Problem Statement

In this work we explore the visual scene graph generation problem. More specifically, we focus on the visual relationship detection problem. Our goal is to detect objects, i.e. localize objects in an image using bounding boxes and classify them, as well as to detect the predicates that connect object pairs. Object and predicate classification is performed using a predetermined set of classes, including a background class that suggests an object's absence or the lack of a relationship between two objects, respectively.

Namely, we search for all the ordered triplets in the form of:

$$\langle object_1, predicate, object_2 \rangle$$

that exist in an image, combined with the localization of those objects. At this point we note that the term predicate is used to describe the verbal phrase without the participation of objects. This terminology is being used throughout this document. The aforementioned triplet can be expressed in the following grammatical form, which will be henceforth preferred:

$$\langle subject, predicate, object \rangle$$

In the context of this work, the term relationship is used exclusively to describe the interactions between two objects and does not take into account relationships of the form:

$$\langle subject, is, attribute \rangle$$

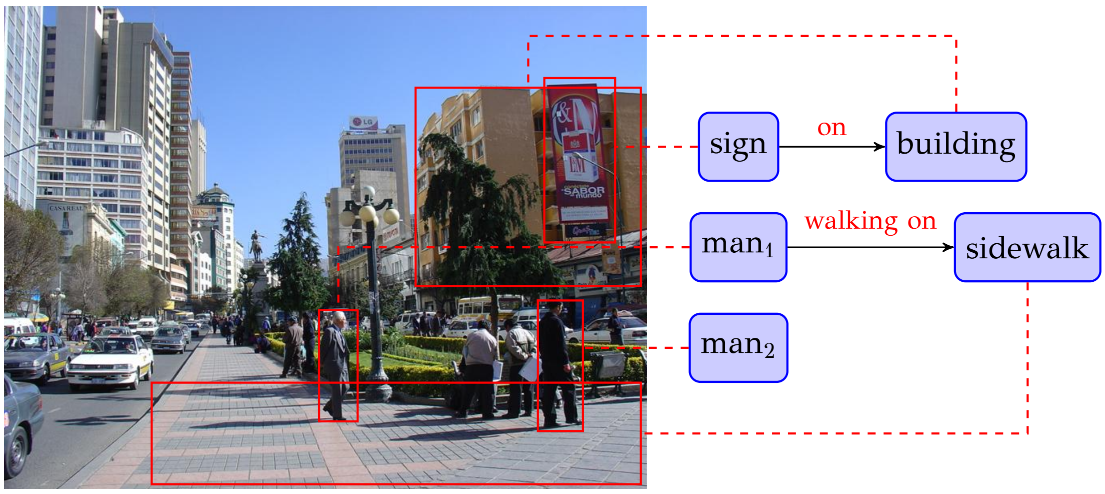
which is included, for example, in Open Images dataset [37] and is commonly referred as attribute detection.

Formally, let $\mathcal{I}$ be an input image, $\boldsymbol{V}$ be the set of localized object regions in image $\mathcal{I}$, describing bounding boxes in $\mathbb{R}^4$ space and $\boldsymbol{E} \subseteq P(\boldsymbol{V}, 2)$ be the set of connections between object pairs, where $P(\boldsymbol{V}, 2)$ denotes the 2-permutations of the set $\boldsymbol{V}$. Furthermore, let $\mathbf{O}$

and $\mathbf{P}$ be the object and predicate labels among the predetermined classes C_O and C_P, respectively.

Using this notation, we form the directed graph $\mathcal{G} = (\boldsymbol{V}, \boldsymbol{E}, \mathbf{O}, \mathbf{P})$, where $\boldsymbol{V}$ indicates the vertices and $\boldsymbol{E}$ indicates the edges. Note that $\mathcal{G}$ is a simple directed graph, i.e. there are neither multiple edges nor loops in the graph. This corresponds to applying multiclass predicate classification instead of multi-label predicate classification. Our goal is to predict the graph elements given an input image. Mathematically, this can be formulated as modeling the system $P(\mathcal{G}\,|\,\mathcal{I})$.

In Figure 1.1 we schematically present the composed graph along with the problem's formalistic description.



(a)

$$\mathcal{G} = \left\{ \begin{array}{l} \boldsymbol{V} = \Big\{o_1 : box_1, o_2 : box_2, o_3 : box_3, o_4 : box_4, o_5 : box_5\Big\} \\ \boldsymbol{E} = \Big\{r_1 : \{o_1, o_2\}, r_2 : \{o_3, o_4\}\Big\} \\ \mathbf{O} = \Big\{sign, building, man, sidewalk, man\Big\} \\ \mathbf{P} = \Big\{on, walking\ on\Big\} \end{array} \right\}$$

(b)

Figure 1.1: Problem Formulation

1.3 Thesis Structure

In the following chapters we discuss related work in the field of image understanding, our system's modeling and architecture, the experimental procedure and methodology we followed, as well as the results of the conducted experiments and the conclusions we drew. This work is comprised of six chapters, including the current introductory chapter.

In chapter 2 we discuss related work in the fields of object detection and scene graph generation and we briefly present the Visual Genome dataset.

In chapter 3 we formulate the system modeling and overall system architecture, as well as the functionality that governs message passing neural networks and graph convolutional networks.

In chapter 4 we discuss the methodology we followed to preprocess the dataset and the experimental settings under which we train and evaluate the system.

In chapter 5 we discuss the evaluation results.

Finally, in chapter 6 we delve into result analysis and extract conclusions regarding the chosen methodology and system modeling. Furthermore, we propose extensions for future work to further increase the current system's performance.

Chapter 2

Related Work

2.1 Image Classification & Object Detection

The evolution of deep learning and convolutional neural networks, which followed as a result of the creation of more efficient training mechanisms and the development of computer hardware brought a revolution in the field of computer vision and artificial intelligence in general. AlexNet [39] was the first in a long series of increasingly sophisticated neural network architectures and image classification systems. It was followed by ZF Net [71], VGGNet [61], GoogLeNet [64] and ResNet V1 [24] and ResNet V2 [26], which achieved ground-breaking performance in ILSVRC competitions and raised the bar in image classification in unprecedented levels.

These networks were a starting point in the field of object detection, where R-CNN (**R**egions with **C**onvolutional **N**eural **N**etwork features) [20] was the first architecture to successfully incorporate convolutional networks, by applying them to a set of regions of interest extracted via Selective Search algorithm [68].

R-CNN was the first in a large family of object detection systems. It was followed by Fast R-CNN [19] and Faster R-CNN [56], both of which decreased the detection speed and increased performance. Faster R-CNN introduced a Region Proposal Network to extract regions of interest and thus, it became the first end-to-end trainable object detection architecture, comprised entirely of convolutional and fully-connected layers.

Beyond the R-CNN network family which mainly focuses on network performance, it is worth mentioning the YOLO (You Only Look Once) network family with YOLO [54] and YOLO9000 [55], as well as the SSD (Single Shot MultiBox Detector) [45] architecture. These models follow a different approach in object detection and belong to the family of single-shot detectors; region extraction is not part of a parallel subnetwork, as in Faster R-CNN, but is instead part of the main network. These networks reach very high inference speeds, achieving object detection in real time with tens of frames per second, while maintaining high performance, since their performance is on par with R-CNN networks.

Instance segmentation is a natural extension to object detection, aiming to detect objects and localize them with non-convex contours. [11, 12, 40] constitute notable examples in the field, followed recently by Mask R-CNN [23], which achieves remarkably greater performance and is an extension to Faster R-CNN in the field of instance segmentation.

2.2 Scene Graph Generation & Relationship Detection

Despite being a classic task with high research interest in the field of computer vision, object detection cannot fully capture the content of an image, since it does not take into account object relationships and the attributes that describe the objects. Over the past few years, the advent of datasets incorporating those elements has resurged the interest in tackling cognitive tasks, such as scene graph generation, image captioning and visual-question answering.

[46] was one of the first efforts in the field of relationship detection, combining convolutional neural networks and object detection models with a language module that models relationship likelihood. Over the next few years, there have been quite a few approaches for relationship detection [51, 10, 43, 41, 69], with increasingly sophisticated methodologies to tackle the problem.

2.2.1 Visual Relationship Detection with Language Priors

The system proposed in [46] is one of the first successful attempts to use convolutional networks in the field of relationship detection. In this work, the task is approached using a visual and a language module.

In the visual module, an R-CNN network [20] extracts the regions of interest and two VGG16 convolutional networks [61] classify the objects and their relationships. All possible pairwise relationships are taken into account. Their bounding boxes are defined as the union of the respective object and subject bounding boxes.

The language module uses word vectors [47] to project the objects in a word vector space. It then projects the relationships in a new relationship vector space that represents the way the objects interact. This module is trained to estimate the relationships likelihood, enhancing overall system performance.

2.2.2 Scene Graph Generation by Iterative Message Passing

Our approach in scene graph generation has a lot in common with [69]. In this work, scene graph generation is modeled as an iterative information exchange system among the objects and their relationships. Its functionality is governed by the following equations:

$$m_i^t = \sum_{j:i\to j} \sigma(\mathbf{v}_1^T[h_i^t, h_{i\to j}^t])h_{i\to j}^t + \sum_{j:j\to i} \sigma(\mathbf{v}_2^T[h_i^t, h_{j\to i}^t])h_{j\to i}^t \tag{2.1}$$

$$m_{i\to j}^t = \sigma(\mathbf{w}_1^T[h_i^t, h_{i\to j}^t])h_i^t + \sigma(\mathbf{w}_2^T[h_j^t, h_{i\to j}^t])h_j^t \tag{2.2}$$

$$h_i^{t+1} = \mathrm{GRU}\big(m_i^t\big), \tag{2.3}$$

$$h_{i\to j}^{t+1} = \mathrm{GRU}\big(m_{i\to j}^t\big) \tag{2.4}$$

where h_i^t and $h_{i\to j}^t$ are the vectors that encode object i and relationship $i \to j$, respectively, $\sigma(\cdot)$ is the sigmoid function, $[\cdot, \cdot]$ denotes vector concatenation and $\mathrm{GRU}(\cdot)$ is a Gated Recurrent Unit [8].

In order to fully understand how this system works, the reader is referred to section 3.3 that formulates the definition and the functionality of a message passing neural network.

2.2.3 Scene Graphs at Cognitive Image Tasks

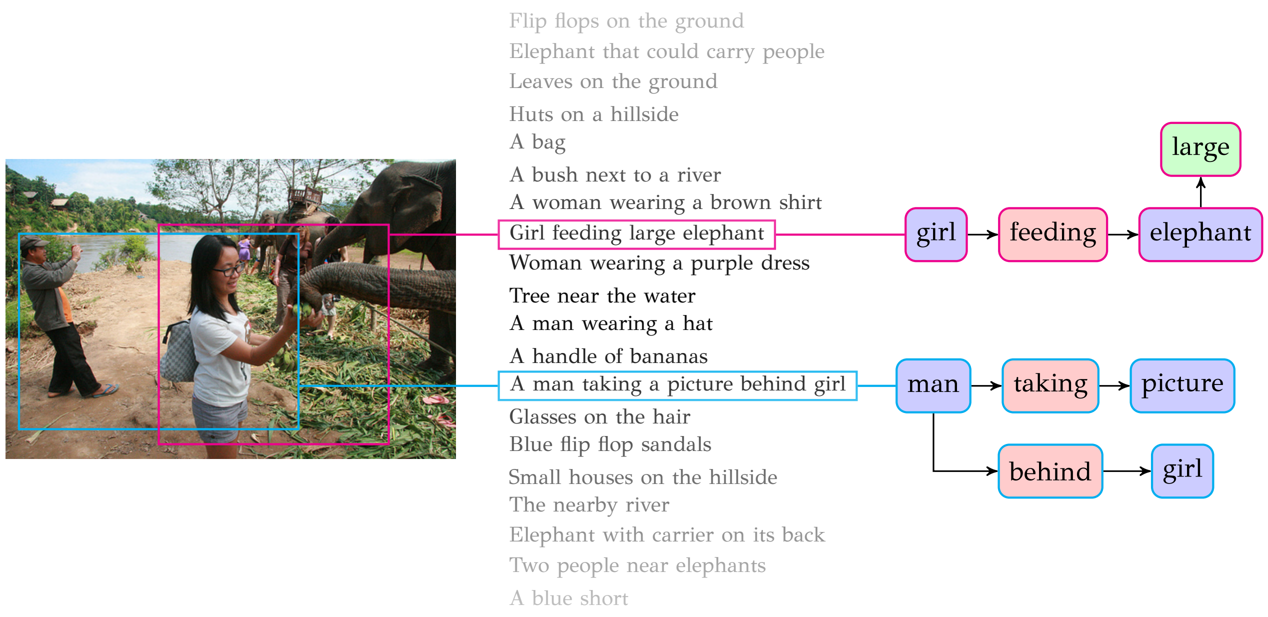
Scene graphs are a compact representation of an image's semantic content that avoid the ambiguity inherently present in text representations. Thus, scene graphs have been used as an intermediate representation in cognitive tasks, such as visual question answering [65], semantic image retrieval [32] and more recently in image generation [29].

2.3 Visual Genome Dataset

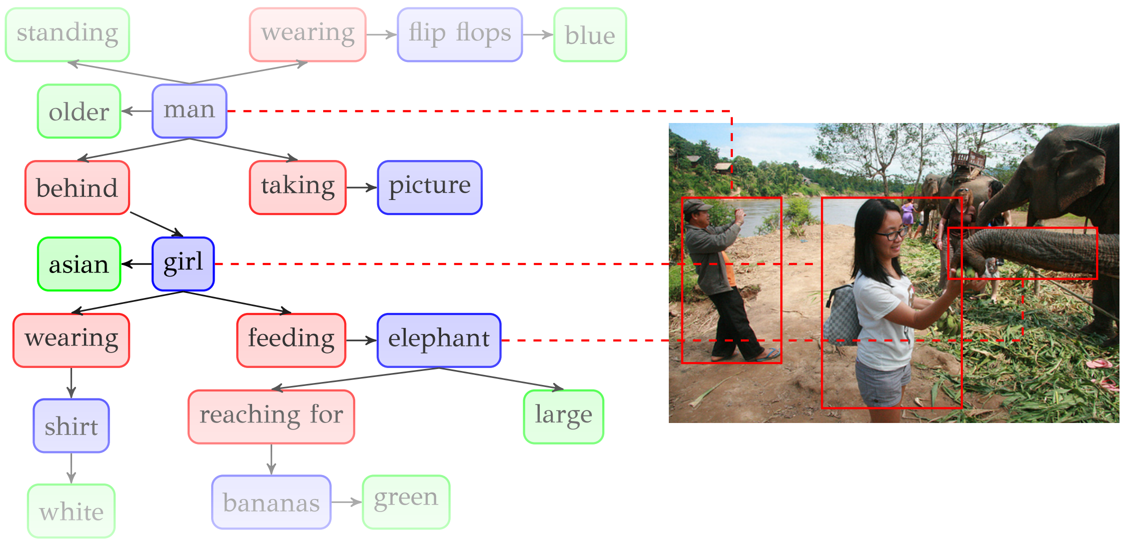
The creation of Visual Genome dataset [38] was a key factor for progress in the field of scene graph generation. Visual Genome is intended to enable the study of computer vision tasks beyond perceptive ones, such as object detection, to cognitive ones. The ability to understand the semantic content of an image and to reason about the visual world is necessary in many higher level tasks, such as relationship detection, visual question answering and semantic image retrieval.

Visual Genome was created from the union of YFCC100M [66] and MS-COCO [44] and contains more than 100,000 images with extensive object, relationship and attribute annotations collected through crowdsourcing techniques. All descriptions are written in free-form text and canonicalized to WordNet [48] synsets. Hence, Visual Genome contains more than 75,000 object categories and more than 40,000 relationship and attribute categories. On average, every image is annotated by 35 objects, 26 attributes and 21 pairwise relationships. Furthermore, every image is annotated by region graphs and captions, as well as question-answer pairs and an image-level scene graph.

The following figures demonstrate indicative examples of images in Visual Genome along with their metadata representation. Figure 2.1 shows an example of high level image descriptions in Visual Genome. Every image is described by a number of localized objects, their attributes and the relationships between them. These elements are combined to generate region graphs; a cognitive representation that is both human and computer understandable, as well as region captions in natural language. Figure 2.2 demonstrates the metadata descriptors by which images are represented in Visual Genome. Every image contains local descriptors, such as region captions and region graphs. These graphs compose a holistic image scene graph that fully describes an image. Finally, every image is characterized by a set of question-answer pairs that are either free type, or are related to a certain region of interest.



(a)



(b)

Figure 2.1: Examples of (a) region captions and region graphs and (b) holistic image scene graph. Adapted from [38]. *Best viewed in color*

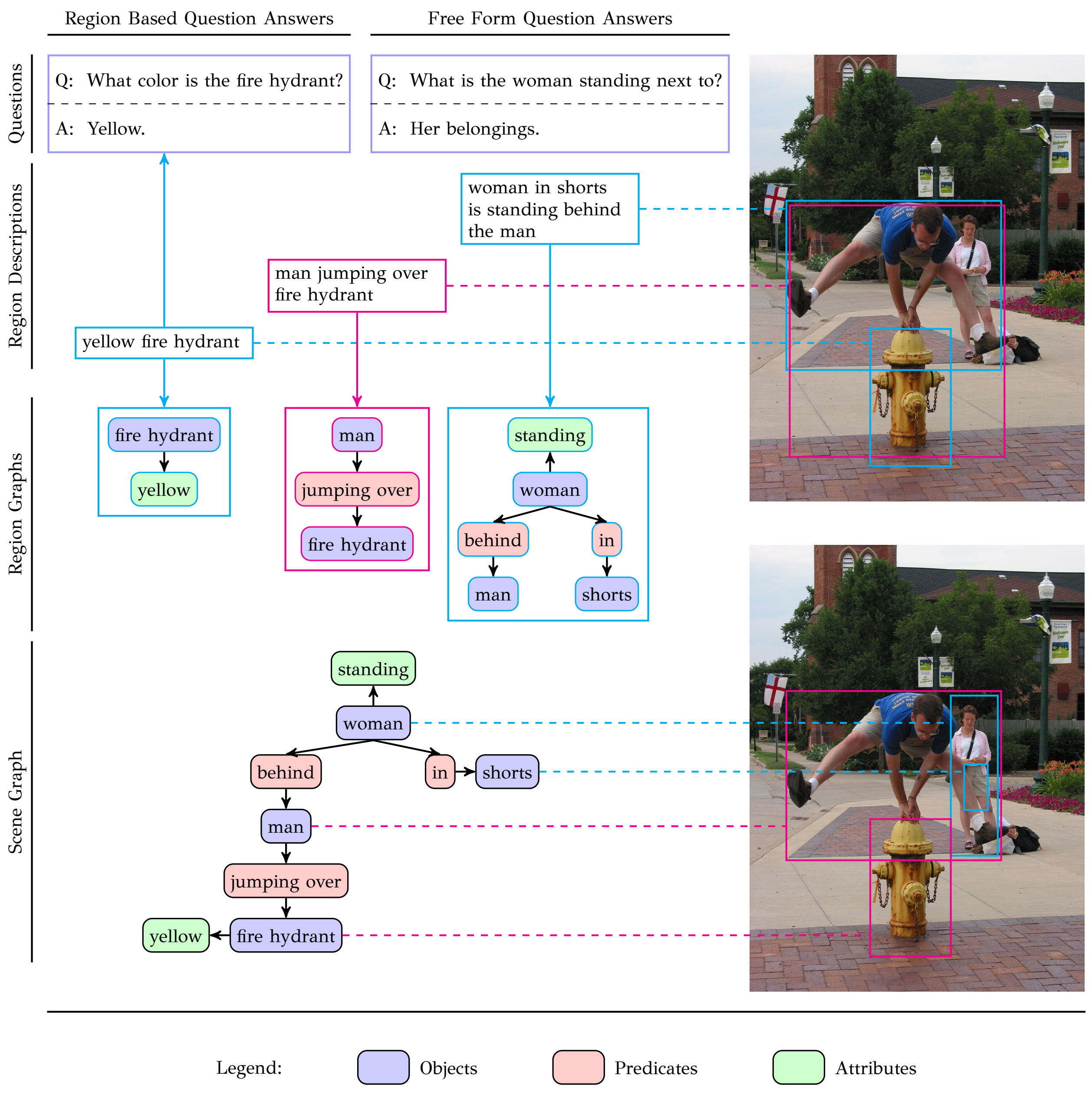


Figure 2.2: Image content representation in Visual Genome dataset. Adapted from [38]. *Best viewed in color*

Chapter 3

System Modeling

3.1 System Modeling

As mentioned in section 1.2, our goal is to extract the scene graph $\mathcal{G} = (\boldsymbol{V}, \boldsymbol{E}, \mathbf{O}, \mathbf{P})$ given an input image $\mathcal{I}$, i.e. modeling the system $P(\mathcal{G} \mid \mathcal{I})$.

In this work, we factorize the scene graph generation problem into three subproblems, namely the object region generation problem, the relationship generation problem and the graph classification problem. Thus, scene graph generation can be formulated as follows:

$$P(\mathcal{G} \mid \mathcal{I}) = \overbrace{P(\boldsymbol{V} \mid \mathcal{I})}^{\substack{\text{Object Region} \\ \text{Generation}}} \underbrace{P(\boldsymbol{E} \mid \boldsymbol{V}, \mathcal{I})}_{\text{Relationship Generation}} \overbrace{P(\mathbf{O}, \mathbf{P} \mid \boldsymbol{V}, \boldsymbol{E}, \mathcal{I})}^{\text{Graph Clasification}} \tag{3.1}$$

This factorization allows us to disentangle scene graph construction from scene graph classification. The object region generation submodule $P(\boldsymbol{V} \mid \mathcal{I})$ is modeled using a convolutional neural network and a region proposal network from a pre-existing object detection pipeline. The relationship proposal submodule $P(\boldsymbol{E} \mid \boldsymbol{V}, \mathcal{I})$ is modeled throughout most of the experiments by taking into account all possible connections/relationships among the objects. However, the large number of potentially noisy proposals at test time can deteriorate network performance. Apart from that, using all possible relationships can prove to be a heavy computational bottleneck. Hence, in order to examine this hypothesis and to test the importance of this submodule, we propose a relationship pruning network that learns to prune unlikely relationships. Finally, the graph classification submodule $P(\mathbf{O}, \mathbf{P} \mid \boldsymbol{V}, \boldsymbol{E}, \mathcal{I})$ is modeled as an iterative information propagation scheme among the connected graph elements, followed by two distinct object and relationship classification components.

3.2 System Architecture

According to the aforementioned system modeling, we can form system architecture, which is schematically represented at Figure 3.1.

The input image $\mathcal{I}$ with dimensions $H \times W \times 3$ goes through a convolutional network that outputs a convolutional feature map with dimensions $H' \times W' \times C$. An object region proposal network receives this tensor as input and extracts P_O object regions, which through a region pooling operation create the object tensor $P_O \times \hat{H} \times \hat{W} \times C$. Respectively, the relationship tensor $P_R \times \hat{H} \times \hat{W} \times C$ is created by calculating the bounding box union for every ordered combination of object regions except self connections.

The object and relationship tensors go through a number of fully connected layers that transform them to the matrices $P_O \times D_O$ and $P_R \times D_R$, respectively. These matrices are the input to a message passing network that exchanges information according to graph connections. The message passing network outputs the transformed matrices $P_O \times D'_O$ and $P_R \times D'_R$ that encode the collected information. Every encoded object and relationship goes through a fully connected layer that computes the final scores and prediction probabilities.

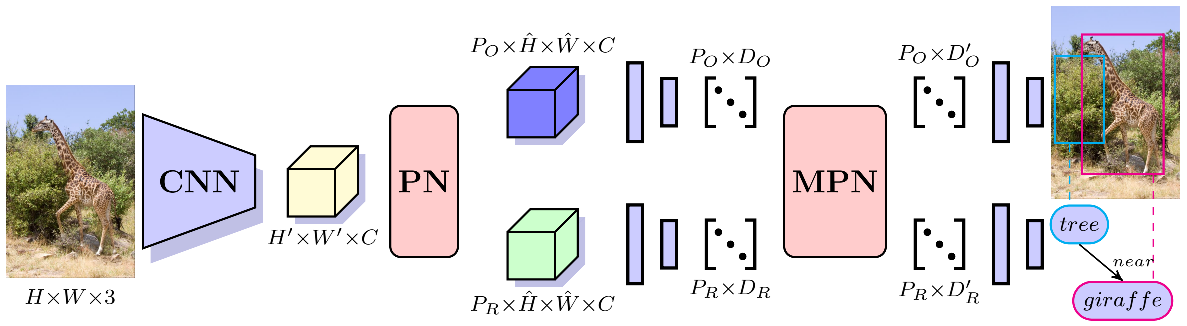


Figure 3.1: System Architecture. **CNN**: Convolutional Neural Network, **PN**: Proposal Network, **MPN**: Message Passing Network

The usefulness of message passing neural networks boils down to the interactions among the connected graph elements and the information exchange among graph vertices and edges. This functionality is diametrically opposed to a system that classifies graph elements in isolation from their neighbouring elements. Using a message passing network results in a more complete graph classification, since each element is defined by its context through an iterative refinement process.

3.3 Message Passing Neural Networks

Message passing neural networks [18] constitute an effort to group a family of neural networks that focus on controlled information exchange among a set of connected graph nodes, under a common mathematical framework. While the original work focuses on solving quantum chemistry problems, message passing neural networks can be reformulated and generalized to solve problems from different scientific fields, such as the field of computer vision and more specifically in this work the problem of scene graph generation.

Inspired by this work, we reformulate message passing neural networks to tackle scene graph generation, according to the following message propagation equations:

$$
\begin{aligned}
m_{v_i}^{t+1} &= \sum_{v_j \in N_{out}(v_i)} M_{\mathcal{V}out}^t\big(h_{v_i}^t, h_{v_j}^t, h_{e_{ij}}^t\big) + \sum_{v_j \in N_{in}(v_i)} M_{\mathcal{V}in}^t\big(h_{v_i}^t, h_{v_j}^t, h_{e_{ji}}^t\big), \\
h_{v_i}^{t+1} &= U_{\mathcal{V}}^t\big(h_{v_i}^t, m_{v_i}^{t+1}\big), \\
m_{e_{ij}}^{t+1} &= M_{\mathcal{E}}^t\big(h_{v_i}^t, h_{v_j}^t, h_{e_{ij}}^t\big), \\
h_{e_{ij}}^{t+1} &= U_{\mathcal{E}}^t\big(h_{e_{ij}}^t, m_{e_{ij}}^{t+1}\big)
\end{aligned} \tag{3.2}
$$

where $h_{v_i}^t$ is the vector that encodes vertex (object) information v_i at time step t, while $h_{e_{ij}}^t$ the vector that encodes edge (relationship) information between vertices v_i and v_j at time step t.

At time step $t = 0$, vertex and edge vectors $h_{v_i}^0$ and $h_{e_{ij}}^0$ respectively, constitute the output of fully connected layers that precede the message passing network. Information is being exchanged for $\mathbf{T}$ steps. Information propagation in a message passing network is abstractly represented in Figure 3.2.

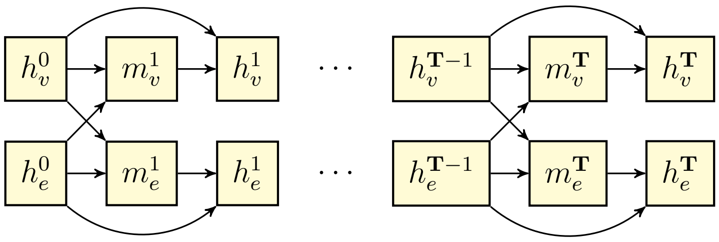


Figure 3.2: Message Passing Neural Network

The creation of a message passing neural network model can be achieved through specifying the functions $M_{\mathcal{V}out}^t$, $M_{\mathcal{V}in}^t$ and $M_{\mathcal{E}}^t$ that define the ways in which the model collects information from its neighbouring nodes and the functions $U_{\mathcal{V}}^t$ and $U_{\mathcal{E}}^t$ that define the ways in which vertices and edges reevaluate their encoded information. In their most general form, these functions are parameterized and differentiable.

In the following subsections we present the models that we experimented on within the context of this work. We note that in some cases, the functions $M_{\mathcal{V}out}^t$ and $M_{\mathcal{V}in}^t$ are concatenated to $M_{\mathcal{V}}^t(h_{v_i}^t, h_{v_j}^t, h_{e_{ij}}^t, h_{e_{ji}}^t)$, while vector $m_{v_i}^{t+1}$ is defined as the sum of all neighbouring region messages; these models treat incoming and outgoing relationships in the same manner.

3.3.1 Molecular Graph Convolutions

This formulation is based on [34]. It is determined by the following message and update functions:

$$\begin{aligned}
M_{\mathcal{V}out}(h^t_{v_i}, h^t_{v_j}, h^t_{e_{ij}}) &= h^t_{e_{ij}}, \\
U_{\mathcal{V}}(h^t_{v_i}, m^{t+1}_{v_i}) &= \rho\Big(\mathbf{W}_1\Big[\rho\big(\mathbf{W}_0 h^t_{v_i}\big), m^{t+1}_{v_i}\Big]\Big), \\
M_{\mathcal{E}}(h^t_{v_i}, h^t_{v_j}, h^t_{e_{ij}}) &= \rho\Big(\mathbf{W}_2\Big[h^t_{v_i}, h^t_{v_j}\Big]\Big), \\
U_{\mathcal{E}}(h^t_{e_{ij}}, m^{t+1}_{e_{ij}}) &= \rho\Big(\mathbf{W}_4\Big[\rho\big(\mathbf{W}_3 h^t_{e_{ij}}\big), m^{t+1}_{e_{ij}}\Big]\Big)
\end{aligned} \tag{3.3}$$

where $\rho(\cdot)$ is the rectified linear unit (ReLU) and $[\cdot, \cdot]$ represents vector concatenation.

An altered version of the previous formulation that we experimented on, in order for the objects to collect more information is as follows:

$$\begin{aligned}
M_{\mathcal{V}out}(h^t_{v_i}, h^t_{v_j}, h^t_{e_{ij}}) &= \rho\Big(\mathbf{W}_5\Big[h^t_{v_j}, h^t_{e_{ij}}\Big]\Big), \\
M_{\mathcal{V}in}(h^t_{v_i}, h^t_{v_j}, h^t_{e_{ji}}) &= \rho\Big(\mathbf{W}_6\Big[h^t_{v_i}, h^t_{e_{ji}}\Big]\Big), \\
U_{\mathcal{V}}(h^t_{v_i}, m^{t+1}_{v_i}) &= \rho\Big(\mathbf{W}_1\Big[\rho\big(\mathbf{W}_0 h^t_{v_i}\big), m^{t+1}_{v_i}\Big]\Big), \\
M_{\mathcal{E}}(h^t_{v_i}, h^t_{v_j}, h^t_{e_{ij}}) &= \rho\Big(\mathbf{W}_2\Big[h^t_{v_i}, h^t_{v_j}\Big]\Big), \\
U_{\mathcal{E}}(h^t_{e_{ij}}, m^{t+1}_{e_{ij}}) &= \rho\Big(\mathbf{W}_4\Big[\rho\big(\mathbf{W}_3 h^t_{e_{ij}}\big), m^{t+1}_{e_{ij}}\Big]\Big)
\end{aligned} \tag{3.4}$$

3.3.2 Gated Graph Neural Network

This formulation is based on [42]. Its functionality is governed by the following functions:

$$\begin{aligned}
M_{\mathcal{V}}(h^t_{v_i}, h^t_{v_j}, h^t_{e_{ij}}, h^t_{e_{ji}}) &= \rho\Bigg(\mathbf{W}_7\Bigg[\rho\Big(\mathbf{W}_6 h^t_{v_i}\Big), \rho\Bigg(\mathbf{W}_5\Bigg[\rho\Big(\mathbf{W}_3\Big[h^t_{v_j}, h^t_{e_{ij}}\Big]\Big), \rho\Big(\mathbf{W}_4\Big[h^t_{v_i}, h^t_{e_{ji}}\Big]\Big)\Bigg]\Bigg)\Bigg]\Bigg), \\
U_{\mathcal{V}}\big(h^t_{v_i}, m^{t+1}_{v_i}\big) &= \mathrm{GRU}\big(m^{t+1}_{v_i}\big), \\
M_{\mathcal{E}}(h^t_{v_i}, h^t_{v_j}, h^t_{e_{ij}}) &= \rho\Bigg(\mathbf{W}_2\Bigg[\rho\Big(\mathbf{W}_0 h^t_{e_{ij}}\Big), \rho\Big(\mathbf{W}_1\Big[h^t_{v_i}, h^t_{v_j}\Big]\Big)\Bigg]\Bigg), \\
U_{\mathcal{E}}\big(h^t_{e_{ij}}, m^{t+1}_{e_{ij}}\big) &= \mathrm{GRU}\big(m^{t+1}_{e_{ij}}\big)
\end{aligned} \tag{3.5}$$

where $\mathrm{GRU}(\cdot)$ is the Gated Recurrent Unit [8].

A modified version that we experimented on was to replace the initial relationship bounding boxes and their feature extraction, therefore, with the concatenated relationship subject and object features. These features are the output of fully connected layers and their conversion to relationship features is accomplished via a fully connected layer that transforms them to the desired dimensions. Using message passing network notation, relationship vectors $h^0_{e_{ij}}$ are defined as follows:

$$h^0_{e_{ij}} = \rho\Big(\mathbf{W}\big[h^0_{v_i}, h^0_{v_j}\big]\Big) \tag{3.6}$$

We also experimented on a second modification, orthogonal to the first one, that aims at simplifying the initial formulation without reducing neither its efficacy nor its efficiency. It is defined as follows:

$$\begin{aligned} M_{\mathcal{V}}(h^t_{v_i}, h^t_{v_j}, h^t_{e_{ij}}, h^t_{e_{ji}}) &= \rho\Big(\mathbf{W}_{\mathcal{V}}\Big[\rho\Big(\mathbf{W}_A\Big[h^t_{v_i}, h^t_{e_{ij}}, h^t_{v_j}\Big]\Big), \rho\Big(\mathbf{W}_P\Big[h^t_{v_j}, h^t_{e_{ji}}, h^t_{v_i}\Big]\Big)\Big]\Big), \\ U_{\mathcal{V}}\big(h^t_{v_i}, m^{t+1}_{v_i}\big) &= \mathrm{GRU}\big(m^{t+1}_{v_i}\big), \\ M_{\mathcal{E}}(h^t_{v_i}, h^t_{v_j}, h^t_{e_{ij}}) &= \rho\Big(\mathbf{W}_{\mathcal{E}}\Big[h^t_{v_i}, h^t_{e_{ij}}, h^t_{v_j}\Big]\Big), \\ U_{\mathcal{E}}\big(h^t_{e_{ij}}, m^{t+1}_{e_{ij}}\big) &= \mathrm{GRU}\big(m^{t+1}_{e_{ij}}\big) \end{aligned} \tag{3.7}$$

3.4 Graph Convolutional Networks

Graph convolutional networks [36] are a reformulated version of convolutional neural networks that operate on graph structured data. Information in a multilayer graph convolutional network is propagated according to the following rule:

$$H^{(t+1)} = \rho\Big(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(t)}W^{(t)}\Big), \tag{3.8}$$

where $\tilde{A} = A + I_N$ is the square adjacency matrix of graph $\mathcal{G}$ with added self-loops though the identity matrix I_N and $\tilde{D}$ the diagonal degree matrix of the matrix $\tilde{A}$ with elements $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$.

This formulation, albeit effective in many problem types, does not treat graph edges as entities and as such, is not directly suitable to the problem of relationship detection and scene graph generation. Thus, we model graph edges as vertices, which results in creating a new bipartite graph with two types of vertices; those who represent objects and can potentially be connected to an arbitrarily large number of vertices and those who represent relationships and can only be connected to their corresponding subject and object.

Furthermore, the initial formulation is restricted to undirected graphs, whilst image scene graphs are directed graphs. However, the triad *subject* $\rightarrow$ *predicate* $\rightarrow$ *object* directionality is predetermined and known a priori. Apart from that, the implied bidirectionality in an undirected graph allows messages to propagate to distant graph nodes and to collect more information from neighbouring nodes. Hence, the adjacency matrix was modeled as a symmetric bipartite adjacency matrix of an undirected graph.

Figure 3.3 shows the adjacency matrix A of a “complete” graph, within the allowed connections. Adjacency matrix’s dimensions are $N^2 \times N^2$, since it is comprised from N objects and $R = N(N-1)$ relationships that model the interactions of each and every one of the N objects with the remaining $N-1$ objects. The adjacency matrix is a bipartite matrix that is composed of square zero matrices on its main diagonal and of non-zero matrices on the off-diagonal parts. This is due to the fact that there is no direct communication between two objects or two relationships. The only types of communication allowed are those between a subject and a predicate and between a predicate and an object. The adjacency matrix is also a symmetric matrix, as noted in the previous paragraph. Finally, non-zero matrix elements in the off-diagonal parts come from the fact that relationships are serially defined between object pairs; the first $N-1$ relationships model the interactions between the first object and the remaining $N-1$, the next $N-1$ relationships model the interactions between the second object and the remaining $N-1$ and so on.

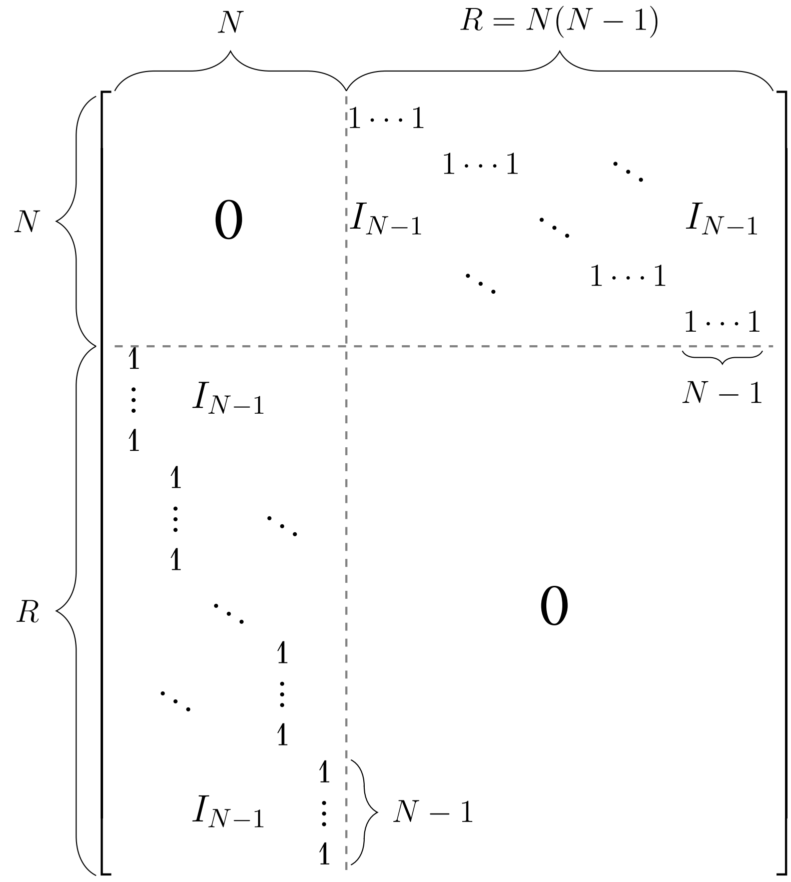


Figure 3.3: Graph Convolutional Network Adjacency Matrix

3.4.1 Formulation as a Message Passing Neural Network

Graph convolutional networks can also be described as message passing neural networks. By modeling graph relationships as vertices, we get the following mathematical formulation for a graph convolutional network, where the term v_i is used to describe the objects as well as the relationships in an image:

$$\begin{aligned} m_{v_i}^{t+1} &= \sum_{v_j \in N(v_i) \cup \{v_i\}} M_{\mathcal{V}}\big(h_{v_i}^t, h_{v_j}^t\big), \\ M_{\mathcal{V}}\big(h_{v_i}^t, h_{v_j}^t\big) &= \big(\deg(v_i)\deg(v_j)\big)^{-1/2} h_{v_j}^t, \\ U_{\mathcal{V}}^t\big(h_{v_i}^t, m_{v_i}^{t+1}\big) &= \rho\big(\mathbf{W}^t m_{v_i}^{t+1}\big) \end{aligned} \tag{3.9}$$

3.5 Relationship Pruning Network

The relationship pruning network is a simple implementation of the relationship generation submodule and was created to test how relationship quality affects overall system performance. This network operates on the full set of relationships between objects. Hence, its application follows object region generation and the initial relationship generation, whilst its output becomes the relationship input to the message passing network. The network prunes relationships only once and is not applied iteratively to the encoded message passing network relationships.

Relationship $h_{e_{ij}}^0$ is pruned on the following condition:

$$\sigma(\mathbf{w}_p^T [h_{v_i}^0, h_{e_{ij}}^0, h_{v_j}^0]) < 0.5$$

where $\mathbf{w}_p$ the (trainable) vector that scores relationships and $\sigma(\cdot)$ the logistic function.

Chapter 4

Experimental Procedure

4.1 Visual Genome Dataset

The dataset we chose to train and evaluate our models is Visual Genome [38]. Visual Genome (version 1.4) contains 108,077 images and every image is annotated on average with 23.28 objects and 21.43 relationships. The crowdsourcing techniques used to create Visual Genome, whilst effective at annotating a large number of images extensively, suffer from a few drawbacks. First of all, a large number of objects is localized poorly; bounding boxes are either too tight and cut off essential parts or too relaxed and contain background noise. Secondly, there exist a lot of highly overlapping bounding boxes that characterize the same object. Finally, using freeform text results in multiple names describing objects and relationships, some of which are the same words but written in a different way (e.g. using capital letters or unnecessary punctuation).

For the aforementioned reasons, we perform a metadata preprocessing step that includes object bounding boxes as well as object and relationship labels. More specifically, entity labels were stemmed, after removing punctuation and non-grammatical characters and converting the letters to lowercase. We chose the 150 most frequent objects and the 50 most frequent predicates as our object and predicate classes. Furthermore, we merged highly overlapping bounding boxes and removed very small boxes. Finally, we removed from the dataset all images that contain no relationships, since model evaluation is performed using relationships.

The resulting dataset is comprised of 84,085 images, which account for 77.8% of the original dataset. Every image contains an average of 12.1 objects and 4.56 relationships. The data split in train, validation and test sets was performed using random sampling to avoid semantic correlations among neighbouring images. Finally, the dataset was split in 60%/10%/30% train/validation/test sets.

The train and validation sets were augmented using mirror images. Thus, these splits effectively doubled in size, resulting in 100,794 train images and 16,948 validation images. Cumulatively, this results in 117,742 images.

The object and relationship frequencies in the dataset present some interesting characteristics. Figure 4.1 demonstrates the large tails in object and relationship frequencies. Figure 4.2 shows the objects and predicates distribution in the three data splits, where the even split among them is visually clear. On the contrary, Figure 4.3 shows the entity distributions using serial data splits. As one can notice, there are quite a few instances of noticeable deviations among the three splits, which justifies the choice to sample the three splits.

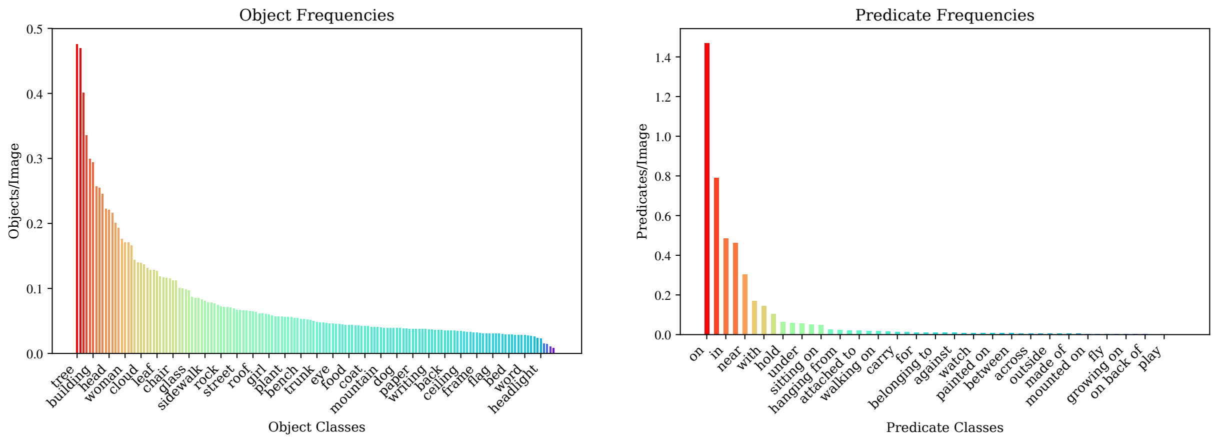


Figure 4.1: Object and Predicate Frequencies

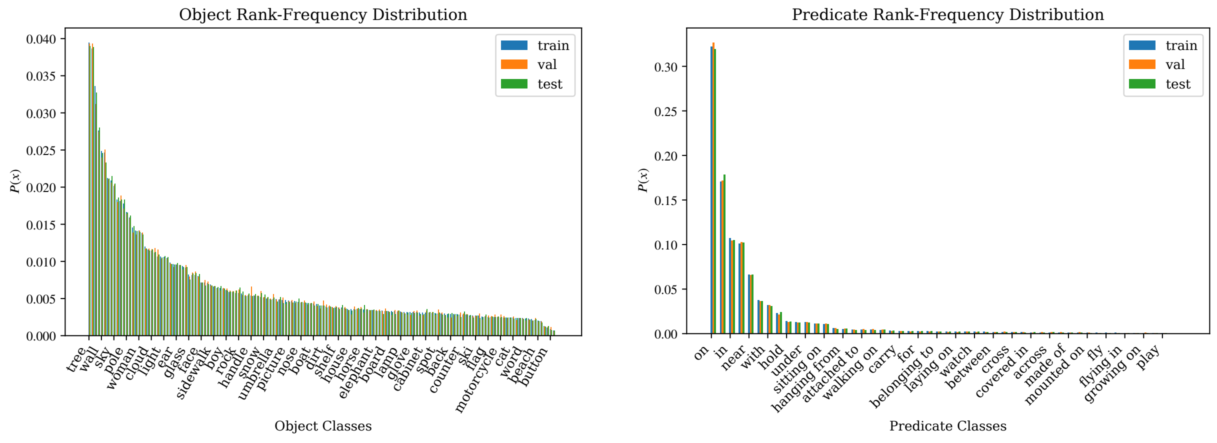


Figure 4.2: Object and Predicate Distribution

4.2 Training

We follow an image-centric mini-batch sampling strategy, similar to the one used in Faster R-CNN [56]. More specifically, each mini-batch is comprised of 1-2 images and 256 regions of interest, a combination of groundtruth regions and proposed regions from the region proposal network.

We use an MS COCO pretrained VGG-16 [61] convolutional neural network from the Faster R-CNN pipeline to extract visual features from images. We use this model in all of our experiments, unless stated otherwise, in order to better compare model performance.

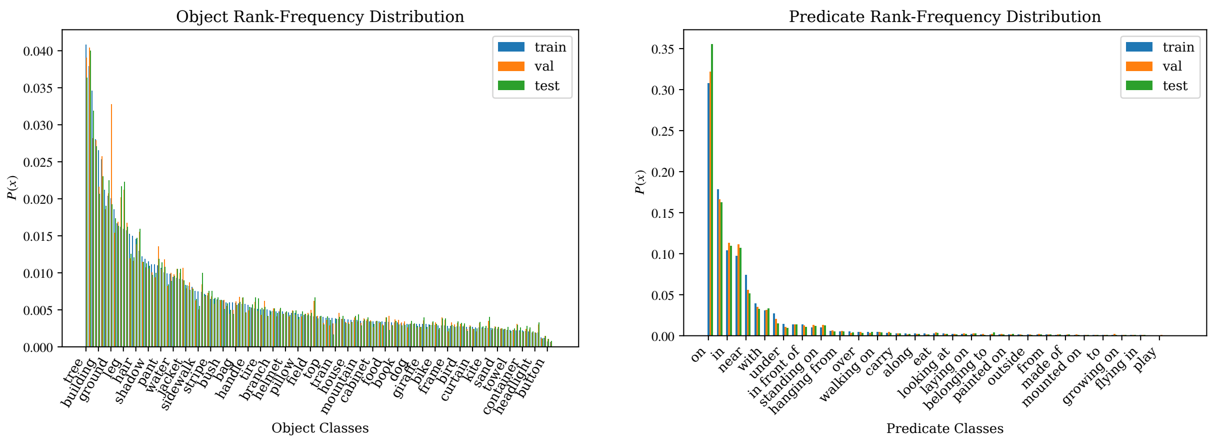


Figure 4.3: Unshuffled dataset Object and Predicate Distribution

Features from regions of interest are extracted using RoI Align [23]. The same algorithm is used to extract features from potential relationships. Relationship bounding boxes are defined as the union of the subject and object bounding boxes.

We experimented on the number of iterations in the message passing layers as well as their output dimensions. The number of iterations varies from 1 to 3, while the output dimensions vary from 256 to 1024, with a default value of 512.

The models were trained using Stochastic Gradient Descent with standard backpropagation. We experimented on a variety of optimization algorithms, including Momentum [52] and Nesterov Accelerated Gradient [49, 50, 63], as well as RMSProp [67] and Adam [35]. We used He initialization [25] to initialize the weights with ReLU activations and Glorot initialization [21] to initialize the remaining weights. Finally, we freeze the convolutional layer weights throughout all of our experiments. We used L_2 regularization and Dropout [62] on fully-connected layers.

Hyperparameter optimization was performed through manual hyperparameter tuning combined with random hyperparameter search [4].

4.2.1 Mini-batch Sampling Strategy

During training we extract 2,000 regions of interest from every image from the set of proposals made by the region proposal network using non-maximum suppression. For every region we calculate its maximum overlap with the groundtruth regions, as well as the corresponding object. Regions with an overlap higher than 0.5 are used in bounding box regression. Hence, we calculate the following bounding box transformation, according to [20]:

$$\begin{aligned} t_x &= (g_x - p_x)/p_w \\ t_y &= (g_y - p_y)/p_h \\ t_w &= \ln(g_w/p_w) \\ t_h &= \ln(g_h/p_h) \end{aligned} \tag{4.1}$$

where:

$\{t_x, t_y, t_w, t_h\}$	= the regression target,
$\{p_x, p_y, p_w, p_h\}$	= the proposed region bounding box,
$\{g_x, g_y, g_w, g_h\}$	= the groundtruth bounding box,

expressed in center coordinates (x, y) and box dimensions (w, h).

Next, we augment the set of relationships by replacing all groundtruth subjects and objects with the regions of interest with an overlap greater than 0.5. We then sample relationships until we have 128 regions. If we run out of relationships, we sample regions that belong to no relationships. Furthermore, we sample 128 background relationships. Finally, if the desired number of regions has not been attained yet, we fill the remaining ones with background regions.

4.2.2 Loss Functions

We use two cross-entropy loss functions for the object and predicate losses and a Huber loss function (smooth L1) for the bounding box offsets. In most experiments, predicate classification do not take true negative relationships into account. Respectively, bounding box regression ignores true negative boxes.

$$\mathcal{L}_{obj} = -\sum_{i=1}^{N_O}\sum_{c=0}^{C_O} y_{ic} \ln p_{ic} \tag{4.2}$$

$$\mathcal{L}_{pred} = -\sum_{i=1}^{N_P}\sum_{c=1}^{C_P} y_{ic} \ln p_{ic} \tag{4.3}$$

$$\mathcal{L}_H(x;\delta) = \begin{cases} 0.5x^2, & \text{if } |x| \le \delta \\ \delta(|x| - 0.5\delta), & \text{otherwise} \end{cases} \tag{4.4}$$

$$\mathcal{L}_{bbox} = \sum_{i=1}^{N_O}\sum_{c=1}^{C_O} y_{ic} \sum_{b\in\boldsymbol{B}} \mathcal{L}_H(t_{ib} - d_{icb}; \delta = 1) \tag{4.5}$$

where y_{ic} is the value of the one-hot encoding vector $\boldsymbol{y}_i$ for object/predicate i and class c, p_{ic} is the probability for the object/predicate i and class c, t_{ib} is the regression target for object i, as defined in Equation 4.1, d_{icb} is the predicted bounding box offset for the object i and class c, $\boldsymbol{B} = \{x, y, w, h\}$, N_O and N_P is the number of objects and predicates in the mini-batch, respectively, and finally, C_O and C_P are the object and predicate classes.

The total loss in every iteration is calculated by the weighted sum of the individual losses plus the regularization loss $\mathcal{L}_{reg}$.

$$\mathcal{L}_{total} = w_{obj} \cdot \mathcal{L}_{obj} + w_{pred} \cdot \mathcal{L}_{pred} + w_{bbox} \cdot \mathcal{L}_{bbox} + \lambda \cdot \mathcal{L}_{reg} \tag{4.6}$$

In most experiments, the weight coefficients are equal to 1.

In the experiments conducted with the addition of a relationship pruning network, we use an additional binary log-loss function, defined as follows:

$$\mathcal{L}_{prune} = -\sum_{i=1}^{N_P} \big(y_i \ln p_i + (1 - y_i) \ln(1 - p_i) \big) \tag{4.7}$$

where y_i is a binary variable that represents the existence of a groundtruth relationship and p_i is the predicted probability for that relationship.

4.3 Evaluation

At evaluation time, we used non-maximum suppression to select 50 regions of interest from the set of proposals. We made predictions on all object pairs, except self-connections.

4.3.1 Evaluation Metrics

The metric of choice for visual relationship detection is $R@k$ (recall at k) [2, 46] and more specifically $R@[k, IoU = 0.5]$, since it accurately reflects the quality of a system. This metric computes the fraction of times the correct relationship is predicted in the top k confident relationship predictions, while simultaneously the predicted subject and object bounding boxes have an IoU (intersection over union) ratio with their respective grundtruth boxes greater than 0.5. The chosen values for k are 20, 50 and 100.

4.3.2 Evaluation Tasks

The trained models were evaluated at three evaluation tasks of increasing difficulty, according to [46, 69]. Despite the fact that the main evaluation task is the final task, all tasks provide useful insight and help to further understand the problem, by adding abstraction layers and removing limitations of the current architecture.

The first evaluation task is **predicate classification**. In this task, the input is an image and the set of localized objects in it. Our aim is to correctly classify the predicates that connect object pairs.

The second evaluation task is **scene graph classification**. In this task, the input is an image and a set of bounding boxes. Our aim is to classify the objects as well as the predicates that connect ordered object pairs.

The third evaluation task is **scene graph detection**. In this task, the input is just an image and our aim is to detect all triplets of the form $\langle subject, predicate, object \rangle$ that exist in it, along with the accurate localization of all relationship subjects and objects.

The inputs and outputs for the three evaluation tasks are presented schematically in Figure 4.4.

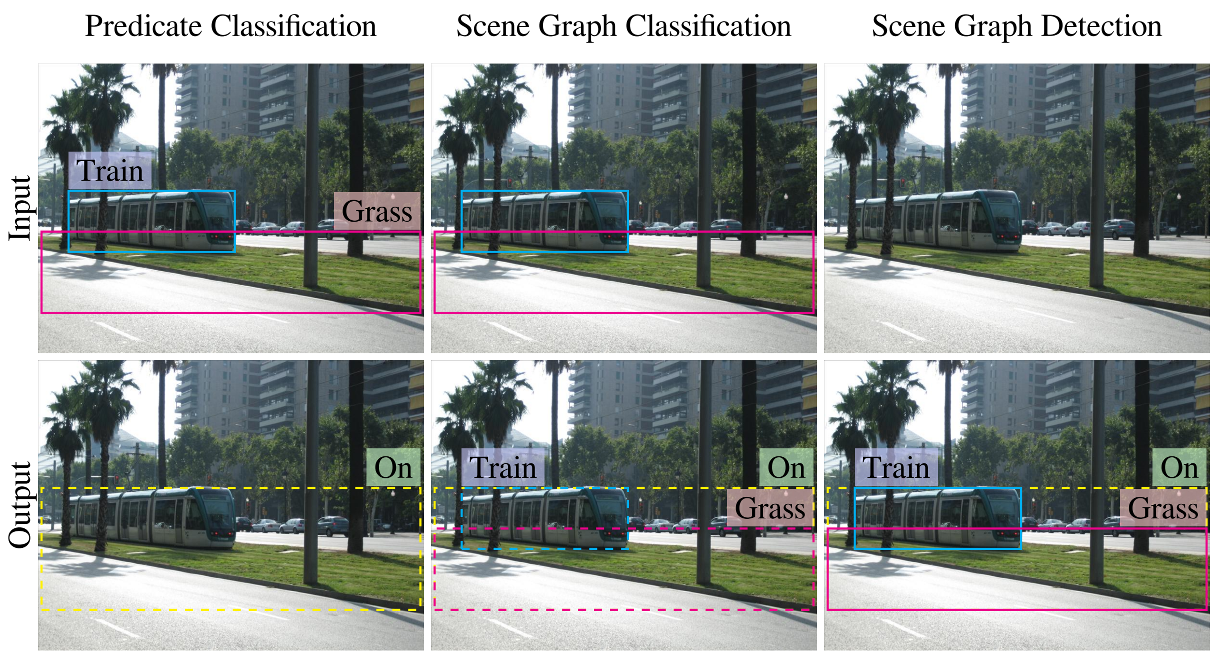


Figure 4.4: Evaluation Tasks Visual Representation

4.4 Implementation Details

This work was implemented in Python 2.7 using TensorFlow [1], software version 1.5. Both the training and the evaluation process were performed using two computers from the university server of the Multimedia Understanding Group. Their specifications can be summarized as follows:

- System #1
  - CPU: Intel® Xeon® E5335 (2.00 GHz)
  - RAM: 7,972 MiB
  - GPU: Nvidia® GeForce® GTX 780 (6,144 MiB)
- System #2
  - CPU: Intel® Xeon® E5-2650 v3 (2.30 GHz)
  - RAM: 64,510 MiB
  - GPU: Nvidia® Tesla® K40c (12,288 MiB)

Chapter 5

Results

5.1 Results Overview

The models were trained on the train set and evaluated on the validation set. Afterwards, we retrained the best models on the combined trainval set and evaluated them frequently on the validation set. Finally, when the models reached their optimal performance, we evaluated them on the test set.

The Gated Graph Neural Network (subsection 3.3.2) family model that had the best performance is the one described in Equation 3.7. The model reached is highest performance at 102,871 iterations. The optimal hyperparameters are presented in List 5.1.

Training Hyperpameters:

- **Optimization Algorithm**: Momentum
- **Optimization Algorithm Parameters**:
  - **momentum**: 0.9
- **Learning Rate**: 10^{-3}
- **Learning Rate Step Decay Period**: 1 epoch (58,871 iterations)

Network Hyperparameters:

- **Message Passing Neural Network Iterations**: 1
- **Regularization Strength**: 0.0
- **Ignore Background Relationship Loss**: True
- **Message Passing Neural Network vertex vector dimensions**: 512
- **Message Passing Neural Network edge vector dimensions**: 512

List 5.1: Optimal Message Passing Neural Network Hyperparameters

The optimal hyperparameters for the graph convolutional network are presented in List 5.2. The network reached its highest performance after 235,484 iterations.

Training Hyperpameters:

Optimization Algorithm: Momentum

Optimization Algorithm Parameters:

momentum: 0.9

Learning Rate: 10^{-3}

Learning Rate Step Decay Period: 2 epochs (117,742 iterations)

Network Hyperparameters:

Graph Convolutional Network Iterations: 2

Regularization Strength: 0.0

Ignore Background Relationship Loss: False

Graph Convolutional Network vertex & edge dimensions: 512

List 5.2: Optimal Graph Convolutional Network Hyperparameters

Molecular Graph Convolutions (subsection 3.3.1) family models did not reach any good performance and are thus not presented in the results table.

Table 5.1 summarizes the performance of the models stated above, along with the models [46] and [69]. For these two models, we present both the original results, as these are published in the respective works, as well as their retrained versions according to our own experimental procedure, for a more meaningful comparison.

We only present the visual part of [46], since the language part is an orthogonal approach and can be independently added to our system. The visual system in [46] is equal to our system without the addition of a message passing component.

	Predicate Classification			Scene Graph Classification			Scene Graph Detection		
Model	$R@20$	$R@50$	$R@100$	$R@20$	$R@50$	$R@100$	$R@20$	$R@50$	$R@100$
VRD‡ [46]	–	1.58	1.85	–	–	–	–	7.11	7.11
IMP [69]	–	44.75	53.08	–	21.72	24.38	–	3.44	4.24
VRD‡† [46]	18.97	29.78	37.74	6.64	9.78	11.95	0.06	0.14	0.27
IMP† [69]	29.91	44.36	53.92	14.69	19.56	22.42	1.65	2.51	3.34
GGNN	32.65	**46.83**	**55.91**	**14.98**	**19.83**	**22.65**	**1.83**	**2.75**	**3.60**
GCN	**32.94**	43.18	48.23	13.06	16.12	17.72	1.76	2.57	3.21

Table 5.1: Evaluation Results at Visual Genome dataset [38]. ‡: Visual Model, †: Implementation according to [69] and retraining according to our experimental procedure

Table 5.1 shows the importance of adding a message passing network, as it significatly boosts network performance in all evaluation tasks. Both GCN and GGNN networks score ten times better in scene graph detection task, while the performance gain in the remaining two tasks is more than 50%. The GGNN network scores higher than the IMP network [69] across all evaluation tasks. The difference is quite clear in predicate classification and scene graph detection, with performance gains ranging from 3.55 to 8.39% and 7.22 to 9.83%, respectively. Finally, the GGNN network has greater performance than the GCN network based on almost every evaluation metric.

5.1.1 Results with the addition of a Relationship Pruning Network

The GGNN network that achieved the highest performance was retrained with the addition of a relationship pruning network presented in section 3.5, in order to test the usefulness of a relationship generation submodule versus using all possible relationships. This model was trained without searching for the optimal hyperparameters. Instead, we mostly used the same hyperparameters as the original GGNN. However, we opted not to ignore background relationship losses. Hence, the chosen hyperparameters are presented in List 5.3.

Training Hyperpameters:

- **Optimization Algorithm**: Momentum
- **Optimization Algorithm Parameters**:
  - **momentum**: 0.9
- **Learning Rate**: 10^{-3}
- **Learning Rate Step Decay Period**: 1 epoch (58,871 iterations)

Network Hyperparameters:

- **Message Passing Neural Network Iterations**: 1
- **Regularization Strength**: 0.0
- **Ignore Background Relationship Loss**: False
- **Message Passing Neural Network vertex vector dimensions**: 512
- **Message Passing Neural Network edge vector dimensions**: 512

List 5.3: Optimal Message Passing Neural Network Hyperparameters with the addition of a relationship pruning network

Table 5.2 summarizes the performance of the new network, abbreviated as GGNN+, compared to its original version.

The GGNN+ network is superior in all evaluation tasks and based on all metrics except R@100 metric in predicate classication and scene graph classification. The performance gain is significant in all tasks when using R@20, and especially so in the scene graph detection task, where we observe an 86% performance gain.

	Predicate Classification			Scene Graph Classification			Scene Graph Detection		
Model	R@20	R@50	R@100	R@20	R@50	R@100	R@20	R@50	R@100
GGNN	32.65	46.83	**55.91**	14.98	19.83	**22.65**	1.83	2.75	3.60
GGNN+	**44.12**	**48.71**	49.56	**19.62**	**21.01**	21.22	**3.42**	**4.61**	**5.35**

Table 5.2: Evaluation Results with the addition of a relationship pruning network

Chapter 6

Conclusions & Future Work

The evaluation of the systems implemented in the context of this work provides valuable insights regarding the functionality and the usefulness of message passing in visual scene graph generation. Based on these insights as well as the knowledge obtained during this work, we propose future extensions to further enhance system performance.

6.1 Conclusions

In this work, we model scene graph generation using three discrete submodules, the object region generation submodule, the relationship generation submodule and the scene graph detection submodule, emphasizing our experiments on the third submodule using a message passing neural network. Based on our experiments, we can safely say that the addition of such a network greatly boosts network performance, as it allows contextual information to propagate among objects and their relationships. Both the GGNN network and the GCN network improve upon the VRD baseline [46].

The addition of a relationship pruning network, while not studied thoroughly, seems to greatly enhance system performance. This is attributed to the robustness of the collected information, since the message routing is performed through a controlled propagation scheme and not towards all directions.

Despite that, even without the addition of relationship pruning network, the models manage to control the abundance of messages and the nodes learn to keep only the valuable information to update their internal representation. We hypothesize that this is due to the fact that even unconnected object pairs can contain meaningful contextual information about one another.

6.2 Future Work

The importance of the object region generation submodule in the overall system performance was hardly investigated. However, since it is the first submodule in our pipeline, its poor performance can hinder the consecutive modules' performance. Therefore, we propose retraining/finetuning the convolutional layers. Furthermore, using a higher quality object detection model can also greatly boost performance. As an example, we note Resnet-101 Faster R-CNN [24].

The relationship pruning network that was implemented in this work constitutes a strong indication on the importance of the relationship generation submodule. We suggest further experimentation on more elaborate relationship generation submodules. The implemented submodule could either prune relationships or propose them from scratch. We hypothesize that graph convolutional networks can also benefit from the existence of such a submodule, since their architecture originally targets semi-supervised learning problems, in which part of the groundtruth graph connections are known a priori.

As we discussed in section 4.3, the chosen evaluation metric is $R@k$. Despite the fact that this metric is a robust evaluation measure, it suffers from a major drawback; it does not compute the recall values for each class independently as a mean-average recall (mAR) metric would. Taking into account the fact that the predicate distribution is heavily skewed, $R@k$ allows the network to perform poorly in rare predicate classes without having a degraded performance, as long as it performs well in frequent predicate classes.

Hence, we propose the usage of a mean-average recall at k ($mAR@k$) metric or a modified mean-average precision (mAP) metric, such as the one used in Open Images [37] Visual Relationship Detection. This metric is a modified version of $mAP@0.5$ used in PASCAL VOC, in which we modify the definition of a false positive to ignore the cases that a predicted relationship does not appear in the groundtruth relationships of an image. Thus, it efficiently solves the problem that unidentified relationships can get falsely penalized, all while removing the drawback that $R@k$ has. Needless to say, one can use these metrics complementary to one another, using a weighted sum evaluation scheme.

Appendix A

Notation Conventions

Formula	**Description**
$\sigma(\cdot)$	Sigmoid function
$\rho(\cdot)$	Rectified Linear Unit
$\odot$	Hadamard Product Operator
$[\cdot,\cdot]$	Vector concatenation
$\mathbf{w}$	Weight Vector
$\mathbf{W}$	Weight Matrix

Table A.1: Mathematical Formulae & Notation Conventions

Bibliography

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems". In: *arXiv: 1603.04467 [cs]* (Mar. 14, 2016). URL: http://arxiv.org/abs/1603.04467 (cit. on p. 24).

[2] Bogdan Alexe, Thomas Deselaers, and Vittorio Ferrari. "Measuring the objectness of image windows". In: *IEEE transactions on pattern analysis and machine intelligence* 34.11 (2012), pp. 2189–2202 (cit. on p. 23).

[3] Yoshua Bengio. "Practical recommendations for gradient-based training of deep architectures". In: *arXiv:1206.5533 [cs]* (June 24, 2012). URL: http://arxiv.org/abs/1206.5533.

[4] James Bergstra and Yoshua Bengio. "Random Search for Hyper-Parameter Optimization". In: *Journal of Machine Learning Research* 13 (Feb 2012), pp. 281–305. ISSN: ISSN 1533-7928. URL: http://www.jmlr.org/papers/v13/bergstra12a.html (cit. on p. 21).

[5] Christopher Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. New York: Springer-Verlag, 2006. ISBN: 978-0-387-31073-2. URL: //www.springer.com/us/book/9780387310732.

[6] Xinlei Chen and Abhinav Gupta. "An Implementation of Faster RCNN with Study for Region Sampling". In: *arXiv:1702.02138 [cs]* (Feb. 7, 2017). URL: http://arxiv.org/abs/1702.02138.

[7] Kyunghyun Cho, Aaron Courville, and Yoshua Bengio. "Describing Multimedia Content using Attention-based Encoder–Decoder Networks". In: *arXiv:1507.01053 [cs]* (July 3, 2015). URL: http://arxiv.org/abs/1507.01053.

[8] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation". In: *arXiv:1406.1078 [cs, stat]* (June 3, 2014). URL: http://arxiv.org/abs/1406.1078 (visited on 10/16/2018) (cit. on pp. 7, 14).

[9] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. "Gated Feedback Recurrent Neural Networks". In: *arXiv:1502.02367 [cs, stat]* (Feb. 9, 2015). URL: http://arxiv.org/abs/1502.02367.

[10] Bo Dai, Yuqi Zhang, and Dahua Lin. "Detecting Visual Relationships with Deep Relational Networks". In: *arXiv:1704.03114 [cs]* (Apr. 10, 2017). URL: http://arxiv.org/abs/1704.03114 (cit. on p. 6).

[11] Jifeng Dai, Kaiming He, and Jian Sun. "Instance-aware Semantic Segmentation via Multi-task Network Cascades". In: *arXiv:1512.04412 [cs]* (Dec. 14, 2015). URL: http://arxiv.org/abs/1512.04412 (cit. on p. 5).

[12] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. "R-FCN: Object Detection via Region-based Fully Convolutional Networks". In: *arXiv:1605.06409 [cs]* (May 20, 2016). URL: http://arxiv.org/abs/1605.06409 (cit. on p. 5).

[13] Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. "Language Modeling with Gated Convolutional Networks". In: *arXiv:1612.08083 [cs]* (Dec. 23, 2016). URL: http://arxiv.org/abs/1612.08083.

[14] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. "Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering". In: *arXiv:1606.09375 [cs, stat]* (June 30, 2016). URL: http://arxiv.org/abs/1606.09375.

[15] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. "The Pascal Visual Object Classes Challenge: A Retrospective". In: *International Journal of Computer Vision* 111.1 (Jan. 2015), pp. 98–136 (cit. on p. 1).

[16] Jonas Gehring, Michael Auli, David Grangier, and Yann N. Dauphin. "A Convolutional Encoder Model for Neural Machine Translation". In: *arXiv:1611.02344 [cs]* (Nov. 7, 2016). URL: http://arxiv.org/abs/1611.02344.

[17] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. "Convolutional Sequence to Sequence Learning". In: *arXiv:1705.03122 [cs]* (May 8, 2017). URL: http://arxiv.org/abs/1705.03122.

[18] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. "Neural Message Passing for Quantum Chemistry". In: *arXiv:1704.01212 [cs]* (Apr. 4, 2017). URL: http://arxiv.org/abs/1704.01212 (cit. on p. 12).

[19] Ross Girshick. "Fast R-CNN". In: *arXiv:1504.08083 [cs]* (Apr. 30, 2015). URL: http://arxiv.org/abs/1504.08083 (cit. on pp. 1, 5).

[20] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *arXiv:1311.2524 [cs]* (Nov. 11, 2013). URL: http://arxiv.org/abs/1311.2524 (cit. on pp. 1, 5, 6, 22).

[21] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. Mar. 31, 2010, pp. 249–256. URL: http://proceedings.mlr.press/v9/glorot10a.html (cit. on p. 21).

[22] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[23] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. "Mask R-CNN". In: *arXiv:1703.06870 [cs]* (Mar. 20, 2017). URL: http://arxiv.org/abs/1703.06870 (cit. on pp. 6, 21).

[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition". In: *arXiv:1512.03385 [cs]* (Dec. 10, 2015). URL: http://arxiv.org/abs/1512.03385 (cit. on pp. 1, 5, 30).

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification". In: *arXiv: 1502.01852 [cs]* (Feb. 6, 2015). URL: http://arxiv.org/abs/1502.01852 (cit. on pp. 1, 21).

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Identity Mappings in Deep Residual Networks". In: *arXiv:1603.05027 [cs]* (Mar. 16, 2016). URL: http://arxiv.org/abs/1603.05027 (cit. on p. 5).

[27] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, et al. "Speed/accuracy trade-offs for modern convolutional object detectors". In: *arXiv:1611.10012 [cs]* (Nov. 30, 2016). URL: http://arxiv.org/abs/1611.10012.

[28] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *arXiv:1502.03167 [cs]* (Feb. 10, 2015). URL: http://arxiv.org/abs/1502.03167.

[29] Justin Johnson, Agrim Gupta, and Li Fei-Fei. "Image Generation from Scene Graphs". In: *arXiv:1804.01622 [cs]* (Apr. 4, 2018). URL: http://arxiv.org/abs/1804.01622 (cit. on p. 7).

[30] 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: (Dec. 20, 2016). URL: https://arxiv.org/abs/1612.06890 (cit. on p. 2).

[31] Justin Johnson, Andrej Karpathy, and Li Fei-Fei. "DenseCap: Fully Convolutional Localization Networks for Dense Captioning". In: *arXiv:1511.07571 [cs]* (Nov. 24, 2015). URL: http://arxiv.org/abs/1511.07571.

[32] Justin Johnson, Ranjay Krishna, Michael Stark, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Li Fei-Fei. "Image retrieval using scene graphs". In: *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*. IEEE, 2015, pp. 3668–3678. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=7298990 (cit. on pp. 1, 7).

[33] Andrej Karpathy and Li Fei-Fei. "Deep visual-semantic alignments for generating image descriptions". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 3128–3137. URL: http://www.cv-foundation.org/openaccess/content_cvpr_2015/html/Karpathy_Deep_Visual-Semantic_Alignments_2015_CVPR_paper.html.

[34] Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. "Molecular graph convolutions: moving beyond fingerprints". In: *Journal of computer-aided molecular design* 30.8 (2016), pp. 595–608 (cit. on p. 13).

[35] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *arXiv:1412.6980 [cs]* (Dec. 22, 2014). URL: http://arxiv.org/abs/1412.6980 (cit. on p. 21).

[36] Thomas N. Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks". In: *arXiv:1609.02907 [cs, stat]* (Sept. 9, 2016). URL: http://arxiv.org/abs/1609.02907 (cit. on p. 15).

[37] Ivan Krasin, Tom Duerig, Neil Alldrin, Vittorio Ferrari, Sami Abu-El-Haija, Alina Kuznetsova, Hassan Rom, Jasper Uijlings, Stefan Popov, Shahab Kamali, et al. "OpenImages: A public dataset for large-scale multi-label and multi-class image classification." In: (2017) (cit. on pp. 2, 30).

[38] 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". In: *arXiv:1602.07332 [cs]* (Feb. 23, 2016). URL: http://arxiv.org/abs/1602.07332 (cit. on pp. 2, 7–9, 19, 26).

[39] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger. Curran Associates, Inc., 2012, pp. 1097–1105. URL: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf (cit. on pp. 1, 5).

[40] Yi Li, Haozhi Qi, Jifeng Dai, Xiangyang Ji, and Yichen Wei. "Fully Convolutional Instance-aware Semantic Segmentation". In: *arXiv:1611.07709 [cs]* (Nov. 23, 2016). URL: http://arxiv.org/abs/1611.07709 (cit. on p. 5).

[41] Yikang Li, Wanli Ouyang, Bolei Zhou, Kun Wang, and Xiaogang Wang. "Scene Graph Generation from Objects, Phrases and Caption Regions". In: *arXiv:1707.09700 [cs]* (July 30, 2017). URL: http://arxiv.org/abs/1707.09700 (cit. on p. 6).

[42] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. "Gated graph sequence neural networks". In: *arXiv preprint arXiv:1511.05493* (2015) (cit. on p. 14).

[43] Xiaodan Liang, Lisa Lee, and Eric P. Xing. "Deep Variation-structured Reinforcement Learning for Visual Relationship and Attribute Detection". In: *arXiv:1703.03054 [cs]* (Mar. 8, 2017). URL: http://arxiv.org/abs/1703.03054 (cit. on p. 6).

[44] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. "Microsoft COCO: Common Objects in Context". In: *arXiv:1405.0312 [cs]* (May 1, 2014). URL: http://arxiv.org/abs/1405.0312 (cit. on pp. 1, 7).

[45] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. "SSD: Single Shot MultiBox Detector". In: (Dec. 8, 2015). DOI: 10.1007/978-3-319-46448-0_2. URL: https://arxiv.org/abs/1512.02325 (visited on 10/11/2018) (cit. on p. 5).

[46] Cewu Lu, Ranjay Krishna, Michael Bernstein, and Li Fei-Fei. "Visual Relationship Detection with Language Priors". In: *arXiv:1608.00187 [cs]* (July 31, 2016). URL: http://arxiv.org/abs/1608.00187 (cit. on pp. 6, 23, 26, 29).

[47] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. "Efficient estimation of word representations in vector space". In: *arXiv preprint arXiv:1301.3781* (2013) (cit. on p. 6).

[48] George A. Miller. "WordNet: A Lexical Database for English". In: *Commun. ACM* 38.11 (Nov. 1995), pp. 39–41. ISSN: 0001-0782. DOI: 10.1145/219717.219748. URL: http://doi.acm.org/10.1145/219717.219748 (cit. on p. 7).

[49] Yurii Nesterov. "A Method for Solving a Convex Programming Problem with Convergence Rate O(1/k^2)". In: *Soviet Mathematics Doklady* 27 (1983), pp. 372–376 (cit. on p. 21).

[50] Yurii Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Applied Optimization. Springer US, 2004. ISBN: 978-1-4020-7553-7. URL: //www.springer.com/us/book/9781402075537 (cit. on p. 21).

[51] Alejandro Newell and Jia Deng. "Pixels to Graphs by Associative Embedding". In: *arXiv:1706.07365 [cs]* (June 22, 2017). URL: http://arxiv.org/abs/1706.07365 (cit. on p. 6).

[52] B. T. Polyak. “Some methods of speeding up the convergence of iteration methods”. In: *USSR Computational Mathematics and Mathematical Physics* 4.5 (1964), pp. 1–17. ISSN: 0041-5553. DOI: https://doi.org/10.1016/0041-5553(64)90137-5. URL: http://www.sciencedirect.com/science/article/pii/0041555364901375 (cit. on p. 21).

[53] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. “On the convergence of adam and beyond”. In: (2018).

[54] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. “You Only Look Once: Unified, Real-Time Object Detection”. In: *arXiv:1506.02640 [cs]* (June 8, 2015). URL: http://arxiv.org/abs/1506.02640 (cit. on p. 5).

[55] Joseph Redmon and Ali Farhadi. “YOLO9000: Better, Faster, Stronger”. In: *arXiv: 1612.08242 [cs]* (Dec. 25, 2016). URL: http://arxiv.org/abs/1612.08242 (visited on 10/11/2018) (cit. on p. 5).

[56] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”. In: *arXiv:1506.01497 [cs]* (June 4, 2015). URL: http://arxiv.org/abs/1506.01497 (cit. on pp. 1, 5, 20).

[57] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. “ImageNet Large Scale Visual Recognition Challenge”. In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y (cit. on p. 1).

[58] Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. “Modeling Relational Data with Graph Convolutional Networks”. In: *arXiv:1703.06103 [cs, stat]* (Mar. 17, 2017). URL: http://arxiv.org/abs/1703.06103.

[59] Sebastian Schuster, Ranjay Krishna, Angel Chang, Li Fei-Fei, and Christopher D. Manning. “Generating semantically precise scene graphs from textual descriptions for improved image retrieval”. In: *Proceedings of the Fourth Workshop on Vision and Language*. 2015, pp. 70–80. URL: http://www-nlp.stanford.edu/pubs/schuster-krishna-chang-feifei-manning-vl15.pdf.

[60] Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. “Training Region-based Object Detectors with Online Hard Example Mining”. In: *arXiv:1604.03540 [cs]* (Apr. 12, 2016). URL: http://arxiv.org/abs/1604.03540.

[61] Karen Simonyan and Andrew Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: *arXiv:1409.1556 [cs]* (Sept. 4, 2014). URL: http://arxiv.org/abs/1409.1556 (cit. on pp. 1, 5, 6, 20).

[62] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15 (2014), pp. 1929–1958. URL: http://www.jmlr.org/papers/v15/srivastava14a.html (cit. on p. 21).

[63] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. "On the importance of initialization and momentum in deep learning". In: *International Conference on Machine Learning*. International Conference on Machine Learning. Feb. 13, 2013, pp. 1139–1147. URL: http://proceedings.mlr.press/v28/sutskever13.html (cit. on p. 21).

[64] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going Deeper with Convolutions". In: *arXiv:1409.4842 [cs]* (Sept. 16, 2014). URL: http://arxiv.org/abs/1409.4842 (cit. on p. 5).

[65] Damien Teney, Lingqiao Liu, and Anton van den Hengel. "Graph-Structured Representations for Visual Question Answering". In: *arXiv:1609.05600 [cs]* (Sept. 19, 2016). URL: http://arxiv.org/abs/1609.05600 (visited on 10/18/2018) (cit. on p. 7).

[66] 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". In: *Communications of the ACM* 59.2 (Jan. 25, 2016), pp. 64–73. ISSN: 00010782. DOI: 10.1145/2812802. URL: http://arxiv.org/abs/1503.01817 (cit. on p. 7).

[67] Tijmen Tieleman and Geoffrey Hinton. "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude". In: *COURSERA: Neural networks for machine learning* 4.2 (2012), pp. 26–31 (cit. on p. 21).

[68] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. "Selective search for object recognition". In: *International journal of computer vision* 104.2 (2013), pp. 154–171 (cit. on p. 5).

[69] Danfei Xu, Yuke Zhu, Christopher B. Choy, and Li Fei-Fei. "Scene Graph Generation by Iterative Message Passing". In: *arXiv:1701.02426 [cs]* (Jan. 9, 2017). URL: http://arxiv.org/abs/1701.02426 (cit. on pp. 6, 23, 26, 27).

[70] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. "Embedding Entities and Relations for Learning and Inference in Knowledge Bases". In: *arXiv:1412.6575 [cs]* (Dec. 19, 2014). URL: http://arxiv.org/abs/1412.6575.

[71] Matthew D. Zeiler and Rob Fergus. "Visualizing and Understanding Convolutional Networks". In: *arXiv:1311.2901 [cs]* (Nov. 12, 2013). URL: http://arxiv.org/abs/1311.2901 (cit. on p. 5).