FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Visually Perceiving Symbolic Representation for Manipulation Tasks in Robotics

Leandro Pereira

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Univ.-Prof. Dr.-Ing. Dongheui Lee

Second Supervisor: Dr. Hyemin Ahn

Third Supervisor: Prof. Anibal Matos

November 12, 2021

Resumo

Para realizar um planeamento de tarefas de manipulação, um robô autónomo deve perceber o estado do ambiente do seu espaço de trabalho para ser capaz de tomar decisões correctas. Isto pode ser feito através do reconhecimento dos objectos e da compreensão da relação visual entre os diferentes pares de objectos na cena. Encontrar a relação entre dois objectos implica identificar o sujeito, o objecto, e um predicado que os relaciona. Para caracterizar estas relações são frequentemente utilizadas representações simbólicas do ambiente observado (por exemplo, livro-sobre-mesa). Trabalhos anteriores não utilizam a detecção de relações para manipulação robótica, pelo que nesta tese propomos um modelo baseado em *deep learning* capaz de reconhecer objectos e prever as relações entre eles num ambiente adequado para a manipulação robótica. Para treinar um modelo de rede neural é muitas vezes necessária uma enorme quantidade de dados, no entanto, isto pode ser difícil e dispendioso de obter. Para enfrentar este problema, propomos um novo conjunto de dados sintéticos cobrindo um total de três tarefas robóticas diferentes que são utilizadas para treinar o nosso modelo baseado em *deep learning*.

Abstract

In order to perform a task planning of manipulation, an autonomous robot must perceive the status of the environment of its workspace to be able to make correct decisions. This can be done by recognizing the objects and understanding the visual relationship between the different pairs of objects in the scene. Finding the relationship between two objects involves identifying the subject, the object, and a predicate relating them. To characterize these relationships is often used symbolic representations of the observed environment (i.e., book-under-cup). Previous works did not use relationship detections for robotic manipulation, therefore in this thesis we propose a deep learning-based model capable of recognising objects and predicting the relationships between them in an environment suitable for manipulation robots. To train a neural network model is often necessary a huge amount of data, however, this can be difficult and expensive to obtain. To tackle this problem, we propose a new synthetic dataset covering a total of three different robotic tasks that is used to train our deep learning-based model.

Contents

List of Figures

List of Tables

Abbreviations and Symbols

AP	Average Precision
CNN	Convolutional Neural Network
CPU	Central Processing Unit
FC	Fully Connected
GPU	Graphical Processing Unit
mAP	Mean Average Precision
NN	Neural Network
SSD	Single Shot Detector
TAMP	Task and Motion Planning
VGG	Visual Geometry Group
VQA	Visual Question Answering

Chapter 1

Introduction

Over the past half century, robots have become part of our everyday life. They help us to clean the house, perform surgical operations and carry out a wide range of manufacturing operations with high speed and precision. The precision is usually obtained using a rigorous mechanical design. Furthermore, the robots operate in controlled and familiar environments, which makes task execution easier and decreases monitoring equipment. The potential for autonomous manipulation applications is huge: robots capable of manipulating the environment can be deployed in hospitals, child and elderly care, factories, outer space, restaurants, service industries, and homes. Even in a specialized environments such as food preparation, this wide deployment scenario and unsystematic environmental changes indicate that an effective manipulation robot must be able to handle environments that the engineers did not anticipate, or that the robot did not encounter before. Until some years ago, robots were only used to perform repetitive tasks. Nowadays, robots are becoming more intelligent with the use of machine learning. Several techniques in the field of deep learning [4], reinforcement learning [5], and imitation learning [6], allowed robots to learn how to perform complicated tasks such as walking or preparing meals with high precision and accuracy.

When it comes to understand the environment where the robot operates, computer vision plays an important role. Computer vision aims to extract relevant information from images or videos collected by cameras in order to gain a high-level understanding of the scenes. It has enabled a big advance in face detection, image captioning, medical image understanding, biometrics, and many more [7, 8, 9]. In robotics, this is called robot vision. Without robot vision the robot is blind and can only move according to its programming being ideal for repetitive tasks. But when it comes to have a multi-purpose robot or a robot that can take decisions, this plays an important role for the robot to understand its surroundings. To create a reliable vision system for a robot an enormous amount of data is necessary to train deep neural networks capable of recognizing scenarios, objects and the actions performed. One of the most challenge problems to solve when modeling a deep neural network is to choose the proper data and the proper format. Having the proper data means that the collected data must correlate with the outcome we want to predict. The data used must be aligned with the problem the robot wants to solve, for example, images of dogs

are not useful when the goal is to implement a facial identification system in the robot. Training a deep learning model requires to have a huge amount of labeled data, this data is usually labeled by humans manually which makes the process inefficient and costly. Moreover, the labeled can contain errors and be inconsistent.

1.1 Problem Statement

In the era of AI, data is a very important resource, and it is known that the quality and the quantity of the dataset will have a direct impact on performance of the model trained. The problem is that many times a big quantity of good quality data is not available, or it is very costly or challenging to obtain.

Simulating data in a 3D virtual environment gives us the possibility of changing every parameter that will have an impact in the final rendered image. We can generate images that cover all possible light scenarios as well as all camera positions. Additionally, the ground-truth label is highly precise and does not contain errors that could be introduced by manual annotation. Contrarily to manual annotations in which we need a lot of human time to annotate thousands of images, in synthetic datasets only CPU time is consumed for the generation of these annotations and the rendering process. Furthermore, we can generate as many images as we want in a much shorter time.

In order to perform a task planning [10], a robot must perceive which objects are in the scene, where they are localized and what their relationships are. Finding the visual relationship between two objects involves identifying the objects and a predicate relating them [11]. To characterize these relationships, symbolic representations of the observed environment are often used, for instance, `book-under-cup`, when the book is localized under the cup. This representation uses an asserting notation compatible with human language which facilitates the human-robot interaction. This allows humans to ask the robot to change the current status of the environment in a intuitive way, for instance, change the relationships `book-on-bookcase` to `book-on-table`. Moreover, as a mid-level learning task, detecting visual relationships may improve many image comprehension tasks such as image captioning and visual question answering. Unlike image understanding based on a single object, visual relationship detection is used to describe two objects which greatly increases the number of possible relationships. Using the representation for relations as a subject-predicate-object, the number of possible relations is $O(N^2K)$ when there are N types of objects and K predicates.

1.2 Related Work

In order to understand the current state of the art of the methodologies to solve the problem presented, we investigated several previous works. The next subsections describe the current work achieved by other researchers. Additionally, we also present works that will be used to solve the stated problem.

1.2.1 Synthetic Dataset

Over the years, computer graphics have improved considerably, making it possible to create virtual images and videos that are very close to reality. Due to the difficulties in acquiring data for machine learning training, many researchers [12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27] have used generated synthetic data to train their models. When it comes to computer vision, the images or videos are generated using game engines or 3D computer graphics software's. Synthetic data has been used successfully in [12] where an engine was used to generate the images to train a neural network capable of recognizing text in pictures.

In [13] free 3D models were used to generate images to train a deep CNN object detector. They showed better results than models trained on real images when these come from different domains. Other works [14, 15] also focused in object detection used synthetic images generated using Blender and Unity 3D. In the solution using Blender, the Cycles engine is used which renders images with higher quality and more realistic. The solutions presented focused in getting as close to reality as possible which is not a problem when only a small number of images is necessary. When a huge number of images need to be generated, the process can be accelerated by losing some image quality.

The CLEVR [19] dataset represents a diagnostic dataset to test a range of visual reasoning abilities. The dataset contains detailed descriptions to represent the reasoning that each question requires. This dataset can be used to conduct rich diagnostics to have a better understanding of the visual capabilities of Visual Question Answering. This requires tight control over the dataset, therefore the researchers generated synthetic images and automatically generated questions. The images have associated ground-truth object locations and attributes, and the questions have an associated machine-readable form. Their images were generated by randomly sampling a scene graph and rendering it using Blender [28]. In this dataset, every scene contains between three and ten objects with random shapes, sizes, materials, colors, and positions. The researchers ensured no objects intersected, that all objects are at least partially visible, and that there are small horizontal and vertical margins between the image-plane centers of each pair of objects. Moreover, the position of the light and the camera were randomly jittered in each image. In this case, simple 3D object shapes were used (cubes, spheres and cylinders) with different materials in eight different colors. They also uses four spatial relationships (left, right, behind, in front). These semantics depend not only on the relative position of the objects but also on the perspective of the camera.

In the last years the number of 3D models available online increased substantially in repositories like Trimble 3D warehouse, providing millions of 3D CAD models covering thousands of object categories. However, many of these models are not well organized or and often grouped in gross categories, moreover, it is not common to see a textual description being provided together with the model. To solve this problem, researchers presented the ShapeNET [29] dataset which is a richly-annotated, large-scale repository of shapes represented by 3D CAD models of objects. The dataset has indexed three million models, with 220000 models classified in 3135 categories. They were able to collect and centralize 3D models that can be used for data-driven methods,

evaluation and comparison of algorithms for fundamental tasks involving geometry and serve as base knowledge for representing real-world objects and their semantics.

1.2.2 Relationship Detection

A paper published in 2016 [30], suggested a method to detect visual relationships between objects in images. The researchers proposed a visual appearance module that learns the appearance of objects and predicates, and fuses them together to jointly predict relationships. They showed that their model was able to detect $O(N^2K)$ relationships by only using $O(N+K)$ relationships. To detect infrequent relationships, this model proposed also a language module that uses pre-trained word vectors to cast relationships into a vector space where similar relationships are optimized to be close to each other. This makes the proposed method be able to detect infrequent or unseen relationships due to the semantic similarity between different relationships. The proposed model outperformed previous methods as Visual Phrases [31] and Image Scene Graphs [32]. In this paper [30], they also proposed a new dataset suitable to detect visual relationships since the existing datasets did not contain sufficient variety of relationships or predicate diversity per object category. One of the weakness of their method is the separation between the extraction of visual appearance features and the final task. To tackle this problem, several papers were published in 2017, [33, 34, 35, 36], however, in most of these papers the author choose to model relationship detection as a multi-class classification problem ignoring the relationship co-occurrence.

In a new paper published in 2018 [37], the author proposed a deep structural ranking framework for visual relationship detection. They proposed a model based on a convolutional neural network which combines different cues to learn the representation for an input instance. In order to facilitate the co-occurrence and incompleteness of visual relationships, they proposed a structure of ranking loss which enforces the annotated relationships to have higher relevance score since the annotated relationships are usually more salient. In their experiments, they used two important tasks to evaluate the proposed method: Predicate detection and Relationship detection. On both learning tasks, the proposed method outperformed the most recent state-of-the-art methods dramatically. In fact, for predicate detection, they were able to improve the state-of-the-art by around 11% according to Recall@100 on the datasets tested and for relationship detection, about 2.4% according to the same metric.

All these works were applied in datasets containing images of environments found by humans in their everyday lives and, therefore, not focused in manipulation relationships.

The closest work to ours was published in 2018 [38], the researchers aimed to help a robot to infer the grasping order of a stack of objects. They proposed an end-to-end neural network that receives an image as input and predicts the object classes, their locations and a manipulation relationship tree. However, they only focused in detecting the stack order of the objects, excluding any other possible relationships between the objects.

1.3 Our Approach

Many approaches that tried to detect visual relationships from images only covered scenarios found by humans in their daily lives, for example, `person-wear-shirt` or `person-ride-horse`. These scenarios are not of interest to manipulation robots because it is not intended that they interact with these kind of objects and predicates (e.g., horse, ride, wear).

In this thesis, we are interested in detecting relationships in objects that will possibly be found in manipulation tasks. We propose a deep learning-based model capable of localizing the objects, predict their category and the relationships between the pairs of objects from an RGB image. Manipulation robots interact with tools and objects that are frequently small and easy to operate, therefore the dataset to train our model must capture the variety of these objects in their different states. Since previous datasets did not cover this kind of environments, we propose a new image dataset covering three possible manipulation tasks. We chose to build a synthetic dataset since a large amount of data is hard and expensive to obtain. To render the dataset images we use Blender [28] to automatically generate the scenes and randomly change the conditions of the environment covering a wider range of possibilities. We use 3D models of objects from the ShapeNET dataset [29] and using Blender API we randomly display them on a table. Lastly, we use Python [39] to generate the annotations of the corresponding generated image.

We use our dataset to train our deep neural network-based model aiming to classify the object category and predict the relationships between the pairs of objects in the scene. Contrarily to the work proposed in [38] we aim to detect every possible relationship between the objects. Our model is divided in two parts: the first is responsible for object detection, for this we use the SSD300 due to its real time properties and good accuracy; the second is responsible for relationship prediction, this model uses a visual appearance cue, responsible to extract the features of the object and a spatial cue to learn the relative spatial location of the objects.

With this information, the robot can infer not only the grasping order of the objects when these are stacked but also be able to organize objects, or even answer to questions about the scenes which can be useful for Task and Motion Planning (TAMP).

Chapter 2

Technical Approach

This chapter explains the technical details of the chosen approach in this thesis. It starts by describing how the synthetic dataset was generated. Secondly, we describe how we modeled our deep learning based-model capable of recognizing objects and the predicate between them.

2.1 Synthetic Dataset

Acquiring data for neural network training is an expensive and labor-intensive task, especially when such data is difficult to access. We proposed a new synthetic dataset that covers the scenarios we are interested in training our neural network model on. To generate the synthetic images we used the 3D Blender software as a tool to automatically generate synthetic images using 3D models from the ShapeNET dataset [29]. We used Blender Python API [28] to automate the scene rendering by automating the loading of the 3D models into the scene and its positioning, and the generation of the ground-truth information. We used Cycles Render Engine since it supports ray-tracing to render the synthetic images which improves their quality.

Real world images embed a lot of information, for instance, the environment, the lightning, the shapes, the texture and so on. In order to make the synthetic images closer to real world images, giving the possibility of transferring the learning to the real world applications, we considered the following aspects during the generation of each scenario:

- Illumination
- Position and orientation of the camera
- Positioning of the objects
- Texture, shape, and materials of the objects
- Number of objects
- Number of relationships.

We also took into consideration of which tasks the robot can perform, in order to decide which 3D models should we use to render the dataset. Regarding this, we proposed three different tasks:

- Task 1: Pouring water in a mug,
- Task 2: Setting a dinner table,
- Task 3: Stacking storage food containers.

For all the tasks we used the object `Table` as the platform where the robot will perform the manipulation. Additionally, we considered the following objects for every task:

- Task 1: Mug, Book, Bottle
- Task 2: Plate, Fork, Knife, Spoon, Wine Glass
- Task 3: Storage Food Containers.

From the ShapeNET dataset, we selected different 3D models for each object category and added them to our model repository (M_0). We used a total of 42 distinct models for Task 1, 33 for Task 2 and a total of 12 for Task 3. We also used 27 models of the object `Table` that are shared among the different tasks. Every category has a different number of models due to different availability of models in the ShapeNET dataset.

To generate a diverse dataset, the 3D scenes must be distinct. We should not only maximize the variety of objects across the dataset but also the different relationships in the dataset. Therefore, we started by randomly selecting the table model and loading it into the scene. Then we randomly chose the number of object models in the scene between 2 or 3 models. After that, we randomly selected a predicate from our list of possible predicates (P_0). For the given predicate, we randomly selected an object category for the subject. For the selection of the object, it is necessary to filter out the object categories that are not physically possible to occur given the predicate and the subject. For example, if the selected predicate is `inside` and the subject category is `mug`, `bottle` must be removed from the object candidates since it is not possible to have the relationship mug-inside-bottle. If the random number of objects is three, we repeat the process by selecting another predicate, a 3D model and applying the same restrictions to make sure that the relationships generated were as intended.

To place the objects on the table in such a way that the predicate between the models is present, we applied geometric constrains to generate the possible 3D space positions for the models, from which we randomly selected a 3D point. After ensuring that all the objects are positioned without intersecting each other and they are inside the table area, we randomly varied the illumination of the scene. This ensures that the images are not biased to a well illuminated environment since this can be applied in scenarios with brighter or dimmer lighting. To increase the range of camera perspectives in our dataset, we set a different position and orientation of the camera for every scene. Then, we confirmed whether all the models are completely visible by the camera, except for the table that can be partially occluded. The bounding box information of the models present in the scene was extracted by using a world-to-camera function that maps every 3D coordinate in the scene to a 2D image-space coordinate. This information, together with the relationships between the different objects in the scene and their categories, is used to generate the annotations for the rendered image. Figure 2.2 shows examples of synthetically generated images.

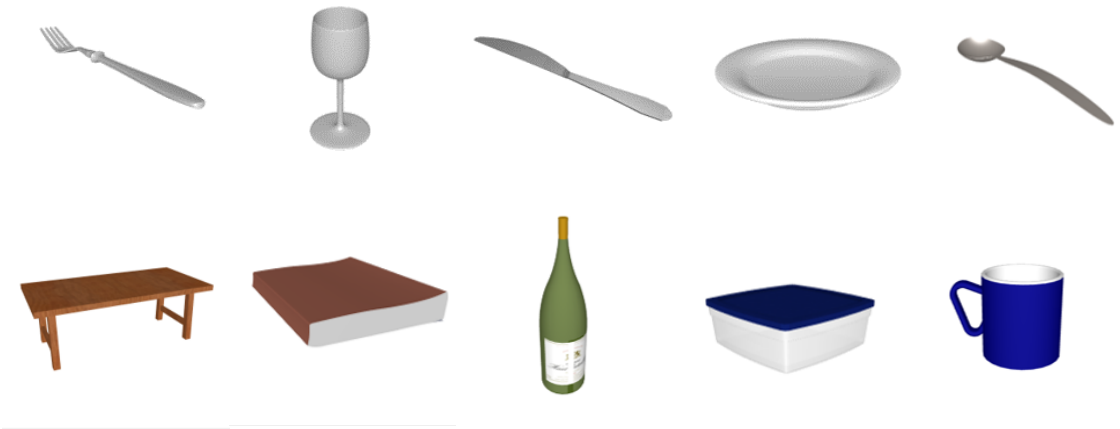


Figure 2.1: Examples of 3D models used in the generation of the synthetic dataset.

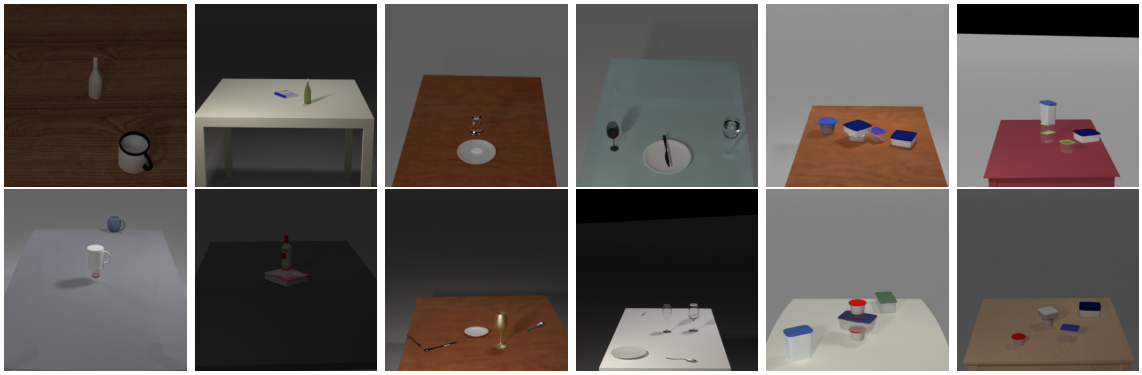


Figure 2.2: Examples of the synthetic generated images for the different tasks

2.1.1 Generated Datasets

The datasets were rendered on a workstation with Intel(R) Core(TM) i7-4790K CPU @ 4.00GHz and with the GPU NVIDIA GeForce RTX 2080 SUPER using the Cycles render from Blender.

2.1.1.1 Manipulation Task 1

For the first manipulation task, we generate 5000 images. The graphs in figure 2.3 shows the distribution of relationships and objects. In the object distribution, we can see that the objects are approximately equally distributed.

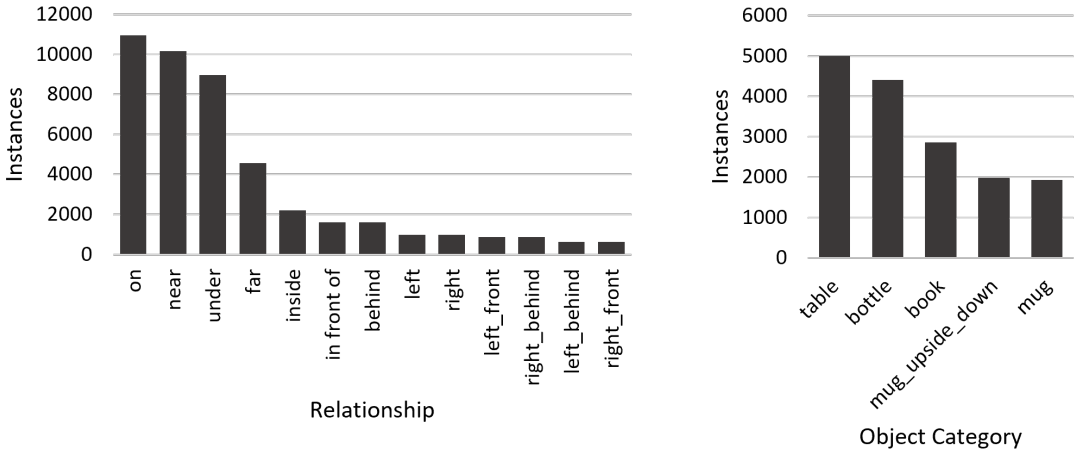


Figure 2.3: (Left) Number of relationship instances per predicate in the dataset generated for Task 1. (Right) Number of object instances per category in the dataset generated for Task 1.

Images	Rel. Types	Rel. Instances	Obj. Instances	Pred. per Image	Obj. per Image
5000	13	45056	16185	9.0112	3.237

Table 2.1: Statistics of the synthetic dataset generated for Task 1

2.1.1.2 Manipulation Task 2

In the second task, more objects are used per image. In addition, higher variety of objects makes the scene more complex with more predicates per image (see Table 2.2). In this task, regarding the relationship distribution, we have similar results to the previous task, however, here we do not have the relationship `inside` given that we do not have objects that can be inside of some other object.

2.1.1.3 Manipulation Task 3

The third task is the simplest when it comes to the variety of object categories, having only two: `table` and `box`. Here the goal is stacking the boxes and organize them.

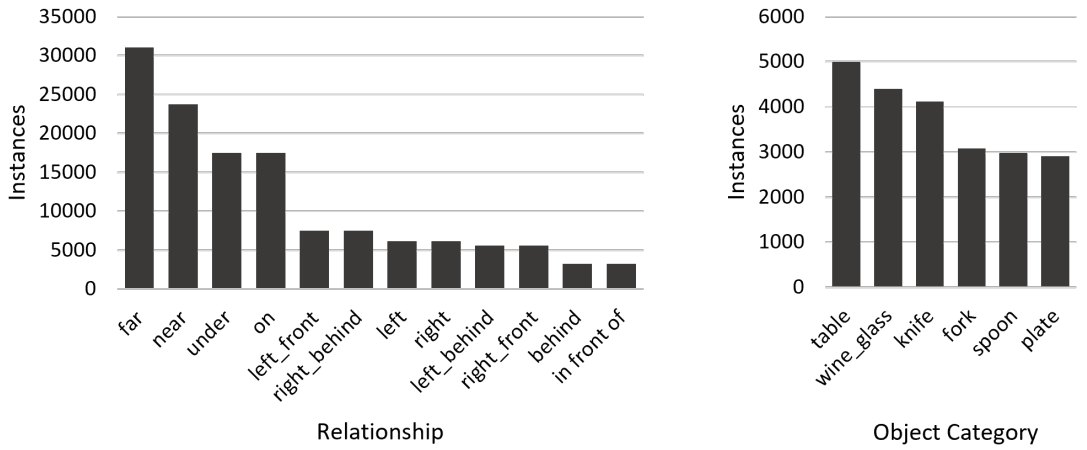


Figure 2.4: (Left) Number of relationship instances per predicate in the dataset generated for Task 2. (Right) Number of object instances per category in the dataset generated for Task 2.

Images	Rel. Types	Rel. Instances	Obj. Instances	Pred. per Image	Obj. per Image
5000	12	134800	22467	26.96	4.4934

Table 2.2: Statistics of the synthetic dataset generated for Task 2

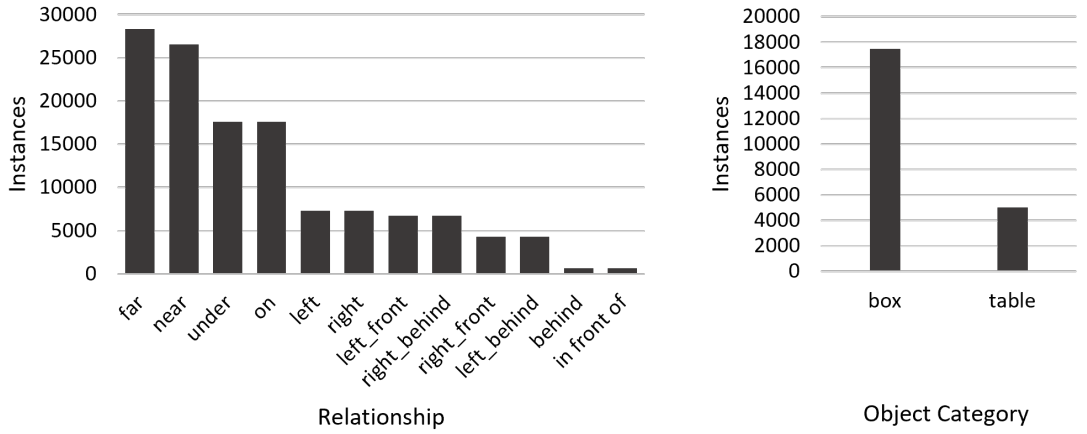


Figure 2.5: (Left) Number of relationship instances per predicate in the dataset generated for Task 3. (Right) Number of object instances per category in the dataset generated for Task 3.

Images	Rel. Types	Rel. Instances	Obj. Instances	Pred. per Image	Obj. per Image
5000	12	127910	22473	25.582	4.4946

Table 2.3: Statistics of the synthetic dataset generated for Task 3

2.2 Neural Network Structure

In this work, we aim to implement a neural network model that receives a RGB image as input and is capable of classifying the objects present in a scene, localize them and predict the relationships between each pair of objects. The final output is a list of symbolic representations of the observed environment. Figure 2.6 illustrates the general goal of our model.

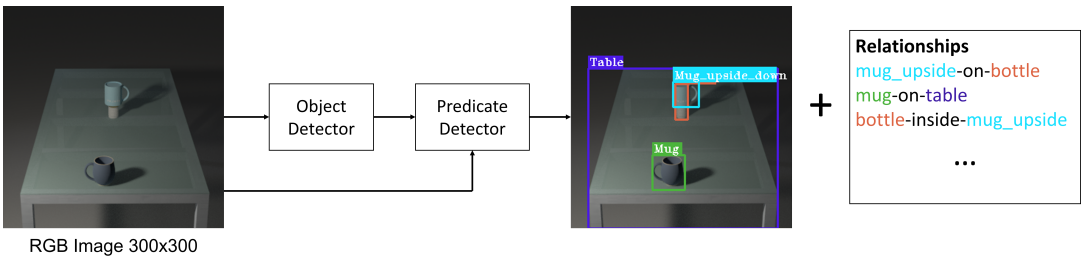


Figure 2.6: General structure of the Neural Network Model

Our proposed method for relationship detection is split in two models: The first is responsible for localizing the objects and predicting their category. The second receives the proposed objects from the first model as an input, and predicts the relationships between the detected objects.

All the developed code related with the neural networks was implemented in Python 3.7.0 using the library Pytorch. We used the Google Colab notebooks to write and execute the code.

2.2.1 Object Detection

For the first module of our neural network, which is responsible for object detection, we based our model in the Single Shot MultiBox Detector (SSD)[1] structure, given its real-time capacity and the well performance demonstrated in [1]

Other architectures for object detection, such as Fast-RCNN [40] and Faster-RCNN [41], use two distinct stages, a region proposal network that performs object localization and a classifier for detecting the categories of the proposed regions. This makes this networks slower than the SSD that only needs one single shot to detect multiple objects within the image. This network is also faster than other single shot detectors (You Only Look Once [42]).

2.2.1.1 Single Shot MultiBox Detector (SSD)

SSD is a pure Convolutional Neural Network (CNN) that can be distinguished in three parts: the base convolutions used from an existing image classification architecture that will provide the low-level feature maps, the auxiliary convolution layers that will supply the higher level features, and the prediction convolution layers that will identify and localize the objects using the feature maps. The SSD300 structure is represented by figure 2.7.

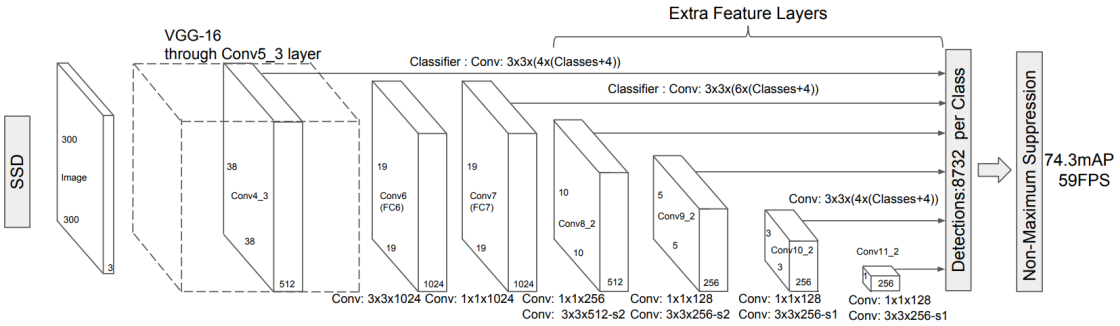


Figure 2.7: Structure of the Single Shot MultiBox Detector. Image credit: [1]

The **base convolution layers** are built from the VGG-16 [2] architecture, all the layers are used except for the fully connected layers, figure 2.8 shows the representation of the layers of the network. VGG-16 [2] is used due to its high performance in classification tasks, as well as the possibility of using these layers already pre-trained on a well founded classification dataset. By using transfer learning from a different but closely related task, we have made some progress in our training task, reducing the time of training. We used one pre-trained on the *ImageNet Large Scale Visual Recognition Competition (ILSVRC)* [43] classification task that is already available in PyTorch. The fully connected layers from the VGG used for classification in the original network are not used here since they are used for object classification, instead they reworked into convolution layers conv6 and conv7 by reshaping its parameters. Due to the different input image size, we add to make some adjustments in the base convolution layers, namely in the 5rd pooling layer from a 2×2 kernel and 2 stride to a 3×3 kernel and 1 stride to avoid the halve dimensions of the feature map from the preceding convolutions layer.

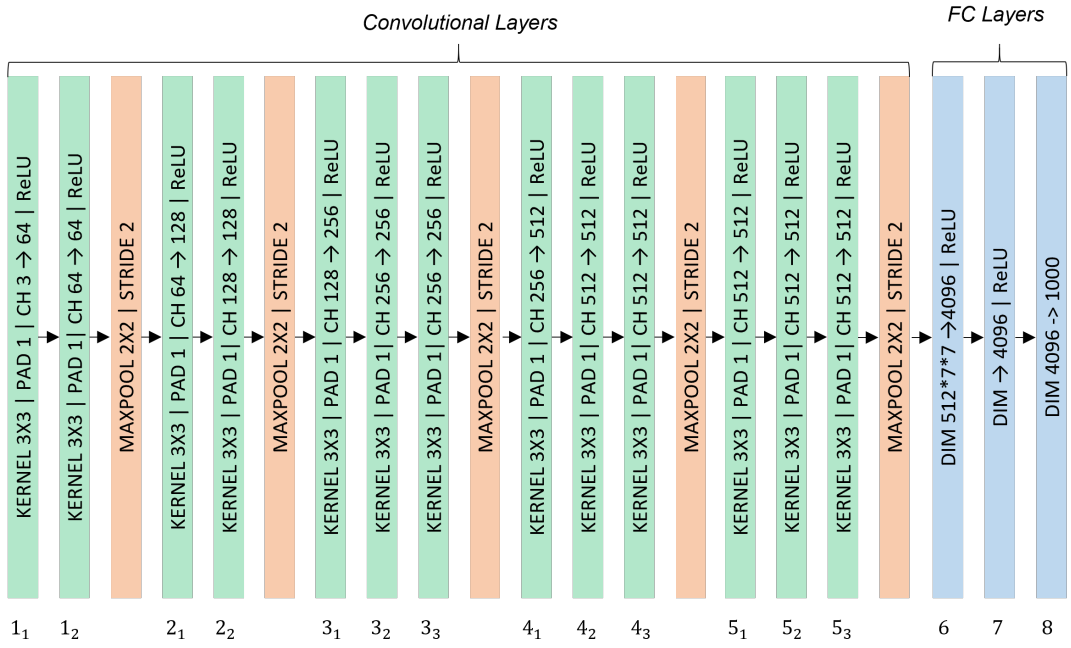


Figure 2.8: Structure of the VGG16 [2]

The output of the convolution layer 5_3 in the original VGG which receives as input images of size 224×224, will be of size $7 \times 7 \times 512$. Therefore, the FC6, shown in figure 2.8, will have an input size of $7 \times 7 \times 512$ and an output size of 4096, the equivalent convolution layer has a kernel size of 7×7, 512 input channels and 4096 output channels, therefore there will be 4096 filters of size $7 \times 7 \times 512$.

The FC7 from the VGG16 has an input size of 4096 and output size of 4096. The input can be seen as an image with dimension 1×1 and 4096 input channels. The equivalent convolution layer will have a kernel of 1×1, 4096 input channels and 4096 output channels, thus there will be 4096 filters of size $1 \times 1 \times 4096$.

Considering that there is a big number of filters and these are large which is computationally expensive, the authors opt to reduce the number of filters and their size by sub-sampling the parameters from the converted convolution layers. Therefore, both convolution layers will have 1024 filters, for conv6 with dimensions $3 \times 3 \times 512$ and for conv7 with a size of $1 \times 1 \times 4096$.

After the base convolution layers we add some **auxiliary layers** to provide additional feature maps that are progressively smaller than the last. We add 4 convolution block with 2 convolution layers each. Here the output size reduction is the result of the stride 2 in every second convolution layer, contrarily to the previous layers that was the result of the max pooling operation.

Object predictions can be quite varied, not only in terms of their type but also in any position they appear, size or shape. However, we should not go so far as to say that there are infinite possibilities for where and how an object can appear. While this is mathematically correct, many options are simply unlikely to occur. Furthermore, we don't need to insist on pixel-perfect boxes. We can simply discretize the mathematical space of potential bounding box predictions into just thousands of possibilities.

Priors or default boxes are fixed pre-calculated boxes which collectively represent the universe of approximate box predictions. These are similar to the anchor boxes from Faster R-CNN [41] except that, in this case, the boxes are applied from low-level feature maps to high-level feature maps, more precisely, the ones from *conv*4_3, *conv*7, *conv*8_2, *conv*9_2, *conv*10_2, *conv*11_2 in figure 2.7.

We design the set of default boxes so that specific feature maps learn to be responsive to particular scales of the objects. If we use m feature maps for prediction the scale of the default boxes for each feature map can be computed as:

$$s_k = s_{min} + \frac{s_{max} - s_{min}}{m-1}(k-1), \quad k \in [1,m] \tag{2.1}$$

where s_{min} is 0.2 and s_{max} is 0.9, meaning that the lowest layer has scale 0.9 and the highest 0.2 and the intermediate have values regularly spaced. For the feature map *conv*4_3 we use a scale of 0.1. We impose the following ratios for the default boxes: $\{1,2,3,\frac{1}{2},\frac{1}{3}\}$ We can also compute the width ($w_k^a = s_k\sqrt{a_r}$) and height ($h_k^a = s_k/\sqrt{a_r}$) for each default box, with a_r representing the ratio of the default box. For aspect ratio 1, there is an additional box with scale $s'_k = \sqrt{s_k s_{k+1}}$. The center of each default box is located at $(\frac{i+0.5}{|f_k|}, \frac{j+0.5}{|f_k|})$ where f_k is the size of the k-th square feature

map and $i, j \in [0, |f_k|]$. The distribution of default boxes can be adapted for different datasets, however we chose to keep the same structure as presented in the original paper. The table 2.4 resumes the prior boxes derived from each feature map.

Feature Map Origin	**Feature Map Dimensions**	**Prior Scale**	**Aspect Ratios**	**Number of Priors per Position**	**Total Number of Priors**
conv4_3	38, 38	0.1	$\{\frac{1}{1}, \frac{2}{1}, \frac{1}{2}\}$ + extra	4	5776
conv7	19, 19	0.2	$\{\frac{1}{1}, \frac{2}{1}, \frac{1}{2}, \frac{3}{1}, \frac{1}{3}\}$ + extra	6	2166
conv8_2	10, 10	0.375	$\{\frac{1}{1}, \frac{2}{1}, \frac{1}{2}, \frac{3}{1}, \frac{1}{3}\}$ + extra	6	600
conv9_2	5, 5	0.55	$\{\frac{1}{1}, \frac{2}{1}, \frac{1}{2}, \frac{3}{1}, \frac{1}{3}\}$ + extra	6	150
conv10_2	3, 3	0.725	$\{\frac{1}{1}, \frac{2}{1}, \frac{1}{2}\}$ extra	4	36
conv11_2	1, 1	0.9	$\{\frac{1}{1}, \frac{2}{1}, \frac{1}{2}\}$ extra	4	4
Total	-	-	-	-	8732 priors

Table 2.4: Summary of the priors calculation for the SSD300

The boxes obtained are an approximated starting point that will be later adjusted to obtain a more precise bounding box. It is necessary to calculate the deviation between the prior box and the predicted bounding box. For that we calculate the four offsets $(g_{cx}, g_{cy}, g_w, g_h)$ using the equations 2.2. The offsets are normalized by its corresponding dimension, height or width, of the prior because a certain offset would be less significant for a larger prior that it would be for a smaller one.

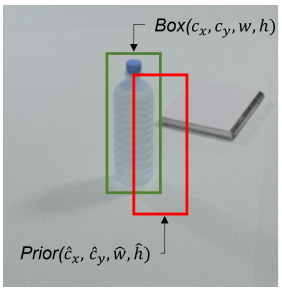


Figure 2.9: Representation of the true bounding box and the prior

$$g_{c_x} = \frac{c_x - \hat{c}_x}{\hat{w}} \quad g_{c_y} = \frac{c_y - \hat{c}_y}{\hat{h}} \quad g_w = \log\left(\frac{w}{\hat{w}}\right) \quad g_h = \log\left(\frac{h}{\hat{h}}\right) \tag{2.2}$$

We need to predict the localization of the bounding box and its class. We use a convolution layer with a 3×3 kernel with padding and stride equals to 1 with four filters for each prior present at the location, these four filters calculate the offset $(g_{cx}, g_{cy}, g_w, g_h)$ for the bounding box predicted

from that prior. For the class prediction it is also used a convolution layer with a 3×3 kernel with padding and stride equals 1 with c filters for each prior at a given location, c corresponds to the number of classes, including a background class. In summary, for each prior out of k for every location on each feature map with size $m \times n$ we compute an offset $(g_{cx}, g_{cy}, g_w, g_h)$ and c class scores for the bounding box. This results in a total of $(c+4)k$ filters applied to each location in the feature map, yielding $(c+4)kmn$ outputs.

During the training we need to find a way to compare the prediction made by network and the actual ground truth of the bounding boxes and the classes. We find the overlap between all the priors and the total number of ground truth objects (N). Then we match every prior with the ground truth box with the highest jaccard overlap, similarly to MultiBox. However, in SSD we then match the default boxes to any ground truth box with Jaccard overlap greater than a threshold (0.5). This is calculated by dividing the area of the intersection of both boxes by its union. The priors with an overlap inferior to 0.5 will be considered negative matches which means they don't contain any object. The positive matches will have the bounding box coordinates of the ground-truth as targets for localization and its ground truth label as the target for class prediction. Negative matches will not receive target coordinates and its target class will be a *background* class. The loss function consists in two terms: the localization loss and the confidence loss.

Localization loss Since there is no target for negative matches, the localization loss will be calculated only for the predicted boxes with positive matches. Additionally, the predicated boxes are in the form of offsets of the the default boxes, therefore we need to calculate the ground-truth boxes in the same format. The localization loss is then given by the smooth L1 loss between the predicted box and the ground-truth box.

$$\mathscr{L}_{loc}(x,l,g) = \sum_{i \in Pos}^{N} \sum_{m \in \{c_x, c_y, w, h\}} x_{ij}^{k} L_{1;smooth}(l_i^m - \hat{g}_j^m) \tag{2.3}$$

where l_i^m is the m-th offset of the four offsets that represent the predicted box from the i-th positive match and $\hat{g}_j^m$ is the correspondent ground-truth box in the offset format. $x_{ij}^p = \{1,0\}$ indicates the matching of the i-th default box to the j-th ground truth box of category p. The smooth L1 is given by the equation 2.4.

$$\mathscr{L}_{1;smooth} = \begin{cases} |x| & \text{if } |x| > \alpha \\ \frac{1}{|\alpha|} x^2 & \text{if } |x| \leq \alpha \end{cases} \tag{2.4}$$

where α is a hyper-parameter which was set to 1. In case N=0, we set the loss to 0.

Confidence loss All the predictions have a ground truth label associated with it. It is not only important to classify correctly the object but also its absence. Considering that there only a few objects in an image, from the thousands of predictions, a big majority will not contain any object. If our negative matches overwhelm the object detections, we will end up with a model less likely to detect objects since it was taught to have a background class more. Instead of using all the negative examples, we sort them using the highest Cross-Entropy Loss and we choose those with

top losses. The number of selected negative examples is a fixed multiple of the positive matches, in this case the authors used a ratio of 3:1.

The confidence loss is the softmax loss over multiple classes confidences (c) which results from the softmax activation plus a Cross-Entropy Loss, referred in equation 2.5, where c_i^p is the score for category p for prior i-th prior.

$$\mathscr{L}_{conf}(x,c) = -\sum_{i\in Pos}^{N} x_{ij}^p log(\hat{c}_i^p) - \sum_{i\in Neg} log(\hat{c}_i^0) \quad \text{where} \quad \hat{c}_i^p = \frac{\exp(c_i^p)}{\sum_p \exp(c_i^p)} \tag{2.5}$$

Total Loss The Multibox Loss is the aggregation of the two losses combined a ratio α, in general this is a learnable parameter. In this particular case the authors chose to use $\alpha = 1$.

$$\mathscr{L} = \mathscr{L}_{conf} + \alpha\mathscr{L}_{loc} \tag{2.6}$$

Due to the large number of predicted boxes, it is very likely that more that one box is predicted for the same object. To solve this problem we sort all the prediction by its class score and then we calculate the Jaccard overlap between candidate in a given class, in case the overlap is higher than a specified threshold we keep the candidate with the highest score and suppress the other - this is called Non-Maximum Suppression (NMS).

2.2.2 Relationship Detection

In this section, we introduce the structure of the proposed deep neural network model for visual relationship detection. To detect the relationships between the objects, we should receive as input a set of annotated objects in a image and output the relationships between each pair of objects. We define a visual relationship instance as `subject-predicate-object`.

We define the set of all annotated object pairs within an image as $\mathscr{A}$. For each element $(s,o) \in \mathscr{A}$, s and o represents the subject and the object, respectively, in the pair, $\mathscr{P}_{(s,o)\in\mathscr{A}}$ is the set of all predicates annotated for the subject-object pair (s,o). We define all the relationships in an image as following:

$$\mathscr{R} = \{(s,p,o)|(s,o) \in \mathscr{A} \wedge p \in \mathscr{P}_{(s,o)\in\mathscr{A}}\} \tag{2.7}$$

We propose a deep convolution network which combines multiple cues to sufficiently learn the representation for an input instance represented in figure 2.10. We use a visual appearance cue and a spatial location cue. In many previous works, a semantic embedding cue was used to integrate the category information in the prediction. However, in our application, the semantic relatedness between objects is approximately the same, given that we are interested in detect spatial relationships in objects that share most of the predicates.

Visual Appearance Cue From a visual appearance, humans can easily localize and identity relationships between objects in an image. In our model, we also use a visual cue, in order to understand the features of the objects that are a associated with a specific predicate. For this,

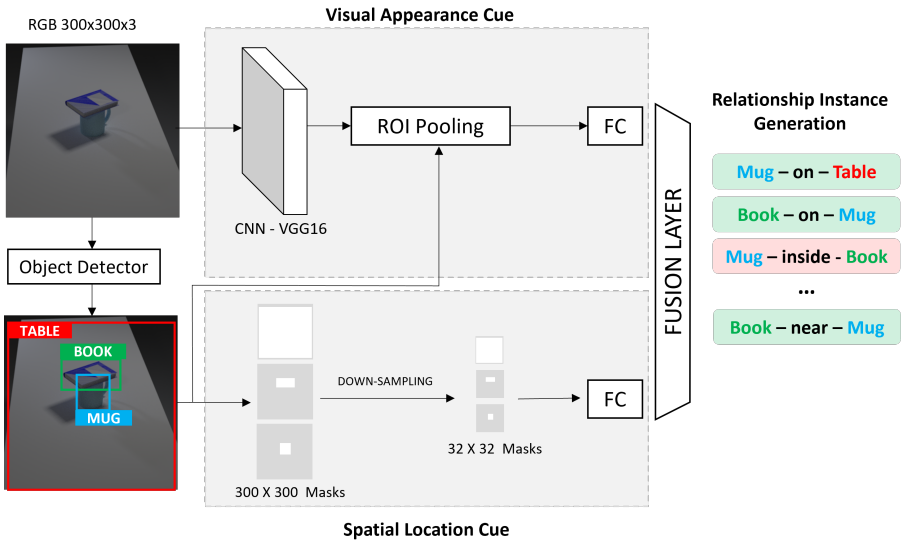


Figure 2.10: Proposed framework for visual relationship detection. Given an image as input the object detection network gives the bounding boxes and its class. The objects features and the spatial masks are extracted followed by a fusion layer to merge both cues and generate multiple relationship instances.

for every relationship $r = (s, p, o) \in \mathscr{R}$ we define the bounding box of the of the `subject` as $b_s = (x_s, y_s, w_s, h_s)$ and the `object` as $b_o = (x_o, y_o, w_o, h_o)$. Additionally we use b_p to denote the union area of b_s and b_p, which represents the predicate. When reasoning about relationships is often useful to capture the surroundings of the predicate and therefore we also add a small margin to b_p.

For the first five layers of the visual cue we used a VGG16 pre-trained in the *ImageNet Large Scale Visual Recognition Competition (ILSVRC)* classification task similarly to the SSD300. This allows us to extract feature maps from images to which we apply ROI Pooling operations by using b_o, b_s and b_p as the Regions of Interest (ROI). This produces an output of fixed dimension neither dependent on the feature map size nor on the proposals size. It is only dependent on the parameters used, in this particular case we chose a size of 7×7. This also allows to share the results of the last convolution layer among the different object proposals which highly reduce the amount of computation. The result of this operation is fed into three fully connected layers and later concatenated into a singe vector. We represent the output of this cue as $V(x)$.

Spatial Location Cue Spatial location is complementary to the visual appearance. Many of the relationships we are trying to predict can be identified by the spatial position of the object and subject in the scene. It is possible to make a good guess about the relationship between two objects without even knowing their categories. Furthermore, it is resilient to photometric variations, for instance, changes in the illumination or noisy images. As shown in [37, 11], dual spatial masks is a good way to explore the spatial information of the objects for a relationship instance. We use a binary mask for the `subject` and the `objects`, derived from the bounding box of each

object. The spatial mask is firstly generated with the size of the original image where all the pixels contained by the bounding box are one and the others are zero. The spatial masks are further down-sampled to the size 32×32 which makes the learning process faster and require less memory usage. In [11] was shown that the size 32×32 is a good balance between fidelity and cost. Both spatial masks from the object and the subject are merged and used as the input of a sequence of three convolutional layers followed by a fully connected layer which reduces the spatial masks into a low-dimensional vector. The results of these operations are concatenated with the vector from the visual cue and fed into a fully connected fusion layer. We represent the output of the spatial cue as $S(x)$.

Loss Function We denote the fused features from the combination of the multiple clues for a single object pair as $f(x,s,o)$ for a relationship instance $r = (s,p,o)$. The affinity between an image x and a relationship r is given by the scores learnt for a p_k predicate and the fused features, represented in equation 2.8.

$$\phi(x,r) = \phi(x,\{s,p,o\}) = w_p \cdot f(x,s,o) \tag{2.8}$$

The output of our predicate function should indicate the likelihood for a relationship to occur. To model this we start by defining the set of relationships instances that do not appear in a given image as:

$$\mathscr{R}' = \{(s',p',o') | (s',o') \in \mathscr{A} \wedge p' \notin \mathscr{P}_{(s',o') \in \mathscr{A}}\} \tag{2.9}$$

Our model should assign a low score to relationship that is very unlikely to occur, for example, `bottle-inside-table`, and a higher score to a relationship that is very likely to occur like `bottle-on-table`. We define this behavior as a ranking loss function:

$$\mathscr{L}(x) = \sum_{r \in R} \sum_{r' \in R'} max(0, 1 - (\phi(x,r) - \phi(x,r'))) \tag{2.10}$$

The loss optimizes a multi-class multi-classification problem. Every pair of objects can have a variable number of labels and it can be labeled with a certain class from a multi-class set. This means that every pair can have multiple predicates and this number is variable across the different pairs.

A prediction is considered correct when the following formula is satisfied:

$$\phi(x,r) - \phi(x,r') \geq 1 \tag{2.11}$$

Algorithm 1 Training Algorithm

Input: Training set of images with objects and relationships annotated
  Train object classifier with SSD300
  Initialize the weights **W**
  while $\mathscr{L}(x)$ not converged **do**
    Calculate the visual appearance cue $V(x)$
    Calculate the spatial cue $S(x)$
    Fuse both cues using the fusion layer $f(V(x), S(x))$
    Calculate the scores $\phi(x, r)$
    Back-propagate and optimize **W** using ADAM
  end while
Output: Trained weights **W**

Chapter 3

Evaluation

In this chapter, we perform a deep evaluation of the different components of our work. In section 3.1, we perform training with different dataset sizes to study its impact in the SSD performance. In section 3.2, we compare the performance of our object detector model in real world data when trained in only synthetic data, real data and the combination of both. In the next section 3.3, we evaluate the performance of our object detector in the generated synthetic dataset. Lastly, we evaluate the performance of the relationship detector network in synthetic and real data. We also run experiments to understand the contribution of the different components of our network model for the final predictions. The speed of the predictions made by our models is also analyzed.

To measure the performance of the model for object classification we use the average precision (AP) for each object class. We compute the area of the Precision-Recall curve for recall between 0 and 1 using 11 equally spaced recall levels. We only considered a valid prediction when its confidence score was above 0.4. When there are two or more detections with a IoU of 50% or higher with the same ground truth, it is only marked as True Positive the detection with the highest IoU and all the others are marked as False Positives (FP). After calculating the AP for every class we summarize the performance of a model across all classes by calculating the mean AP (mAP) which is simply the average AP across all classes. We also only considered the top 100 predictions per image even if there were more predictions with a score higher than 0.4 .

By following previous works [38, 30, 37], the evaluation metrics for the relationship model that we use are the Recall @ 100, Recall @ 50 and Recall @ 25. This metric (Recall @ K) computes the fractions of times the correct relationship is predicted in the top K confident relationship predictions. For example, for our 13 predicates and with an average of 4 objects per image, the total possible number of prediction is $4 \times 13 \times 4$, therefore the Recall @ 10 for a random guess is 0.05. All of our experiments on neural networks were conducted in Google Colab in an environment equipped with the GPU Tesla K80 and Intel(R) Xeon(R) CPU @ 2.30GHz. For the evaluation of the deep learning models we used exclusively synthetic data, with 4000 images for training and 1000 for testing.

3.1 Performance of SSD with Different Dataset Sizes

To understand the impact of the dataset size in the performance obtained in the object detection we trained our model with different dataset sizes from 500 images up to 4000. In this case we are not interested in evaluate the network model but rather understand the impact of changing the size of the dataset in performance. We used our SSD300 object detector to perform this evaluation, the training was executed using a fixed learning rate of 0.002. We use the SGD optimizer to train the whole network with a 0.9 momentum and 0.0005 weight decay, we also set the batch size to 32. The results obtained are show in figure 3.1 where we plotted the mAP over the number of epochs trained.

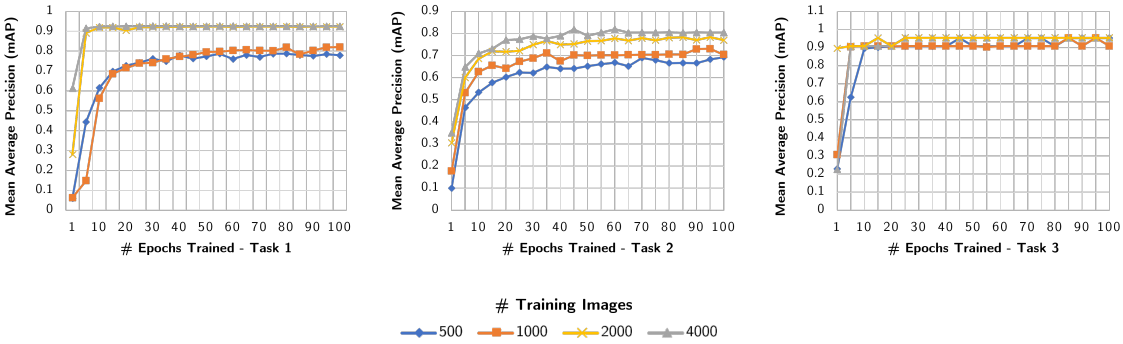


Figure 3.1: Plots of the mean average precision of object detection over the number of trained epochs for different training dataset sizes. The plots were made for every task.

For Task 1, we can observe that having 2000 images for training is enough and a larger data set will not improve the performance of the object detector.

In Task 2, there are a total of six objects, therefore we can see an improvement in the performance of the detector until the largest dataset, however the performance gain starts becoming smaller for bigger datasets, we can see that different between 2000 and 4000 is smaller than between 1000 and 2000. When there are more classes, more training data is necessary to obtain the best performance.

For Task 3 where we only have two object classes the detector is capable to perform well while training in only 500 images.The increase of the dataset size, doesn't give a significant increase in the performance, due to the low number of classes all the objects are seen enough times allowing to extract all the required information from only 500 images.

3.2 Performance of SSD in Real World Data

In this section, we aim to study the effect of using synthetic data when the acquisition of correctly label data from real world is difficult to obtain. We performed three trainings with different datasets for every task.

Firstly, we trained our object detector with only synthetic data using 4000 images, secondly we trained using synthetic data for the first 50 epochs and then purely real world images for the

next 50 epochs and lastly we trained for 100 epochs solely in real data. In this training we used a real world dataset containing a total of 50 images, 25 for training and 25 for testing that we manually annotated. We perform the evaluation of the networks trained in different datasets in the test dataset containing 25 real world images.

In figure 3.2, 3.3 and 3.4, we plotted the mAP vs Training epoch for every training dataset and for every task. Using only real data, performs the worst for task 1 and 3 due to its very reduced size, using synthetic and real data performs always the best, however for task 3 this difference is very small, the detector was able to learn from synthetic data quite well. The tables 3.1, 3.2 and 3.3 also show the average precision for each object after the 100 epochs of training and the mAP for each dataset.

Dataset	book	bottle	mug	mug upside down	table	mAP
Only Synthetic	0.64	0.73	0.57	0.33	0.93	0.64
Synthetic (99.3 %) + Real World (0.7%)	**0.82**	**0.85**	**0.98**	0.33	**0.95**	**0.79**
Only Real (25 images)	0.33	0.09	0.53	**0.55**	0.83	0.47

Table 3.1: Average precision and mAP for Task 1 trained in different datasets and evaluated in the real world dataset

Dataset	fork	knife	plate	spoon	wine glass	table	mAP
Only Synthetic	0.00	0.02	0.00	0.04	0.00	**1.00**	0.18
Synthetic (99.3 %) + Real World (0.7%)	**0.62**	0.42	**0.91**	**0.43**	0.22	**1.00**	**0.60**
Only Real (25 images)	0.52	**0.51**	0.69	0.33	**0.26**	**1.00**	0.55

Table 3.2: Average precision and mAP for Task 2 trained in different datasets and evaluated in the real world dataset

Dataset	Box	Table	mAP
Only Synthetic	0.89	**1.00**	0.94
Synthetic (99.3 %) + Real World (0.7%)	**0.90**	**1.00**	**0.95**
Only Real (25 images)	0.64	0.28	0.46

Table 3.3: Average precision and mAP for Task 3 trained in different datasets and evaluated in the real world dataset

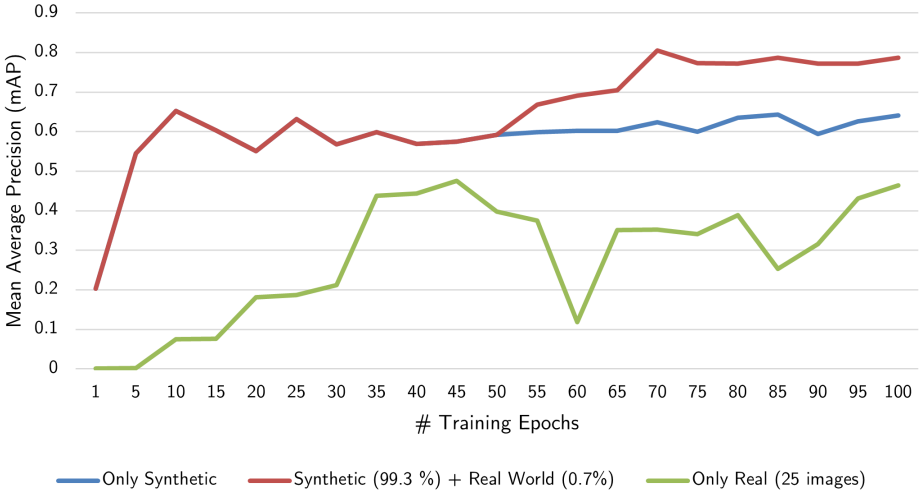


Figure 3.2: Plot illustrating the effect of including synthetic data while training the neural network for task 1

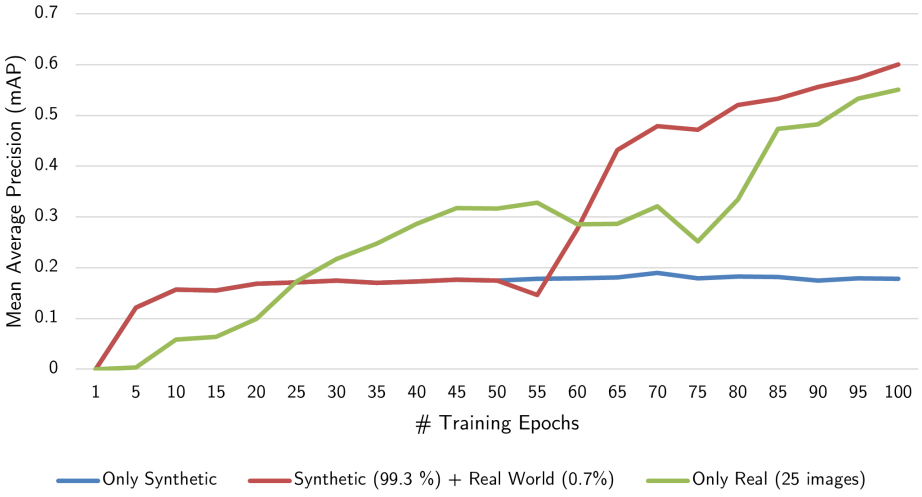


Figure 3.3: Plot illustrating the effect of including synthetic data while training the neural network for task 2

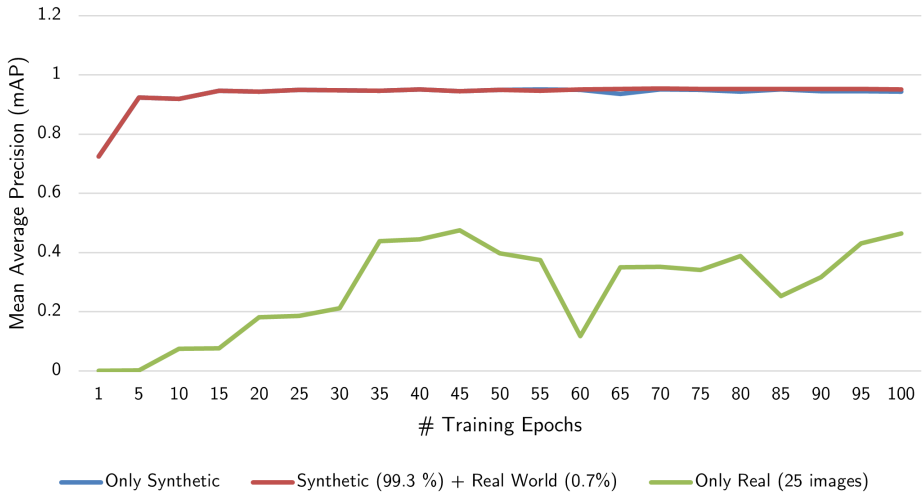


Figure 3.4: Plot illustrating the effect of including synthetic data while training the neural network for task 3

3.3 Performance of SSD in Synthetic Data

In order to find the best hyper-parameters for our model, we performed Random Search which is usually faster that Grid Search [44] however, it does not guarantee the best performance. We performed the evaluation with random hyper-parameters until we were satisfied with the results. Random search can outperform a grid search, especially if only a small number of hyper-parameters affect the performance of the machine learning algorithm.

We performed a training using a fixed learning rate of 0.002. We use the SGD optimizer to train the whole network with a 0.9 momentum and 0.0005 weight decay, we also set the batch size to 32. We evaluated the performance of the network for every task. The AP for every object category and the mAP for every task are present in Table 3.4. Figures 3.5, 3.6, 3.7, show qualitative results of the model's performance.

Dataset	Average Precision per Object						mAP
Task 1	book	bottle	mug	table	mug_upside_down		0.925
	0.907	0.903	0.907	1	0.909		
Task 2	fork	knife	plate	spoon	table	wine_glass	0.804
	0.908	0.696	0.909	0.595	1	0.718	
Task 3	box	table					0.954
	0.909	1					

Table 3.4: Average precision of the object detector for every task evaluated in the synthetic dataset

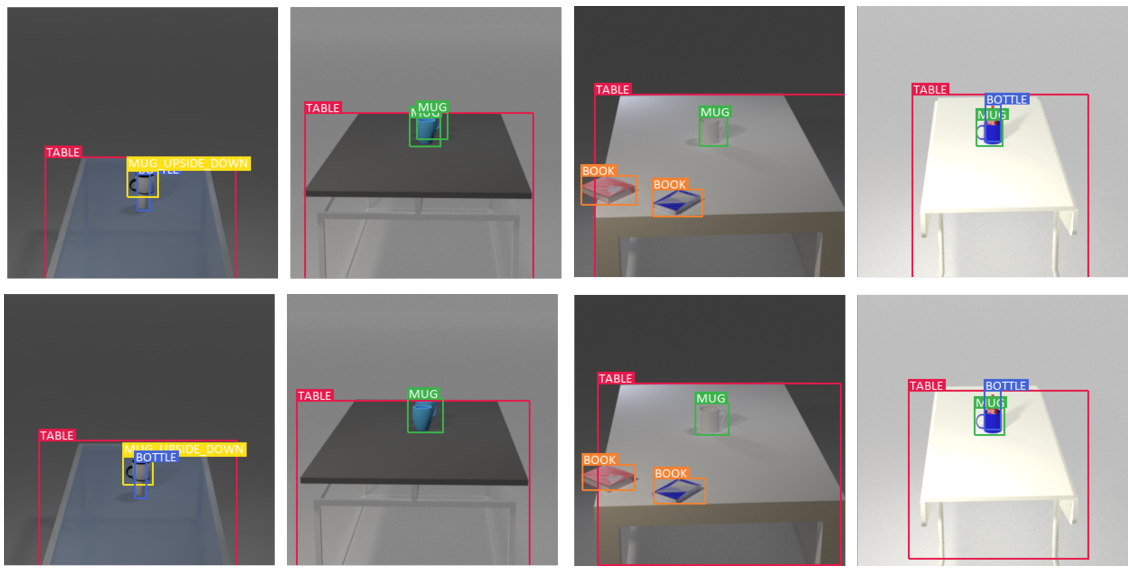


Figure 3.5: Qualitative results for the object detection model. On top are the ground-truth and in the bottom the predictions made by the model for task 1.

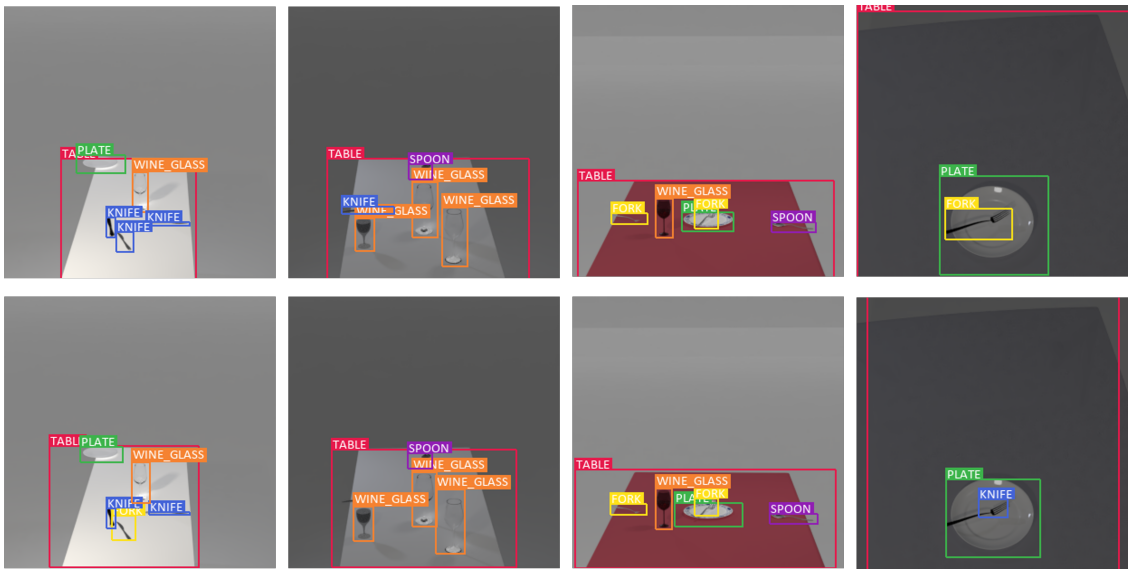


Figure 3.6: Qualitative results for the object detection model. On top are the ground-truth and in the bottom the predictions made by the model for task 2.

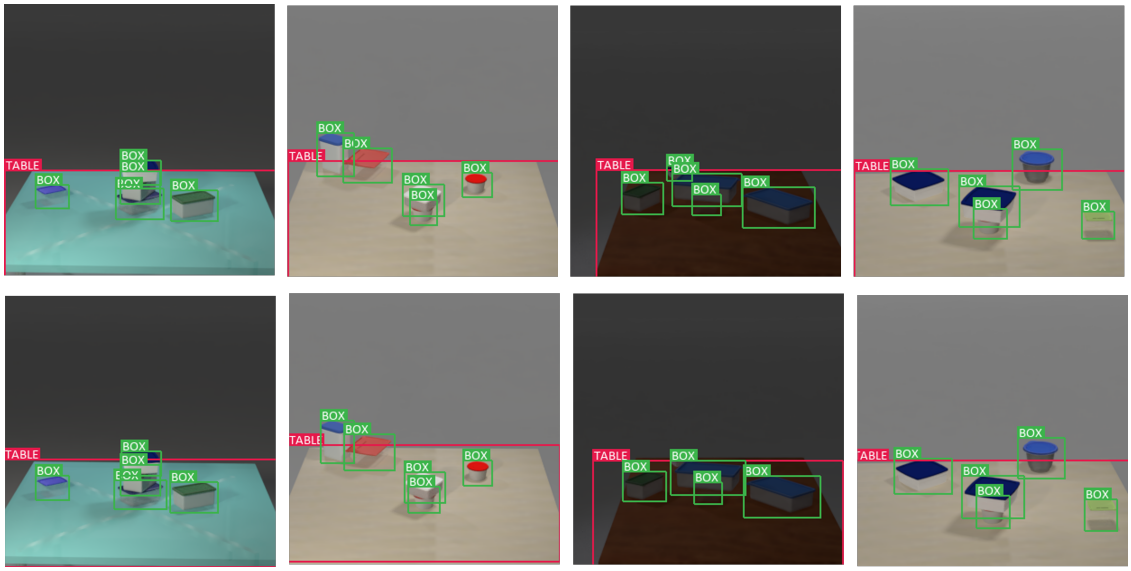


Figure 3.7: Qualitative results for the object detection model. On top are the ground-truth and in the bottom the predictions made by the model for task 3.

3.4 Performance of Relationship Detection

Detecting a visual relationship involves not only predicting the predicate between two object but also recognizing and classifying both objects. To study our model performance in these different tasks, we measured the predictions based in different conditions:

1. **Predicate Detection**: We use the ground-truth of the objects classification, their bounding boxes and the image as the input of the network, the output is the predicate between the different pairs of objects from the ground-truth. The result is considered correct if the predicate is correct.

2. **Relationship Detection**: The input is solely the image and the output is a set of relationships `subject-predicate-object`. Each triplet is treated as an whole. The result is considered correct if both objects are correctly detected which means the category is right and the IoU between the bounding box predicted and the ground truth is equal or greater than 0.5, and the predicted predicate is correct.

As shown before, in order to learn the relationship instance of a pair of objects, our model uses multiple cues to deeply understand the visual and spatial connection between different object. In this section, we show the experiments performed to study all the network components to understand how they affect the final performance. We use V_1 to denote only using the union of the object and subject bounding boxes (relationship bounding box). We use S to denote the spatial location cue and V_2 as the use of the bounding boxes of the object and subject independently. The visual appearance cue is denoted by the combination of V_1 and V_2.

	Predicate Detection			Relationship Detection		
	R@25	R@50	R@100	R@25	R@50	R@100
S	98.61	99.18	99.88	92.23	95.75	96.46
V_1	84.08	95.15	99.20	77.83	88.62	93.73
V_2	90.41	96.64	99.11	84.42	90.33	93.38
$V_1 + V_2$	91.90	97.78	99.40	86.39	94.01	96.14
$V_1 + V_2 + S$	**99.55**	**99.86**	**99.94**	**93.09**	**96.24**	**96.54**

Table 3.5: Component analysis in our dataset for Task 1

In our comparison of the different components of the visual appearance cue, V_2 has a higher contribution than V_1 in all tasks for the total performance of the visual appearance cue. The spatial cue has the majority contribution to the total model performance. In task 1 the visual cue only contributes less than 1% in the different metrics. In task 2, the increase is more significant, being 5.46%, 11.5% and 13% for R@25, R@50 and R@100 respectively in predicate detection. In the last task, the contribution of the visual part is also quite significant with a 5.4%, 9% and 8.85% increase in the respective metrics in predicate detection.

	Predicate Detection			Relationship Detection		
	R@25	R@50	R@100	R@25	R@50	R@100
S	56.37	85.52	88.37	45.50	63.23	70.97
V_1	30.63	54.40	74.89	29.43	44.60	58.00
V_2	54.22	72.17	91.09	44.55	55.72	65.36
$V_1 + V_2$	55.30	75.93	93.27	44.30	57.36	71.83
$V_1 + V_2 + S$	**59.45**	**95.40**	**99.91**	**46.46**	**69.08**	**79.41**

Table 3.6: Component analysis in our dataset for Task 2

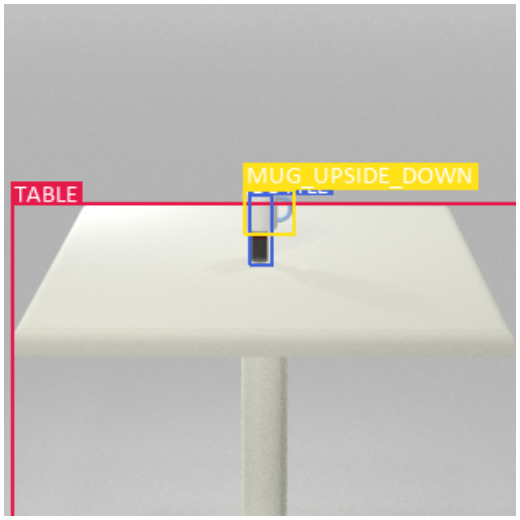
	Predicate Detection			Relationship Detection		
	R@25	R@50	R@100	R@25	R@50	R@100
S	60.55	87.39	91.71	54.61	83.58	93.25
V_1	37.93	60.29	81.21	36.21	56.48	77.85
V_2	57.28	75.40	92.14	53.94	71.89	88.80
$V_1 + V_2$	57.89	78.63	93.90	52.87	73.52	90.42
$V_1 + V_2 + S$	**63.79**	**95.31**	**99.83**	**57.21**	**90.35**	**96.69**

Table 3.7: Component analysis in our dataset for Task 3

3.4.0.1 Qualitative Results

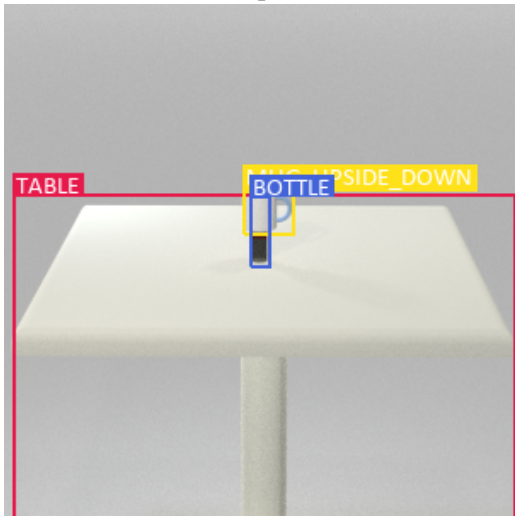
In Figure 3.8, 3.9 and 3.10 we present some qualitative results for the predictions made in synthetic data using our best model. The correct predictions are marked in green and the wrong ones in red. We show the results for the predicate detection and the relationship detection. Only the predictions with score above 1 must considered if we look at our margin loss function 2.10 which have margin 1.

Predicate Detection



bottle - on - table	16.85
table - under - bottle	13.14
bottle - near - mug_down	7.21
mug_down - near - bottle	7.14
mug_down - on - bottle	5.90
bottle - inside - mug_down	5.08
bottle - under - mug_down	2.13
bottle - in front of - mug_down	0.99
mug_down - behind - bottle	0.17

Relationship Detection



bottle - on - table	17.60
mug_down - on - table	17.43
table - under - mug_down	13.94
table - under - bottle	13.46
bottle - near - mug_down	7.21
mug_down - near - bottle	7.14
mug_down - on - bottle	5.90
bottle - inside - mug_down	5.08
bottle - under - mug_down	2.13
bottle - in front of - mug_down	0.99
mug_down - behind - bottle	0.17

Figure 3.8: Qualitative Results of the proposed method in our synthetic dataset with the confidence scores for Task 1. The model used was trained in synthetic data. The correct predictions are shown in green while the wrong predictions are in red. Note: mug_down refers to mug_upside_down

.

Predicate Detection

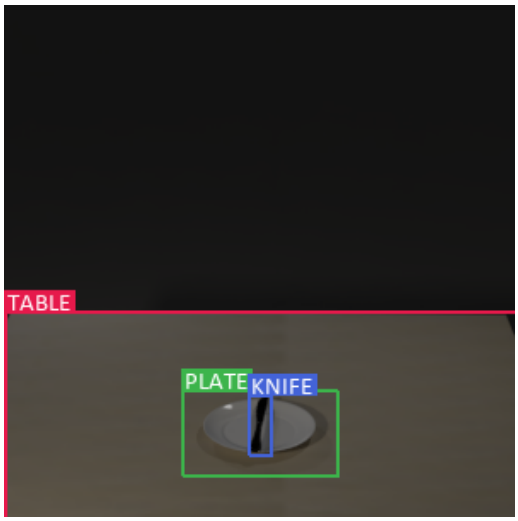


plate - near - knife	10.56
table - under - plate	10.41
knife - near - plate	9.77
plate - on - table	8.85
plate - under - knife	3.26
knife - on - plate	2.90

Relationship Detection

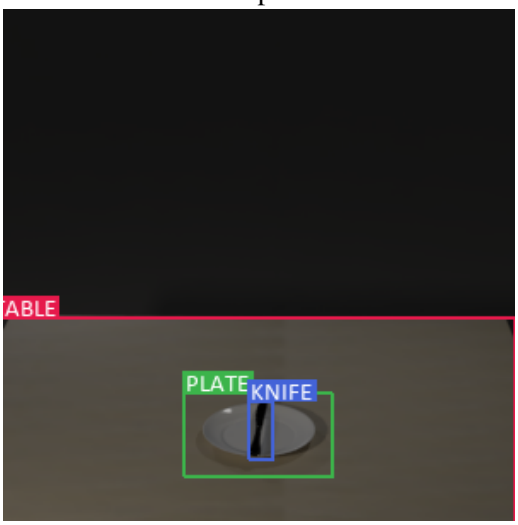


table - under - plate	10.56
plate - near - knife	10.06
table - under - knife	9.87
knife - near - plate	9.29
knife - on - table	9.13
plate - on - table	8.92
knife - on - plate	3.30
plate - under - knife	3.17

Figure 3.9: Qualitative Results of the proposed method in our synthetic dataset with the confidence scores for Task 2. The model used was trained in synthetic data. The correct predictions are shown in green while the wrong predictions are in red.

Predicate Detection

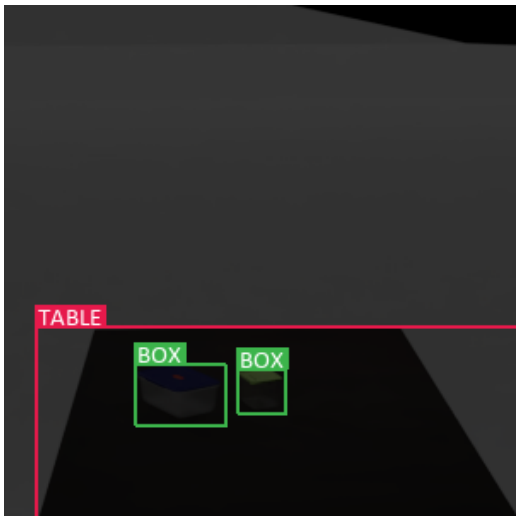


table - under - box_1	11.57
table under - box_2	10.80
box_1 - on - table	9.60
box_1 - near - box_2	8.27
box_2 - on - table	7.86
box_2 - near - box_1	7.57
box_2 - left - box_1	5.84
box_1 - right - box_2	5.77
box_2 - left_front - box_1	0.39

Relationship Detection

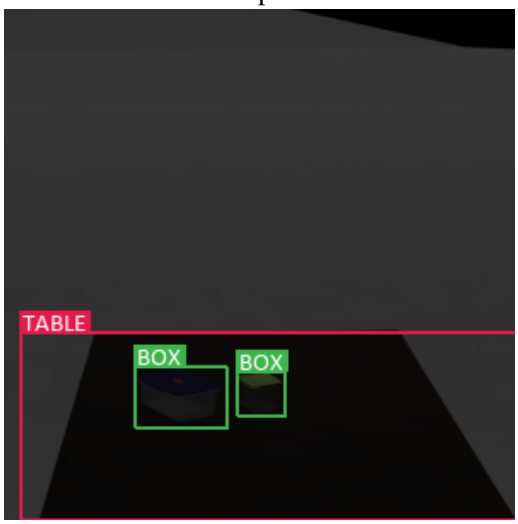


table - under - box_2	11.67
table - under - box_1	10.92
box_2 - on - table	9.72
box_2 - near - box_1	8.22
box_1 - on - table	7.97
box_1 - near - box_2	7.58
box_1 - left - box_2	5.64
box_2 - right - box_1	5.47
box_1 - left_front - box_2	0.42
box_2 - right_behind - box_1	0.09

Figure 3.10: Qualitative Results of the proposed method in our synthetic dataset with the confidence scores for Task 3. The model used was trained in synthetic data. The correct predictions are shown in green while the wrong predictions are in red.

In Figure 3.11, 3.12 and 3.13, we show qualitative results for our predicate and relationship detection in real world images. We use our predicate detection model trained in only synthetic data and our object detector trained in real and synthetic data. Since we don't have the ground-truth of the relationships for real data, additionally to the green an red marked predictions, we also use yellow to mark predictions that can be subjective.

Predicate Detection

mug - on - table	17.44
book - on - table	15.98
mug - near - book	12.91
book - near - mug	12.09
table - under - mug	11.63
table - under - book	10.90
mug - behind - book	3.60
book - under - mug	0.80
book - in front of - mug	0.69

Relationship Detection

mug - on - table	17.09
book - on - table	15.98
mug - near - book	12.80
table - under - mug	12.01
book - near - mug	11.55
table - under - book	10.73
mug - behind - book	3.02
book - in front of - mug	1.53
mug - on - book	1.06

Figure 3.11: Qualitative Results of the proposed method in real world images with the confidence scores for Task 1. The model used was trained in real and synthetic data. The correct predictions are shown in green while the wrong predictions are in red.

Predicate Detection

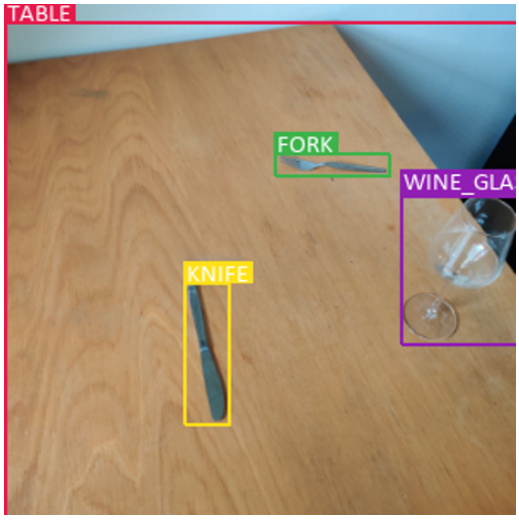


table - under - fork	9.26
table - under - knife	8.96
knife - on - table	8.92
fork - on - table	8.59
fork - far - knife	7.88
table - under - wine_glass	7.25
wine_glass - on - table	7.24
knife - far - fork	5.55
wine_glass - far - knife	4.81
fork - right_behind - knife	4.69
fork - near - wine_glass	4.49
knife - far - wine_glass	4.44
fork - left_behind - wine_glass	3.78
knife - left_front - fork	3.72
wine_glass - right_behind - knife	3.46
wine_glass - near - fork	3.13
knife - left_front - wine_glass	3.04
wine_glass - far - fork	2.57
wine_glass - right_front - fork	2.36
wine_glass - near - knife	2.19
wine_glass - right - knife	2.13
knife - left - wine_glass	1.99
wine_glass - right - fork	1.93
knife - near - wine_glass	1.78
fork - far - wine_glass	1.26
fork - left - wine_glass	1.10
fork - behind - knife	0.91
knife - in front of - fork	0.25

Relationship Detection

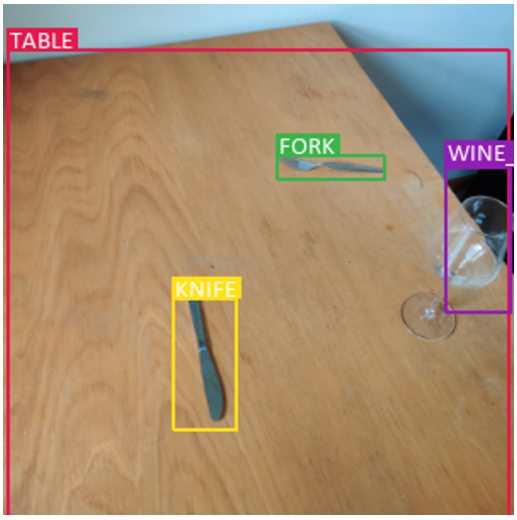


table - under - knife	9.29
knife - on - table	9.27
table - under - fork	9.13
fork - on - table	8.66
fork - far - knife	7.78
table - under - wine_glass	7.72
wine_glass - on - table	7.41
wine_glass - far - knife	6.28
wine_glass - right_behind - knife	5.93
knife - far - fork	5.36
fork - right_behind - knife	5.25
knife - far - wine_glass	5.11
knife - left_front - wine_glass	4.73
knife - left_front - fork	3.98
fork - left_behind - wine_glass	3.74
wine_glass - right - fork	3.02
wine_glass - far - fork	2.97
wine_glass - near - fork	2.95
fork - near - wine_glass	2.86
fork - far - wine_glass	2.82
wine_glass - right_front - fork	1.91
fork - left - wine_glass	1.31
knife - near - wine_glass	0.97
wine_glass - near - knife	0.87
knife - left - wine_glass	0.56
fork - behind - knife	0.47
knife - in front of - fork	0.43
wine_glass - right - knife	0.21
knife - near - fork	0.19

Figure 3.12: Qualitative Results of the proposed method in real world images with the confidence scores for Task 2. The model used was trained in real and synthetic data. The correct predictions are shown in green while the wrong predictions are in red, the yellow is used to mark subjective detections.

Predicate Detection

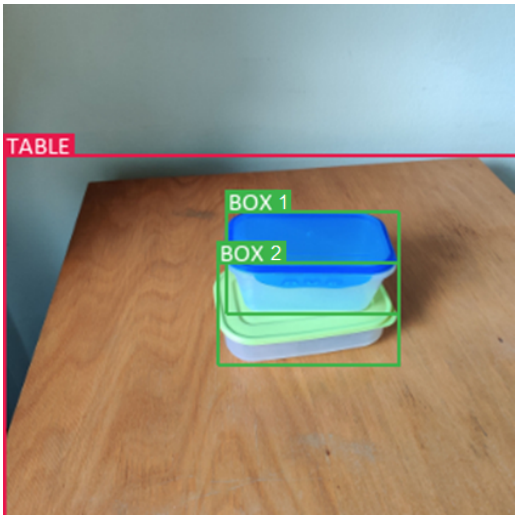


table - under - box_2	14.59
table - under - box_1	12.51
box_2 - on - table	10.40
box_1 - on - table	8.62
box_1 - near - box_2	7.00
box_2 - near - box_1	6.99
box_1 - on - box_2	1.56
box_1 - under - box_2	1.24
box_2 - under - box_1	0.73
box_2 - under - table	0.00

Relationship Detection

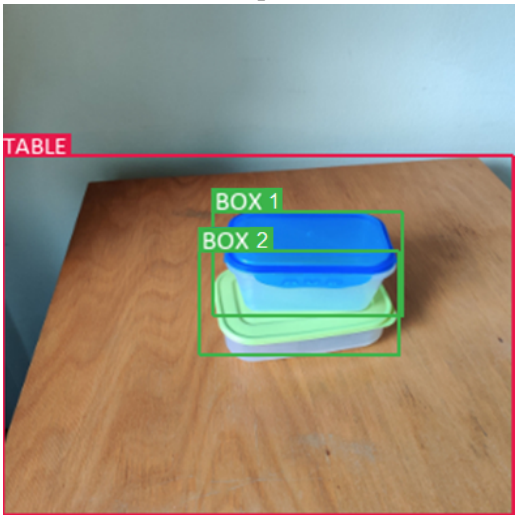


table - under - box_2	14.22
table - under - box_1	12.04
box_2 - on - table	10.16
box_1 - on - table	8.48
box_2 - near - box_1	7.36
box_1 - near - box_2	7.22
box_1 - on - box_2	3.33
box_2 - under - box_1	2.49
box_1 - under - box_2	2.25
box_2 - on - box_1	1.69

Figure 3.13: Qualitative Results of the proposed method in real world images with the confidence scores for Task 3. The model used was trained in real and synthetic data. The correct predictions are shown in green while the wrong predictions are in red.

3.5 Real-Time Performance

When it comes to robotic manipulation, real-time constraints play a very important role for the success of the system. In this section we evaluate the timing performance of our network and compare it with similar works.

According to paper regarding the SSD300 [1], the average time to detect the objects in a single image is 21.74ms on Titan X. In our work the average time for the object detection is 27.45 ms with the graphics card NVIDIA Tesla K80.

In the previous work [38], they also used SSD300 to detect the objects and reported the detection time as in the original paper. In their model, the relationship predictor takes 5.5ms per image in average. In our case, it takes 5.8ms, however our model is capable of detecting different relationships and not only the grasping order of the objects.

Model	Graphics Card	Object Detection (ms)	Pred. Detection (ms)	Total (ms)
SSD300 [1]	TitanX	21.74	-	-
[30]	TitanX	-	-	122
[38]	TitanXp	21.74	5.5	**27.24**
Ours	Tesla K80	27.445	5.80	33.24

In a graphics card that takes a total of 30 ms to detect the objects and the relationships, it is possible to make predictions of 33.3 frames per second which is suitable for real time applications.

3.6 Scene Graphs

Scene Graphs is a data structure that can represent clearly the objects, attributes and relationships between objects in a scene [45]. With the development of computer vision technology we are no longer just interested in simply recognizes objects in an image; we also aim to have a higher level of understanding and reasoning of a scene.

Our model for object and relationship detection can be used to generate scene graphs that can be employed in visual question answering [46], organizing objects or to infer the correct grasping order of the objects [38].

Figure 3.14, shows an example of a scene graph generated from the relationships predicted by our model.

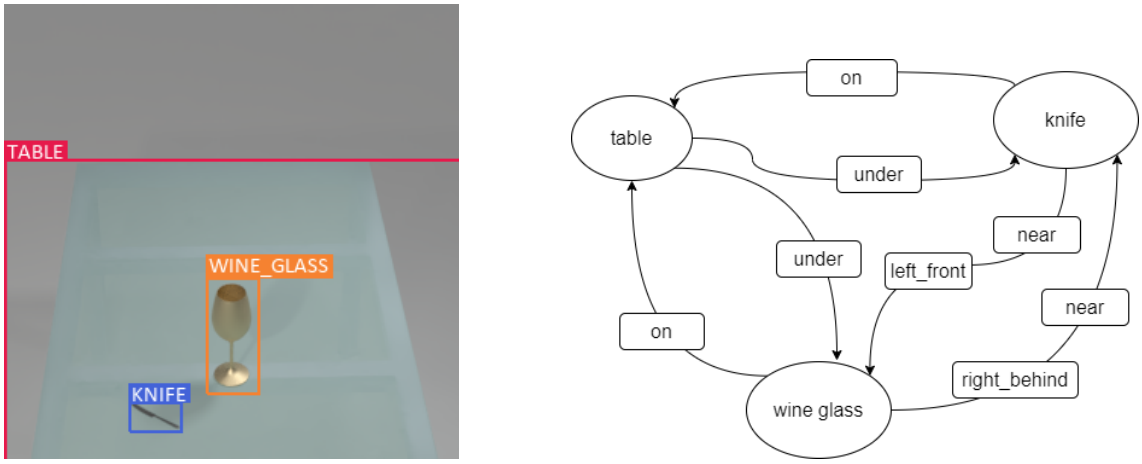


Figure 3.14: The figure illustrates an example of a scene graph generated from the objects detected in the figure on the left

Chapter 4

Discussion

In this chapter, we discuss the results of our experiments as well the quality of the generated and captured datasets. We also discuss the results of the neural network models, the contribution of each component and the experiments conducted in these models.

4.1 Synthetic Dataset

3D Models All the 3D models chosen to render our scenes were obtained from free sources such as the ShapeNET dataset that gathers annotated free 3D models from different sources. The advantage of using free 3D models is that they can be used free of charge for non-commercial purposes. However, these models tend to be quite simple and lack detail, sometimes the textures of the object are simply a color instead of a more realistic texture, also the shape of the objects are simple and can be easily identified as 3D models and not photos of real objects. To solve this problem, there are several models with great detail available online. However, these are paid and for that reason we did not use them in this thesis. Using models closer to real objects will make the generated images more realistic, which is likely to improve the performance of the models in real environments when they are trained on synthetic datasets. Another problem related to the 3D models is the limit of available 3D models with different textures and shapes, in this case we used a total of 114 models, however using a greater diversity of models makes our model prediction more generalized and able to make correct predictions on a wider range of different objects.

Blender Scene Each scene rendered in blender was used to render a synthetic image, applying variations in light and camera perspective to get closer to real scenes and cover more possibilities. In the generated images a white background was used, however, in the real world the background often contains people standing close to the camera or other elements present in the room such as chairs or tables. Varying the background randomly would help to create a model that is more robust to this background changes in real-world applications.

Rendered Relationships To detect the relationships between different objects, we hard-coded spatial constraints that we defined for a given relationship. For example, if the position of one object is greater than another object given the size of the objects the relationship on is generated. By visual inspection, the relations generated in the final dataset were good, without any relations that should not exist or missing annotations that should have been generated. This is one of the advantages of generating synthetic data because the annotations are always correctly annotated as long as the algorithm is implemented flawlessly. Data labeling done by humans can introduce random and non-consistent errors.

Object Positioning To position the objects on the table, spatial constraints were used, not taking into account the physical constraints of the objects. For example, if the objects are colliding or if the object is suspended in the air, in our case, to avoid collisions, the objects were placed at a distance greater than the sum of the sizes of the two objects. Furthermore, some positions will be practically impossible to appear in the real-world. For example, a mug inside a bottle is not reasonable, even if it fits inside. Applying physical laws to the generated scenes may make it easier to generate images without the use of extensive hard-coded constraints. However, our final datasets appeared to not have any object collisions or objects in unrealistic positions.

Rendered Datasets We rendered 3 different datasets for the different tasks. In our experiments we generated statistics that help us understand the content of the generated datasets. Our datasets contain only 12 or 13 relations unlike previous works [30] using more than 6000, however in this context the number of possible relations that objects can have is much more limited than in datasets covering environments with humans. We could increase its number by adding, for example, comparative relationships, like object 1 is bigger than object 2. Here we also limit the number of objects per image to 4 for the first task and 5 for the second and third tasks.

The objects and the relationships between them are randomly selected by our algorithm and therefore one would expect the total number of objects to be similar for all categories and the different relations to appear approximately the same number of times. However, this is not observed by the graphs showing the distributions of objects and relations in the datasets (Figure 2.3, 2.4, 2.5). Relationships like `on`, `under`, `near` or `far` appear more often than others because for each pair of objects a relation about the distance between them is instantiated. Also, each object will be on the table and therefore the relations `under` and `on` are generated for each object. Many relationships are mirrored as the result of their spatial opposition, for example, if object 1 is on the left of object 2, object 2 will be on the right of object 1 making the sum of these two predicates the same. The rest of the relationships are approximately evenly distributed. However, the relationships `behind` and `in front of` appear slightly less due to the limit considered between `behind` and `left-behind`, for example. From this we can infer that the area of the table where the object can be located relatively to another object is smaller for the relationships `behind` and `in front of` than for other relationships like `left` or `right`.

4.2 Neural Network

Dataset size Increasing the size of the dataset shows generally an improvement of the object detector performance. However, for a given complexity of a scene, there is a limit where having more data has none or almost no effect in the final performance. A smaller number of images may be enough for the network to be at its maximum capacity for its depth. Increasing the networks depth will make it more likely to overfit and perform worse in the testing data.

Synthetic to Real In our synthetic to real experiments, we show that adding a small percentage of real data to the synthetic data generally improves the network performance on real images. Using only real data generally performs worse than using both synthetic and real data, and even worse than using only synthetic. The small size of the real-world dataset does not give enough information to the deep learning-based model to generalize well. By analyzing the graphs, 3.2, 3.3 and 3.4, we can see that in task 1 using real data has a significant positive impact. The mAP increases considerably at epoch 50 when training with real images is started. In task 2, the knowledge acquired by the model when trained on synthetic data is quite low. Thus, when real data is introduced, there is a significant increase in the performance of the network. Moreover, the mAP curve of synthetic+real data is similar to the curve of only real data from epoch 50 onwards. In task 3, the model is able to generalize the synthetic objects quite well making the introduction of real data have almost no impact on the performance obtained. The model is able to learn on synthetic data and perform well on real data due to the low number of classes in this task.

Object Detector In this thesis, we used the SSD300 object detector due to its real-time performance and good accuracy. However, we could have used another objector detector such as Faster-RCNN as in some previous works [30]. Additionally, we used a VGG-16 [2] as a base network for the object detector and relationship detector, we could also have used ResNet50 but as shown in [38], VGG-16 performs slightly better.

Object Detection Performance It is clear that our object detector performs generally well, obtaining a mAP greater than 0.9 for tasks 1 and 3. By visual inspection, the detector fails to recognize objects that are nearly occluded by other objects, or to estimate the bounding box perfectly equal to the ground-truth. In task 2, the mAP is only 0.8, we can notice that the objects `fork`, `knife` and `wine glass` have lower AP compared to the other objects contributing for the low mAP. The `fork` and `knife` are usually quite small and difficult to recognize in an image with only 300x300 pixels, as they do not show enough detail. To solve this problem, one could use images with a higher resolution. The low AP for the wine glass is due to its transparent material making it hard to capture the visual features. Moreover the features learned for the glass are constantly changing depending on the background, a possible solution would be to augment images containing glasses to help the network to generalize the object independently of the background.

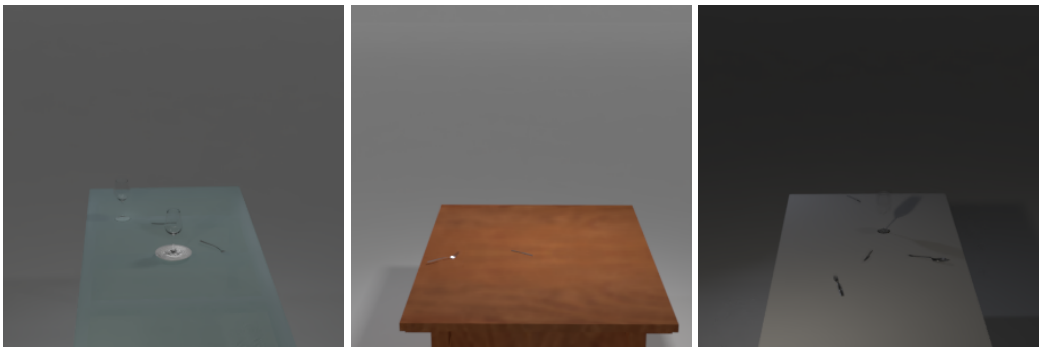


Figure 4.1: Examples of generated images with difficult visibility of the wine glass and small cutlery

Real Time Constraints Our experiments showed that our model is suitable for real-time applications. We evaluated our models in a slower graphics card than previous works, however this was still enough to be used in real-time applications which in important in robotic manipulation tasks.

Relationship Detection In the study of the different components of our model we can observe that the spatial cue has a huge impact in the networks performance. From our perspective, the reason for this is the fact that all the relationships learned describe spatial information of the subject relatively to the object, this information is learned by our spatial location component.

Qualitative Results Generally, our qualitative results for relationship detection show good results. In the relationship prediction, in real world images this performance is remarkably worse given that the training of the predicate detector was performed in synthetic data. We can also notice that the relationship `under` and `on` is frequently detected when the object are above or below each other but not in contact. This means that the visual appearance cue was not able to learn that the object need to be in contact to predict the mentioned relationships. Adding the predicates `above` and `under` could potently improve the performance considering that the network would have to learn to distinguish between both predicates. Currently, if the objects are localized above each other is very likely that they have the relationship on, since this is very common for this spatial configuration.

Zero-shot Other relationship detection works analyse the zero-shot learning due to the incompleteness of annotations in images. Having our data synthetically generated allows us to annotate all possible relationships between every object. This avoids the problem of detecting relationships that were never seen before during the training process. During the training all the possible combinations of objects and predicates are seen by the network.

Chapter 5

Conclusion

This thesis allowed us to gain more knowledge of the current state-of-the-art of data synthesis for neural network training and in the area of object relationship detection. By bringing together several works by different authors we were able to adapt these methods to a new application, in this case to visually perceiving symbolic representation for robotic manipulation.

From this research we can draw the following conclusions:

- Generating a synthetic dataset allows to obtain a big amount of data, without any wrong annotations in a relatively small amount of time, the time of rendering is only determined by the GPU performance.
- The limited number of 3D models available decreases the generalization made by the network, which increases the difficulty to transfer learning to real world.
- Neural networks trained in uniquely synthetic data may have a low performance in real world data compared when evaluated in synthetic data. To solve this problem training with a very small amount of real data can substantially improve the performance.
- Scenes that are more complex, with more object categories, require more data to achieve the same performance in object detection than scenes that are less complex.
- Having a deep-learning based model with a visual appearance cue and a spatial cue is enough to detect relationships between objects in 2D RGB images. Moreover, the model is suitable for real time application for the number of objects and relationships present in this work.
- Adding more predicates between the objects would improve the information extracted from the environment which would increase the number of possible robotic manipulation tasks.

5.1 Future Work

Although we have achieved positive results with performance similar to the state of the art, there are several improvements that could be done in order to increase the usability of our method in

a real environment. To improve the autonomy of robotic manipulation, we could improve our framework by increase the extracted information from the visual input. Next, we present some suggestions that could be implemented in the future.

5.1.0.1 End-to-End Network

We chose to train two different networks, one for object detection and another for relationship detection between the objects previously detected. Both networks share the same weights in the first layers since they used the same VGG-16. This was chosen in order to decrease the complexity of the networks implementation. However merging these networks and perform an end-to-end training could improve the overall performance [38]. Moreover, we could use the network to make real-time predictions of objects classes and their relationships. In the current implementation, the image is passed in the object detector and the output of this model together with the image are used as input of the relationship detector.

5.1.0.2 RetinaGAN

The gap between the simulated images and the real world is the reason that makes the deployment of models in real world trained in synthetic data difficult. To overcome this visual reality gap we could use pixel-level domain adaptation, these methods employ generative adversarial networks to translate the synthetic images to the real world domain [47], the down side of this method is that a GAN can change the image by removing information necessary for the given task, for example, it is important to preserve the scene features that directly interact with the robot.

[48] proposed a new network model called RetinaGAN that aims to adapt simulated images in realistic ones with object detection consistency. RetinaGAN is an extension of CycleGAN [49], an approach that learn bidirectional mapping between two different domains. The new approach trains with a frozen object detector that provides object consistency loss. Similarly to CycleGAN the model uses unpaired data without labels.

To keep the object detection invariance, they use a pre-trained and frozen EfficientDet model on each image and compute the perception consistency loss. This loss penalizes the generator for discrepancies in object detections between translations. The Perception Consistency Loss is given by:

$$\begin{aligned} L_{prcp}(x,y,F,G) = {} & L_{prcp}(x,G(x)) + \frac{1}{2}L_{prcp}(x,F(G(x))) \\ & + \frac{1}{2}L_{prcp}(G(x),F(G(x))) + L_{prcp}(y,F(y)) \\ & + \frac{1}{2}L_{prcp}(y,G(F(y))) + \frac{1}{2}L_{prcp}(F(y),G(F(y))) \end{aligned} \tag{5.1}$$

Compared to the CycleGAN loss, the RetinaGAN loss will only have this additional component for object invariance yielding the following loss function:

$$L_{RetinaGAN}(G,F,D_x,D_y) = L_{CycleGAN}(G,F,D_x,D_y) + \lambda_{prcp}L_{prcp}(x,y,F,G) \tag{5.2}$$

The researchers trained a Q2-policy to perform a pushing task in simulation and obtained 90% of success however when deployed to real they obtained 0% success, by applying RetinaGAN they where able to create a policy that achieved 90% of success rate in the real domain. This shows the effective of using a RetinaGAN to transfer simulated learning to real world.

This work could be used in this thesis to improve the success rate in the real domain by transferring the synthetic images to a more realistic domain.

5.1.0.3 Pose Estimation

In robot manipulation, in order to grasp objects with a robot hand or gripper a list of good grasps may be calculated in advance based of geometric information and the physical model of the objects grasped. The requirement for the full description of the physical model of the object makes this approach unsuitable for exploration of unknown scenes through image and depth sensors only, as they often provide partial and noisy information.

A paper published in 2020 [3], proposed a new approach for 6D object pose estimation from 2D RGB images. The method is highly accurate, efficient and scalable achieving a performance of 97.35% in terms of the ADD(-S) metric on the 6D pose estimation benchmark dataset Linemod using RGB input, being the current state of art for 6D object pose estimation from RGB input. The method proposes a end-to-end architecture which improves the real time performance, achieving 27 FPS in a multi-object scenario (up to eight objects).

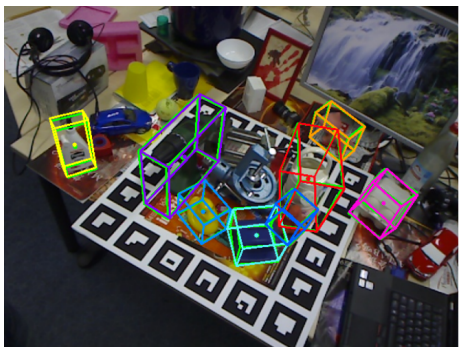


Figure 5.1: Example prediction for qualitative evaluation of the model performing single shot 6D multi object pose estimation while running end-to-end at over 26 FPS. Green 3D bounding boxes visualize ground truth poses while our estimated poses are represented by the other colors. Source: [3]

A 6D pose gives the information of the translation and rotation of an object, this information could be used together with our network to not only classify the object, their 2D bounding box and their relationships but also their 6D pose which is very useful for autonomous robotic manipulation. Moreover, the translation and rotation of the objects can be used to improve the prediction of relationships between the objects, particularly the spatial ones.

References

[1] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. *Lecture Notes in Computer Science*, page 21–37, 2016. URL: http://dx.doi.org/10.1007/978-3-319-46448-0_2, doi:10.1007/978-3-319-46448-0_2.

[2] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015. arXiv:1409.1556.

[3] Yannick Bukschat and Marcus Vetter. Efficientpose: An efficient, accurate and scalable end-to-end 6d multi object pose estimation approach, 2020. arXiv:2011.04307.

[4] Harry A. Pierson and Michael S. Gashler. Deep learning in robotics: A review of recent research. *CoRR*, abs/1707.07217, 2017. URL: http://arxiv.org/abs/1707.07217, arXiv:1707.07217.

[5] Jens Kober, J. Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *Int. J. Rob. Res.*, 32(11):1238–1274, September 2013. URL: https://doi.org/10.1177/0278364913495721, doi:10.1177/0278364913495721.

[6] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey of learning methods. *ACM Comput. Surv.*, 50(2), April 2017. URL: https://doi.org/10.1145/3054912, doi:10.1145/3054912.

[7] Xin He, Zhi-Ming Liu, and Ji-Liu Zhou. Real-time human face detection in color image. In *Proceedings of the 2003 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.03EX693)*, volume 5, pages 2915–2920 Vol.5, 2003. doi:10.1109/ICMLC.2003.1260064.

[8] Haoran Wang, Y. Zhang, and Xiaosheng Yu. An overview of image caption generation methods. *Computational Intelligence and Neuroscience*, 2020, 2020.

[9] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen A. W. M. van der Laak, Bram van Ginneken, and Clara I. Sánchez. A survey on deep learning in medical image analysis. *CoRR*, abs/1702.05747, 2017. URL: http://arxiv.org/abs/1702.05747, arXiv:1702.05747.

[10] Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated Planning: Theory and Practice*. The Morgan Kaufmann Series in Artificial Intelligence. Morgan Kaufmann, Amsterdam, 2004.

[11] Ruichi Yu, Ang Li, Vlad I. Morariu, and Larry S. Davis. Visual relationship detection with internal and external linguistic knowledge distillation, 2017. arXiv:1707.09423.

[12] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Synthetic data and artificial neural networks for natural scene text recognition, 2014. arXiv:1406.2227.

[13] Xingchao Peng, Baochen Sun, Karim Ali, and Kate Saenko. Learning deep object detectors from 3d models, 2015. arXiv:1412.7122.

[14] Param S. Rajpura, Hristo Bojinov, and Ravi S. Hegde. Object detection using deep cnns trained on synthetic images, 2017. arXiv:1706.06782.

[15] Kai Wang, Fuyuan Shi, Wenqi Wang, Yibing Nan, and Shiguo Lian. Synthetic data generation and adaption for object detection in smart vending machines, 2019. arXiv:1904.12294.

[16] Jonathan Tremblay, Aayush Prakash, David Acuna, Mark Brophy, Varun Jampani, Cem Anil, Thang To, Eric Cameracci, Shaad Boochoon, and Stan Birchfield. Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018.

[17] Chaitanya Mitash, Kostas E. Bekris, and Abdeslam Boularias. A self-supervised learning system for object detection using physics simulation and multi-view pose estimation, 2017. arXiv:1703.03347.

[18] Hironori Hattori, Yasodekshna Vishnu Naresh Boddeti, Kris M. Kitani, and Takeo Kanade. Learning scene-specific pedestrian detectors without real data. In *Proceedings of (CVPR) Computer Vision and Pattern Recognition*, pages 3819 – 3827. IEEE, June 2015.

[19] 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, 2016. arXiv:1612.06890.

[20] Apostolia Tsirikoglou, Joel Kronander, Magnus Wrenninge, and Jonas Unger. Procedural modeling and physically based rendering for synthetic data generation in automotive applications, 2017. arXiv:1710.06270.

[21] Adrien Gaidon, Qiao Wang, Yohann Cabon, and Eleonora Vig. Virtual worlds as proxy for multi-object tracking analysis, 2016. arXiv:1605.06457.

[22] Matthias Müller, Vincent Casser, Jean Lahoud, Neil Smith, and Bernard Ghanem. Sim4cv: A photo-realistic simulator for computer vision applications. *International Journal of Computer Vision*, 126(9):902–919, Mar 2018. URL: http://dx.doi.org/10.1007/s11263-018-1073-7, doi:10.1007/s11263-018-1073-7.

[23] John McCormac, Ankur Handa, Stefan Leutenegger, and Andrew J. Davison. Scenenet rgb-d: 5m photorealistic images of synthetic indoor trajectories with ground truth, 2017. arXiv:1612.05079.

[24] Yi Zhang, Weichao Qiu, Qi Chen, Xiaolin Hu, and Alan Yuille. Unrealstereo: Controlling hazardous factors to analyze stereo vision, 2018. arXiv:1612.04647.

[25] Weichao Qiu and Alan Yuille. Unrealcv: Connecting computer vision to unreal engine, 2016. arXiv:1609.01326.

[26] N.Mayer, E.Ilg, P.Häusser, P.Fischer, D.Cremers, A.Dosovitskiy, and T.Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. arXiv:1512.02134. URL: http://lmb.informatik.uni-freiburg.de/Publications/2016/MIFDB16.

[27] Philipp Fischer, Alexey Dosovitskiy, Eddy Ilg, Philip Häusser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. *CoRR*, abs/1504.06852, 2015. URL: http://arxiv.org/abs/1504.06852, arXiv:1504.06852.

[28] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. URL: http://www.blender.org.

[29] Angel X. Chang, Thomas A. Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qi-Xing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. Shapenet: An information-rich 3d model repository. *CoRR*, abs/1512.03012, 2015. URL: http://arxiv.org/abs/1512.03012, arXiv:1512.03012.

[30] Cewu Lu, Ranjay Krishna, Michael Bernstein, and Li Fei-Fei. Visual relationship detection with language priors, 2016. arXiv:1608.00187.

[31] Mohammad Amin Sadeghi and Ali Farhadi. Recognition using visual phrases. In *CVPR 2011*, pages 1745–1752, 2011. doi:10.1109/CVPR.2011.5995711.

[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 *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3668–3678, 2015. doi:10.1109/CVPR.2015.7298990.

[33] Bo Dai, Yuqi Zhang, and Dahua Lin. Detecting visual relationships with deep relational networks, 2017. arXiv:1704.03114.

[34] Yikang Li, Wanli Ouyang, Xiaogang Wang, and Xiao'ou Tang. Vip-cnn: Visual phrase guided convolutional neural network, 2017. arXiv:1702.07191.

[35] Hanwang Zhang, Zawlin Kyaw, Shih-Fu Chang, and Tat-Seng Chua. Visual translation embedding network for visual relation detection, 2017. arXiv:1702.08319.

[36] Xiaodan Liang, Lisa Lee, and Eric P. Xing. Deep variation-structured reinforcement learning for visual relationship and attribute detection, 2017. arXiv:1703.03054.

[37] Kongming Liang, Yuhong Guo, Hong Chang, and Xilin Chen. Visual relationship detection with deep structural ranking. In *AAAI*, 2018.

[38] Hanbo Zhang, Xuguang Lan, Xinwen Zhou, Zhiqiang Tian, Yang Zhang, and Nanning Zheng. Visual manipulation relationship network for autonomous robotics. In *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*, pages 118–125, 2018. doi:10.1109/HUMANOIDS.2018.8625071.

[39] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009.

[40] Ross Girshick. Fast r-cnn, 2015. arXiv:1504.08083.

[41] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2016. arXiv:1506.01497.

[42] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016. doi:10.1109/CVPR.2016.91.

[43] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge, 2015. arXiv:1409.0575.

[44] Petro Liashchynskyi and Pavlo Liashchynskyi. Grid search, random search, genetic algorithm: A big comparison for nas, 2019. arXiv:1912.06059.

[45] Xiaojun Chang, Pengzhen Ren, Pengfei Xu, Zhihui Li, Xiaojiang Chen, and Alex Hauptmann. Scene graphs: A survey of generations and applications, 2021. arXiv:2104.01111.

[46] Qi Wu, Damien Teney, Peng Wang, Chunhua Shen, Anthony Dick, and Anton van den Hengel. Visual question answering: A survey of methods and datasets, 2016. arXiv:1607.05910.

[47] Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. *CoRR*, abs/1612.05424, 2016. URL: http://arxiv.org/abs/1612.05424, arXiv:1612.05424.

[48] Daniel Ho, Kanishka Rao, Zhuo Xu, Eric Jang, Mohi Khansari, and Yunfei Bai. Retinagan: An object-aware approach to sim-to-real transfer. *CoRR*, abs/2011.03148, 2020. URL: https://arxiv.org/abs/2011.03148, arXiv:2011.03148.

[49] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017.