



universität
wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

**Scene2Model framework: Relation prediction
using logical tensor networks**

verfasst von / submitted by

Neli Petkova BSc

angestrebter akademischer Grad /

in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien, 2023 / Vienna, 2023

Studienkennzahl lt. Studienblatt /
degree programme code as it
appears on the student record sheet:

UA 066 926

Studienrichtung lt. Studienblatt /
degree programme as it appears on
the student record sheet:

**Wirtschaftsinformatik /
Business Informatics**

Betreut von / Supervisor:

**o. Univ.-Prof. Dr.techn.
Dimitris Karagiannis**

Contents

1	Abstract	4
2	Introduction	5
2.1	Scene2Model framework	5
2.2	OMiLAB - Smart Innovation Environment [1] [2]	7
3	Literature Review	10
3.1	Object detection models	10
3.1.1	Convolutional Neural Network Backbones	12
3.1.2	One-stage object detection	12
3.1.2.1	Single-shot multi-box detectors (SSD) [3]	13
3.1.2.2	RetinaNet [4]	16
3.1.2.3	YOLO - You Only Look Once [5]	18
3.1.3	Two-stage object detection	24
3.1.3.1	R-CNN [6]	24
3.1.3.2	SPP-net [7]	29
3.1.3.3	Fast R-CNN [8]	31
3.1.3.4	Faster R-CNN [9]	36
3.1.3.5	Mask R-CNN [10]	40
3.2	Visual relation detection	47
3.2.1	Visual phrases [11]	48
3.2.2	Visual relationship detection (VRD) with language priors [12]	50
3.2.3	VRD with with deep relational networks [13]	53
3.2.4	VRD with with deep structural ranking [14]	55
3.2.5	VRD with visual translation embedding network [15]	59
3.2.6	VRD with Internal and External Linguistic Knowledge Distillation [16]	61
3.2.7	VRD with logical tensor networks [17]	64
3.3	Scene2Model approach [18] [19]	68
4	Implementation Approach and Technology Selection	70
4.1	Agile Modeling Method Engineering	70
4.2	Object detection	72
4.3	Visual relationship detection	74

5	Implementation	78
5.1	UML diagrams	80
5.1.1	Use case diagram	80
5.1.2	Activity diagrams	81
5.1.3	Component diagram	92
5.1.4	Deployment diagram	93
5.2	Design thinking setup	94
5.3	Hardware and software prerequisites	96
5.3.1	Object detection & visual relationship identification . .	96
5.3.2	Mobile application	97
5.3.3	Modelling and Meta-modelling tool	98
5.4	Mobile application	99
5.5	Object detection using YOLOv4 [20]	101
5.6	Relation prediction using logical tensor networks	105
5.7	Modelling language creation	114
5.8	Conceptual model generation	115
6	Conclusions	117
6.1	Thesis Results	117
6.2	Open Topics	118
	Appendices	119
A	User manual	119
B	Source code	129
C	Statutory Declaration	130
D	Abstract	131
D.1	Abstract in English	131
D.2	Abstract in German	131

1 Abstract

The continuous need for driving business innovation and process optimization requires for manners to close the gap between expert knowledge and the machines or individuals that will be performing the actual work. To address this challenge, we present our Scene2Model framework that enables the extraction and formalization of domain knowledge directly from business experts by motivating them to create a physical representation of their innovation ideas in a guided design thinking workshop. The emerged physical scene is then transformed into a digital conceptual model using object detection and relation prediction tools. The output of the Scene2Model framework is a universal digital model (an abstract generic representation that captures the fundamental concepts and relationships applicable across various domains or industries), which is machine processable and can be used for checking the feasibility of otherwise resource-intensive and thus high-cost solutions.

2 Introduction

2.1 Scene2Model framework

Domain expertise and business insights do not usually coexist with technical skills in the common team structure in big or small corporations. [21] So when thinking about innovations or optimization, they are normally triggered top down from the management layers, but have to be executed in practice by employees or machines. The gap between the human specialist knowledge and intention and the underlying infrastructure or machines e.g. in production can be bridged through building structured executable conceptual representations of the real world such as domain models. A domain model is in fact conceptualized expert knowledge about some area of life with a certain level of abstraction. The essential information is presented in a systematic way using a predefined vocabulary and acknowledging the connections between the different model components. [21]

As noted, the main shortcoming in the realization of business-driven improvements and process optimizations comes from the fact that the people, having the domain insights and making the strategic decisions, do not overlap with the ones capable of conceptual modelling and translating knowledge into the an abstract and structured form, necessary for the planned execution. [22] That is exactly where the inspiration for our Scene2Model framework came from. Its main purpose is to provide a common understanding about the domain in question, that is both usable by business experts, technical professionals and computers and all that in a universally applicable and automatised fashion. The Scene2Model framework achieves this by facilitating a design thinking workshop, in which the domain specialists create a physical representation of the their innovation ideas or a scene, which is then transformed into a digital conceptual model with the help of an object detection and relation prediction software. The final digital model is abstract, formally structured and machine understandable and can serve as basis for simulation and proof checking else expensive and time consuming projects.

To better understand how the Scene2Model framework functions, we will briefly introduce its components and the logical flow of events. Firstly, as mentioned, the essential prerequisite our Scene2Model framework has is general knowledge about the domain. Preliminary information about the domain components, the logical connections between them and all known facts and constraints, that define the scope of relations and any dependencies or lim-

itations that persist, has to be gathered in the first phase of preparation by surveying the business experts. Once we have a formal specification of the domain, containing a set of objects and relations, that bind them together, a set of figures, to be used as graphical representation for the domain objects, is selected and printed out on paper, marking the second phase of preparation. The paper figures are photographed from different angles and fed into an object detection model for training. The possible connections between entities are formally defined using expressions in first order logic and have no physical representation other than the fact, that related object should be spatially closely aligned to each other. This sums up the preparation or the training phase of the Scene2Model framework. What follows is a live guided design thinking workshop with the domain experts, that are motivated to clench up their brains and consider possible improvements and innovations. To document their thinking process, they are encouraged to use the mentioned paper figures and place them on a canvas to form a coherent physical representation of the emerged ideas, wherein related entities should be closely situated near each other to indicate connectedness. At the end of the workshop a snapshot of the paper figure composition, also referred to as the physical model, is created. The image, capturing the physical model is provided as input to the recognition part of our Scene2Model framework, which solves two tasks - object detection and relation prediction. The recognised object instances and the relations, that connect them, are converted to a universal digital model by the model generation unit, which rounds up the scope of our Scene2Model framework. The created general model can be further translated into more specific models like e.g. TU Graz's EcoViz [23] using model transformation tools, but this is beyond the reach of the framework, presented in this thesis.

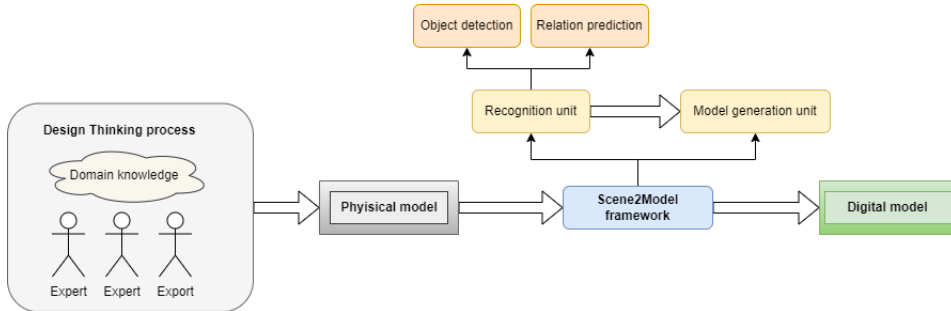


Figure 1: Scene2Model framework overview

2.2 OMiLAB - Smart Innovation Environment [1] [2]

This thesis is conducted as part of Vienna hub of the non profit organization OMiLAB (Open Models Laboratory), which is an open digital ecosystem with the initial goal of supporting the process and providing guidance and support at the development and administration of conceptual modeling methods. The main objective was to provide an open innovation and collaboration space for stakeholders involved in the research, education and design of modeling methods and model-driven digital products. From an overall perspective OMiLAB aims to elevate the process of modeling method development by establishing an agile conceptualization framework in a dedicated open ecosystem and with the help of a methodology that caters to both novices and experienced shareholders. [1]

The evolving digital transformation and the rising interest in IoT (Internet of Things) and cyber-physical systems (CPS) has driven OMiLAB to broaden its scope from serving mostly modelling interested parties to providing an adequate environment for enabling digital innovation. The innovation process can be fundamentally broken down into three steps: (1) an idea for improvement is born in the head of a domain expert - the idea is still unstructured, (2) the idea is formally specified using domain semantics - the idea is now a concept, and (3) the concept is tested in practice, a digital twin proof-of concept or a demo is created. Inspired to smartify this process, the emphasis in OMiLAB has been lately put on a new key asset of the common ecosystem - the smart innovation environment for digital engineers. (Fig. 2)

The smart innovation environment enables the idea creation and improvement identification by establishing the so called digital design thinking workspace, which provides a design thinking toolkit, assistance at the organization of innovation workshops with business experts and lastly the necessary technology for digitising the results. At conceptualization level, OMiLAB offers modelling and metamodeling tools with various interoperability channels to agilely create conceptual models from the gathered insights, which serve as both human- and machine understandable semantically rich formal representation of the improvement plan in place, and thus bridge the semantic gap between the design thinking results of the innovation workshops and the engineering of the digital twins proofs-of-concepts. The digital twins (robotic and IoT devices) and the underlying infrastructure for programming them as well an integration platform, which acts as a container for interoperability connectors with external systems or services, is provided by the last com-

ponent of the smart innovation environment - the digital experimentation workspace. (Fig. 2)

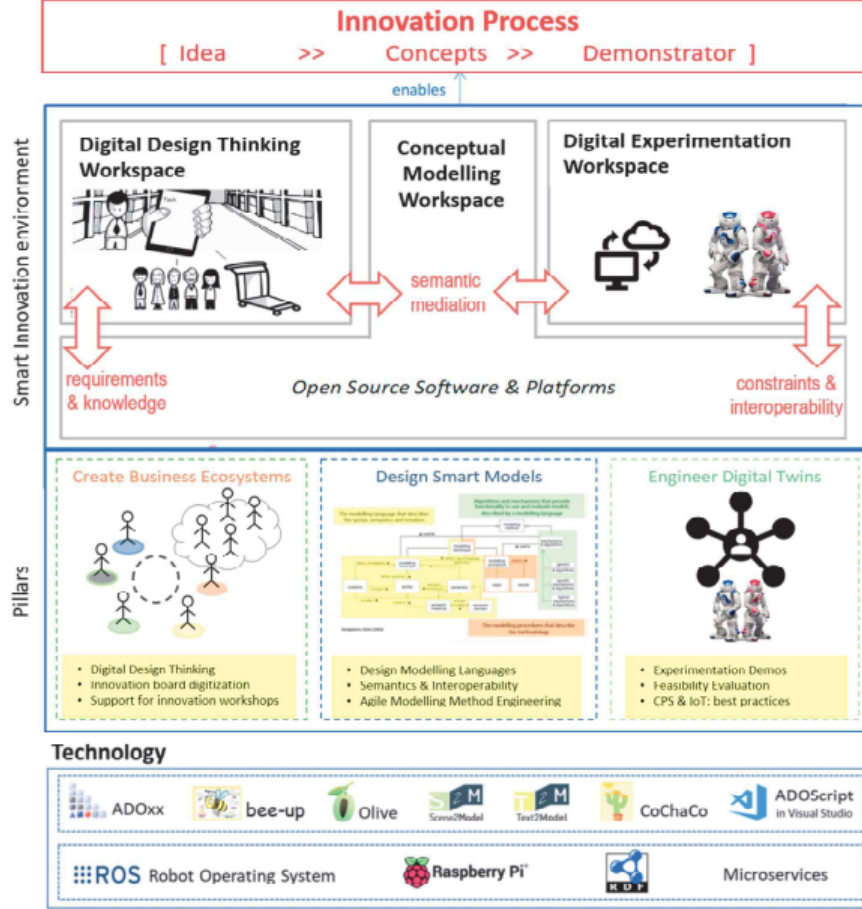


Figure 2: The smart innovation environment structure of OMiLAB [2]

The smart innovation environment, described above, has made this research possible by providing the necessary software and hardware resources and workspace configurations for digital design thinking and conceptual modeling. The preliminary phase of the Scene2Model framework, proposed in this thesis, relies on extracting business insights in an innovation workshop, which is facilitated by the digital design thinking workspace, emerged at OMiLAB. The digital design thinking workspace made available the physical artefacts to be used in the workshops - extensive sets of general (persons and build-

ings e.g.) and more specific paper figurines (e.g. medical equipment), which are part of a professionally designed toolkit by SAP AppHaus [24], used for creating storyboards with the aim of achieving a common understanding of the problem to be solved and validating the proposed ideas. The necessary infrastructure for capturing the results of the design thinking process (e.g. a web camera) and for the training of the digitisation unit (the recognition unit for object detection and relation identification) has been also generously provided by the Vienna node of the Open Models Laboratory (OMiLAB). Additionally, with a focal point on conceptualization processes and model-driven digital products, the conceptual modeling workspace at OMiLAB supplied the modelling and metamodeling tools, as well as an integration platform and interoperability facilitators for the agile domain-specific modeling method development and conceptual model creation in our model generation unit. Last but not least, without the guidance and support of the OMiLAB collaboration network throughout the whole innovation process, none of this would have been possible.

3 Literature Review

3.1 Object detection models

Object detection is a computer vision task, which deals with localizing objects within images, videos and other multimedia and allocates them to a pre-defined set of classes. [25] Object detection models require a large amount of labeled images as input for the training phase in order to detect and classify instances of the same classes, defined in the previous stage, on new visuals. Labeling an image includes annotating all objects of interest with a bounding box (the coordinates of a rectangular area, which covers the object tightly) and providing a class name for each instance. [26] State of the art object detection algorithms mostly rely on deep learning models, that apply convolutional neural networks (CNN), in contrast to other deep learning techniques, designed for solving other tasks. [26] Deep learning is a subset of machine learning, which requires no human interaction when it comes to feature extraction thus prevalent in visual tasks. Neural networks, which are the backbone of deep learning algorithms, are modeled to imitate neural activity in the human brain in that signals are transformed and passed along different paths in a complex network of neurons based on a sophisticated logic. [27] Simply put, neural networks apply magic in a black box to transform information input for the aim of answering some question and encoding a decision of the initial problem. CNN as a subset of neural networks have a specific structure, including three types of main layers: (1) convolutional layer, (2) pooling layer and (3) fully-connected layer, and are especially successful at dealing with image processing and other visual tasks due to their structure. [28] The convolutional layer is the core component of a CNN and plays a crucial role, as most of the computation is performed inside of it. It works like magnifying glass in that it takes a small region (typically a 3x3 matrix) of the image, checks if the feature (the object class is divided in building blocks or features) is present and repeats the process, moving across the whole image and building a so called feature map. At first, the features are as simple as colors and edges, but with the progression of the image detection process, the intended object class is detected as a whole. [29]

Depending on how much of the input data is annotated, we can differentiate between supervised learning - the whole training data is labeled [30], unsupervised - none of it is labeled [31], and semi-supervised - some of it is annotated and some - not [32]. Object detection algorithms can be also cat-

egorized based on the used approach. One-staged object detectors or single shot detectors perform object localization and classification in just one step, whereas two-staged object detectors first generate the region proposals (areas, in which objects are located with a high probability) and perform the content prediction and fine tuning of the bounding boxes in the second stage. [26] One-stage detectors are faster due to the simpler architecture in comparison to two-stage detectors that have higher localization and recognition accuracy. [33] Both approaches provide certain advantages, so the decision on which one is more beneficial to use depends on the specific scenario. For real-time applications and embedded systems you would opt for speed and resource efficiency, whereas when time and computing power are not a real constraint and accuracy is the biggest priority like in medical applications, more reliable two-stage detectors are a better option. [33] Popular two-stage detectors include Faster RCNN [9] and Mask RCNN [10] and for single-stage detection single-shot multi-box detectors (SSD) [3], YOLO [5] and RetinaNet [4] present a preferred choice.

The most often used benchmark dataset for evaluating object detection model accuracy is MSCOCO (Microsoft Common Objects in Context) [34], which consists of 324 000 images, split into training, test and validation set of images with annotations of instances of objects of 80 distinct categories.

The most popular metric for analyzing the performance of object detection models is mAP (mean average precision) [35] and takes values between 0 and 1. mAP is calculated based on the recall and precision given a IoU (Intersection over Union) threshold - an average precision is calculated inside each class and the mean is build across the average precision of the classes. IoU represents the ratio between the area of intersection of the two bounding boxes (ground truth and predicted bounding box) and the area of union, therefore the higher the better precision of detection. Precision and recall refer to how well you can differentiate between true positives out of all positive and out of all predictions respectively. As false positive in the scope of object detection is understood that the model has detected an object, but the object is not there with regards to the ground truth. [36] Analogously, a true positive is a correct prediction. mAP seeks to maximize the combination between recall and precision, encapsulating the trade-off between both metrics, ensuring that given a high recall (correctly detected / positives w.r.t ground truth) false positives are not disregarded, but are penalized instead, and that given a high precision (correctly detected / all predictions), missed instances (not detected) are also taken into consideration when calculating

the robustness of an object detection model. [35]

3.1.1 Convolutional Neural Network Backbones

The term backbone in the context of convolutional neural networks and object detection refers to an already trained CNN, which performs well at simpler visual computing tasks like object classification, and due to the transferability in deep learning (the ability to acquire and reuse knowledge) is often used at the front of object detection models for feature extracting. [37] Backbone CNN take images as input and unpack them into lines, dots, edges etc. (low-level features) in the first layers and build up on those in later layers to construct and learn higher-level features in order to eventually recognize the intended object. The most used backbone CNN include AlexNet [38] (used in RCNN [6]), VGG-16 [39] (used in Faster RCNN [9] and SSD [3]), GoogLeNet or Inception V1 [40] (used in RetinaNet [4] and Faster RCNN [9]) and DarkNet [41] (used in YOLO [5]).

3.1.2 One-stage object detection

Deep learning and in particular deep convolutional neural networks have shown to be the most efficient technology for solving visual computing tasks like object detection. [25] The object detection process consists of two main components: (1) object localization (the output is a bounding box - the coordinates of a rectangular area around the object) and the classification of the detected object (the output is the class assignment of the object of interest). In two-stage object detectors the object localization step is separated into two stages: (1) region of interest (RoI) proposal and (2) classification of the object in the proposed RoI and bounding box regression (fine-tuning of the RoI). One-stage object detectors on the other hand yield both tasks in one single step using one single deep convolutional neural network by skipping the RoI proposal step and directly tackling the prediction of the bounding box coordinates and the class allocation at once. [26]

One-stage object detectors are growing in popularity due to the increasing need for highly time and cost efficient deep learning models for real-time image processing and other visual tasks. In the following sections we will take a look at a few examples like single-shot multi-box detectors (SSD) [3], YOLO (You Only Look Once) [5] and RetinaNet [4], that show similar or better accuracy, compared to state-of-the-art two-stage models, and are

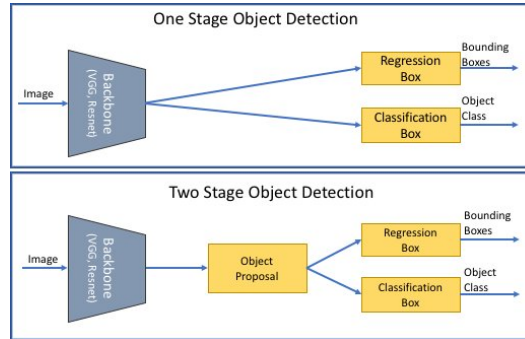


Figure 3: One-staged vs. two-staged object detection [42]

faster and require significantly less computing power due to the simpler model architecture.

3.1.2.1 Single-shot multi-box detectors (SSD) [3]

Single-shot multi-box detectors (SSD) propose a novel solution to the object detection problem using one single convolutional network. SSD is the first deep network based object detector that does not re-sample pixels or features for bounding box hypotheses and is as accurate as approaches that do. [3] SSD compensate for skipping the region extraction step by incorporating accuracy improvement strategies like the usage of multi-scale feature maps to facilitate object prediction at different aspect ratios and adopting a fixed set of default bounding boxes as a starting point for the regression, which speeds up the bounding box fine-tuning. These enhancements have allowed SSD to outperform other object detection models on accuracy, while still maintaining real-time speed. (Fig. 4)

SSD use VGG-16 [43] for feature extracting and multiple convolutional layers with reducing in size feature maps and multi-box prediction for the actual object detection. (Fig. 5)

The convolutional layers in SSD split the image into a grid (e.g. the first SSD layer uses 38x38 grid) and apply a convolution filter across the grid to make a prediction for each grid cell. The prediction is based on a predefined set of default bounding boxes also known as anchor boxes. [44] Default boxes are generated from the ground truth and represent the most often observed shapes with regard to the height-width ratio of objects in the given dataset. (e.g. people figures usually have height-width ratio of approximately 0.4 and

Method	mAP	FPS	batch size	# Boxes	Input resolution
Faster R-CNN (VGG16)	73.2	7	1	~ 6000	$\sim 1000 \times 600$
Fast YOLO	52.7	155	1	98	448×448
YOLO (VGG16)	66.4	21	1	98	448×448
SSD300	74.3	46	1	8732	300×300
SSD512	76.8	19	1	24564	512×512
SSD300	74.3	59	8	8732	300×300
SSD512	76.8	22	8	24564	512×512

Figure 4: SSD performance and speed comparison to state-of-the-art object detection models [3] The results are conducted on on Pascal VOC2007 test. FPS - frames per second (the metric for speed), mAP - mean average precision (the metric for accuracy), SSD300 - SSD with input images of size 300x300, SSD512 - SSD with input images of size 512x512

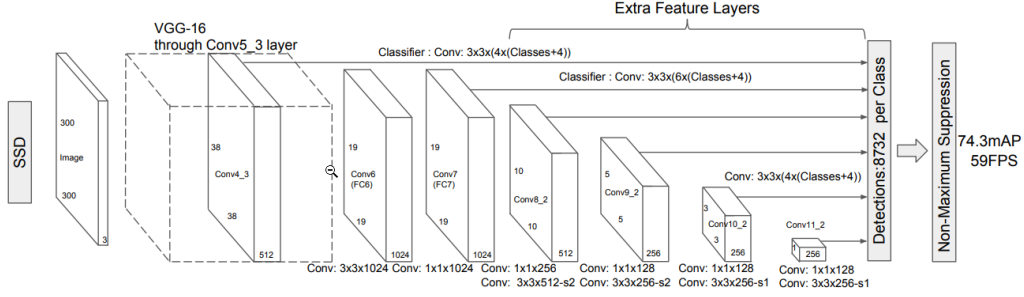


Figure 5: Single-shot multi-box detectors (SSD) architecture [3]

cars - about 2.6). As already discussed, in convolution layers a small sized filter (3x3 matrix) slides through the input (the image split into a grid), and builds a feature map of predictions. If we for instance have 3 default bounding boxes (e.g. horizontal rectangle, vertical rectangle and a square shape), a 38x38 grid and 15 object classes, the output of the convolutional layer would be a prediction, that contains the confidence score for each class + 1 reserved spot, if there is no object, and the offset of the ground truth coordinates from the default bounding box. (Fig.6) The convolutional filter (e.g. a 3x3 matrix) moves across the grid and performs this computation for each grid cells (38x38) and for every default bounding box, resulting in an output matrix of size [38x38x3 (default filters) x20 (15 object classes + 1 for no class + 4 for bounding box offset)]. The match with the highest confidence score is selected for bounding box refinement, whereby the confidence score is a function of the IoU (intersection over union) between the default bounding

box and the ground truth bounding box and as match is considered an IoU greater than 0.5. (Fig.7) As a consequence, to avoid over-fitting the number of default bounding boxes should be kept as small as possible (typically between 4 and 10).

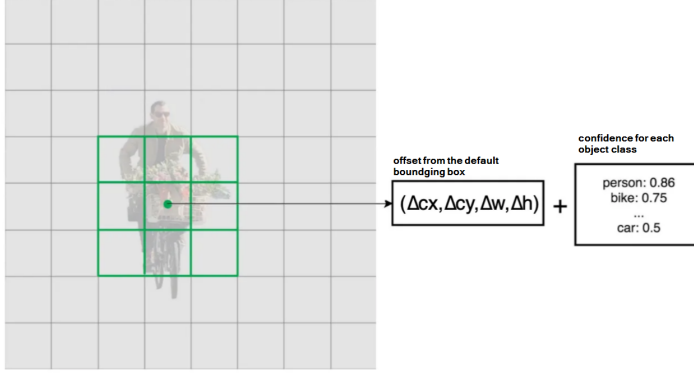


Figure 6: The output of SSD for a 3x3 convolutional filter and 8x8 grid [45] The offset of the default bounding box from the ground truth is defined as [delta centroid x, delta centroid y, delta width, delta height]

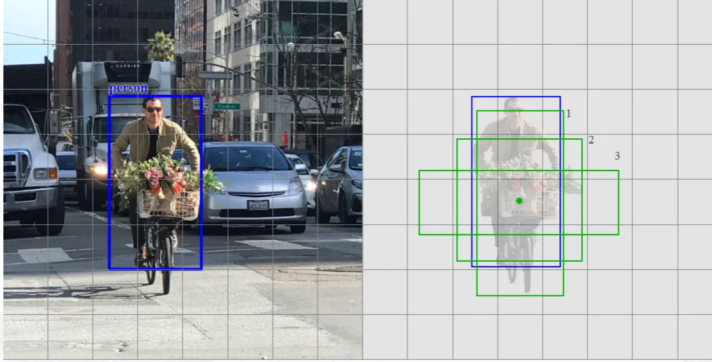


Figure 7: Positive match for class person and default bounding boxes 1. and 2. [45] The ground truth is colored in blue, the default bounding boxes - in green.

Additionally, the grid size decreases in dimensions over the different SSD layers, resulting into so called multi-scale feature maps, which allows for different sized objects to be detected - smaller sized object are detected in

the front layers (with higher resolution feature maps) and bigger-scale object - in the later ones (with lower resolution feature maps). Here it should be also noted, that SSD does not perform as well as Faster RCNN e.g. at recognizing small-sized objects due to the fact, that the features in the leftmost layers are still as simple and basic as dots and edges, which affects the learning and recognition accuracy negatively.

3.1.2.2 RetinaNet [4]

RetinaNet is an one-stage object detection model, which uses ResNet-101-FPN (FPN on top of ResNet) as backbone for feature extracting. (Fig.8).

ResNet or residual neural network is a deep learning model, designed to support deeper CNN architectures without facing the traditional challenge of the exploding or vanishing gradient.[46] Neural networks are trained using the stochastic gradient descent on the backpropagation error, seeking to find the weights that minimize the loss function. Too many layers will eventually reduce the gradient until it disappears (leaning towards 0), which deteriorates performance, or degenerates the model with every additional layer. ResNet tackles that problem by adopting shortcut connections or skip connections, which stack a number of convolutional layers, called residual blocks, and reuse the activation of the layer directly before the block and pass it as output to the subsequent layer, which smooths out the gradient flow during backpropagation, enabling CNN to scale to more than 100 layers.

FPN stands for feature pyramid networks and is an architecture, used for multi-scale feature extracting in visual processing models. [47] Object detectors address the challenge of the localization and recognition of different-sized objects by creating feature maps with diverse scales (the input image is split into varying in size grid shapes), usually decreasing in spatial dimensions (bottom to top layer), and using the feature maps from all layers to make predictions. Similar to SSD, FPN uses a fixed set of anchors (default bounding boxes in SSD), compares them to the ground truth and calculates an offset, which is then minimized in the bounding box regression. Due to the visual similarity with pyramids, such architectures are referred to as feature pyramids. Higher resolution feature maps are responsible for the detection of small objects and lower resolution feature maps - for bigger ones. Small object detection typically suffers low accuracy in that case due to the fact that higher resolution feature maps (that can only match small-scale items)

are constructed in the first layers, where the features are still too immature (consisting of e.g. only dots and edges), which has also been observed in single shot multi-box detectors (SSD). FPN has adopted a top-down pathway and lateral connections between the feature maps on the different layers to semantically enrich the pyramidal feature hierarchy on the lower levels by combining and leveraging low-resolution, semantically strong features with high-resolution, semantically weak features and thus increase the accuracy at all scales. [47] (Fig.8) The top-down pathway zooms out and hypothesizes features with more expressive power on lower resolution layers by passing them down from higher resolution layers. The lateral connections merge the findings of the bottom-up created feature maps (the original feature maps created by ResNet) and the top-down ones (created by the top-down pathway) by using simple element-wise addition.

Aside from the briefly introduced backbone, consisting of ResNet and FPN, which is responsible for the feature map extraction from an image input, RetinaNet’s architecture consists of two additional convolutional sub-networks for the object classification (class subnet) and for the bounding box regression (box subnet), which deliberately follow a simple design. (Fig.8)

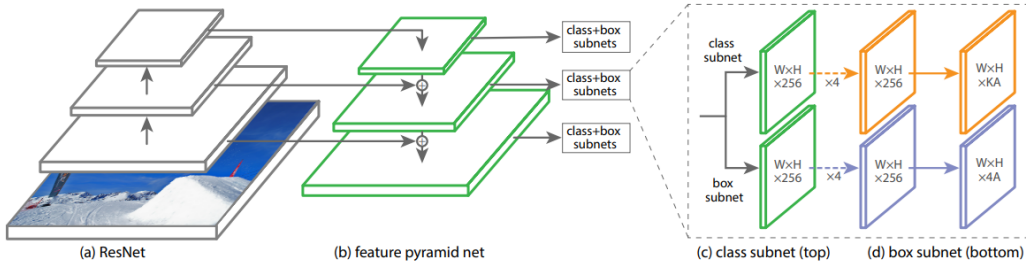


Figure 8: The RetinaNet architecture [4]

The simplicity of the RetinaNet’s model architecture is absolutely intentional, as the creators’ main focus was to address, what they refer to as the main disadvantage of one-stage models against their counterparts (two-staged models) - the class imbalance. The class imbalance between foreground and background classes consists in the fact that one-stage detectors evaluate a huge number of locations per image (in the range of $10^4 - 10^5$), out of which hardly any contain an object (foreground classes) and the rest are only background image details. This imbalance leads to inefficient learning during training, as most locations are easy negatives (negatives = image regions

with no objects identified or background regions, easy vs. hard = examples with large errors), which bring no useful information to the model, but rather overwhelm the training and tend to degenerate the model. [4] RetinaNet proposes a focal loss function for tackling the class imbalance problem without sampling or reducing the number of locations to evaluate. [4] Other models typically address this issue by filtering out easy examples and focusing the training on hard negatives, whereas in contrast RetinaNet seeks to minimize the contribution of inliers (easy examples with small errors) by assigning them with lower weights, so that even if there is a lot of them, their impact on the total loss is brought to a minimum.

The usage of a focal loss function for confronting the challenge of the class imbalance especially in one-stage detectors, where region proposal is not performed separately, but rather a huge number of locations (image regions) has to be evaluated for object localization and recognition, has shown to be so effective, that RetinaNet outperforms state-of the art one- and two staged models on the COCO dataset. The COCO dataset is a good baseline, because it also contains a lot of small objects, proving to have closed the accuracy gap at smaller scales too. (Fig. 9)

	backbone	AP	AP ₅₀	AP ₇₅	AP _S	AP _M	AP _L
<i>Two-stage methods</i>							
Faster R-CNN+++ [16]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [20]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [17]	Inception-ResNet-v2 [34]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [32]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	52.1
<i>One-stage methods</i>							
YOLOv2 [27]	DarkNet-19 [27]	21.6	44.0	19.2	5.0	22.4	35.5
SSD513 [22, 9]	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 [9]	ResNet-101-DSSD	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet (ours)	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
RetinaNet (ours)	ResNeXt-101-FPN	40.8	61.1	44.1	24.1	44.2	51.2

Figure 9: RetinaNet performance compared to state-of-the art one- and two-staged models on COCO test-dev [4]

3.1.2.3 YOLO - You Only Look Once [5]

YOLO or You Only Look Once is an object detection model, which uses only one convolutional network (one-staged model) to target both object localization and classification. The initial model architecture of YOLO consists of 24 convolutional layers, followed by 2 fully connected layers, and is influ-

our case the regression input are the image pixels directly and the desired result contains the bounding box coordinates and the corresponding class predictions.

YOLO approaches the problem of object detection by first dividing the input picture into a grid shape ($S \times S$), in which each grid cell is responsible for localizing a fixed number of bounding boxes (B) and their corresponding confidence scores (the level of certainty, that there is an object in the bounding box, multiplied by the intersection over union (IoU) between the predicted box and any ground truth box) on one hand and on the other hand for building one class probability map (a vector, containing the conditional probabilities for each class to be encountered in that grid cell - valid for all proposed bounding boxes). A bounding box prediction consists of 5 values - x (coordinate of the x -axis of the box centroid), y (coordinate of the y -axis of the box centroid), w (width), h (height), and the confidence score mentioned, hence the final prediction of YOLO is a $S \times S$ (B times $5 + C$) shaped matrix. (Fig. 11)

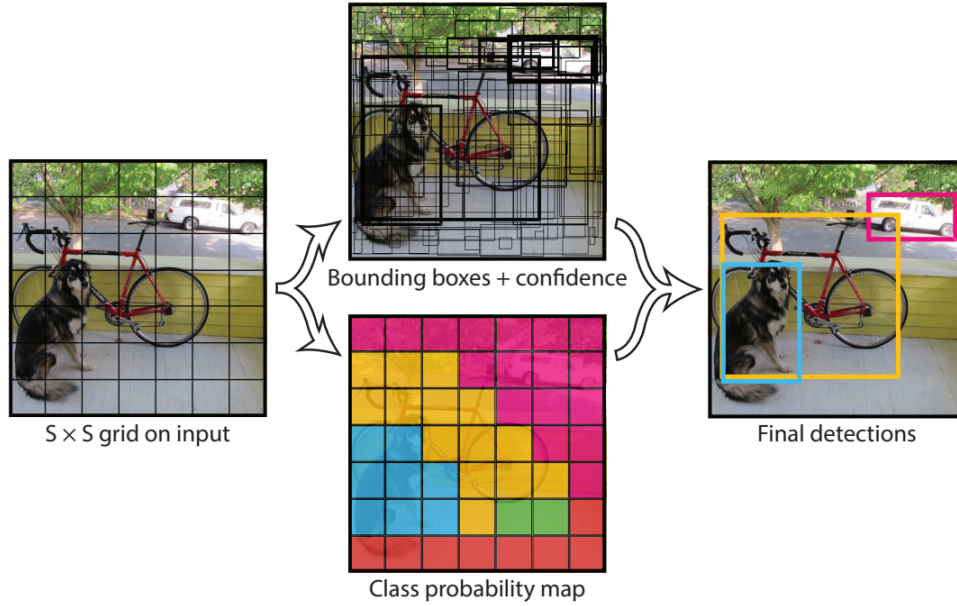


Figure 11: YOLO prediction process [5]

The fact that each grid cell predicts multiple but always the exact same number of bounding boxes has two important consequences, the first

of which is that a lot of redundant boxes are created for a single object. To remove the duplicates and choose the most suitable bounding box, YOLO uses a technique called non-max suppression (NMS). NMS works as follows: first, the best bounding box for an object is chosen, based on the intersection over union (IoU) with the ground truth (highest IoU = best box), then all extra boxes are deleted (a box is deleted, if it overlaps too much - having an IoU bigger than a given threshold - with the already approved best bounding box for that object) and the process is repeated for all classes. The other consequence is that YOLO does not perform well in the recognition of clustered objects (objects tightly located near each other), because (a) the number of objects to be detected by a single grid cell is limited, (b) close by objects are naturally reduced by NMS as duplicates and (c) all bounding boxes, proposed by one grid cell share the same class probability map, thus one grid cell can not predict objects of different classes.

YOLO generally performs well with regard to inference speed (runs at real time speed of 45 fps [5]), but has several limitations, when it comes to accuracy. Since the initial version of YOLO, released in 2015, multiple iterations have been released, proposing different improvements.

Similar to SSD [3], YOLO mainly struggles with the localization of small objects due to the fact that reduction layers erase some of the already scarce and semantically poor information about small-scale items during down sampling and max pooling, limiting the foundation for learning. One iteration of YOLO (YOLO v3 [48]) tackles that challenge by adopting FPN (feature pyramid network), similar to RetinaNet, which facilitates the creation of multi-scale semantically rich feature maps (bottom layers, carrying high resolution features, are enriched top-down) and thus the detection of various in size objects. Another measure to improve the detection of small-sized object has been adopted in YOLOv2 [49] and namely the multi-scale training, which consists in the resizing of the network on the fly during training rather than adjusting the image inputs, which is possible due to the fact that the model contains only convolutional and pooling layers. A side effect of the multi-scale training is the trade-off between accuracy and speed - with smaller size, the network is faster, but has poorer accuracy. (Fig. 12)

In YOLO v2 another key improvement is the introduction of anchor boxes or priors, which represent the most common shapes and ratios, that objects have, and by aligning well with the dataset, enable the detection of a broader range of sizes and shapes. The selection of good anchor boxes is crucial, because YOLO then predicts the displacement (the offset between the bounding

Detection Frameworks	Train	mAP	FPS
Fast R-CNN [5]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[15]	2007+2012	73.2	7
Faster R-CNN ResNet[6]	2007+2012	76.4	5
YOLO [14]	2007+2012	63.4	45
SSD300 [11]	2007+2012	74.3	46
SSD500 [11]	2007+2012	76.8	19
YOLOv2 288×288	2007+2012	69.0	91
YOLOv2 352×352	2007+2012	73.7	81
YOLOv2 416×416	2007+2012	76.8	67
YOLOv2 480×480	2007+2012	77.8	59
YOLOv2 544×544	2007+2012	78.6	40

Figure 12: YOLOv2 performance comparison on Pascal VOC 2007[49]

box and the ground truth) and not the box coordinates directly, and small offsets are simpler and thus easier to predict. In YOLO v2 [49] as well as in later versions [48] anchor boxes are selected using k-means clustering with distance metric based on the IoU instead of the standard choice - the Euclidean distance. In comparison to the initial version, where only one class probability map is created per grid cell, YOLO v2 makes class predictions and calculates objectness (equivalent to the confidence score in YOLO) for each anchor box. YOLO v3 predicts objectness with the help of logistic regression. [48]

In the initial version YOLO trains on images with resolution 224×224 pixels and performs detection on 448×448 pixels re-scaled pictures. In the second iteration of YOLO on top of the backbone training for image classification), which is again done on the same input size (224×224 pixels), a fine-tuning phase with smaller learning rate and higher resolution (448×448 pixels) images has been adopted for the last few epochs, which has improved the accuracy on ImageNet with up to 4% mAP (mean average precision).[49]

Another significant improvement on stability and accuracy (mAP increase of 2%) in YOLOv2 has been brought by the introduction of batch normalization on all convolutional layers, which helped with the challenge of over-fitting without the need of dropout, so it could be removed from the model.

Throughout the YOLO versions, the authors have experimented with different loss functions. In YOLO v1 and YOLO v2 the loss function is the typical one for linear regression - based on the sum of squared errors between the bounding boxes and the ground truth. In YOLO v3 the authors

tested a focal loss function to address the class imbalance problem (similar to RetinaNet), but this worsened the accuracy by 2% mAP. [48] YOLO v4 uses CIoU loss, which is calculated jointly based on the intersection area between the prediction bounding box and the ground truth, the distance between the centres of the boxes and the distinction in their aspect ratios. [20]

An important breakthrough has been made in YOLO v4 with the adoption of what they call "a bag of freebies" - a series of steps that can be taken to improve the model's performance without increasing latency at inference time. [20] They include tweaks in the data management and data augmentation, by e.g. generating new images out of the existing ones using filters and distortions, cropping, rotating and re-scaling etc. (Fig. 13) The YOLO v4 authors have come up with a new data augmentation technique called mosaic representation, which creates patched images by combining segments of four different pictures, facilitating the detection of objects outside of their normal context.

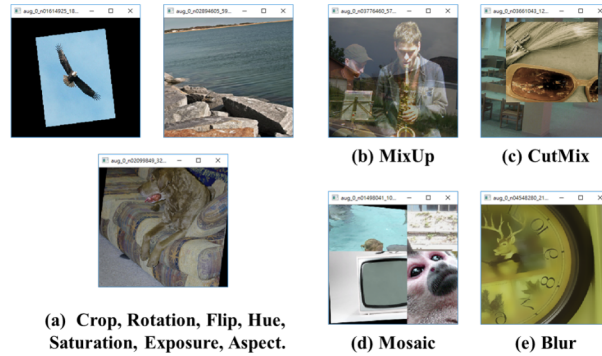


Figure 13: Data augmentation methods[20]

The backbone for feature extracting has also changed in the different YOLO versions - the initial one used a custom network inspired by GoogleNet [40] with reduction layers (Fig. 10), YOLOv2 switched to Darknet-19 [41], a variant of the VGGNet [43] architecture with 19 convolutional layers, which is simpler and faster in comparison to the initial backbone. YOLOv3 and YOLOv4 rely on variants of ResNet [46] architectures with shortcut or skip connections - YOLOv3 uses Darknet-53 [41] with 53 convolutional layers and YOLOv4 - CSPNet (Cross Stage Partial Network) with 54 convolutional layers.

Many unofficial version of YOLO have been released since YOLO v4 was

out in 2020, but critics argue that they were not peer-reviewed and tested to an adequate extent, thus they have remained unofficial.

3.1.3 Two-stage object detection

Two-staged object detectors divide the task of object recognition in two: (1) region proposal or the selection of image segments, that with high a likelihood contain an object and (2) bounding box regression or the fine-tuning of the bounding box coordinates in the proposed regions and object classification, consisting in the assignment of a class label to the identified object and a confidence score or the level of certainty the model has, that the prediction is true. In comparison to one-staged models two-staged object detectors tend to be slower, more resource intensive, but have higher level of accuracy, which makes them inappropriate of for real-time applications like autonomous driving and embedded system, but very much suitable in the medical sphere e.g., where accuracy is the highest priority. [33]

3.1.3.1 R-CNN [6]

R-CNN is a two staged model for object detection, which combines region proposal with convolutional neural networks (CNN), and has been named in a very straightforward fashion R-CNN (regions with CNN features). R-CNN consists of three main modules: (1) a module that generates region proposals, (2) a feature extraction module and (3) and object classification module. (Fig. 14) During region proposal, a set of 2400 image segments is selected as region candidates, using a method called selective search. [50]

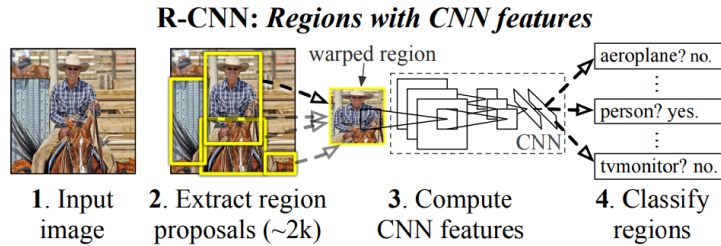


Figure 14: R-CNN system overview[6]

Selective search is a search algorithm, that combines the assets of image

segmentation, which makes use of information about the image structure to cluster pixels into regions, and exhaustive search, the tactic of which is to examine all potential possible locations and in the case of selective search - all possible image conditions for region building like color, texture and so on.

Image segmentation deals with the problem of splitting an image into regions or the building of visually significant groupings based on some similarity factor (e.g. color, motion, intensity etc.). Selective search utilizes Felzenszwalb's algorithm for graph-based image segmentation [51], which builds a graph, in which the nodes are the image pixels and the weight edges between the nodes are represented by the dissimilarity factor between them (intentionally reversed, as nodes having bigger visual difference should be far from each other, so that they are assigned to different groups). The algorithm works as follows: (1) initial state: every nodes is a group itself (2) conditioned the smallest distance between two nodes in different groups is smaller than the minimum internal difference of the two groups (which is governed by a factor k , if k =small, the model has preference from small groups and if k =large - for large), then the groups should be merged and (3) continue until no further groups can be combined. (Fig. 15)

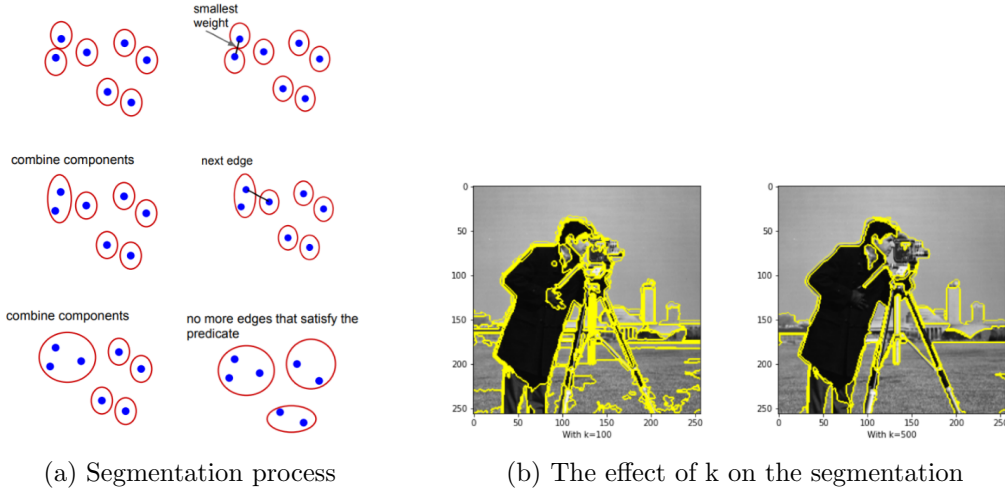


Figure 15: Felzenszwalb's algorithm [51]

Selective search uses the image segmentation method, proposed by Felzenszwalb [51] as a starting point and then applies hierarchical groupings to capture all possible scales (it joins regions based on their similarity and continues

until the whole image becomes one single region). As there is a variety of reasons for regions to constitute to an object, the region grouping strategies are diversified using four different similarity measures: it joins regions based on color, texture, size and fill.

Back to R-CNN, once phase one is done, we have around 2000 regions with arbitrary size and aspect ratios, which have to be fed to the convolutional neural network (CNN) for the feature extraction. The creators of R-CNN chose to use the AlexNet standard architecture for object classification, which requires input of fixed size 227x227 pixels [52]. The creators of R-CNN explored a few strategies to achieve the desired shape (Fig. 16), but in the end chose to use the warping mechanism, shown in Fig. 16 (D). This alteration of the original regions distorts the natural dimensions of the object and unfortunately attributes to a loss of accuracy. The addition of 16 pixels of context padding (second row in Fig. 16) around the region proposal showed an improvement of 3-5 mAP (mean average precision).

AlexNet was the the first widely acknowledged application of deep learning and the first to smash the competition in the ImageNet Large Scale Visual Recognition in September 2012 challenge by accomplishing an top-5 error rate of 16.4% (the first result under 25%) and an advantage of 9.8% advantage to the second best competitor, which skyrocketed the attention and further development of machine learning and in particular convolutional neural networks (CNN) for visual computing tasks. [52] AlexNet was the first CNN to leverage parallel GPU processing (AlexNet uses two GPUs simultaneously) and the first one to use ReLu (Rectified Linear Unit) activation function instead of the standard sigmoid, both of which conducted to making the training time manageable, compensating for the network depth and the resulting from it high number of parameters (60 million). The model consists of five convolutional layers and three fully connected linear layers, the last of which is a 1000 node softmax layer, which produces the class probability map of the 1000 classes (in ImageNet), necessary for the object classification task. (Fig. 17) The last key feature, employed by AlexNet for the first time and worth mentioning, is overlapping pooling, which summarizes a group of activations but in contrast to traditional pooling, when moved ahead, the new window does not slide away by the whole window size, but rather overlaps with the previously traversed one to some extent. This tactic has improved the top-1 and top-5 error rates by 0.4% and 0.3% and has shown to reduce overfitting. [52]

Back to R-CNN, in order to deploy AlexNet for the feature extraction, the



Figure 16: Different strategies for transforming the arbitrary shaped regions to have required size for the CNN input (227x227 pixels)[6]

(a) the original region proposal (b) the image region is scaled into the tightest square, including additional image context around it, keeping the original aspect ratios (c) same as (b), but the context around the region is excluded (d) the image region is twisted to match the required shape (anisotropically - without keeping the original aspect ratios). The first row has no context padding, the second row has context padding of 16 pixels

last fully connected layer with the softmax activation function is removed, leaving a fully connected 4096-dimensional layer as the last one, which means that the produced feature vector has 4096 dimensions. (Fig. 17)

Having a feature vector for all proposed image regions, the next step would be to classify the object, contained in the RoI (region of interest). R-CNN applies multiple support vector machines (SVM), one for each class from the dataset, on the feature vectors to calculate the class probabilities and chooses the class with the highest score. As previously discussed, regions, containing an object, are considered as positive examples and ones, that hold only background - as negative examples. Regions with an intersection over union (IoU) of less than 0.3 with any given ground truth bounding box are

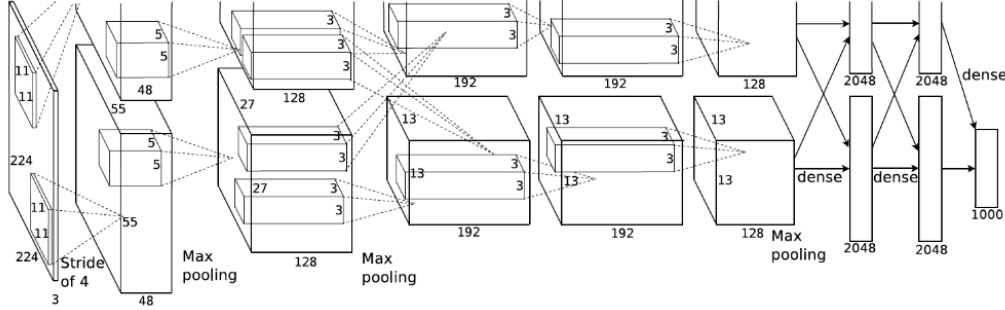


Figure 17: AlexNet architecture [52]

seen as negative examples. The distinction between positive and negative examples is important, because bounding box regression is performed only on positive examples and the loss function should also ignore the negative examples. Additionally, the SVM are trained on positive examples from the ground truth and a 5000 image subset of hard negative examples (using all increased mAP by only 0.5%, but doubled the SVM training time), taken from the proposed regions, which is a popularly used technique in classifiers called hard negative mining. In hard negative mining for SVM, the focus is set on selecting background image segments, which partially contain and object or have noisy texture and are thus hard (in contrast to easy clear empty backgrounds), so that the classifier can improve and recognize them after training.

Back to R-CNN, after running the support vector machine models on the 2000 or so region proposals per image and have a class probability vector per region, we can discard the negative examples and assign the positive examples to the class with the highest score. This still leaves us with a lot of redundant regions, that cover one and the same object. R-CNN utilizes non-maximum suppression (NMS) to remove all extra regions by first selecting the best image segment (having highest IoU with the ground truth) for an instance of some class and then eliminating all additional regions, that have a high overlap (surpassing some threshold) with the best image region, and repeat iteratively for all classes on all images. Finally, R-CNN takes the best region candidates and applies class-specific linear regression to fine tune the bounding box coordinates, aiming to find a transformation function from the region coordinates to the ground truth by minimizing the regularized squared errors. The concrete ground truth box to use for the regression of

each prediction is selected based on their overlap (highest overlap = best fit) as measured by the intersection over union (IoU) of both boxes. Predictions, which can not be assigned to a ground truth are abandoned for the training of the bounding box regression model.

R-CNN plays an important role in two-staged object detection, as it was the first model to adopt region-based convolutional neural networks for object detection in 2014. Many extension of R-CNN have been researched in the next couple of years, seeking to improve speed, accuracy and resource utilization, some of which will be briefly reviewed in the next sections including Fast R-CNN, Faster R-CNN, Mask R-CNN and Cascade R-CNN.

3.1.3.2 SPP-net [7]

One of the main drawbacks of R-CNN is that feature extraction using CNN is repeated 2000 times or so per image, although a lot of the region proposals overlap or are contained in each other, thus do not introduce new knowledge for the training and could be potentially reused and attributes to the model's slow runtime. Another inconvenience, that was briefly touched upon, is the necessity to warp the image segments into the required size before feeding them to the CNN, which distorts the original dimensions of the object and as a consequence decreases detection accuracy. SPP-net proposes a solution to both problems. [7] The fixed input size constraint comes from the fully connected layers in the CNN, as the other type of layers in AlexNet - the convolutional layers have no such limitation. This shortcoming can be overcome by employing a spatial pyramid pooling layer (SPP) instead of traditional pooling on the last convolutional layer just before the first fully connected one, which is exactly what the creators of SPP-net propose and where its name in fact comes from. The SPP layer can handle different sized inputs and will produce a fixed-length output.

To understand spatial pyramid pooling, first the role of pooling in convolution layers has to be examined. Convolution layers apply different filters on input images to produce feature maps, that describe the existence of the features in question (e.g. color, edges, simple shapes) in the image. (e.g. $\hat{\cdot}$ shape is the feature which activates filter#66 in Fig. 18 and filter#118 is activated by V shape.)

A pooling layers is traditionally adopted after each convolution layer to reduce the dimensions of the generated feature maps, which improves re-

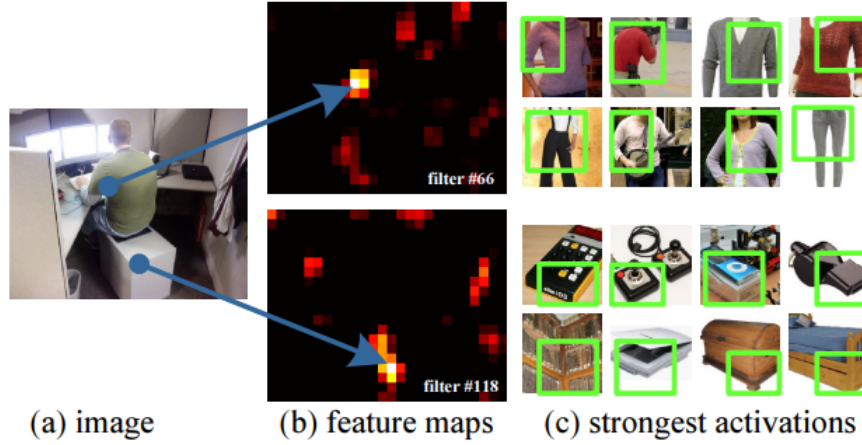


Figure 18: Visualization of feature maps [7]
(a) an image from Pascal VOC 2007 (b) the feature maps for filter#66 and filter#118 and (c) the receptive fields or image areas, having strong activations on that filters

source utilization and speed during training, and to deal with the intrinsic limitation of convolutional feature maps - the fact that they link the presence of a certain feature with their location in the image. Pooling works similar to the convolution operation - a filter of size e.g. 2×2 slides over the image with a given speed (stride=the number of pixels to shift forward each step) and produces a single output for every traversed image location (either the maximum value or the average over all pixels in the window). As a result the feature map, transformed by the pooling layer, is smaller but proportional to the initial feature map and naturally different sized inputs would still produce different sized feature maps.

Compared to traditional pooling, which uses a pooling window and stride of fixed size, spatial pyramid pooling layers adapt their sizes proportionally to the input image, in order to always generate an output with the same size. Given we have an input feature map of size 14×14 and we want to reduce that down to 4×4 using pooling, we can calculate both the window size and stride by dividing 14 by 4. The stride size is selected by rounding that number down and the window - by rounding it up to the next whole number. (stride = 3, window size = 4×4) Additionally, another key feature of SPP is that it applies pooling on the input multiple times and creates a number of feature maps

with different resolutions, typically reducing in dimensions, which visually resemble a pyramid and hence the name. The different-sized feature maps are then merged back together to form a fixed-length representation, that is then passed to the following fully connected convolutional layers. (Fig. 19)

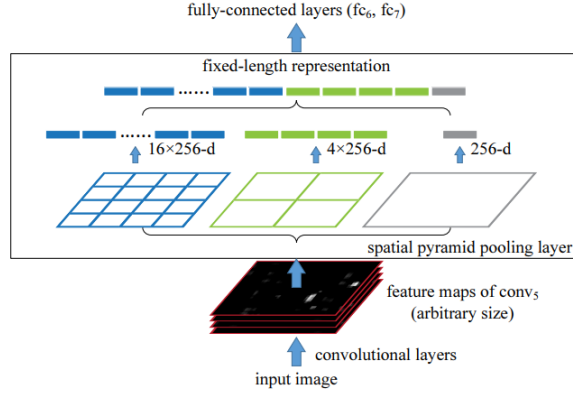


Figure 19: Spatial pyramid pooling layer [7]

The number of filters in the last convolutional layer = 256. The SPP layer produces three feature maps (4x4, 2x2 and 1x1) and then flattens and merges them to the required fixed-length representation

Back to SPP-net in the context of R-CNN and object detection, the authors of [7] have proposed a model, based on R-CNN, which replaces the convolutional neural network (AlexNet) with SPP-net, and applies it only once per image to create a feature map, representing the picture in its entirety, rather than applying the CNN once for every region of interest (RoI), exploding to around 2000 executions per image. Spatial pyramid pooling is then applied on the region candidates' feature maps, which are derived from the big picture feature map rather than created separately as in R-CNN, to ensure the fixed-length output, expected by the following fully connected layers. (Fig. 21) The SVM for object classification and the bounding box regression models are kept unchanged. This alteration has improved training speed 3 times and prediction speed by a factor of 102, while retaining similar or better accuracy than R-CNN. (Fig. 20)

3.1.3.3 Fast R-CNN [8]

The main shortcoming of R-CNN is its resource and time inefficiency due

	SPP (1-sc) (ZF-5)	SPP (5-sc) (ZF-5)	R-CNN (ZF-5)
ftfc ₇	54.5	<u>55.2</u>	55.1
ftfc ₇ bb	58.0	59.2	59.2
conv time (GPU)	0.053s	0.293s	14.37s
fc time (GPU)	0.089s	0.089s	0.089s
total time (GPU)	0.142s	0.382s	14.46s
speedup (vs. RCNN)	102×	38×	-

Figure 20: Detection results (mAP) on Pascal VOC 2007 [7]

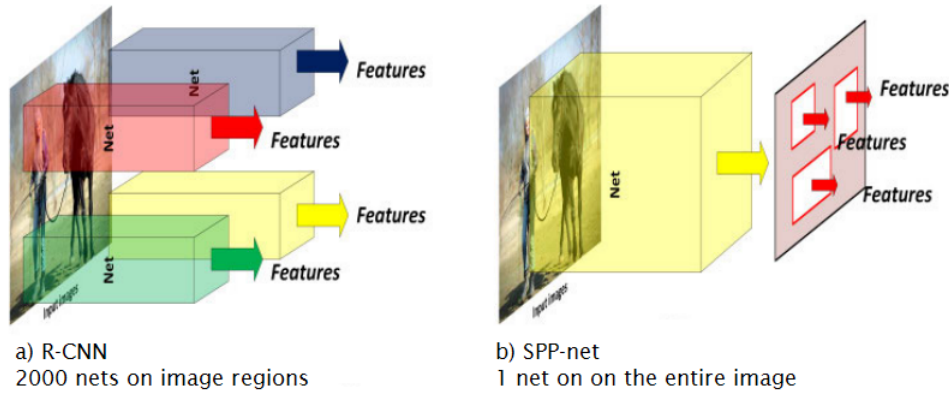


Figure 21: Comparison between R-CNN and SPP-net framework [7]

to (a) the model consisting of a multi-phase pipeline, which can not be parallelized, as the output from one component is required as input for the training of the next and (b) the suboptimal setup with regard to feature extraction and the fact that a CNN forward pass is performed around 2000 times per image for all region proposals separately, when in reality there are a lot of redundant or overlapping regions, where feature computation could be shared. Ross Girshick, one of the creators of R-CNN, released an improved version less than an year later (in 2015) called Fast R-CNN. Fast R-CNN is 9 times faster than R-CNN with training time of 9.5 hours in comparison to R-CNN's 84 hours and 146 times faster at test time, which means that Fast R-CNN needs only 0.3 seconds (excluding region proposal) for object detection, while also accomplishing a greater accuracy on the PASCAL VOC 2007 dataset (66.9% mAP) compared to R-CNN (58.5% mAP). (Fig. 22)

Fast R-CNN addresses the speed problem similar to SPP-net by (1) performing feature extraction only once per image for its entirety and then using

	Fast R-CNN			R-CNN			SPPnet
	S	M	L	S	M	L	[†] L
train time (h)	1.2	2.0	9.5	22	28	84	25
train speedup	18.3×	14.0×	8.8×	1×	1×	1×	3.4×
test rate (s/im)	0.10	0.15	0.32	9.8	12.1	47.0	2.3
▷ with SVD	0.06	0.08	0.22	-	-	-	-
test speedup	98×	80×	146×	1×	1×	1×	20×
▷ with SVD	169×	150×	213×	-	-	-	-
VOC07 mAP	57.1	59.2	66.9	58.5	60.2	66.0	63.1
▷ with SVD	56.5	58.7	66.6	-	-	-	-

Figure 22: Comparison R-CNN, SPP-net, Fast R-CNN on VOC07 [8]

S - AlexNet (used in R-CNN)

M - VGGM-1024 - VGG model with the same depth as AlexNet, but wider

L - VGG16 - VGG model with 16 layers (twice as deep as AlexNet) [39]

the already computed feature map to construct the image regions' feature maps and (2) by introducing a RoI (region of interest) pooling layer, which works analogically to the spatial pyramid pooling (SPP) layer in SPP-net with the only difference, that a RoI pooling layer generates one single feature map instead of multiple reducing in size in the case of SPP. The RoI pooling layer, as seen in SPP-net, replaces the last max pooling layer on the last convolutional layer, preceding the two fully connected (FC) layers, to ensure the fixed sized input, required by the FC layers. In contrast to SPP-net, which has kept the SVM for object classification and the bounding box regression model for coordinates fine-tuning as designed originally in R-CNN, Fast R-CNN sees another improvement candidate and replaces both models with two sibling fully connected convolutional layers on top of the existing architecture. The two new FC layers end with a softmax layer and a linear regression layer respectively and output a class probability vector (with length $C + 1$, where C is the number of classes and one value is reserved for the no-object scenario or a background segment) and the fine-tuned bounding box coordinates. (Fig. 23)

Fast R-CNN combines all stages of object detection as seen in R-CNN (feature extraction, object classification and bounding box regression) into a single convolutional network and leaves only the region proposal as a stand-alone task, performed using selective search in advance. The main advantage of unifying all models into one network, is that the feature maps do not need to be stored on the disk in between tasks, whereas R-CNN had to cache them

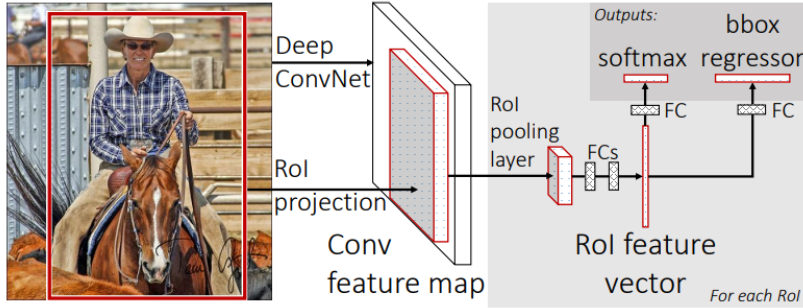


Figure 23: Fast R-CNN model architecture [8]

to later feed them to the SVM model for classification, which spares hundreds of gigabytes of storage space. Another benefit of multi-task training as compared to stage-wise training in R-CNN is the increase in accuracy by around 1% mAP at pure classification. Fast R-CNN achieves that improvement by adopting a multi-task loss function with a robust L1 loss for the regression (instead of the L2 loss used in R-CNN), that takes advantage of the fact that tasks influence each other through a shared representation in the CNN

Training the weights of the layers below the SPP (spatial pyramid pooling) layer in the normal setup is so inefficient, that SPP-net just does not do it and instead only fine-tunes the last two fully connected layers, which seemed sufficient for the shallower CNN (AlexNet with 8 layers) used. [7] The creator of Fast R-CNN explored ways to overcome that problem, as Fast R-CNN uses a deeper CNN and namely VGG16 with 16 layers, and he strongly believed, that not training these 13 layers (16 - 2 fully connected - 1 softmax) would greatly impair the model accuracy.

The inefficiency of backpropagation through the SPP layer comes from the fact, that for each RoI (region of interest) a forward pass has to be performed on a feature map, encoding the entire image, given regions come from a different picture each time. Fast R-CNN proposed an approach called hierarchical sampling, which consists in first sampling a set of images and then choosing a fixed number of regions, belonging to them, to process in one batch for the SGD (stochastic gradient descent) in backpropagation. RoIs from the same image can share computation and memory during both forward pass and backpropagation, which makes training 64 times faster (for a batch size of 64 regions per 1 image) than the case of processing regions in no specific order, with no regard to the image they stem from, as seen

in R-CNN and SPP-net. This speedup made learning (updating the weights through backpropagation) on all network layers manageable and has led to an increase in accuracy from 61.4% to 66.9% mAP, in comparison to only fine-tuning the fully connected layers, as observed in SPP-net. (Fig. 25) Due to the fact that training is very time consuming, Girshick has done some experiments and shown, that updating the weights on the layers up to conv3_1 has no significant effect (improves accuracy minimally at a high cost (training time rises 7 times)), so he decided to freeze the first four layers during training and only learn through backpropagation on the remaining 9 convolutional layers. [8]

	layers that are fine-tuned in model L			SPPnet L
	$\geq \text{fc6}$	$\geq \text{conv3}_1$	$\geq \text{conv2}_1$	$\geq \text{fc6}$
VOC07 mAP	61.4	66.9	67.2	63.1
test rate (s/im)	0.32	0.32	0.32	2.3

Figure 24: Effect of restricting which layers are fine-tuned for VGG16 [8]

The final aspect about Fast R-CNN worth mentioning is that it uses truncated SVD (singular value decomposition) to perform dimensionality reduction on the last two fully connected layers, as their computation attributes for almost 45% of the detection time. By applying truncated SVD on the last FC layers, the runtime was reduced by 30% at the cost of a tiny decline in accuracy (from 66.9% to 66.6% mAP).

To summarize, Fast R-CNN is an improved version of R-CNN, which has fundamentally accelerated both training and prediction time and minimized space utilization, while also achieving an increase in accuracy by:

1. unifying the different training stages (feature extraction, classification and bounding box regression) into a single CNN, so features do not need to be cached on disk anymore
2. adopting a multi-task loss function, enabled by (1)
3. introducing an RoI pooling layer, so that images do not have to be warped anymore to ensure a fixed size (warping distorted objects' natural aspect ratios and thus had a bad impact on accuracy)
4. extracting a feature map only once for an entire image and deriving the RoIs' feature maps from it, which saves a lot of time and computational resources

-
5. learning through backpropagation on all layers, even the ones below the SPP layer
 6. applying truncated SVD on the last two fully connected layers to reduce their dimensions, which led to a 30% runtime speedup

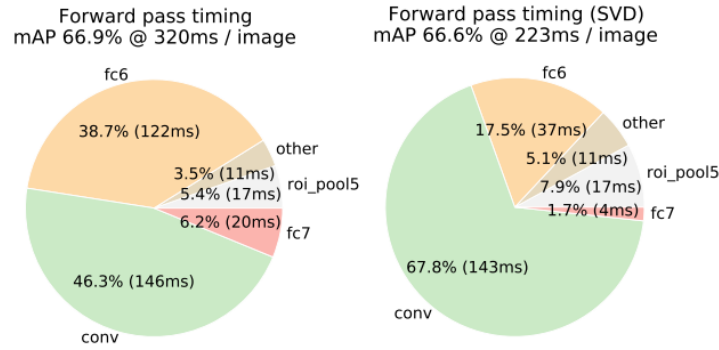


Figure 25: The effect of truncated SVD on Fast R-CNN detection time [8]

3.1.3.4 Faster R-CNN [9]

Fast R-CNN proposed a lot of improvements to R-CNN, but has left a considerable bottleneck in the face of the selective search algorithm, used for the region proposal, which is relatively slow, and was thus the main improvement target for the next iteration of R-CNN - Faster R-CNN, which was released later the same year as Fast R-CNN (2015). [9] Fast R-CNN has already achieved an almost runtime speed at test time, but only if the region proposal time is taken out of the equation [8], which is the exact reason for slow selective search to be replaced with a deep fully convolutional neural network (FCN) called RPN (region proposal network) in Faster R-CNN, which ultimately shares convolutional computation at test time with the object detection network, and thus needs almost no extra time for the region proposal (10ms per image in comparison to 2 seconds for selective search). As Fast R-CNN is already a very good solution for object classification and bounding box regression with outstanding accuracy and speed, which has adopted noble advancements to R-CNN like enabling the training of all layers and sharing the convolutional computation of features between regions, coming

from the same image, for both forward pass and backpropagation, the creators of Faster R-CNN have decided to keep it unchanged to the original version. With that being said, Faster R-CNN has finally managed to unify object localization and classification into one single unified neural network for object detection by removing the only stand-alone component in Fast R-CNN - the algorithm for region proposal - and taking over the task internally to the CNN. (Fig. 26)

Inspired by selective search and the fact that it performs region proposal by grouping pixels based on primitive features like color or intensity, the creators of Faster R-CNN decided to explore the idea of exploiting the already computed entire-image feature maps, generated as part of the classification task in Fast R-CNN, instead of running a separate external algorithm. [9] That encouraged the adoption of RPN (region proposal network) - a fully convolutional network for RoI (region of interest) generation, which shares convolutional layers with the object classification network (Fast R-CNN), both of which jointly form a single model, that carries out end to end object localization and detection. (Fig. 26)

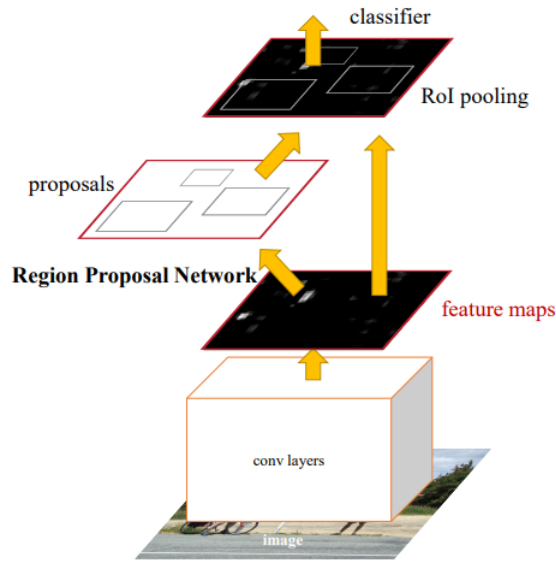


Figure 26: Faster R-CNN model architecture [9]

RPN generates regions of interest (RoI) by sliding a 3x3 window (filter) in a convolutional fashion over the full-image feature map, created by the

backbone CNN, and evaluating multi-scale multi-ratio anchor boxes, all centered in the middle of the filter. The evaluation consists in RPN employing two sibling 1x1 convolution layers for binary classification and bounding box regression for each location in the feature map. The output is respectively a vector holding 2 values (the objectness scores for foreground (the region contains an object) and background (no object)) for the binary classifier and a vector holding 4 values - the offset coordinates between the fine-tuned region and the tested anchor box, for the bounding box regressor. (Fig. 28 (a))

To ensure that variously sized and shaped objects are considered for the region proposal, RPN uses 9 different anchor boxes: 3 common aspect ratios (a wide rectangle - 2:1, a tall rectangle - 1:2 and a square - 1:1) in 3 scales: 128x128, 256x256 and 512x512 pixels, (Fig. 28 (b)) and applies them on input images of maximal size 1000x600 pixels. (The images are resized to have shorter side = 600px or a longer side = 1000px and shorter side of less than 600 pixels) Alternative approaches to using a single-scale image, single-size feature filter (sliding window) and multiple anchor boxes, having different aspect ratios and scales, as seen in Faster R-CNN, include the usage of multiple scales of the same input image, referred to as image pyramids, or using feature filters of different size, also known as feature pyramids, to cover both small and larger-sized objects, as shown in Fig. 27. Both mentioned approaches are slower and computationally more expensive.

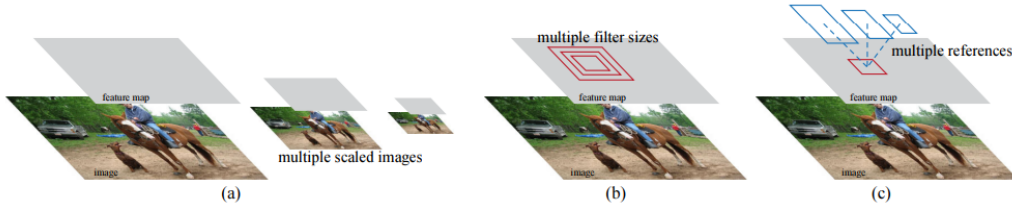


Figure 27: Different strategies for covering multiple scales and sizes [9]
(a) image pyramid - different scaled images, (b) feature pyramid - different sized filters, (c) anchor box pyramid - multiple scaled anchor boxes

Given a stride of 2x2 (the number of rows and columns traversed per filter slide), a biggest input image of 1000x600 pixels, VGG as backbone network for the feature extraction (as seen in R-CNN), the resulting feature map for the entire image will have a size of $1000/16 = 62$ ($2^4 = 16$, as VGG has 4 max pooling layers, each of which reduces the feature map dimensions twice for stride = 2) by 37, so around 60x40 or about 2400 locations to

evaluate for the region proposal. RPN will go through all 2400 grid cells and make 9 proposals for every reference box (anchor box), each of which consists of 6 values - 2 for the binary classification (the objectness scores for the cases of any given object contained in the region (foreground class) versus no object or a background class) and 4 values for the offset between the fine-tuned box and the reference box coordinates. (Fig. 28 (a)) This results in maximum $2400 \times 9 = 21600$ region proposals, out of which the ones containing no object or the negatives (anchor boxes with intersection over union (IoU) of less than 0.3 with any ground truth) and the cross-boundary boxes are eliminated. The remaining 6000 or so RoIs (region of interest) per image are additionally reduced to 2000, by filtering out redundant boxes, covering one and the same object instance, using non-maximum suppression (NMS), coming to approximately the same amount of RoIs, that selective search produced.

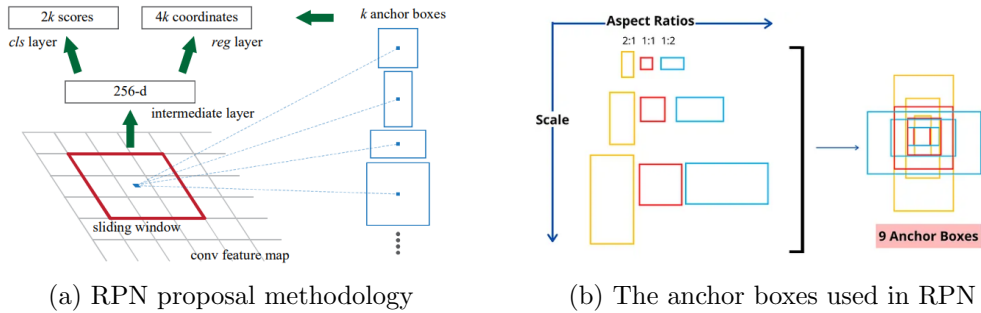


Figure 28: Region proposal network (RPN) [9]

One important aspect about the adoption of a region proposal network (RPN), which uses anchor boxes and makes relative to them proposals, is the fact that predictions are translation invariant, meaning that if an object is detected in one location and is moved somewhere else, the network will still be able to recognize it and make the same prediction regardless, which is not a given for all region proposal algorithms. (e.g. the MultiBox method is not translation invariant. [53])

Both components of Faster R-CNN - RPN for region proposal and Fast R-CNN for object classification and bounding box regression, can be trained separately and used as standalone networks in a two-stage object detection model. Nevertheless, the creators of Faster R-CNN saw the opportunity to share the convolutional layers (13 in VGG16) between both models and

formed a single unified, deep-learning based object detection system [9] with a higher accuracy as compared to Fast R-CNN, rooting in the fact that the convolutional layers are trained jointly, receiving valuable insights from both networks, and 10 times lower recognition time, caused by region proposal being the major slowdown factor with runtime of 2 seconds per image in Fast R-CNN, whereas in the new setup it comes almost free of cost - at 10ms per image.

Different strategies for training the convolutional layers, that RPN and Fast R-CNN share, have been explored and following 4-step method called alternating training, has shown to be the most favourable:

1. initialize the convolutional layers with weights, pretrained on ImageNet for general image classification, and train RPN
2. use the region proposals, generated in step 1, as input for the Fast R-CNN and once again initialize the convolutional layers with the same weights, pretrained on ImageNet for general image classification, and then fine tune all layers
3. train RPN again, but this time using the updated convolutional layers from step 2 and fine tune only the layers unique to RPN (keep the shared convolutional layers fixed)
4. train Fast R-CNN again and fine tune only the layers unique to Fast R-CNN (keep the shared convolutional layers fixed)

To summarize, Faster R-CNN is faster and more accurate than Fast R-CNN, mainly due to the fact that the relatively slow external algorithm for region proposal - selective search, has been replaced with RPN (a fully convolutional deep neural network). Speaking in numbers, Faster R-CNN (RPN+VGG with shared convolutional layers) has achieved accuracy of 69.9% mAP as compared to Fast R-CNN, having 66.9%, on the PASCAL VOC 07 dataset. When it come to the runtime, Faster R-CNN takes only 198ms per image for the recognition as opposed to Fast R-CNN needing almost 10 times more time or 1830ms per image, all coming from the region proposal.

3.1.3.5 Mask R-CNN [10]

Mask R-CNN is a shiny version of Faster R-CNN, which performs object

method	# proposals	data	mAP (%)
SS	2000	07	66.9 [†]
SS	2000	07+12	70.0
RPN+VGG, unshared	300	07	68.5
RPN+VGG, shared	300	07	69.9
RPN+VGG, shared	300	07+12	73.2
RPN+VGG, shared	300	COCO+07+12	78.8

Figure 29: Accuracy results on PASCAL VOC 2007 [9]

Fast R-CNN is the model, used for object classification and bounding box regression for all rows in the table, applied on top of the compared region proposal algorithms

model	system	conv	proposal	region-wise	total	rate
VGG	SS + Fast R-CNN	146	1510	174	1830	0.5 fps
VGG	RPN + Fast R-CNN	141	10	47	198	5 fps
ZF	RPN + Fast R-CNN	31	3	25	59	17 fps

Figure 30: Runtime comparison - Fast R-CNN and Faster R-CNN [9]

localization and detection and outputs additionally to the class assignment and bounding boxes coordinates, a pixel-precise segmentation mask for all object instances on the picture, solving a more advanced task called image segmentation. (Fig. 31) Image segmentation consists in splitting a visual into multiple sections (sets of pixels), based on their content. The subject can be further divided into semantic segmentation and instance segmentation. Semantic segmentation is used for classifying image parts into different categories and instance segmentation additionally differentiates between the different occurrences of the objects from the same category. A classic example of image segmentation can be seen on Fig. 32, where the the visual is separated based on two categories - person or background and on the right side the characteristic for instance segmentation distinction between the different instances of the same class (person) can be seen.

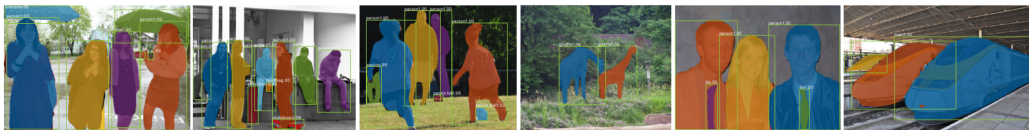


Figure 31: Mask R-CNN example outputs - a segmentation mask, bounding box and a class label for each object instance [10]

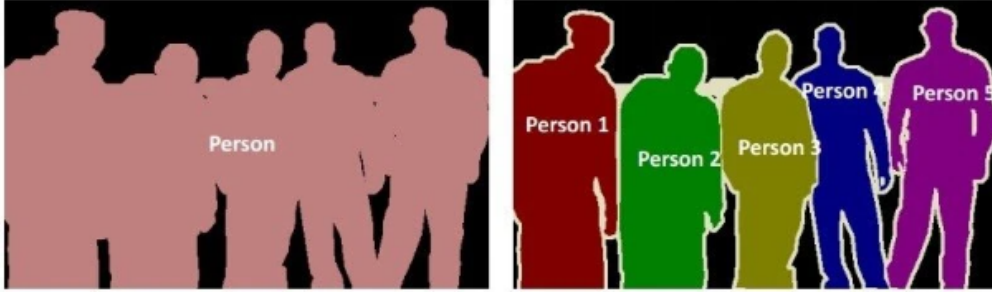


Figure 32: Image segmentation [10]

To extend the existing model for the segmentation mask generation, the creators of Mask R-CNN have added yet another fully convolutional neural network (FCN) on top of the existing Faster R-CNN infrastructure, this time parallel to the last two sibling fully connected layers, that are responsible for the final object classification and bounding box regression, so the twin branches in Faster R-CNN have become triplets in Mask R-CNN.

The main difference between Mask R-CNN and Faster R-CNN is the adoption of a RoI Align layer as a replacement to the RoI pooling Layer, first introduced in Fast R-CNN. The RoI Align layer is an enhanced version of the RoI pooling layer, which instead of rounding down when calculating the position of the region proposal on the feature map, uses bi-linear interpolation to get a more precise representation of the RoI. A mask is in fact a crisp layout of an object, so while losing information in the order couple of pixels by quantization (rounding down) does not affect object detection much, it is suboptimal for a rigorous task like semantic instance segmentation.

Given an image of size 512x512 pixels, the backbone - VGG-16, would create a 16x16 feature map, so the scaling factor is $512/16 = 32$. Each grid cell in the feature map corresponds to a 32x32 pixels region in the original image. Imagine the RPN has made a proposal of size 200x145 with top left corner with coordinates (296,192) for the grey-white cat in the picture (Fig. 34), so a RoI feature map of size $200/32 = 6.25 \times 145/32 = 4.53$ needs to be extracted from the entire-image feature map starting at coordinates $(296/32 = 9.25, 192/32 = 6)$. Here comes the difference between RoI Pool and RoI Align - RoI Pool would cut a feature map of size 6x4 with origin at (9,6), whereas RoI Align would not round down but rather slice the feature map precisely. The next thing, that happens is sampling down the resulting feature map into

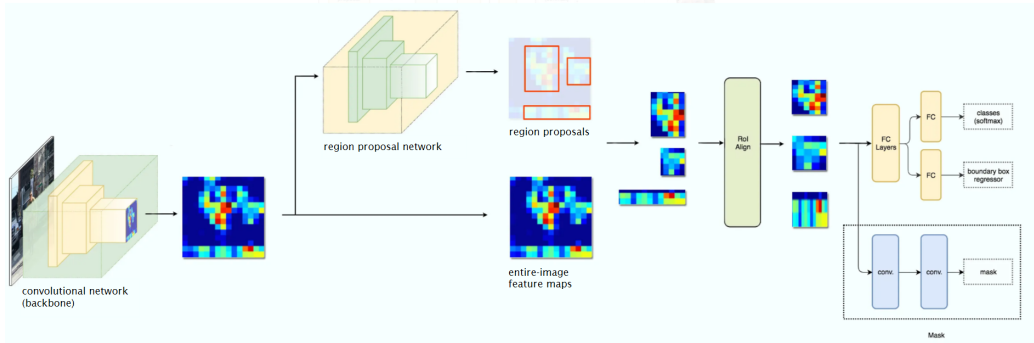


Figure 33: Mask R-CNN addition to Faster R-CNN [10]

a fixed size representation, required by the following fully connected layers. Given the RoI feature maps have to be reduced to a size 3×3 , the 6×4 feature map has to be divided into 9 bins (3×3) with size $6/3 = 2 \times 4/3 = 1.33$, which is again rounded down to bins of size 2×1 . This leads to again losing information about the last row of the feature map. (Fig. 35) Bounding box prediction is robust to such small displacements, so RoI Pool is a satisfactory mechanism for deriving a reduced in size fixed-length representation of a RoI from an entire-image feature map in the context of object detection, where pixel precision is not crucial, as objects rarely have a rectangular shape, so bounding boxes do not really closely outline the borders of objects anyway. For semantic image segmentation such misalignments are understandably not favorable, thus the adoption of RoI Align is of considerable importance.

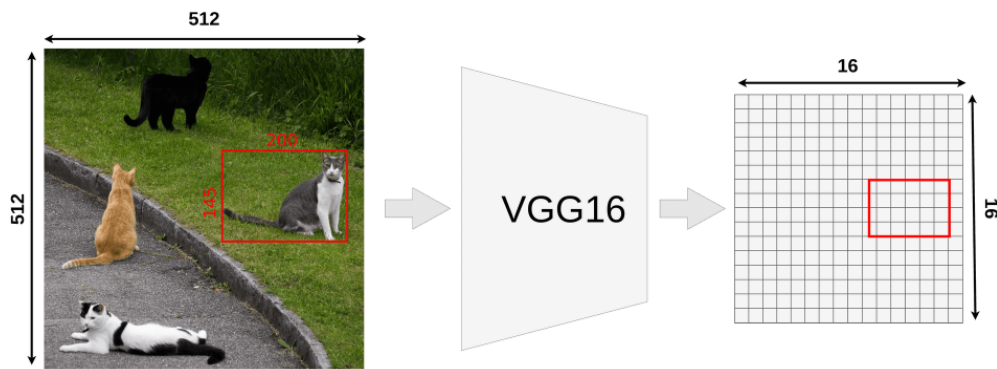


Figure 34: Feature map generation

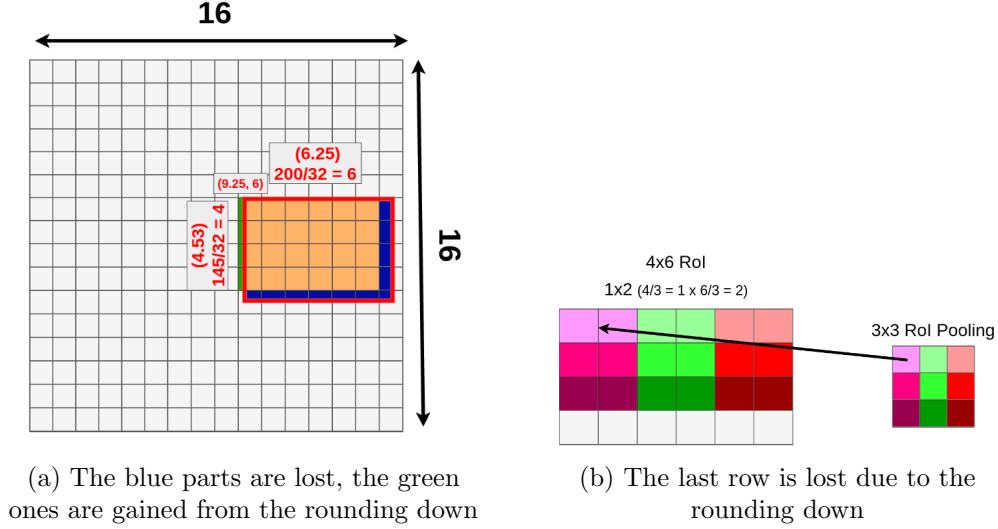


Figure 35: Quantization loss of RoI Pool

As a comparison RoI Align would extract the RoI feature map precisely without rounding down and would apply bi-linear interpolation to create the 3x3 fixed length output, necessary for the fully connected layers. In particular, when creating the 9 bins required for the 3x3 down-sampled version of the RoI feature map RoI Align would again choose the exact number including the decimal places instead of taking only the integer part of the bin size, so the region of interest, having size 4.53x6.25, would be divided in 9 bins of size $4.53/3=1.51 \times 6.25/3 = 2.08$. To perform the bi-linear interpolation four sampling points are created inside each bin, evenly distanced from each other and the bin borders. A value is assigned to each sampling point, which is the average of the values of the four nearest grid cells multiplied by the weighted distance to their centres, so the closer grid cell will affect the sampling point value most and the furthers - the least. (Fig. 36) Finally, the 4 sampling points per bin are aggregated using maximum or average pooling to create the resulting 3x3 feature map. The adoption of RoI Align layer, which eliminates the misalignments between the region proposals and the down-sampled fixed-sized feature maps, resulting from the quantization performed in RoI Pool layers, has a major effect (3 to 5 % rise in AP) on the model accuracy with regard to the segmentation mask creation. (Fig. 37)

In reality, choosing RoI Align over RoI Pool layer improves the accuracy

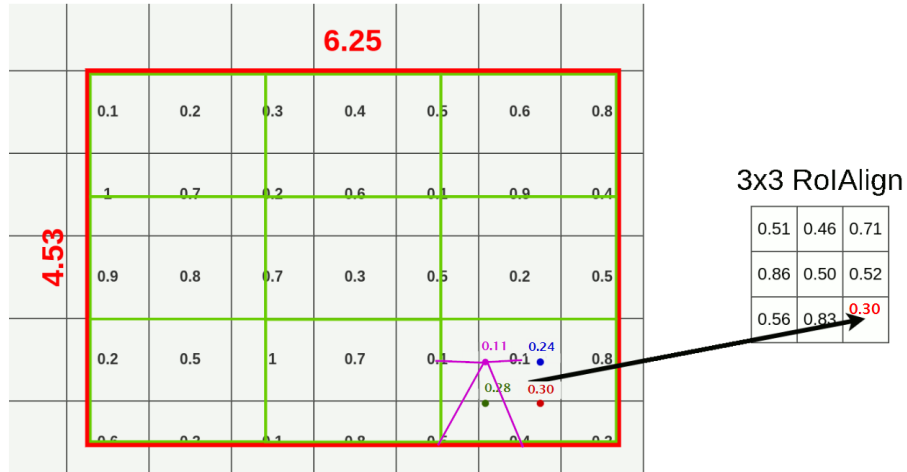


Figure 36: Bi-linear interpolation in RoI Align

of not only segmentation mask generation, but also the bounding box fixation in object detection, which is very intuitive, as RoI Align performs the same task as RoI Pool, but more precisely and coming with almost no additional cost, when it comes to runtime or computing resource consumption. Just swapping RoI Pool with RoI Align increases the accuracy of object detection by 1.1% AP in Faster R-CNN and adding up the mask segmentation part into the multitask training makes Mask R-CNN even more accurate with an overall advantage of 2% AP. (Fig. 38)

	align?	bilinear?	agg.	AP	AP ₅₀	AP ₇₅
<i>RoIPool</i> [12]			max	26.9	48.8	26.4
<i>RoIWarp</i> [10]		✓	max	27.2	49.2	27.1
		✓	ave	27.1	48.9	27.1
<i>RoIAlign</i>	✓	✓	max	30.2	51.0	31.8
	✓	✓	ave	30.3	51.2	31.5

Figure 37: The effect of choosing RoI Align versus RoI Pool layer on accuracy (AP on segmentation mask creation). The model was trained on COCO trainval35k, tested on minival [10]

Back to the model architecture of Mask R-CNN, other than switching from RoI Pool to a RoI Align layer, a fully connected neural network (FCN)

for the segmentation mask creation has been added parallel to the bounding box regressor and object classifier branches. The placement of the new FCN is very much intentional, as this allows for the instance segmentation task to be fully decoupled from the classification. The new network is simple - consists of two convolutional layers and outputs K two dimensional binary masks, one for each class (K = the number of classes), by applying per-pixel sigmoid. A binary mask is a down-sampled representation of an image, in which every grid cell, corresponding to e.g. a 16×16 pixels region in the image, is marked with either 1 or 0, depending on whether the grid cell is inside the borders of the object instance in question or not. Once the object classification in the neighbouring branch is finished, the according mask for the selected class is selected and output as result for the instance segmentation. Mask R-CNN has shown to be highly generalizable, which was proven by the creators via the task of human pose estimation on the COCO keypoint dataset. [10]

To summarize, Mask R-CNN is an extension to Faster R-CNN, which improves model accuracy (both bounding box AP and mask AP) by replacing RoI Pool with a more precise RoI Align layer, and additionally outputs a high-quality segmentation mask for each object instance, generated in simple fully convolutional neural network (FCN) parallel and thus fully decoupled from the bounding box regression and object classification branches.

	backbone	AP^{bb}	AP_{50}^{bb}	AP_{75}^{bb}	AP_S^{bb}	AP_M^{bb}	AP_L^{bb}
Faster R-CNN+++ [19]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [27]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [21]	Inception-ResNet-v2 [41]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [39]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	52.1
Faster R-CNN, RoIAlign	ResNet-101-FPN	37.3	59.6	40.3	19.8	40.2	48.8
Mask R-CNN	ResNet-101-FPN	38.2	60.3	41.7	20.1	41.1	50.2
Mask R-CNN	ResNeXt-101-FPN	39.8	62.3	43.4	22.1	43.2	51.2

Figure 38: The effect of choosing RoI Align versus RoI Pool layer on accuracy (AP on object detection) on test-dev [10]

3.2 Visual relation detection

Visual relation detection (VRD) is a computer vision task, which consists in identifying the interactions or connections between objects in an image. VRD is the natural addition to object detection, when it comes to the higher level challenge of scene understanding in visuals, which is crucial for building a bridge between the fields of computer vision and semantic perception. Visual relations can be described using a triplet structure, containing a pair of objects, more precisely - a subject, the doer of the action, and an object, the actor receiving the action, and a predicate or a relation, which corresponds to some action or a state, that connects the aforementioned objects, very similar to a simple sentence in natural language. (e.g. in the relation "girl" - "rides" - "bicycle", the subject is "girl", the object is "bicycle" and the predicate is "rides".)

Speaking of visual relation detection, we can differentiate between three tasks, depending on whether the aim is to predict only the predicate or the verb in the sentence, whole triplets phrases (subject, predicate, object), or the entire scene context in the picture and building a graph, which depicts the interconnections between all actors and relations. All three tasks rely on a predetermined set of objects, that are already assigned to a class with a certain confidence score, provided by an object detection model. Predicate prediction deals with identifying the predicate, given a set of objects, having already acknowledged, that a relation persists, and knowing which role each object takes in the relation (a subject or an object). Phrase detection is a broader and more resource intensive task, as it involves detecting and classifying whole phrases ("monkey-eats-banana" as one word instead of the triplet "monkey" - "eats" - "banana"), which results in an enormous number of triplet combination. Scene detection is the most complex task out of all three, as additionally to all triplets separately, the whole context of the image should be anticipated. This involves considering the interrelations between object instances as participants to potentially multiple different relations, that make sense together as a whole.

The topic of visual relation detection has gained a lot of interest in the last 7 years, as it brings us closer to comprehensive machine perception of images by transforming visual data into processable structured knowledge, that can be reasoned upon and used in conjunction with natural language processing to solve more advanced tasks like visual question answering, image captioning and image generation. Though it has shown to be easy for

humans, it is a challenging computer vision problem, which is still unsolved in the literature to a satisfactory extent. In the following we will go through different visual relation detection models, that take advantage of visual and spatial cues as well internal and external knowledge about the domain (internal knowledge - extracted from the test set, external knowledge - gathered from common sense knowledge datasets).

3.2.1 Visual phrases [11]

The authors of [11] were the first ones to introduce the term visual phrases in 2011 as a composite of two objects in a relationship, when it comes to visual relation detection. They strongly believed and showed, that visual phrases are easier to learn and achieve greater detection accuracy in comparison to the conventional approach of decomposing the problem into its constituents and first detecting the objects independently and then reasoning over the most suitable relation. They also argue that the appearance of objects in a specific composition may completely differ from the same objects, pictured standalone or in some other relation. For instance in the relation "person riding a horse", a person would always have a similar posture and one of his legs would not be visible, whereas the same object in the relation "person lying on the couch" would look completely different, which leads to the conclusion that visual composites are less difficult to recognize than their compounds. The one challenge phrase detection faces is the limited training data, as training sets for visual composites are usually much smaller than those for objects. The other possible issue comes from the fact that the number of phrases increases exponentially with the number of object classes. To address these matters the authors of [11] proposed the approach, illustrated in Fig. 39, which consists in taking advantage of both an object detection model and a visual phrase detection model and then applying a decoding mechanism to filter and prioritize the results of the detectors and provide the final prediction. The phrase detection method from [11] uses deformable parts models (DPM)[54] both for the object detection and phrase detection tasks. DPM models were the best-performing technique for object detection until the appearance of deep learning and the adoption of CNNs for visual computing tasks later in 2012. DPM learn on low-level features based on histograms of oriented gradients (HOG) in comparison to CNNs extracting convolutional feature maps. The other main difference between DPM and CNN is that DPM is a graphical model of a joint probability distri-

bution (a Markov random fields optimisation), so a standard mathematical model, whereas CNNs are "black-box" non-linear classifiers. [55] Back to phrase detection, in [11] the DPM models are trained for each phrase and object class separately and have a proper accuracy for phrases and objects with low variance in the appearance. Multi-class detectors generally require some type of decoding to eliminate disambiguity on which predictions are relevant, so decoders mainly amplify some predictions and suppress others based on the local context. The proposed in [11] novel decoding procedure accounts precisely for surrounding factors without solving difficult inference problems by considering specific properties of interacting objects in various abstraction levels. Simplified, the introduced decoding mechanism does not rely on complex computations and inference strategies but rather expects for the extracted object features to account for nearby objects and thus is not necessary to build pairwise terms in the scoring function and solve complex combinatorial problems.

Overall addressing the problem of visual relation detection by training only on the feasible combinations between a pair of objects and a predicate has shown to have advantages in some cases over the approach of first performing object detection and then reasoning over the most appropriate relation. Recognition using visual phrases has proven to be more beneficial, when the number of objects and resulting composites is manageable, when there is at least some training data, containing the triplets and last but not least, when the variance in the appearance of objects in the selected relations is relatively small.

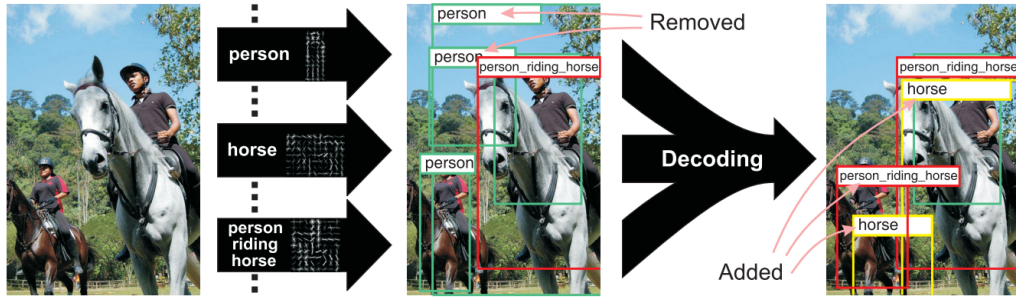


Figure 39: Recognition using visual phrases [11]

3.2.2 Visual relationship detection (VRD) with language priors [12]

Visual relationship detection (VRD) consists in localizing and recognizing pairs of objects in an image and determining the connection or predicate between them. Although intuitively similar to the task of object detection, VRD is more challenging due to the semantic space of viable relations being a lot bigger than that of objects. [12] Additionally, a long-tailed distribution of relations is commonly observed in VRD, meaning that most relations only have a few examples in the training set, which is generally expected considering the fact that relations require two specific objects to co-exist in an image, leading to a greater imbalance of rare relations and a scarce set of examples for VRD to learn from.

The visual relationship detection model, proposed in [12], uses R-CNN [6] for object detection and then trains a visual module, that focuses on learning the appearance of objects and predicates on their own and combines the insights to predict relationships, and a language model, which accounts for semantic similarities between relations in natural language and can infer unseen ones based on frequently occurring combinations. (Fig. 40)

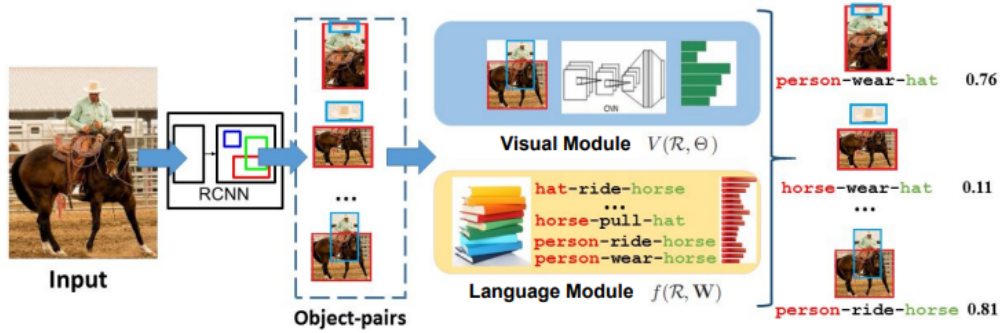


Figure 40: Visual relationship detection pipeline as of [12]

In a bit more detail the visual appearance module consists of two R-CNN models - the first one, used for object detection, is trained on the ground truth bounding boxes and outputs a number of bounding box and a confidence score vectors of size N (N = the number of object classes) for each input image and the second one, used for predicate detection, is trained on a joint vector, formed by appending the bounding boxes all possible pairs of

objects (potential participants in a relation), and outputs a vector, holding the likelihoods for each predicate to be connecting the objects in question, for each object pair. The outputs of the two R-CNN networks are fused together using multiplication (the confidence scores of the two objects, provided by the object detection, multiplied together with the likelihood for the relation), so the impact of each component is proportional to the final relationship prediction.

The language module is inspired by the authors’ impression, that some relationships are semantically similar to one another. (e.g. person-ride-horse and person-ride-camel are seemingly alike). To represent this correlation, the language module is trained to project relations into an embedding space, where semantically related relation are closely located to each other in the vector space. This is achieved by defining the relationship projection function as the concatenated projections of the participant objects multiplied by a parameter, which is learnt during training, plus a bias term. The projection function is optimized using a heuristic, prescribing that the distance between two relationships should be proportional to the vector distances between their components (the participant objects and the predicate). [12] Additionally, the language module utilizes a rank loss function to boost relationships, that occur frequently in the training set and suppress the likelihood score of implausible ones.

Back to the big picture, the predictions of the visual module and the language module are combined using a ranking loss function in order to choose the best suitable relationship, as a classification function does not perform well due to the large number of possible relationships. The resulting model is evaluated using three different scenarios: (1) predicate detection - the object classes and bounding boxes are given, only the predicate needs to be assigned, (2) phrase detection - a relationship as a whole (a composite, consisting of a pair of objects and a predicate) and bounding box has to be identified on the image, and (3) relationship detection - the objects and predicate have to be recognized separately, a bounding box should be allocated for each object. (Fig. 41)

The metric used for evaluation is recall @ 100 and recall @ 50, which account for how well the model recognizes the correct relationships in the top 100 and top 50 predictions respectively. Using mAP for testing the precision of visual relation detection models is not suitable, as it would require to exhaustively annotate all possible relations, else correct predictions, not found in the ground truth, would be penalized unnecessarily. The model

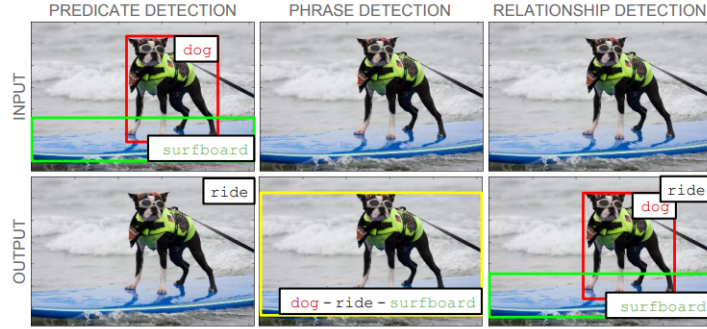


Figure 41: Evaluation scenarios for the model proposed in in [12]

proposed in [12] performed well on the predicate detection task, achieving 47.87 recall @ 100 and about 30% worse on the relationship and the phrase detection tasks, which comes from the challenge of localizing correctly two objects simultaneously, increasing the impact of the object detection errors on the overall prediction. Previously introduced models like Visual Phrases [11] and Joint CNN [56] perform poorly on all three tasks, due to the fact that they train an individual detector for every possible relationship and do not address infrequent ones at all. (Fig. 42) The authors of [12] additionally insisted that visual relation detection models should be capable of performing zero-shot predictions, which include recognizing never seen before relations, as relationships on real life images naturally follow a long tail distribution, and therefore some relationships would not exist in the training dataset, but could be found in the test dataset. Previous work is not able to solve this problem at all and the model, introduced in [12] can infer some unseen relationships based on similar ones and as a result scores 8.45 recall @ 100 for predicate detection and again twice as bad for phrase and relation detection due to object detection error propagation. [12]

The other important contribution of [12] is the introduction of the benchmark visual relationship dataset (VRD), which contains 5000 images, depicting 100 object categories and 70 different predicates. The presented dataset follows a log scale distribution of the number of instances to the number of relationships (what we previously referred to as long-tailed distribution) and covers a high number of links between pairs of objects, which can be divided into 5 subcategories: (1) action (e.g. drive, ride), (2) spatial (e.g. on top of, next to), (3) preposition (e.g. with), (4) comparative (e.g. taller than) and

	Phrase Det.		Relationship Det.		Predicate Det.	
	R@100	R@50	R@100	R@50	R@100	R@50
Visual Phrases [6]	0.07	0.04	-	-	1.91	0.97
Joint CNN [44]	0.09	0.07	0.09	0.07	2.03	1.47
Ours - V only	2.61	2.24	1.85	1.58	7.11	7.11
Ours - L only	0.08	0.08	0.08	0.08	18.22	18.22
Ours - V + naive FC	6.39	6.65	5.47	5.27	28.87	28.87
Ours - V + L only	8.59	9.13	9.18	9.04	35.20	35.20
Ours - V + L + Reg.	8.91	9.60	9.63	9.71	36.31	36.31
Ours - V + L + K	17.03	16.17	14.70	13.86	47.87	47.87

Figure 42: Evaluation results for visual relationship detection [12]

(5) non-action verb (e.g. wear, has). (Fig. 44) Opposed to previously introduced datasets [11], which focus on object detection or image retrieval, the VRD dataset covers a wide variety of different relationships, a high number of which have very few occurrences, but in the same time in order to be used for benchmarking visual relation detection and to dissociate from the task of object detection, the predicates per object category are kept relatively high. Additionally, the test set includes 1878 relationships, which do not occur in the training set, allowing for zero-shot learning to be evaluated.

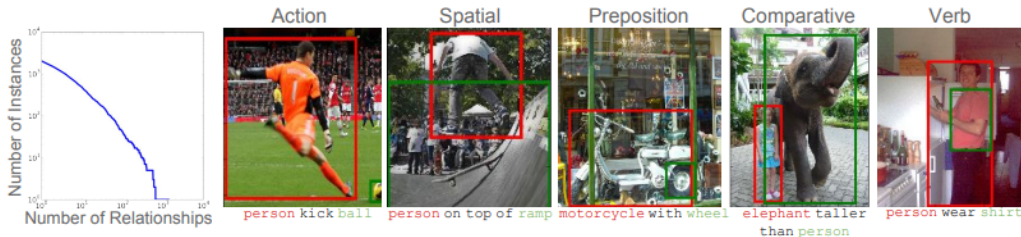


Figure 43: Visual relationship dataset, proposed in [12]

3.2.3 VRD with deep relational networks [13]

In computer vision, recognizing individual objects in an image is often not sufficient to fully understand its contents. Relationships between objects are also important, and can provide majorly significant information for more advanced computer vision tasks such as image captioning. While remarkable progress has been made in object recognition, scene classification, and attribute detection through the use of deep learning techniques, visual relationship detection remains an open problem in the literature. One way

to address visual relationship detection is to handle it like a classification problem, with each combination of objects and relationship predicates, also referred to as visual phrases, considered as a separate class. However, this approach is impractical due to the exponentially growing number of classes (phrases) and their imbalanced nature, consisting in some phrases having very few to no samples, whereas others are abundantly more represented in the training data. An alternative approach is to consider each relationship predicate as a separate class, which is also suboptimal, as there would be high diversity within each class, making it difficult to handle even with the computing capacity of deep networks.

The proposed in [13] framework formulates the output prediction as a triplet in the form of subject, predicate, and object, and jointly infers their class labels by exploiting two types of relations among them - spatial configuration and statistical dependency. These relations are not only ubiquitous and informative but are also often more reliable than visual appearance. Unlike previous relational models such as conditional random fields (CRFs) [57], the proposed model is formulated using a Deep Relational Network (DR-Net), which embeds statistical inference into a deep neural network through iteration unrolling. This new formulation extends the expressive power of deep neural networks to relational modeling and allows the model parameters to be learned in a discriminative fashion using the latest deep learning techniques. The proposed framework outperforms not only classification-based methods but also CRFs based on deep potentials on two large datasets. The major contributions of [13] are DR-Net, a novel formulation that combines the strengths of statistical models and deep learning, and an effective framework for visual relationship detection, which improves the state-of-the-art in this area.

In comparison to other approaches such as Visual Relationship Detection (VRD) [12] and Visual Phrases [11], DR-net offers several advantages. VRD focuses on detecting object-object relationships and typically uses hand-crafted features and rule-based methods to infer relationships. DR-net, on the other hand, leverages deep learning techniques to automatically learn and represent relationships from data, enabling more flexible and data-driven modeling.

Visual Phrases, on the other hand, primarily focuses on detecting relationships described by short phrases, typically involving two objects and a preposition (e.g., "man on horse"). While Visual Phrases provide a compact representation of relationships, they often rely on explicit phrase annotations,

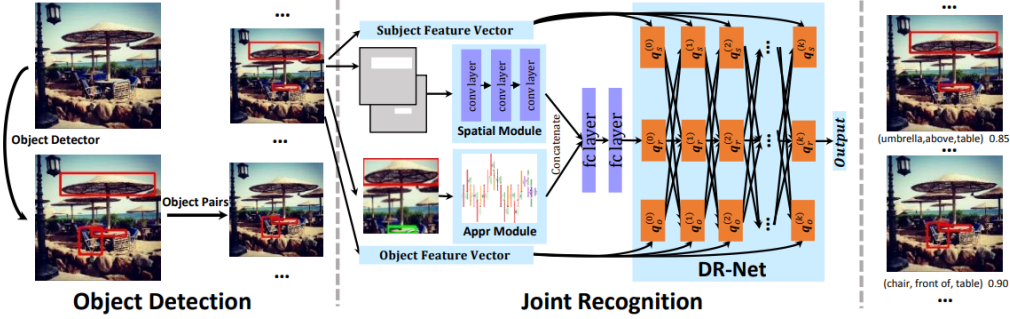


Figure 44: Visual relationship dataset, proposed in [13]

which can be costly and limited in coverage. In contrast, DR-net aims to learn relationships in an end-to-end manner from raw visual data, without relying on explicit linguistic annotations.

DR-net combines the advantages of deep learning with relational modeling, allowing it to capture both low-level visual features and high-level semantic relationships between objects. By considering the spatial and semantic dependencies between objects, DR-net can achieve more accurate and detailed relationship detection, enabling richer and more comprehensive understanding of visual scenes.

Overall, DR-net provides a powerful framework for visual relationship detection by leveraging deep learning techniques to capture complex relationships between objects in an end-to-end manner. It offers a data-driven and flexible approach compared to rule-based methods like VRD, and it extends beyond the limitations of explicit phrase annotations in Visual Phrases, enabling more comprehensive and context-aware relationship understanding in visual scenes.

3.2.4 VRD with with deep structural ranking [14]

Visual Relationship Detection, is a computer vision task that aims to detect and classify relationships between objects in images or visual scenes. It involves identifying the objects present in an image and determining the relationships or interactions between them.

Deep Structural Ranking (DSR) [14] is an approach or methodology that can be applied within the context of VRD. It leverages deep learning techniques and employs a ranking-based framework to address the challenge of

detecting and ranking relationships among objects in a visual scene.

In VRD with DSR, a deep neural network model is trained to learn representations of objects and their relationships from labeled training data. The network takes an input image and outputs a set of candidate object pairs along with their predicted relationship scores. These scores indicate the likelihood or confidence of a particular relationship existing between the objects.

The proposed method focuses on separately modeling the object and predicate aspects and then combines their outcomes to infer the relationship between them. To handle the inherent diversity of each predicate, the authors of [14] propose the integration of multiple cues as input features. Specifically, the network leverages visual appearance cues, spatial location cues, and semantic embedding cues, which are fused into a unified feature vector. (Fig. 45)

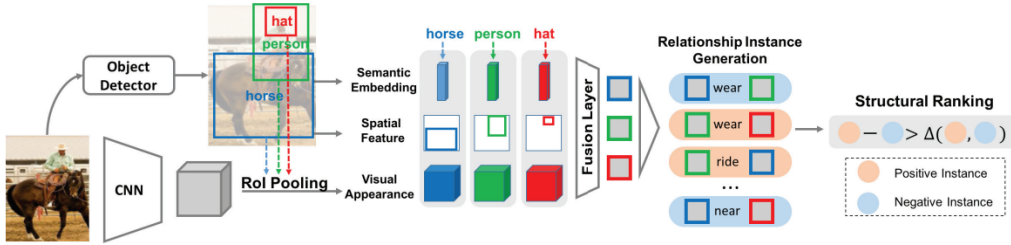


Figure 45: Visual relationship detection with deep structural ranking - framework architecture [14]

Building upon the fused feature representation, to handle the ranking aspect, DSR assigns a ranking score to each object pair based on their predicted relationship scores. This ranking score reflects the relative importance or strength of the relationship compared to other object pairs. By incorporating ranking into the model, DSR aims to prioritize and highlight the most relevant and significant relationships in a given visual scene.

VRD with DSR uses a structural ranking loss function, which ensures that annotated relationships receive higher relevance scores compared to other possible relationships. While some relationships in the image may not have been annotated, they generally hold less significance than the annotated ones. Furthermore, the structural loss is integrated with the probability of the predicates conditioned on their subject and object, aiming to mitigate

the impact of incomplete annotations.

The application of DSR in VRD helps to address the challenge of accurately detecting and ranking relationships between objects in complex visual scenes. By utilizing deep learning and ranking techniques, it enables more precise and context-aware relationship detection, allowing for a richer understanding of the interactions between objects.

The authors of [14] have conducted numerous experiments to demonstrate that the proposed approach achieves superior performance compared to existing state-of-the-art methods when evaluated on two extensively employed datasets (Visual Relationship Dataset [12] and Visual Genome [58]).

Several baseline methods and state-of-the-art comparison methods was used to evaluate VRD with deep structural ranking. The baseline methods included JointCNN and JointBox [56], which aimed to predict relationship components jointly or classify joint bounding boxes into predicates, respectively. The state-of-the-art methods included VR-V, VR-LP, VTE, VRL, ViPCNN, and DR-Net, which employed various techniques such as visual appearance, language priors, visual translation embedding, reinforcement learning, message passing structure, and deep relational networks for visual relationship detection.

Comparative analysis revealed that JointCNN performed the poorest, indicating the difficulty of training a network with supervision from multiple components simultaneously. JointBox achieved better results by focusing solely on predicting predicates. The performance improvement observed between VR-V and VR-LP demonstrated the benefit of integrating category information and language priors. Methods that combined feature learning and visual relationship detection as joint learning tasks consistently outperformed those relying solely on visual appearance. However, multi-task learning strategies, as seen in VTE and ViPCNN, did not yield significant performance gains due to the increased training complexity and difficulties in balancing the contribution of each object function.

DR-Net [13], which exploited the dependencies between objects and relationships using spatial masks, outperformed other methods by effectively utilizing spatial configurations. Nevertheless, VRD with DSR surpassed all recent state-of-the-art approaches in both predicate and relationship detection tasks. It achieved an 11% improvement in Recall@100 for predicate detection and a 2.4% improvement for relationship detection compared to the previous best performance. The proposed method also outperformed DR-Net across all evaluation metrics, indicating the significance of consider-

ing the co-occurrence and incompleteness of predicates in visual relationship detection.

Methods	Predicate Det.		Relationship Det.	
	R@50	R@100	R@50	R@100
JointCNN	1.47	2.03	0.07	0.09
JointBox	25.78	25.78	-	-
VR-V	7.11	7.11	1.58	1.85
VR-LP	47.87	47.87	13.86	14.70
VTE	44.76	44.76	14.07	15.20
VRL	-	-	18.19	20.79
ViP-CNN	-	-	17.32	20.01
DR-Net	80.78	81.90	17.73	20.88
Ours	86.01	93.18	19.03	23.29

Figure 46: VRD with DSR: Performance comparison of the baseline and state-of-the-art methods on the Visual Genome dataset [14]

Methods	Predicate Det.	
	R@50	R@100
JointBox	46.59	46.77
VTE	62.63	62.87
Ours	69.06	74.37

Figure 47: VRD with DSR: Performance comparison of the baseline and state-of-the-art methods on the Visual Genome dataset [14]

In summary, VRD with deep structural ranking combines deep learning techniques with a ranking-based framework to detect and rank relationships between objects in visual scenes. By incorporating ranking scores, it provides a mechanism to highlight the most important relationships and helps address the challenges posed by incomplete annotations, leading to an improved accuracy and context-awareness in visual relationship detection tasks.

3.2.5 VRD with visual translation embedding network [15]

VRD with visual translation embedding network (VTransE) is a state-of-the-art method proposed for the challenging task of visual relationship detection. The main goal of visual relationship detection is to identify and classify relationships between objects in images. Traditional approaches have primarily focused on object detection and recognition, often neglecting the crucial aspect of modeling relationships between objects. VTransE addresses this limitation by explicitly modeling the spatial relationships between objects and leveraging contextual information for accurate relationship classification.

VTransE introduces a novel architecture consisting of three key components: subject appearance modeling, object appearance modeling, and spatial modeling. The subject and object appearance models capture the visual features of the individual objects in the image, while the spatial model encodes the relative positions and orientations of the objects.

To model subject and object appearances, VTransE utilizes a convolutional neural network (CNN) to extract deep visual features from the objects' regions of interest. These features are then used to learn discriminative representations for the objects, facilitating more accurate relationship classification.

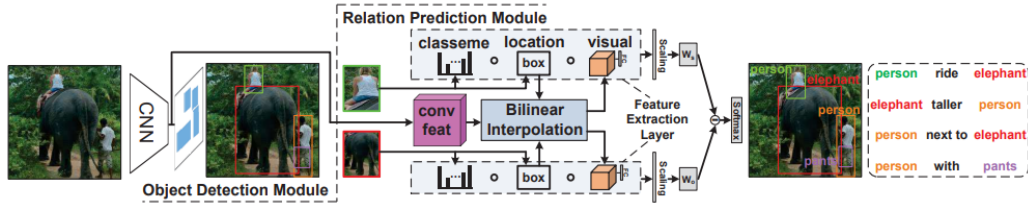


Figure 48: The VTransE network overview [15]

The spatial model in VTransE leverages a graphical model called a spatial transformer network. This model learns to predict the spatial transformations that align the subject and object regions, enabling the extraction of informative spatial features. By capturing the relative spatial configurations between objects, VTransE enhances its ability to recognize and classify complex relationships accurately.

VTransE further incorporates contextual information by exploiting scene-level context. It introduces a context-aware attention mechanism that dynamically weights the importance of different object pairs based on their

contextual relevance. This attention mechanism helps the model focus on relevant relationships and improve the overall performance of visual relationship detection.

Extensive experiments conducted on benchmark datasets demonstrate the effectiveness of VTransE in visual relationship detection. The results show significant improvements in both precision and recall compared to previous state-of-the-art methods. VTransE achieves superior performance in capturing complex relationships and handling challenging scenarios, such as occlusions and varying object scales. (Fig. 49)

Dataset	VRD [27]						VG [23]					
Task	Phrase Det.		Relation Det.		Retrieval		Phrase Det.		Relation Det.		Retrieval	
Metric	R@50	R@100	R@50	R@100	Rr@5	Med r	R@50	R@100	R@50	R@100	Rr@5	Med r
VisualPhrase [37]	0.54	0.63	–	–	3.51	204	3.41	4.27	–	–	11.42	18
DenseCap [19]	0.62	0.77	–	–	4.16	199	3.85	5.01	–	–	12.95	13
Lu's-V [27]	2.24	2.61	1.58	1.85	2.82	211	–	–	–	–	–	–
Lu's-VLK [27]	16.17	17.03	13.86	14.70	8.75	137	–	–	–	–	–	–
VTransE	19.42	22.42	14.07	15.20	7.89	41	9.46	10.45	5.52	6.04	14.65	7
VTransE-2stage	18.45	21.29	13.30	14.64	7.14	41	8.73	10.11	4.97	5.48	12.82	12
Random	0.06	0.11	7.14×10^{-3}	1.43×10^{-2}	2.95	497	0.04	0.07	1.25×10^{-3}	2.50×10^{-3}	3.45	1.28×10^4

Figure 49: Performance comparison on the VRD dataset [15]

In summary, VTranse presents an advanced architecture for visual relationship detection by integrating subject and object appearance modeling, spatial modeling, and context-aware attention mechanisms. The experimental results highlight its ability to accurately detect and classify relationships between objects in images, surpassing existing methods. The contributions of VTranse significantly advance the field of visual relationship detection and pave the way for further research in understanding visual scenes and their underlying relationships.

3.2.6 VRD with Internal and External Linguistic Knowledge Distillation [16]

Visual Relationship Detection (VRD) with Internal and External Linguistic Knowledge Distillation [16] is an advanced approach that aims to improve the accuracy and performance of visual relationship detection models by incorporating both internal and external sources of linguistic knowledge.

In VRD, the goal is to detect and understand the relationships between objects in visual scenes. By leveraging internal linguistic knowledge distillation, which includes textual descriptions, annotations, or labels associated with objects and relationships in the training data, the model can learn to better interpret and recognize these relationships.

Furthermore, external linguistic knowledge distillation involves utilizing additional linguistic information from external sources. This can include annotations from existing visual relationship datasets like VRD [12] and Visual Genome [58] or linguistic databases like Wikipedia, which provide semantic and contextual information related to the visual relationships. By incorporating this external knowledge, the model gains a more comprehensive understanding of the relationships between objects in the visual scenes.

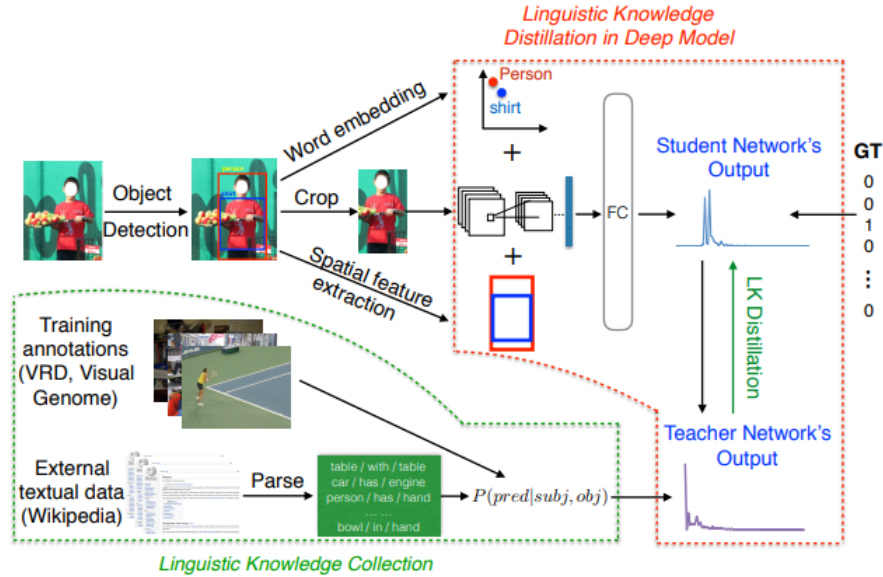


Figure 50: Linguistic Knowledge Distillation Framework [16]

The proposed in [16] end-to-end deep neural network is trained on rich visual features, derived from the cropped image of the joint bounding box of the object pair, spatial features, calculated from the two bounding boxes and word-embeddings of semantic features, stemming from the object detection class confidence scores, and is also further enhanced by a linguistic knowledge collection using the teacher-student framework to improve the accuracy of prediction and to augment the generalization capabilities of the model. (Fig. 50) The student network is trained not only to accurately predict the object and relation labels but to also replicate the nuanced confidence of the teacher network in assigning scores to different predicates.

The performance of the deep neural network has been evaluated on different combinations of input features and with or without linguistic knowledge distillation to assess their individual contributions. In Part 1 of Fig. 51 the baseline model (Baseline: L only) is exclusively trained and evaluated on the VRD dataset, using the linguistic knowledge extracted from the training data. The results have demonstrated the model’s exceptional predictive capabilities, surpassing in accuracy the current state-of-the-art method by a substantial amount. (achieving a recall of 51.34% compared to 47.87% (VRD-Full [12]) for R@100/50, k=1 on the entire VRD test set) This outcome suggests that augmenting the data-driven model with distilled knowledge is both reliable and precise. However using only external knowledge, sourcing from the VRD dataset, with no regard to the visual, spatial and semantic features, that the image holds, does significantly worse on the zero-shot recognition task, which proves the necessity of integrating both data-driven and knowledge-based methods, as suggested in VRD with internal and external linguistic knowledge distillation.

The advantages of incorporating Linguistic Knowledge (LK) distillation on the other hand are evident across the various feature settings. The data-driven neural networks that assimilate linguistic knowledge exhibit significant performance improvements compared to the data-driven models alone. Moreover, the data-driven models that incorporate linguistic knowledge show an improved recall on the zero-shot test set. The networks with LK distillation demonstrate superior generalization capabilities, surpassing both the data-driven baselines and knowledge-only baselines by a substantial margin. Additionally, using the combination of the teacher (for predicting predicates for object pairs present in the training data) and student networks (for all remaining pairs) contributes to improved model performance and effectiveness.

	Entire Set			Zero-shot		
	R@100/50 ² k=1	R@100 k=70	R@50 k=70	R@100/50 k=1	R@100 k=70	R@50 k=70
Part 1: Training images VRD only						
Visual Phrases [26]	1.91	-	-	-	-	-
Joint CNN [6]	2.03	-	-	-	-	-
VRD-V only [19]	7.11	37.20 ³	28.36	3.52	32.34	23.95
VRD-Full [19]	47.87	84.34	70.97	8.45	50.04	29.77
Baseline: U only	34.82	83.15	70.02	12.75	69.42	47.84
Baseline: L only	51.34	85.34	80.64	3.68	18.22	8.13
U+W	37.15	83.78	70.75	13.44	69.77	49.01
U+W+L:S	42.98	84.94	71.83	13.89	72.53	51.37
U+W+L:T	52.96	88.98	83.26	7.81	40.15	32.62
U+SF	36.33	83.68	69.87	14.33	69.01	48.32
U+SF+L:S	41.06	84.81	71.27	15.14	72.72	51.62
U+SF+L:T	51.67	87.71	83.84	8.05	41.51	32.77
U+W+SF	41.33	84.89	72.29	14.13	69.41	48.13
U+W+SF+L: S	47.50	86.97	74.98	16.98	74.65	54.20
U+W+SF+L: T	54.13	89.41	82.54	8.80	41.53	32.81
U+W+SF+L: T+S	55.16	94.65	85.64	-	-	-
Part 2: Training images VRD + VG						
Baseline: U	36.97	84.49	70.19	13.31	70.56	50.34
U+W+SF	42.08	85.89	72.83	14.51	70.79	50.64
U+W+SF+L: S	48.61	87.15	75.45	17.16	75.26	55.41
U+W+SF+L: T	54.61	90.09	82.97	9.23	43.21	33.40
U+W+SF+L: T+S	55.67	95.19	86.14	-	-	-

Figure 51: Predicate Detection on VRD Testing dataset [12]

"U" - the union of the two objects' bounding boxes, "W" - the word vector embeddings of the object detection classes (the semantic features), "SF" - the spatial features of the bounding boxes, "L" - linguistic knowledge distillation, "S" - student network, "T" - teacher network

In summary, visual relationship detection with internal and external linguistic knowledge distillation provides a promising approach for enhancing the accuracy and generalization of deep neural networks in detecting relationships between objects in visual data. By effectively combining visual and linguistic information, the method contributes to the advancement of the field and opens up opportunities for more sophisticated visual understanding tasks.

3.2.7 VRD with logical tensor networks [17]

The field of Semantic Image Interpretation (SII) focuses on automatically extracting high-level information about the content of visual scenes, including objects, their attributes, and the relationships between them. This is achieved by constructing a scene graph that represents the objects as labeled nodes and their relationships as labeled edges. SII enables various applications such as visual question answering, image captioning, complex image querying, and robot interaction.

Visual Relationship Detection (VRD) is a specific task within scene graph construction, where visual relationships are represented as triples consisting of a subject, predicate, and object. Supervised learning techniques are commonly used to detect visual relationships, relying on large amounts of annotated training data. However, the significant effort required for detailed annotation leads to a lack of information in the datasets. (e.g. missing co-occurring relations - "person rides horse" is labeled, but "person on horse" not, and completely absent object pairs) As a result, zero-shot learning techniques need to be employed to predict visual relationships involving unseen instances during training. These techniques leverage similarities with known triples in the training set or utilize background knowledge such as linguistic information and logical constraints about the relationships to enhance the generalization capabilities of models and enable more accurate predictions, even for relationships not seen during training.

The authors of [17] present a solution to the zero-shot learning problem in visual relationship detection by employing Logic Tensor Networks (LTN) [59]. Logical Tensor Networks (LTN) are a framework based on Statistical Relational Learning that combines logical reasoning with statistical modeling. LTN leverage relational data, such as visual relationships, and incorporate logical constraints to learn from the data and make predictions. They represent knowledge using first-order logic and use tensors to encode uncertainty and uncertainty propagation.

In the context of visual relationship detection, LTN are employed to tackle the zero-shot learning problem. By utilizing LTN in visual relationship detection, the models can leverage logical constraints and similarities with already seen triples to make predictions about unseen relationships. This approach allows for generalization and inference based on prior knowledge, similar to how humans reason and learn. The integration of logical knowledge with data-driven approaches and linguistic information can enhance the perfor-

mance of visual relationship detection systems, especially when dealing with unseen instances and complex datasets.

Visual relationship detection with logical tensor networks (VRD with LTN)’s approach [17] has been tested on the Visual Relationship Dataset (VRD) [12], which consists of 4000 training images and 1000 test images that have been annotated with visual relationships. Notably, there are 1877 relationship instances that occur exclusively in the test set, which are utilized for evaluating the zero-shot learning scenario.

The evaluation of VRD with LTN performance is conducted on several standard tasks within the Visual Relationship Detection (VRD) domain. The first task is phrase detection, which involves predicting a correct triple consisting of a subject, predicate, and object, along with localizing them within a single bounding box encompassing both subject and object. To be considered a true positive, the predicted triple must have labels that match those of the ground truth, and the predicted bounding box should overlap with the corresponding ground truth bounding box by at least 50%. The ground truth bounding box is determined by taking the union of the subject and object ground truth bounding boxes.

The second task is relationship detection, where the objective is to predict a correct triple / relationship along with bounding boxes encompassing the subject and object involved in the relationship. A predicted triple is considered a true positive if both predicted bounding boxes have an overlap of at least 50% with their corresponding ground truth bounding boxes, and the labels of the predicted triple should match those in the ground truth.

Lastly, the predicate detection task focuses solely on predicting a set of correct binary predicates between a given set of bounding boxes, without considering the performance of an object detector. The assessment revolves around the ability of LTN to accurately predict binary predicates.

The evaluation of VRD with LTN’s performance on these tasks has been conducted using two types of grounded theories: (1) T_{expl} is a model trained using only positive and negative examples for the class and relation predicates (positive example for class predicate X - object A is labeled with class X in the training data, negative examples - object A is not part of any of the other classes, positive example for relation Y - object pair B is labeled with relation Y, negative examples - object pair B can not be of any of the other relation types.) and (2) T_{prior} , a model, that learns by incorporating both examples and logical constraints, allowing for the assessment of the impact of logical constraints in comparison to a conventional machine learning approach.

The performance of the two LTN models (T_{expl} and T_{prior}) on the relationship and phrase detection tasks can be only truly compared with competing methods, which use the same object detection model, as object detection errors are propagated and strongly influence the relation detection results. Such methods are the ones introduced in [60] (RESCAL, Multi-WayNN, ComplEx, DistMult) and Visual Relationship Dataset (VRD) [12] and the comparison demonstrates that LTN handle object detection errors more effectively. In contrast, the predicate detection task is relatively easier as it is independent of object detection. Here, the effectiveness of LTN becomes apparent among all state-of-the-art solutions, showing significant performance improvements. (Fig. 54) While as mentioned a comprehensive comparison with other competitors is only feasible in the predicate detection task due to the use of different object detectors, both LTN models effectively exploit the data structure to infer relationship similarity and surpass VRL [61] and Context-AwareVRD [62].

Furthermore, the LTN model T_{prior} , trained with both data and constraints, outperforms the model T_{expl} , trained solely with data. This indicates that logical constraints about the domain effectively exclude certain binary predicates for a given subject or object bounding box. (e.g. if the relations is "wear", the object can not be a vehicle or a building) This characteristic of LTN is crucial and brings the advantage of LTN over the Logical Knowledge Distillation (LKD) approach [16], as T_{prior} achieves superior recall@50 and recall@100 in predicate detection. The performance in other tasks remains comparable, despite the fact that LKD employs a more recent object detector with better performance than the one used in VRD with LTN. LKD relies on statistical dependencies (co-occurrences) between subjects/objects and relationships, whereas VRD with LTN leverage logical knowledge, which can express more detailed information and enable more accurate reasoning on visual relationships. VRD with LTN also has a scalability advantage against LKD, as LTN has a linear complexity in comparison to LKD having a quadratic time complexity.

These results validate the ability of LTN to generalize to never-before-seen relationships in visual relationship detection and the capability of LTN to utilize logical constraints in the background knowledge and enhance results in a zero-shot learning scenario. This significant achievement highlights the potential of logical constraints, which are general in nature and can be probably extracted from online linguistic resources, in compensating for the lack of information in datasets due to annotation efforts.

Task	Phrase Det.	Phrase Det.	Relationship Det.	Relationship Det.	Predicate Det.	Predicate Det.
Evaluation	R@100	R@50	R@100	R@50	R@100	R@50
VRD [19]	3.75	3.36	3.52	3.13	8.45	8.45
RESCAL [2]	6.59	5.82	6.07	5.30	16.34	16.34
MultiwayNN [2]	6.93	5.73	6.24	5.22	16.60	16.60
ComplEx [2]	6.50	5.73	5.82	5.05	15.74	15.74
DistMult [2]	4.19	3.34	3.85	3.08	12.40	12.40
$\mathcal{T}_{expl}$ (hmean, newfeats)	15.91 ± 0.54	11.00 ± 0.34	14.65 ± 0.54	10.01 ± 0.38	74.71 ± 0.73	56.25 ± 0.75
$\mathcal{T}_{prior}$ (hmean, newfeats)	15.74 ± 0.44	11.40 ± 0.31	14.43 ± 0.41	10.47 ± 0.32	77.16 ± 0.60	57.34 ± 0.78
LKD [31]	17.24	12.96	15.89	12.02	74.65	54.20
VRL [18]	10.31	9.17	8.52	7.94	-	-
Context-AwareVRD [35]	11.30	10.78	10.26	9.54	16.37	16.37
WeaklySup [23]	7.80	6.80	7.40	6.40	21.60	21.60

Figure 52: VRD with LTN: Performance comparison of the baseline and state-of-the-art methods on the Visual Relationship Dataset [17]

Overall, logical tensor networks offer a powerful framework that combines statistical modeling with logical reasoning to improve the detection of visual relationships, particularly in scenarios involving zero-shot learning where the models need to generalize beyond the training data. VRD with LTN has demonstrated that the combination of logical knowledge and explicit data exceeds state-of-the-art solutions that rely solely on visual and spatial data or linguistic knowledge. This significant accomplishment demonstrates the feasibility of utilizing logical constraints as a means to address the information gaps within datasets arising from annotation efforts.

3.3 Scene2Model approach [18] [19]

The Scene2Model approach was first introduced in [18] as a framework for the automatic transformation of the results from a haptic design thinking method into digital diagrammatic models. The authors of [18] have conducted a proof-of-concept implementation to evaluate the user acceptance of the proposed tool, which has received an overall positive feedback when questioning the study group participants. [19] We will be further building onto the proposed Scene2Model tool by addressing some of the open questions like the adoption of a relation prediction software for the detection of logical connections between the domain objects and the implementation of a user-friendlier graphical interface such as a mobile application.

Haptic design focuses on creating tactile and sensory experiences through physical interactions. Haptic storyboards serve as visual representations of these interactions, aiding in the design and communication of haptic experiences. The Scene2Model tool [19] offers an innovative approach to transform haptic storyboards into diagrammatic models, automating the process and allowing designers to generate tangible representations of haptic interactions.

The Scene2Model tool encompasses several key components. Designers create haptic storyboards that illustrate the sequence of haptic interactions or tactile experiences. These storyboards serve as the input for the tool. Computer vision algorithms analyze the haptic storyboard images, extracting information about the spatial arrangement, text annotations, and haptic cues depicted. This process involves object recognition and feature extraction techniques. Machine learning algorithms then infer diagrammatic models based on the extracted information, employing shape recognition, spatial mapping, and semantic reasoning techniques. The inferred models undergo refinement based on additional design considerations, allowing designers to modify the representation, adjust spatial relationships, and refine the haptic cues. Visualization tools enable designers to perceive and evaluate the resulting diagrammatic models.

The Scene2Model tool has the potential to significantly impact the haptic design process. Firstly, it enhances time efficiency by automating the transformation of haptic storyboards into diagrammatic models, reducing the time and effort required to create formal digital representations of haptic interactions. This allows designers to allocate more time for ideation and refinement. Secondly, the tool promotes design consistency by providing a standardized approach to transforming haptic storyboards. It helps main-

tain a coherent representation of haptic experiences across different design iterations and design teams. Lastly, the tool enhances the accessibility of haptic design by enabling designers with varying levels of expertise to create diagrammatic models of haptic interactions. This broadens participation in the design process and encourages collaboration among multidisciplinary teams.

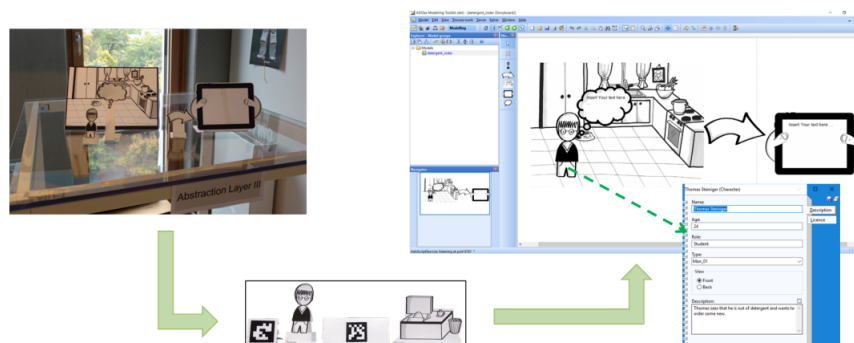


Figure 53: Visualization of the Scene2Model approach [19]

The Scene2Model tool presents an innovative approach to transform haptic storyboards into diagrammatic models. By leveraging computer vision and machine learning techniques, the tool automates the conversion process and provides designers with tangible representations of haptic interactions. Its applications in haptic design range from concept visualization and communication to design exploration and user experience evaluation. With its potential to enhance time efficiency, design consistency, and design accessibility, the Scene2Model tool holds promise for advancing the haptic design process and fostering innovation in this field.

4 Implementation Approach and Technology Selection

In this chapter we will describe the approach, that has framed the development process of the Scene2Model framework and namely the Agile Modeling Method Engineering approach (AMME). We will also unveil the thought process and criteria utilized to make informed choices among the reviewed technologies and solution approaches for the challenging computer vision tasks of object detection and visual relationship detection.

4.1 Agile Modeling Method Engineering

The Scene2Model framework relies on object detection and relation prediction technologies to ultimately create a domain-specific modelling method, capable of capturing the emerged in a live design thinking workshop innovation ideas and concepts. The development of the S2M framework has been performed by closely following the concepts the Agile Modeling Method Engineering approach (AMME) [63].

Agile modeling method engineering [63] is an approach that aims to enhance the flexibility and adaptability of software development methodologies by incorporating principles from agile methodologies into the process of designing and creating modeling methods. This methodological approach recognizes the need for continuous evolution and refinement of modeling methods to meet the changing requirements and challenges in software development projects.

The core idea of agile modeling method engineering is to apply agile principles such as iterative development, collaboration, and customer feedback to the process of designing and evolving modeling methods. The traditional approach to modeling method engineering often involves upfront planning and a sequential development process, which can be rigid and less responsive to changing project needs. In contrast, agile modeling method engineering emphasizes an iterative and incremental approach, where the development of modeling methods is done in small increments, allowing for frequent feedback and adaptation.

The key activities in agile modeling method engineering include requirements elicitation, design, implementation, and evaluation. These activities are carried out in short iterations, typically ranging from weeks to months,

to ensure regular feedback and continuous improvement. Agile modeling method engineering also emphasizes collaboration and communication among stakeholders, including developers, domain experts, and end-users, to gather requirements, validate the effectiveness of the modeling methods, and incorporate feedback in a timely manner.

One of the main advantages of agile modeling method engineering is its ability to accommodate changing requirements and incorporate new knowledge and insights during the development process. By adopting an iterative and incremental approach, modeling methods can be refined and adapted based on the evolving needs and emerging insights from the development team and end-users. This flexibility allows for greater agility in responding to dynamic project environments and improves the overall quality and effectiveness of modeling methods.

Furthermore, agile modeling method engineering promotes a customer-centric approach, where the needs and expectations of end-users are prioritized and incorporated into the design and evaluation of modeling methods. This customer-focused perspective ensures that the resulting modeling methods are aligned with the specific requirements and goals of the software development project, leading to higher user satisfaction and better outcomes.

In conclusion, agile modeling method engineering offers a dynamic and flexible approach to designing and evolving modeling methods in software development. By embracing iterative development, collaboration, and customer feedback, this approach enables the creation of modeling methods that are more responsive to changing project needs and aligned with user requirements. The adoption of agile principles in modeling method engineering enhances the overall agility and effectiveness of software development processes and contributes to the success of software projects.

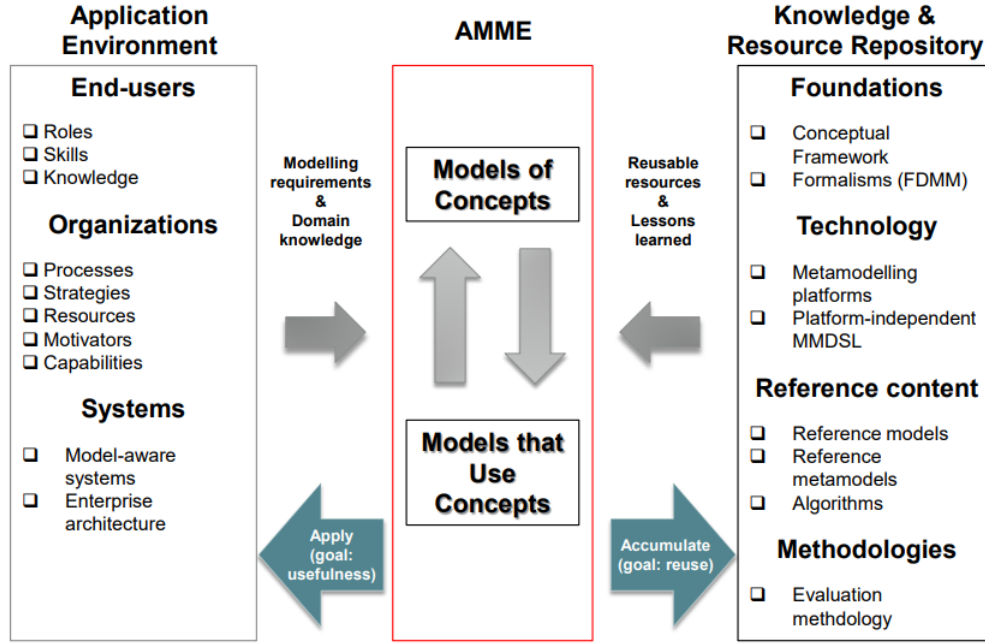


Figure 54: Agile Modeling Method Engineering (AMME) framework [63]

4.2 Object detection

We have reviewed many different algorithms for solving the computer vision task of recognizing already known objects on images and have chosen the YOLOv4 algorithm for object detection after some serious considerations, as it is renowned for its ability to perform real-time object detection without compromising on precision.

When selecting the most suitable object detection model it is important to consider factors such as specific application requirements, computational resources, and trade-offs between accuracy and speed. In our case, having a powerful GPU machine at our disposal, has given us the freedom to explore and consider also more advanced and resource intensive technologies. In the Scene2Model framework object detection comes into play in a live workshop setting, thus detection speed is of primary importance. Precision is also crucial, as object detection errors are propagated to the other elements of the solution and most importantly the performance of the relation recognition module strongly relies on the correct identification and classification of all objects.

Having said all of this, we are strongly confident that YOLOv4 is an excellent candidate for tackling the challenge of object detection in an ad-hoc setting due to several compelling reasons. First and foremost, its outstanding speed enables rapid inference times, making it ideal for real-time applications. By employing a single neural network to predict bounding boxes and class probabilities, YOLOv4 eliminates the need for complex region proposal networks, resulting in faster and more efficient detection. Moreover, YOLOv4 achieves a balance between speed and accuracy, making it well-suited for real-time scenarios where both factors are crucial like ours. It incorporates advanced architectural improvements against previous YOLO versions [5] [49] [48], such as CSPDarknet53 as backbone, to capture fine-grained details and handle objects at various scales, leading to improved detection accuracy. YOLOv4’s adaptability and generalization capabilities are also noteworthy. It performs consistently well across different object classes and datasets, making it versatile for a wide range of applications and domains. Additionally, its lightweight and efficient backbone network, CSPDarknet53, ensures fast inference and training times, making it suitable for resource-constrained environments. The availability of pre-trained models, open-source implementations, and a supportive community contribute to the ease of adoption and customization of YOLOv4.

In summary, YOLOv4’s combination of speed, accuracy, adaptability, efficiency, and community support make it a very suitable choice for real-time object detection tasks, enabling efficient and responsive detection.

4.3 Visual relationship detection

The task of visual relationship detection can be addressed in three ways as reviewed in Section 3.2. In the first method, called predicate detection, object detection and relationship detection are completely decoupled, so object detection is performed first and the recognized objects including bounding box and class assignment are fed to the relationship detection model to predict the most suitable relationship between the given objects. [12] The other two methods internalize the task of object detection in the visual relation detection model and predict whole phrases (a relationship or a composite, consisting of two objects and a predicate, with a single bounding box region, covering both objects) [11] or relationships (additionally to phrase detection, the relation participant objects are localized separately from the whole relationship - a bounding box is assigned for each of them) [12].

In the scope of this research, when we speak of visual relationship detection, we would be referring to the task of predicate detection, which requires a functioning and well-performing object detection model to start with. The decision to approach the tasks of object detection and relation detection separately comes from the fact, that we already have an excellently performing object detection model, scoring a mAP of almost 100%, and the task of predicate detection is considerably easier than phrase and relationship detection, given a reliable object detection model [12].

We have chosen Logic Tensor Networks (LTN) for visual relation detection due to their ability to capture complex visual relationships and to incorporate logical reasoning and contextual information, resulting in a robust and comprehensive solution. Our preferred technology offers several advantages over alternative visual relationship detection tools. Firstly, LTN provide a formal logical framework that can handle uncertainty and ambiguity inherent in visual data, enabling a more interpretable and transparent approach. Additionally, the combination of VRD and LTN leverages the power of deep learning for feature extraction and spatial reasoning, resulting in improved accuracy and performance compared to other methods that may lack these capabilities. [17]

The application of a visual relationship detection tool in the Scene2Model approach, is required in the digitization of the results of the design thinking workshop, the setup of which is in our full control, which means that the VRD training dataset is created specifically for the domain problem in question and traditionally challenging circumstances like the long tail distribution of

relationships in real world can be minimized. [11]

Although inherently easier than phrase detection, predicate detection is still a very challenging task, which is still open in the literature. As we will be applying predicate detection in a laboratory setup, it is in our full control to simplify it by adding cues like arrow paper figurines as complementary object classes, pointing out that a relation between two objects persists, and even annotating the arrows with a label to specify the concrete type of relationship.

The different approaches to modeling a relationship between two objects in a workshop setting are illustrated in Table 1. We have identified three possible ways to represent, that a relationship between two object holds: (1) by placing a labeled arrow between the two objects, (2) by placing an empty arrow between the two objects and (3) by positioning the objects close by, so that their belonging together is spatially visible. Generally speaking, the easier it is to recognize a relationship, the bigger the preliminary workload for the digital engineering team, that facilitates the design thinking workshops and the less smooth and intuitive the design thinking process for the business experts is. More specifically, the simplest approach is to use labeled arrows and to place them between the two objects. This makes the relation detection pretty straightforward, as the arrow contains the whole information about the relation participants and their roles and the correct predicate. On the flip side, this requires to additionally train the object detector to identify arrows and to also be able to recognize which direction the arrow is pointing. Moreover, the hand-written label on the arrow should be interpreted by a text recognition module and mapped to a valid predicate from the domain ontology. Additionally, the workshop participants should be instructed on how to place the arrows, so that a relationship is identified correctly, and which predicates are allowed. This method is not satisfactory, as leaving this much free space for human error may compromise the visual relation detection accuracy. The next possibility for modeling a relation is to use arrows but this time without labeling them. This requires using a more sophisticated prediction detection module in place of the text recognition unit, which should apply reasoning methods and assign the best suitable predicate based on the local context. As a result the workshop participants do not need to worry about choosing a valid predicate from the predefined set, which makes the solution more stable to human error than the previously mentioned approach. The last and most advanced way of representing a relationship between a pair of objects in design thinking workshop setting

consists in just placing the objects close by or so that they overlap, thus it requires no visible cues for extra annotation and as a result eliminates the extra training for the recognition of the arrows and the extra briefing for the workshop participants, on how to place the arrows, so that they are detected correctly. This leads to requiring more progressive relation identification and predicate detection mechanisms, which should take into account the global context of the whole scene and should be able to evaluate potential relations based on spatial cues and predefined logical constraints about the domain.

Having reviewed all options, we have decided on taking the last approach, as it requires least preparation on the part of the digital innovation team for the facilitation of the workshops (no extra training for the detectors and briefings) and provides the most comfortable and intuitive setting for the business experts participating, while in the same time eliminating all possibilities for a human error, which might lead to compromised detection accuracy.

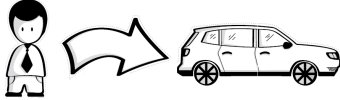
Visual cue	Prerequisites	Relation representation	Required technologies
Labeled directed connector	- multiple images of all objects and arrows from different angles for training - bounding box coordinates and class labels for all objects and arrows on all images for training - briefing for the workshop participants on how to place the arrows - briefing for the workshop participants on which predicates are allowed		- object detector - locates and classifies all objects on an image - relation identifier - acknowledges whether a relationship between two objects persists and if so determines the object roles in the relation (subject or object) - text recognition tool - interprets the handwritten text on the arrows
Non-labeled directed connector	- multiple images of all objects and arrows from different angles for training - bounding box coordinates and class labels for all objects and arrows on all images for training - briefing for the workshop participants on how to place the arrows		- object detector - locates and classifies all objects on an image - relation identifier - acknowledges whether a relationship between two objects persists and if so determines the object roles in the relation (subject or object) - predicate detector - assigns the best suitable predicate based on the local context
No visible cue	- multiple images of all objects from different angles for training - bounding box coordinates and class labels for all objects on all images for training		- object detector - locates and classifies all objects on an image - relation identifier - acknowledges whether a relationship between two objects persists - predicate detector - assigns the best suitable predicate based on logical constraints about the domain

Table 1: Different approaches to modeling a relationship between two objects in a design thinking workshop setting

5 Implementation

The Scene2Model (S2M) framework, created in the scope of this research thesis, poses as a possible solution to the challenge of digitalizing a haptic scene into a universal executable conceptual model in the global context of innovation, which requires expert knowledge to be extracted from business specialists, made explicit and transformed into a formal abstract executable representation of the underlying domain concept in question. Previous work on the Scene2Model scenario has been conducted by the authors of [18], who were also generously providing helpful advice and support during the whole process of implementation.

In this section, we will go through the implementation of the S2M tool and its representation in both the physical and digital space in detail. The physical aspect of the solution consists in the facilitation of real-life design thinking workshops and their building blocks or the physical artefacts used for representing different concepts from the problem domain. As mentioned in Section 2.2 the whole design thinking setup – from the tangible objects – the physical design thinking toolkit, which includes general and more specific paper figurines, through the laboratory configuration including the required hardware and software to the guidance and expertise on how to organize a business innovation think tank, has been pre-determined and thankfully provided by the Vienna node of OMiLAB. [2]

The real contribution of this master thesis lays in the digital space of the Scene2Model framework and contains following components: (1) an automated server for object detection training and testing, responsible for learning the pre-defined elements of the domain and recognizing them, when aligned together in a haptic scene (2) a mobile application functioning as a graphical user interface, easing the preparation of the training dataset for object detection (the process of uploading of the images and labeling of the contained objects in them, required for learning) and the image submitting procedure of the physical scene for recognition, (3) a visual relationship prediction module in charge of identifying the connections between object pairs, relying on spatial features and a semantic knowledge base, (4) a modelling language, created dynamically based on the results of the object detection and relation identification, to facilitate the design of the desired domain-specific conceptual model and (5) an extension software for importing the recognized scene components (objects and relations) in the modelling tool and transforming them into valid elements of the aforementioned modelling

language to ultimately form a conceptual model, outlining the initial ideas and serving as an executable abstract representation of the domain in question, which can be further translated into various specific models using conversion tools. (Fig. 55)

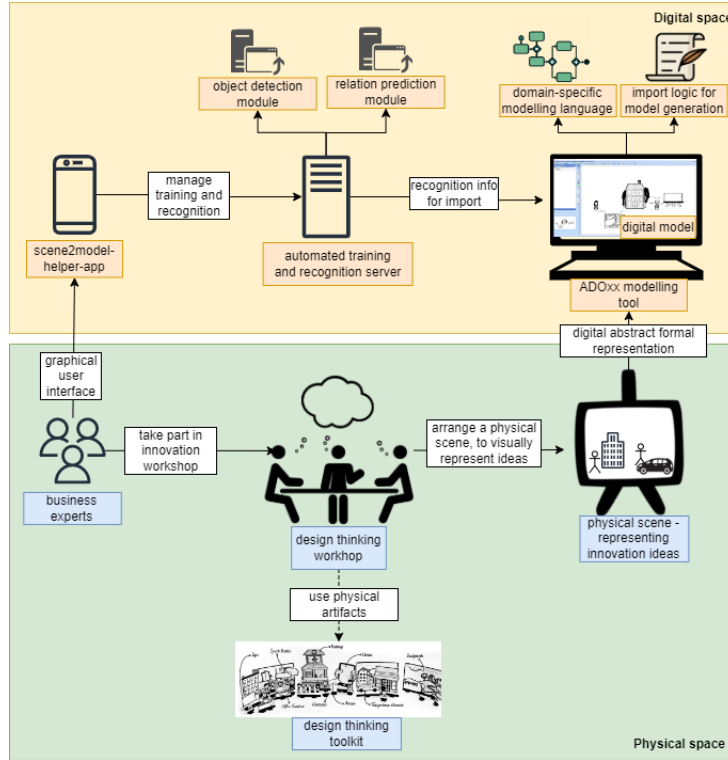


Figure 55: Scene2Model framework

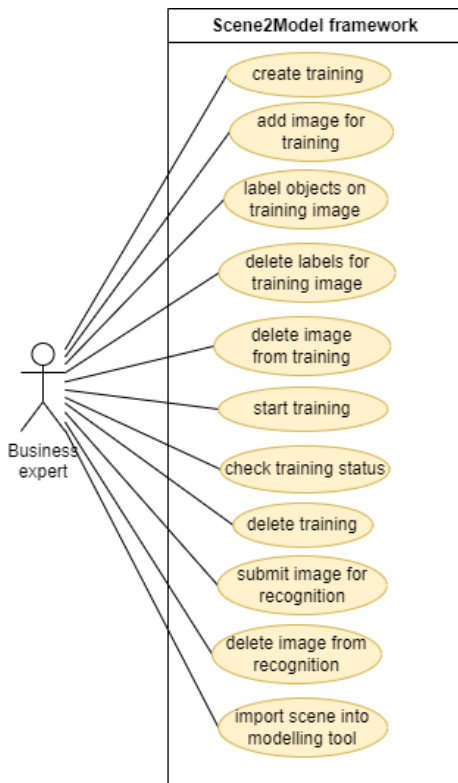
In the following first we will diagrammatically present the structure and interactions between components as well as the physical deployment of the emerged software system followed by a review of the physical design thinking setup and the required hardware and software for the S2M realization. Next we will describe the implementation of the mobile application along with the object detection training and recognition server, succeeded by a review of relation identification and prediction tool. Finally we will go through the module, where everything comes together, or more concretely the generation of the domain-specific modelling language and the logic for mapping the objects of the physical space to digital concepts and the generation of the pursued conceptual model.

5.1 UML diagrams

UML (Unified Modeling Language) [64] diagrams are a set of graphical notations used for visualizing, specifying, constructing, and documenting software systems. They provide a standardized way to depict various aspects of a system, such as its structure, behavior, and interactions between different components. UML diagrams serve as a communication tool among stakeholders, enabling better understanding and analysis of complex software systems. [64]

5.1.1 Use case diagram

Use case diagrams are a type of UML diagram [64] that depict the interactions between actors (users, systems, or external entities) and a system under consideration. They illustrate the functionality and behavior of a system from a user's perspective. Use case diagrams are useful for identifying, organizing, and visualizing the different use cases or functionalities that a system should provide to its users. [64]



5.1.2 Activity diagrams

An activity diagram is a type of UML diagram [64] that models the flow of activities or processes within a system. It represents the workflow or behavior of a system by showing the sequence of activities, decisions, and concurrent actions. Activity diagrams are particularly useful for visualizing complex business processes, algorithmic workflows, or the behavior of a software system at a high level of abstraction. They can capture both the sequential and parallel flows of activities, and include elements such as actions, decision points, control flow, concurrency, and looping. [64]

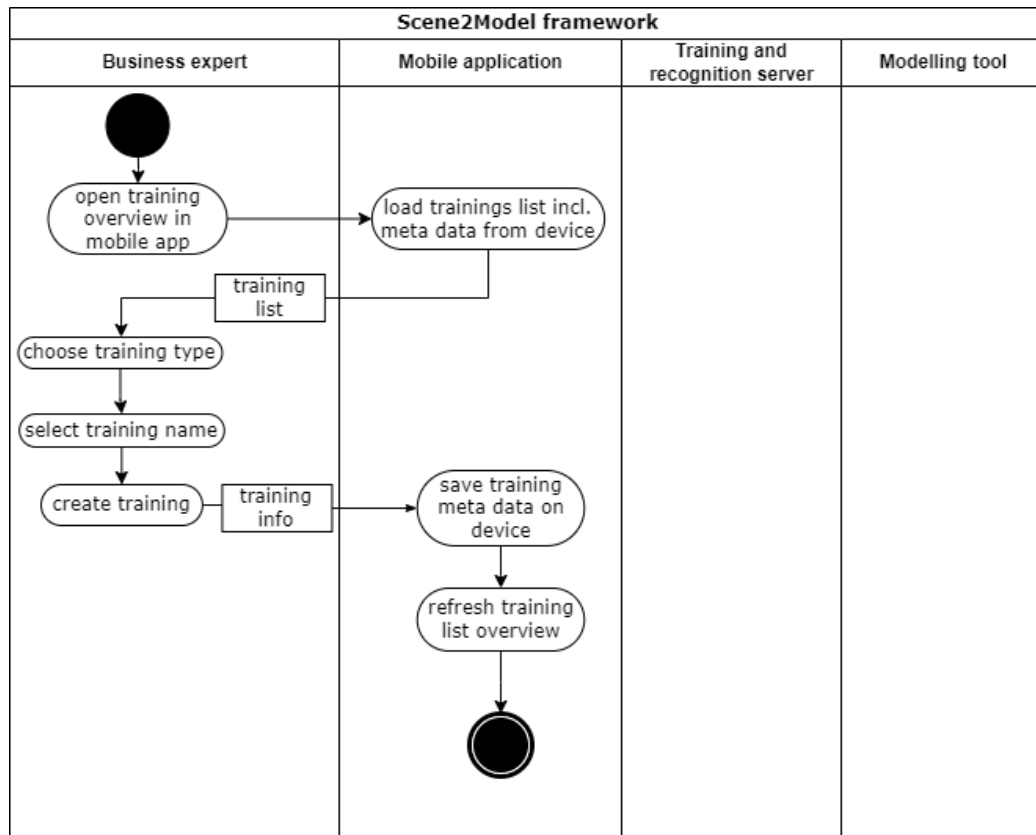


Figure 56: Activity diagram - create training

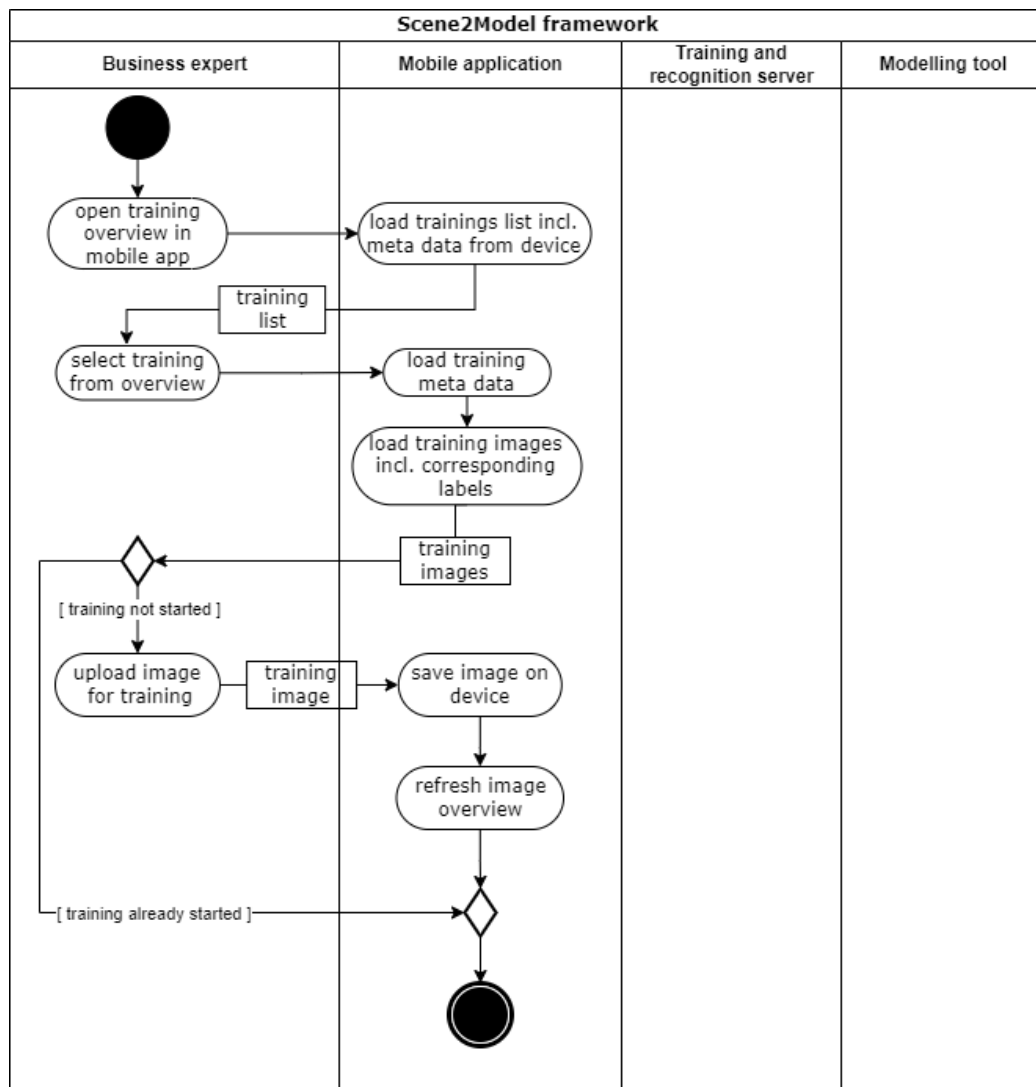


Figure 57: Activity diagram - add image for training

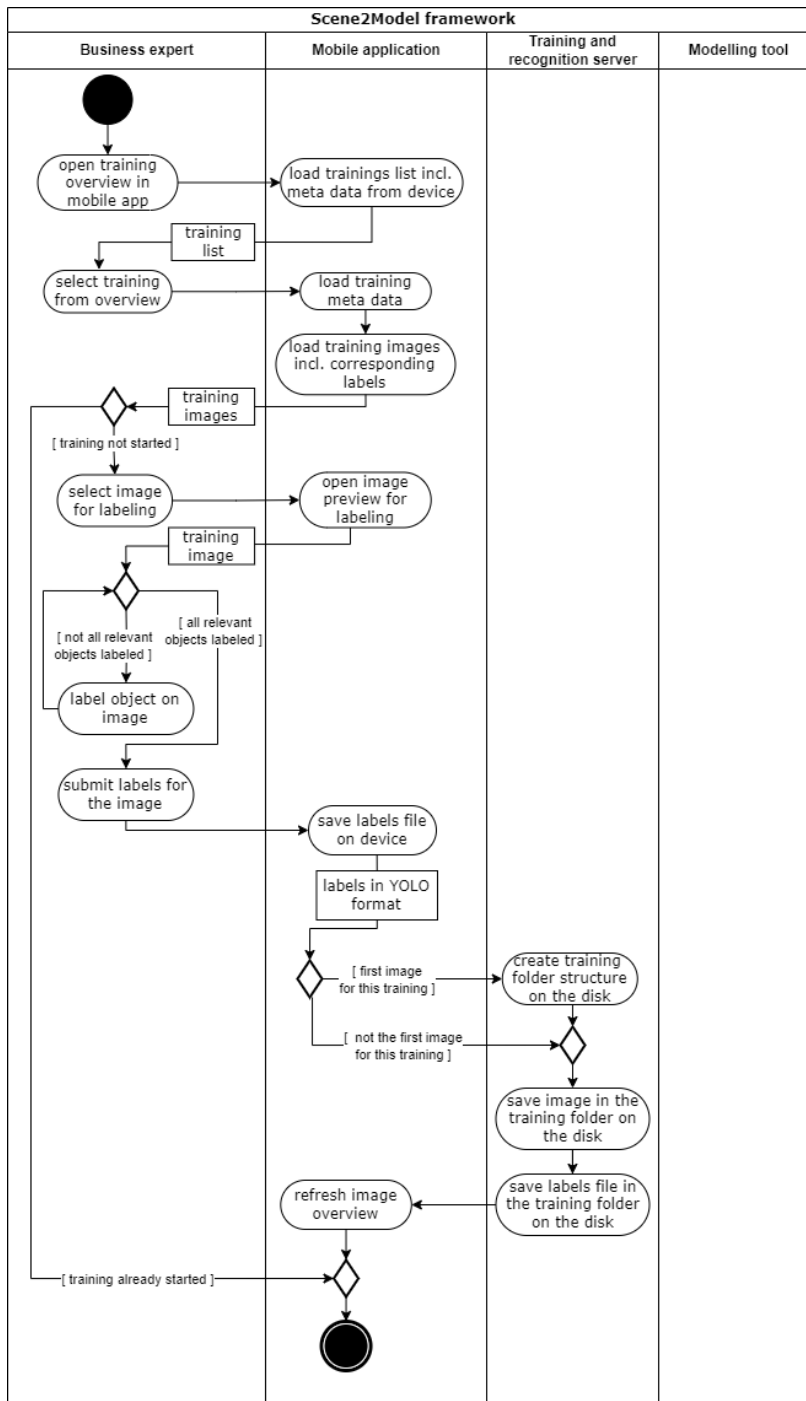


Figure 58: Activity diagram - label objects on training image

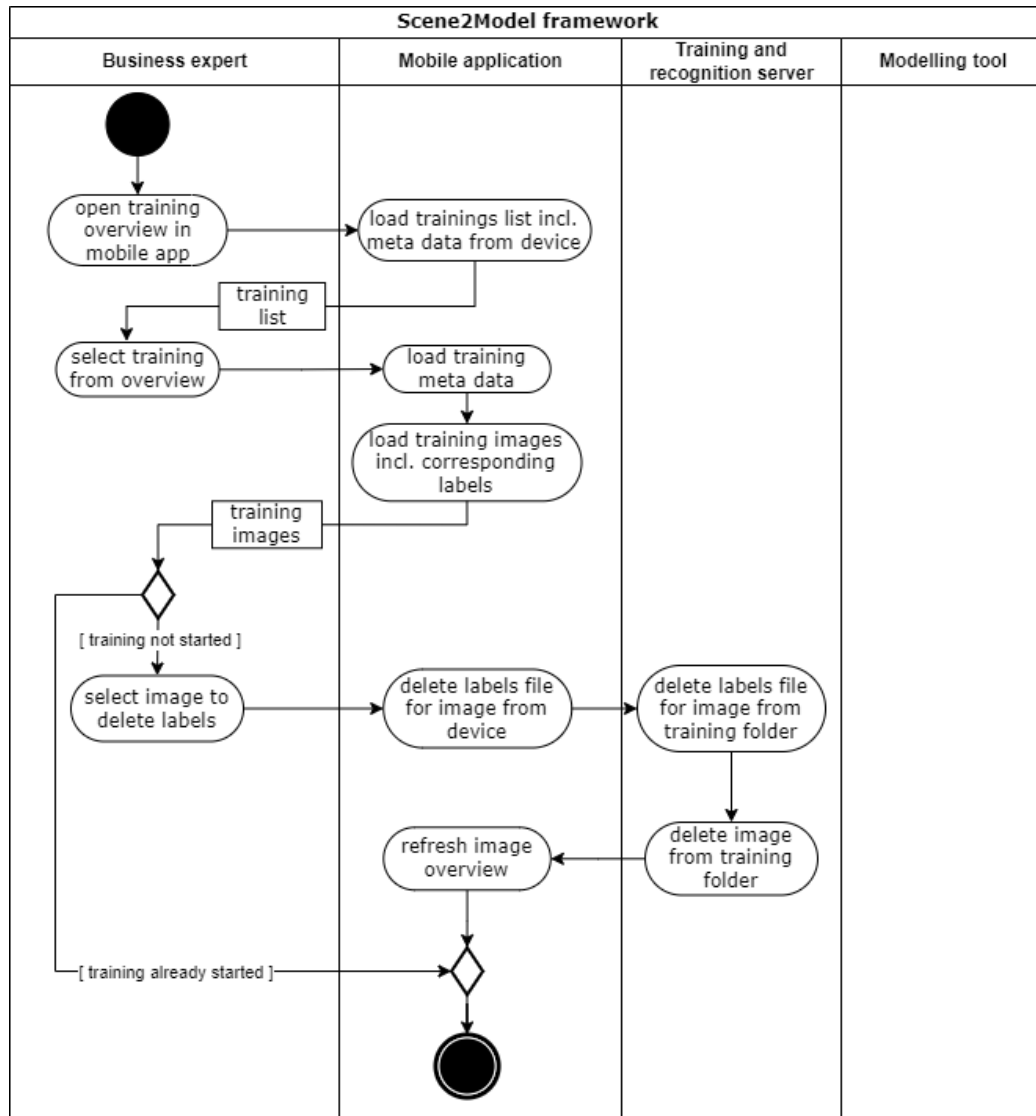


Figure 59: Activity diagram - delete labels for training image

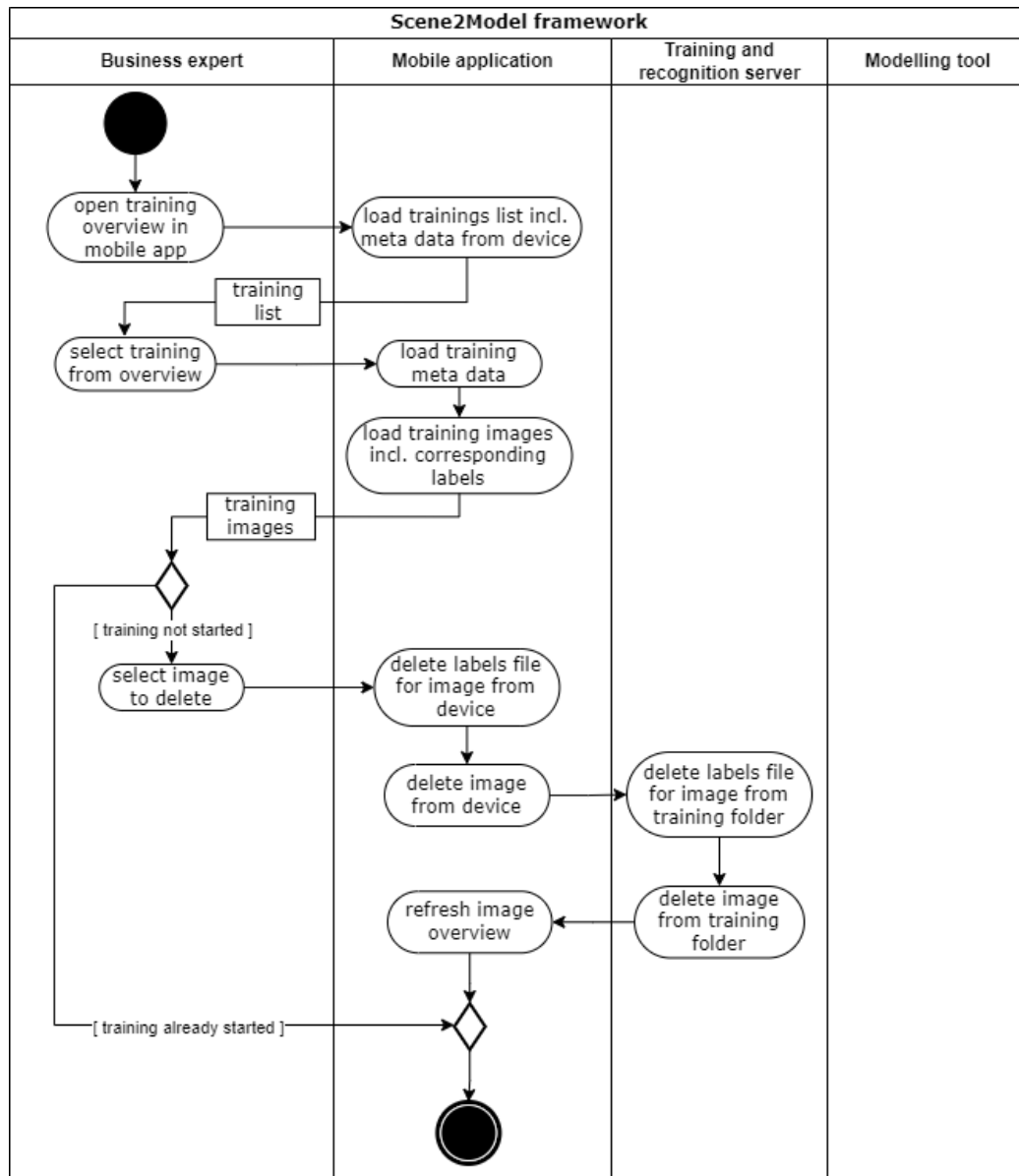


Figure 60: Activity diagram - delete image from training

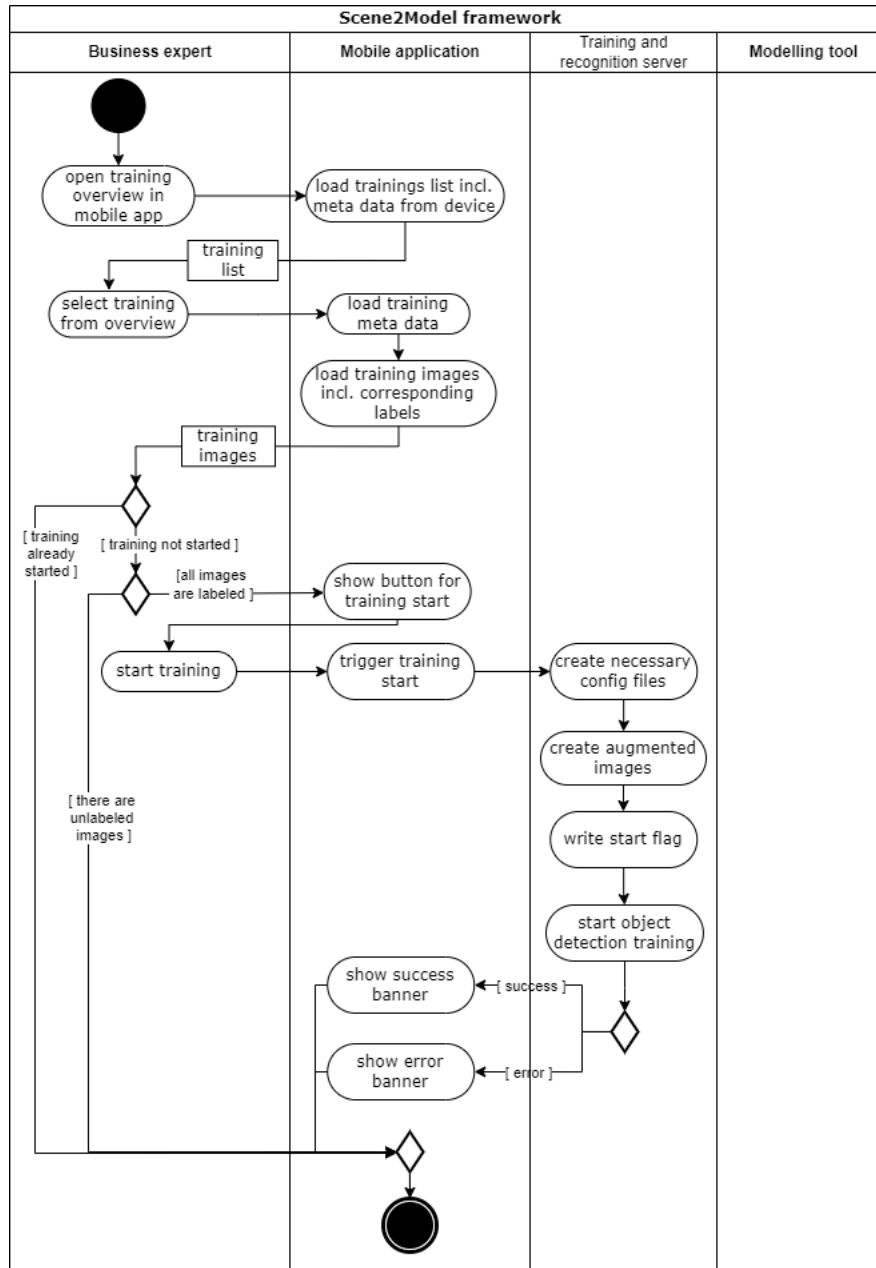


Figure 61: Activity diagram - start training

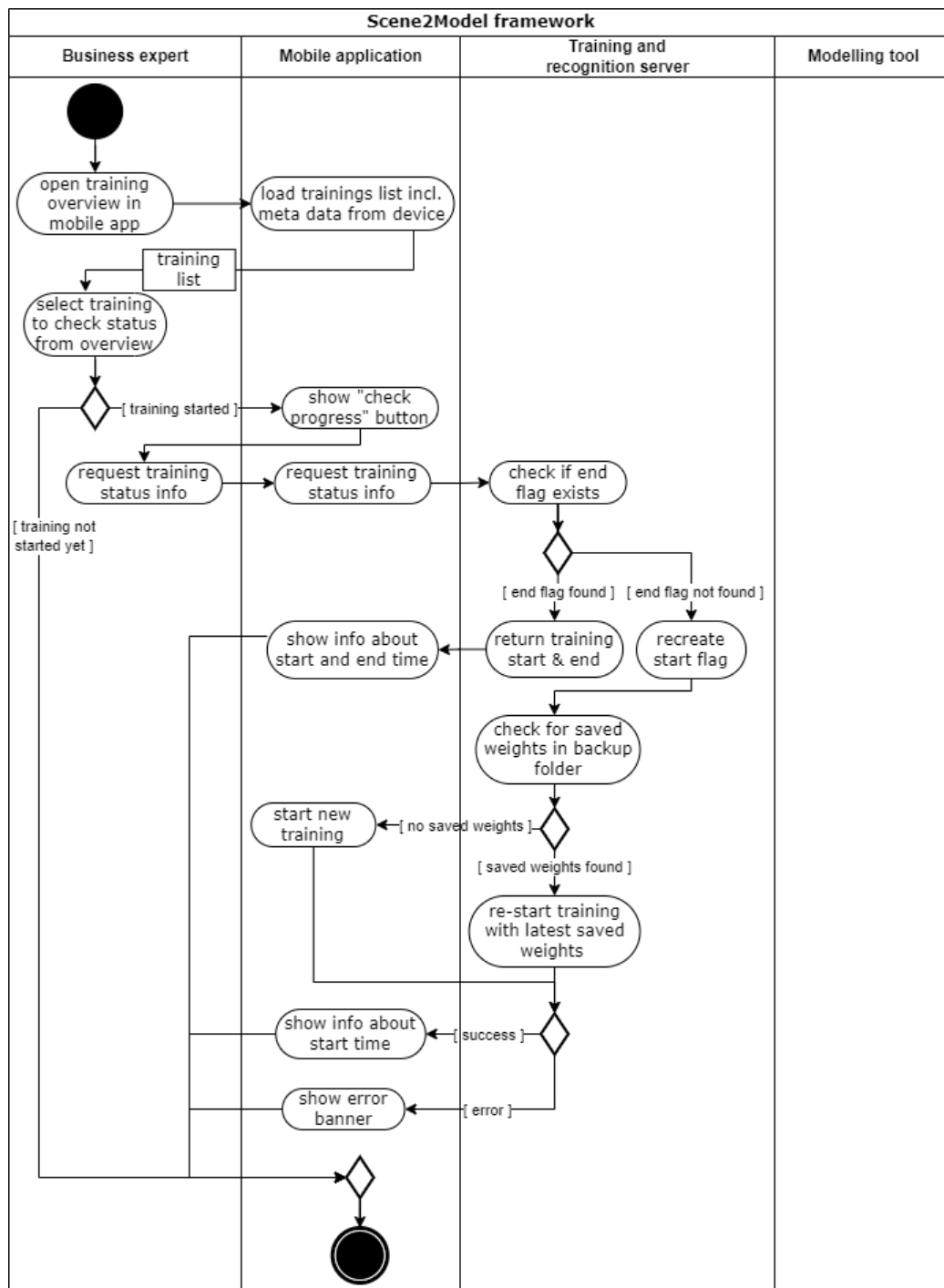


Figure 62: Activity diagram - check training status

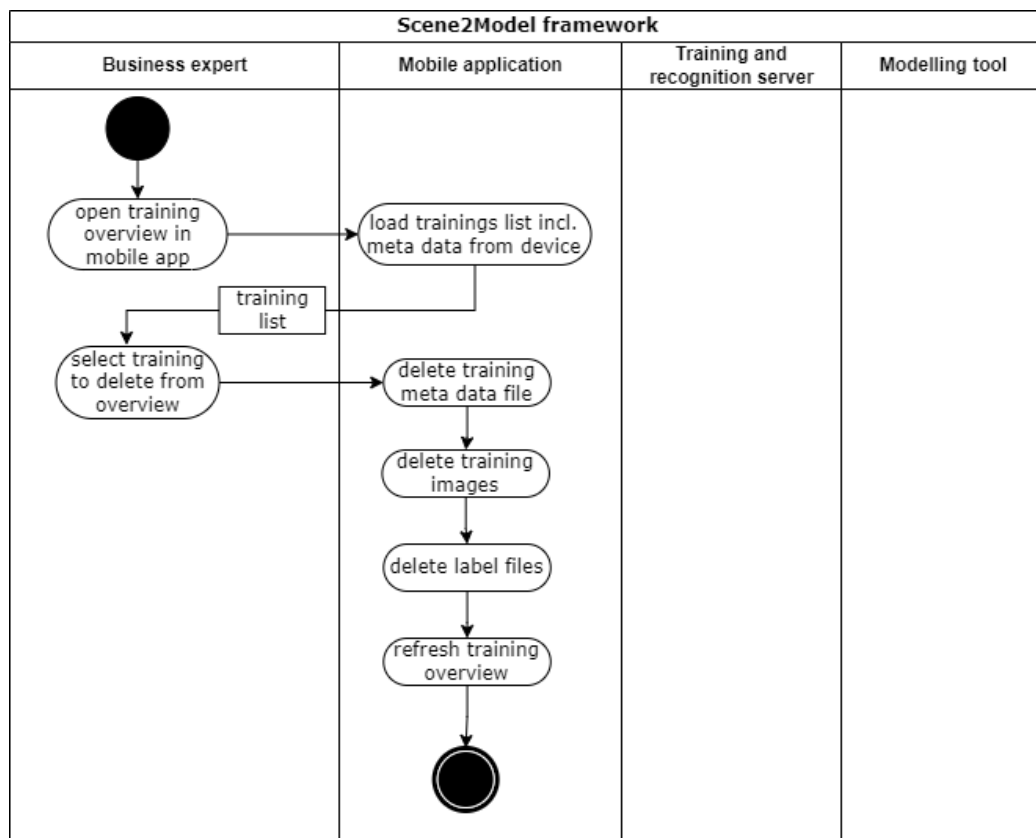


Figure 63: Activity diagram - delete training

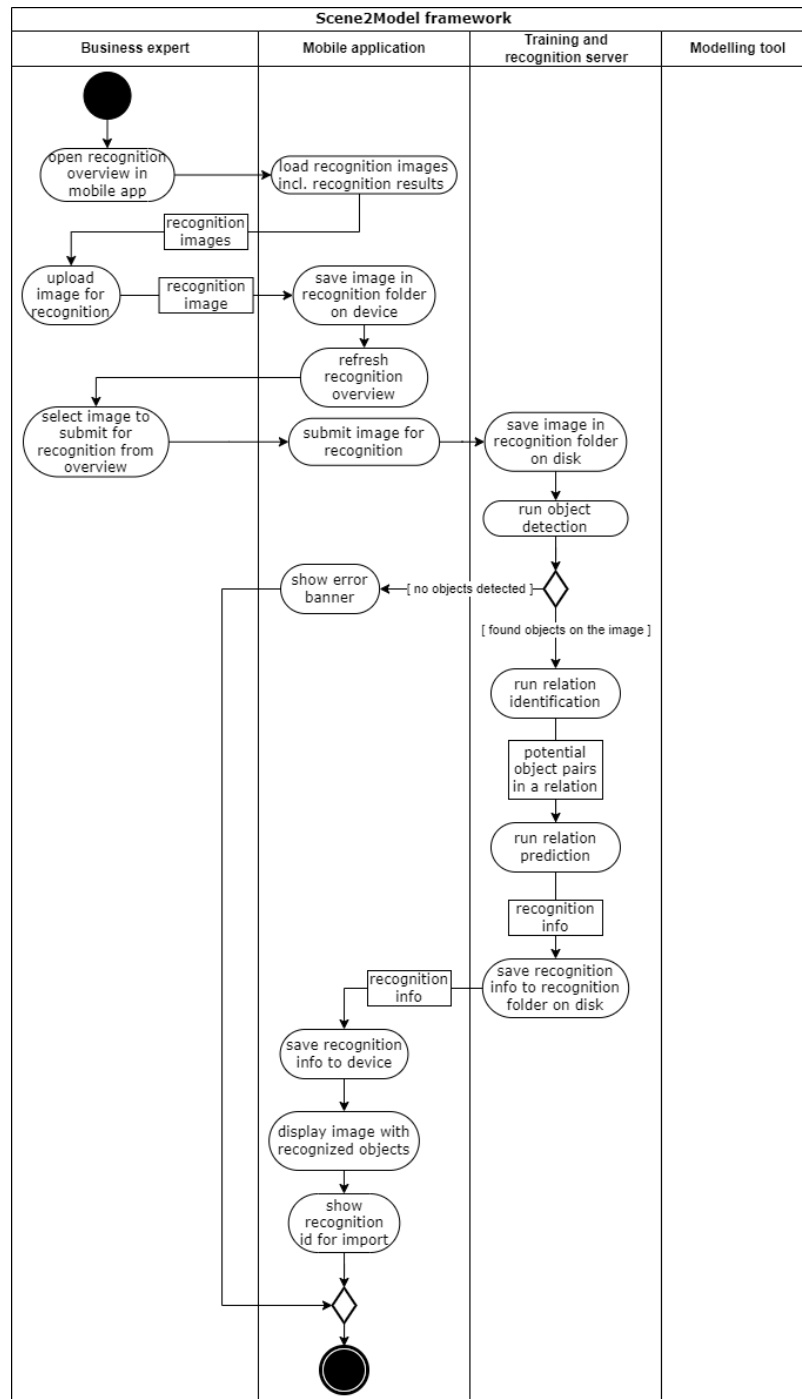


Figure 64: Activity diagram - submit image for recognition

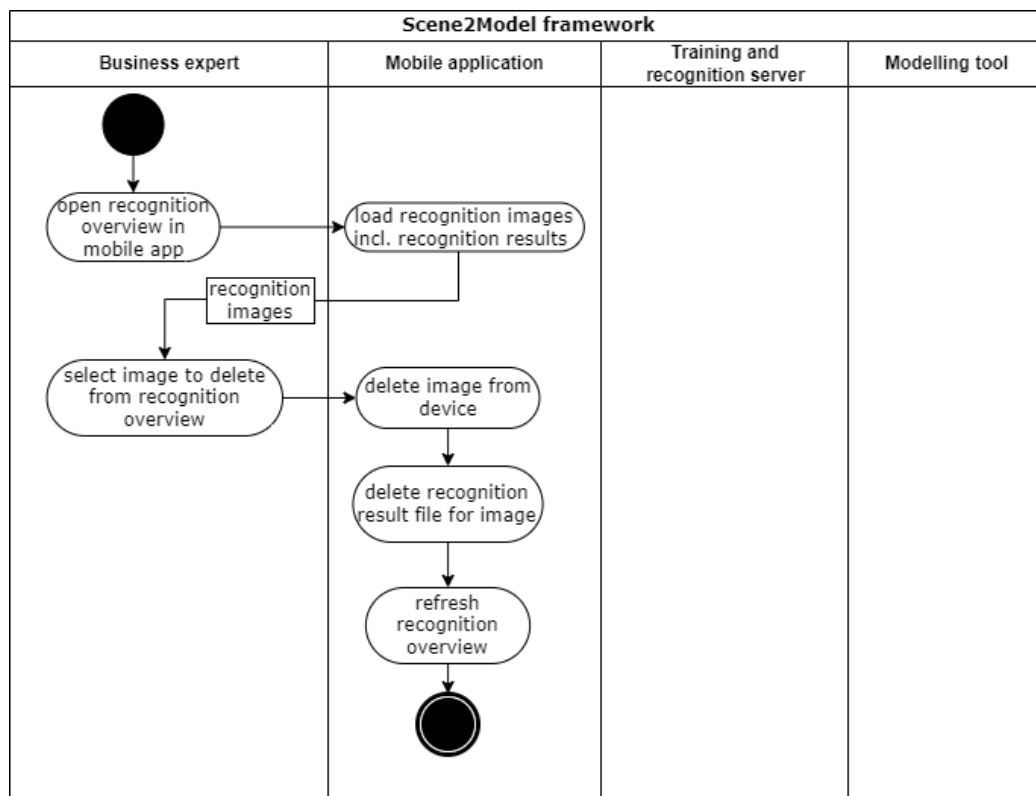


Figure 65: Activity diagram - delete image from recognition

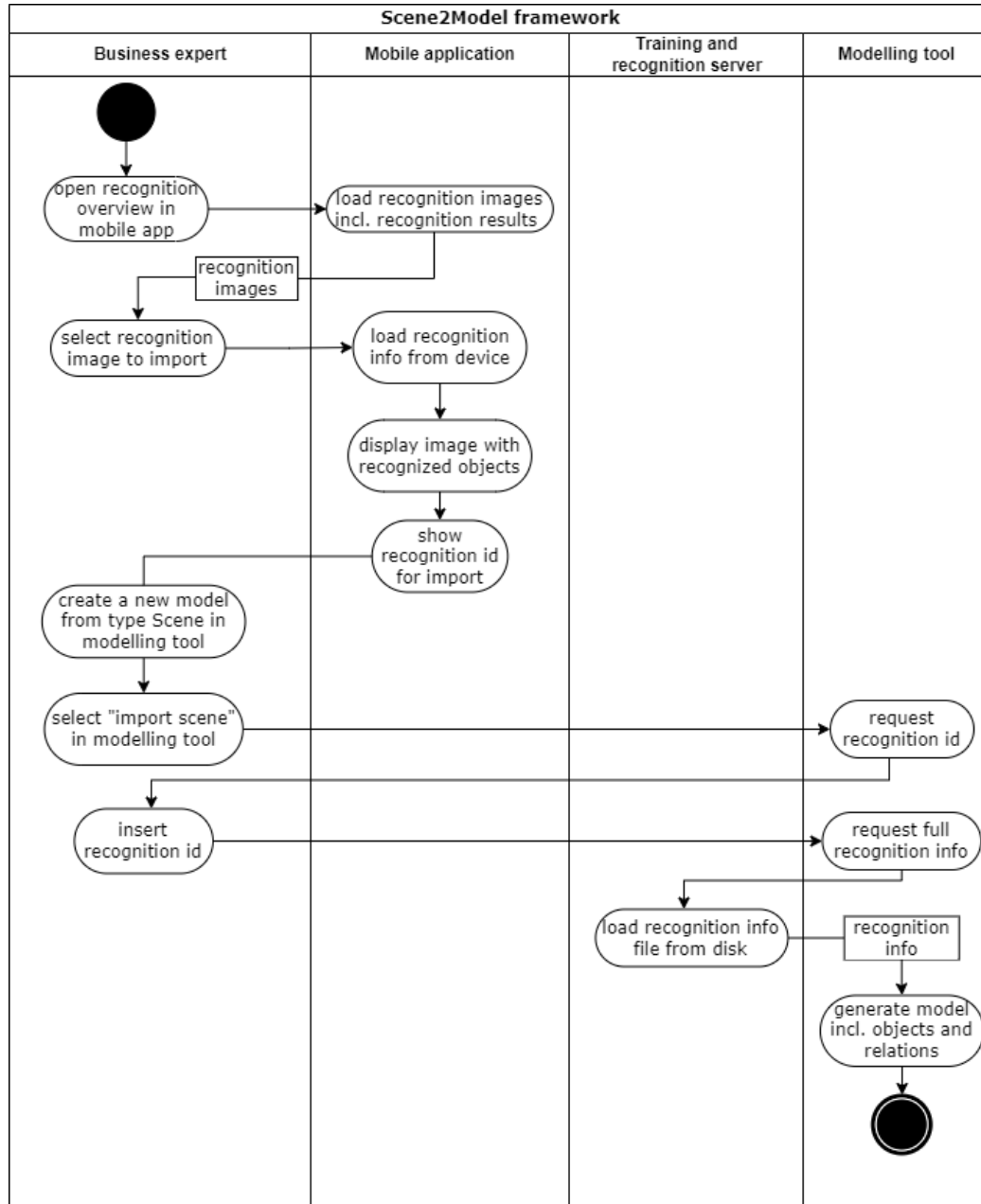
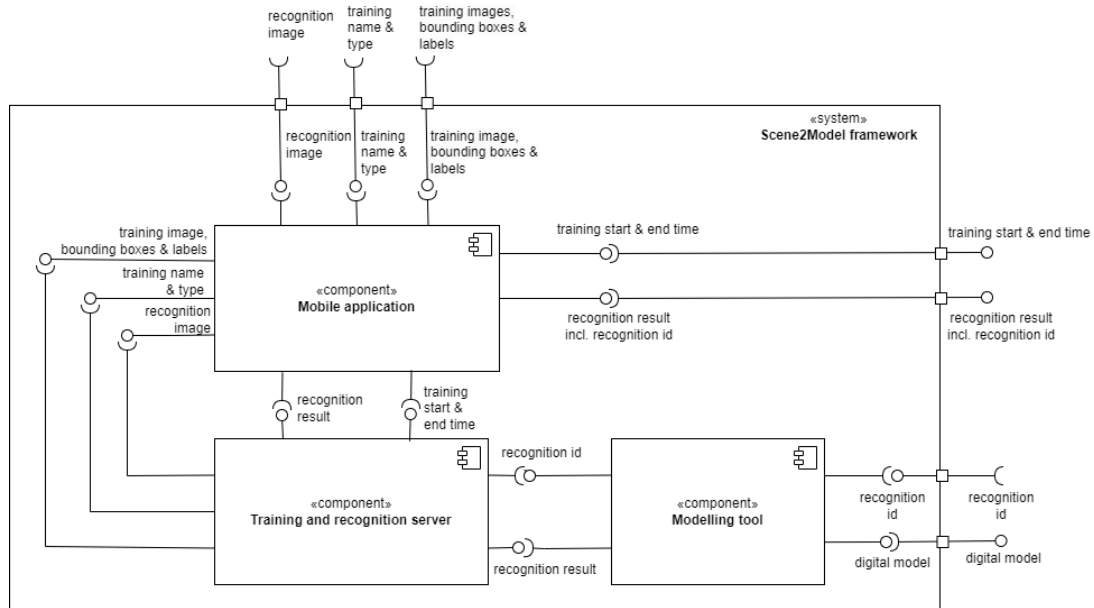


Figure 66: Activity diagram - import scene into modelling tool

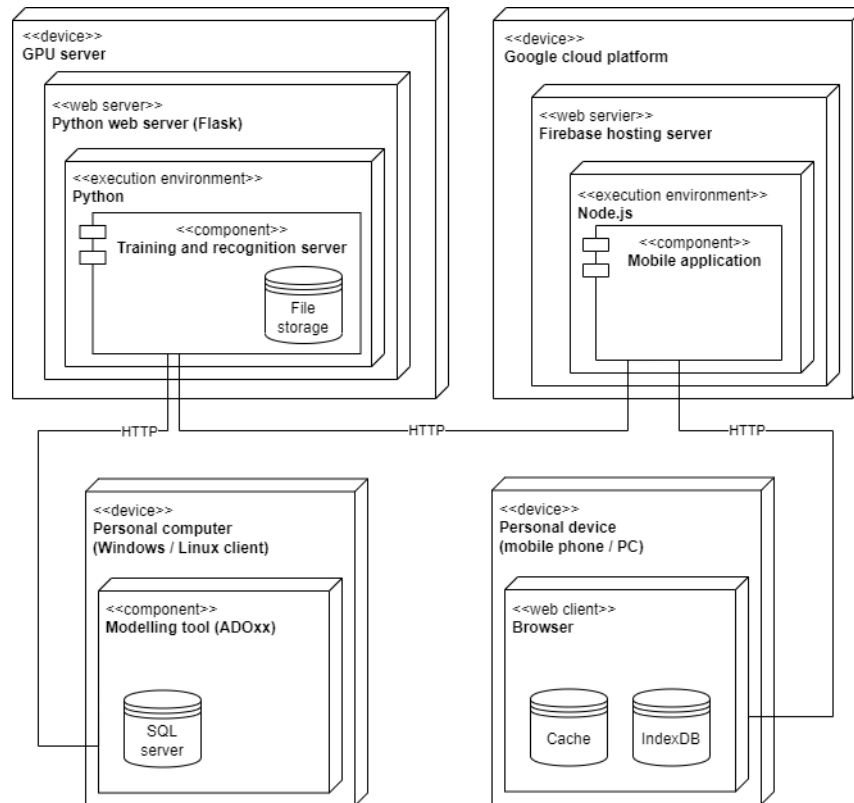
5.1.3 Component diagram

A component diagram is a type of UML diagram [64] that illustrates the organization and dependencies of software components within a system. It represents the physical or logical structure of a system, showcasing the various components and their relationships. Components in the diagram represent individual software modules, libraries, executables, or any other units of code. Component diagrams are helpful in understanding the modular architecture of a system, its dependencies, and the interfaces through which components interact with each other. They aid in system design, analysis, and implementation. [64]



5.1.4 Deployment diagram

A deployment diagram is a type of UML diagram [64] that depicts the physical deployment of software artifacts (such as modules, components, or executables) on hardware nodes within a system. It illustrates the hardware infrastructure on which the software system is deployed, including servers, computers, devices, and network connections. Deployment diagrams show the mapping between software components and the hardware nodes they run on, as well as the communication paths between them. They help visualize the deployment configuration and aid in understanding system scalability, reliability, and performance considerations. [64]



5.2 Design thinking setup

The initial phase of an innovation process requires available knowledge about the domain to be made explicit and structured in a practical abstract representation, containing only the fundamentals aspects of the domain elements. [65] The instrument of choice for extracting insights about the innovation target - some area of life or a business process with a suboptimal setup, is a design thinking workshop, which serves as a creative way to encourage collaborative problem solving and address existing pain points or potential business improvements. [66] The aforementioned innovation think tanks can physically take place anywhere and have the form of an open discussion. To document the emerging ideas, the concepts and the connections between them should be visually represented on a white canvas with the help of paper figurines from a professional design thinking toolkit called Scenes, developed by SAP AppHaus [24], or if some elements are missing in the standard sets, custom images can be printed out and used in addition.

The Scenes general catalogue includes objects from categories such as characters, buildings, vehicles, devices and furniture. (Fig. 76)

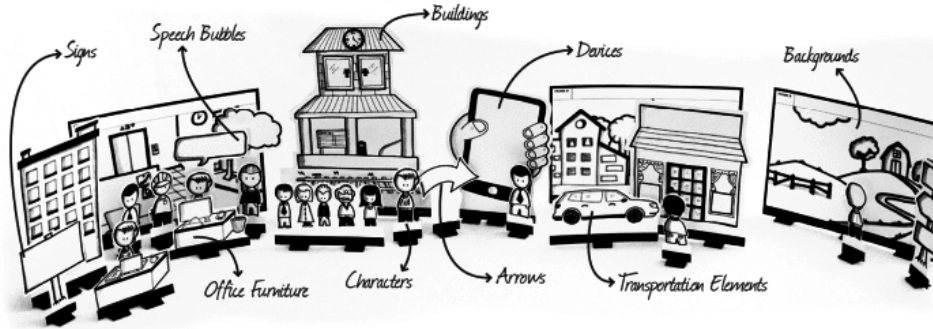


Figure 67: Scenes - the design thinking toolkit by SAP AppHaus [24]

When it comes to rules on how to order the objects in space, there is only one guideline and namely, that related items should be placed close to each other or may even overlap, if this makes sense. We will refer to the resulting configuration as a physical scene. At the end of the design thinking session, when the setup is final, the scene, representing the problem spaces, is photographed and uploaded to the mobile app, described in the next section. An example can be seen on Fig. 68

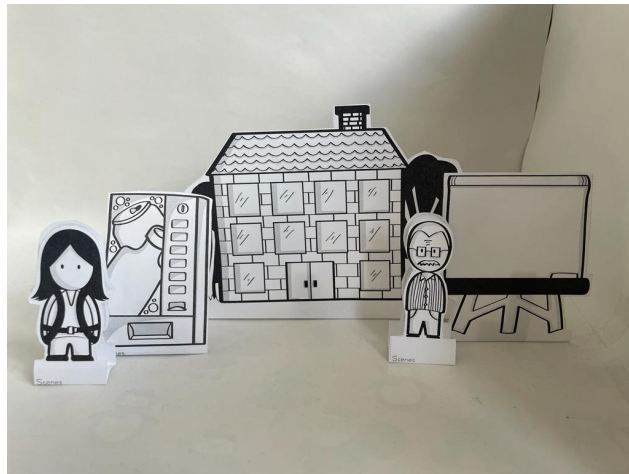


Figure 68: An example physical scene - the result of a design thinking session

5.3 Hardware and software prerequisites

In this section, we will go through the hardware and software requirements, that should be met for the implementation and usage of the Scene2Model framework.

5.3.1 Object detection & visual relationship identification

The object detection module is based on a convolutional neural network (CNN), the training of which is computationally expensive, which means that a powerful machine with a GPU chip is advisable. The CNN trainings have been performed on the supercomputer, physically located in OMiLAB in Vienna [1], and take between 30 minutes and an ten to twelve hours approximately. Apart from the hardware capacity the runtime strongly depends on the CNN type of choice (whether the full YOLOv4 [20] is used or the tiny version) and the number of different object classes.

The visual relationship prediction tool learns a simple MLP (multi-layer perceptron) [67] or a non-convolutional neural network, for every possible object class and relation type. This leads to requiring high computing power and preferably a GPU similarly to the object detection module. The training time is again proportional to the domain size (number of object classes and relation types), but this time it runs in the range of seconds (around 70 seconds for 28 object classes and 16 relations).

Both object detection and visual relationship prediction training have been performed on a device with following specifications:

- CPU: AMD Ryzen Threadripper 1920X, 12 cores, 3.5 GHz to 4.0 GHz
- RAM: 64 GB
- GPU: Nvidia GeForce RTX 2080 Ti (11 GB)
- SSD: 1000 GB

The object detection and visual relationship recognition software has been implemented in Python and requires following modules to be installed. The software has been tested with the specified versions, but will most probably work with newer versions as well.

-
- python==3.8
 - numpy==1.18.5
 - opencv-contrib-python==4.5.1.48
 - opencv-python==4.4.0.46
 - Pillow==7.2.0
 - flask==2.0.1
 - imageio==2.9.0
 - imgaug==0.4.0
 - Flask-Cors==3.0.10
 - ltn==2.1
 - requests==2.28.1

5.3.2 Mobile application

The development of the mobile application can and has been performed on a regular Windows client with following specifications:

- CPU: Intel(R) Core(TM) i5-10310U, 4 Cores, 1.70GHz
- RAM: 16 GB
- SSD: 235.48 GB

The software prerequisites for development and deployment are following:

- NodeJS 12 or higher
- npm (the package manager for JavaScript)
- optional but advisory: Visual Studio Code as code editor
- @capacitor/core - install over npm - npm install -g @capacitor/core
- @capacitor/cli - install over npm - npm install -g @capacitor/cli

-
- @ionic/cli - install over npm - npm install -g @ionic/cli
 - firebase-tools - install over npm - npm install -g firebase-tools

The application has been created using Capacitor [68], which is a cross-platform native runtime enabling the development of hybrid web apps that run natively on iOS, Android, Desktop and more using a single standard web code base. This for a fact means, that the testing can be done both in the browser of the client, where the implementation was performed, or on an actual smartphone. As the main purpose of the mobile application is to ease the image handling for the object detection training (capturing and labeling the objects) and as a graphical user interface for uploading pictures of physical scenes (the results of a design thinking workshop) for recognition, the mobile device, on which the application is used, should have a functioning camera and some free space, as the captured images are temporarily stored on the device as well.

5.3.3 Modelling and Meta-modelling tool

ADOxx [69] is the software of choice for both the implementation of a domain-specific modelling language on the meta-modelling level and for the creation of conceptual models on the modelling level. ADOxx can be installed on a Windows / Linux / macOS client and has following basic hardware requirements:

- Processor: Pentium 4 or equivalent processor (or higher)
- Speed: At least 1,8 GHz (higher recommended)
- RAM: At least 1 GB (more recommended)
- Hard disk capacity: Approx. 400 MB for the installation of ADOxx At least 40 MB for every new (empty) ADOxx database (depends on the DBMS) Plus approx. 400–700 MB for every 1,000 models in a database
- Screen resolution: At least 1024 x 768 (1280 x 960 or higher recommended)
- Colour depth: At least 65,536 colours / 16-bit (16 million colours / 32-bit recommended)

5.4 Mobile application

In the scope of this master thesis a cross-platform application has been implemented using Capacitor [68] and Ionic [70]. Capacitor is a free and open source platform, which enables the realisation of native iOS, Android, Desktop and Progressive Web Apps, all only using a single standard web code base and thereby ensuring compatibility with contemporary web browsers. The core functionality of Capacitor revolves around the transformation of existing web applications into native applications that can be executed on various platforms. This is achieved by establishing connections between the web application and the underlying native platform functionalities through JavaScript. On the frontend side, the UI (user interface) / UX (user experience) toolkit utilized, is Ionic, which is an open source mobile application stack, that leverages web technologies such as HTML, CSS, and JavaScript — and offers integrations for common JavaScript libraries like Angular, React, and Vue. The scene2model-helper-app uses Angular code base, as it is the framework, that has been closely tied to Ionic from the very beginning.

The scene2model-helper-app serves as a graphical interface for preparing the necessary image data set for the domain object detection training as well as for the communication with the automated training and recognition server. The mobile application facilitates the transfer of the labeled pictures, required for training, to the server over the exposed REST API (application programming interface) and also provides a front end for labeling the contained objects on each image. A detailed overview on how to use the mobile application as an end user can be found in Appendix A.

It is important to note, that the application is session based, which means that one can only see his own trainings and can only use those for recognition. This means that user registration is not required to use the application, but rather a new session is built based on the client (browser, device etc.) used for accessing the app. The consequence is, that if you use the same device, but a different browser or open an incognito window, you will be treated as a completely different user and will not be able to use your already trained objects for recognition.

All information about the trainings (name, type (Yolov4 or Yolo4-tiny) etc.) as well as all uploaded or instantly captured images together with their labels and recognition information, returned from the server, is persistently stored as JSON files (all meta-data) or JPEG files (the images) on the client (mobile phone, browser etc.) in the default temporary location. On a mobile

device, if you uninstall the application, you will be asked, if you would like to keep the data or delete it. Same goes for the browser - all temporary files are saved inside an IndexedDB file storage, which can be cleared from the browser developer tools.

As already mentioned, Capacitor enables the development of web-based applications that run equally well across iOS, Android, Desktop and as Progressive Web Apps. Although the scene2model-helper-app has been built and tested for native iOS and installed on a physical iPhone 12, we have acknowledged the advantages of Progressive Web Apps (PWA) and decided to provide it primarily as a PWA application.

Progressive Web Apps (PWA) [71] represent a significant advancement in web application development, merging the best attributes of both web and native applications. These applications utilize modern web technologies, including HTML, CSS, and JavaScript, to provide users with an enhanced and immersive experience similar to that of traditional native apps. Progressive Web Apps offer a compelling alternative to traditional web and native applications, providing enhanced user experiences, offline functionality, push notifications, installation capabilities, cross-device compatibility, and robust security measures. These features make PWAs an increasingly popular choice for developers seeking to deliver highly engaging and accessible applications to a broad range of users. [71]

The scene2model-helper-app has been physically deployed as Progressive Web App (PWA) on Firebase [72], which is a comprehensive mobile and web application development platform made available by Google, that provides a wide range of services spanning various areas, including back end infrastructure, database management, authentication, storage, hosting, analytics, and more. Hosting a Progressive Web App on Firebase brings benefits such as simplified deployment, scalability, HTTPS security, continuous deployment, custom domains, analytics and monitoring, integration with Firebase services, and cost-effectiveness. [73]

The deployment of the scene2model-helper-app is super easy and straightforward. The first step is to build the app for production using "ionic build -prod" command on the Windows cmd inside the root folder of the project, which updates the web directory. The second and last step is to push the changes to Firebase using "firebase deploy". After executing those two commands, the mobile app should live, running the newly deployed version.

5.5 Object detection using YOLOv4 [20]

The object detection module of the Scene2Model framework has been implemented using Python and is based on YOLOv4 - a popular object detection model, consisting of a single convolutional neural network, known for its speed and accuracy. [20] YOLO is renowned for its ability to perform real-time object detection without compromising on precision, which is achieved by employing a single neural network to directly predict bounding boxes and class probabilities for multiple objects at once in an image. [5]

Aside from the full YOLOv4 model, we have also incorporated a more lightweight option of the same model and namely YOLOv4-tiny, which has a more minimalistic architecture and reduced model size compared to YOLOv4, leading to superior speed and lower computational requirements. [74] Additionally, YOLOv4-tiny exhibits faster training times, making it advantageous when working with large datasets or when time is of the essence. (If a YOLOv4 training takes 12 hours, a YOLOv4-tiny training will be completed in about an hour) While it may sacrifice some accuracy compared to larger models, YOLOv4-tiny excels in detecting larger objects, making it suitable for applications where detecting prominent objects is the primary objective. [74] That is exactly why we advise on using sharp images with few details (e.g. black and white icons) for the choice of paper figurines, that represent the domain objects, and use a contrasting canvas as background, so that the items are easily distinguishable.

Now that we have briefly discussed the underlying technology, next we will address the concrete configuration we are using for training and testing the object detection model. For the training we are exploiting pre-trained weights, which are weights learned from training on a large dataset. Using pre-trained weights brings several advantages including faster convergence during training, reduced computational requirements, and improved detection performance. Pre-trained weights provide a starting point with learned feature representations that can be fine-tuned on a specific dataset, enabling efficient training on limited resources while still achieving accurate object detection. By leveraging pre-trained weights, our goal is to save time and effort in training from scratch while benefiting from the knowledge acquired from larger datasets.

Another noteworthy aspect is the adoption of image augmentation techniques during training. Using augmentation during training provides numerous benefits such as improved model generalization, enhanced robustness

to variations in the input data, and prevention of overfitting. Augmentation methods like random cropping, scaling, rotation, and color jittering introduce variations to the training data, expanding its diversity and helping the model learn to handle different object sizes, orientations, and lighting conditions. This helps the object detection model to generalize better to unseen data and improves its ability to detect objects accurately in real-world scenarios.

One more strategy for optimizing the efficiency of training we have used is to introduce the same image multiple times in different resolutions. This allows the model to adapt to various input image sizes commonly encountered in real-world applications, ensuring robust performance across different dimensions. Additionally, training YOLOv4 on multiple resolutions enables the model to capture and learn features at different scales, enhancing its ability to detect objects of varying sizes. Moreover, training on different resolutions can improve the model’s efficiency, as it can be deployed at the appropriate resolution for specific deployment scenarios, optimizing inference speed and resource utilization.

Choosing the right batch size is the last interesting invariant, which affects the training speed, computational resource usage and recognition performance. There is no one-size-fits-all, but a smaller batch size can introduce more variability and noise into the gradient estimates, which might result in slower convergence or less stable training. On the other hand, using a larger batch size can smooth out the gradient estimates but may require longer training time. If the dataset is large, using larger batch sizes can speed up the training process. Smaller datasets might benefit from smaller batch sizes to prevent overfitting and encourage better generalization. We have experimented with different batch sizes and monitored the training progress, validation performance, and memory usage and determined 64 as the optimal batch size for our general setup, which keeps the right balance between computational efficiency (a larger batch size can lead to better GPU utilization and more efficient training, as it allows for parallel processing and faster computations) and training stability. We are always performing a maximum 6000 iterations (weight updates) during training, as it has empirically shown to be a suitable number for our task. Generally, the rule of thumb is to either take the number of classes multiplied by 2000 or a minimum of 6000, which is what we have decided on in order to avoid an exploding training duration with a larger number of object classes. Training YOLOv4-tiny for 6000 iterations takes about 1 hour and 12 hours for the full version.

The object detection module is configured and used in a completely au-

tomated fashion. We have implemented a REST interface in Python, that creates all the necessary files and structures on the GPU machine on demand. The object detection training requires multiple images of each object from different perspectives, preferably having good image quality and on a simple and contrasting background without any distraction objects. The pictures can be captured and labeled through the scene2model-helper-app application. (More information on how to use the mobile application as an end user can be found in Appendix A.) The mobile application is responsible for preparing the labels and bounding boxes in the required YOLO format and transferring the images and labels to the server. (In YOLO a bounding box is represented by four values [x_center, y_center, width, height]. x_center and y_center are the normalized coordinates of the center of the bounding box. The width and height values are also normalized.)

Once the end user has uploaded all training images and labeled all objects on them, the training start can be triggered from the mobile app. After receiving this signal, on the server all configuration files are prepared as necessary and image augmentation is applied on the pictures to increase the diversity and variability of the training data. As already stated, by applying various transformations such as flipping, scaling, rotation, and brightness adjustment, augmentation helps improve the model's ability to generalize to different object appearances, orientations, and lighting conditions, resulting in a more robust and accurate object detection performance.

Subsequently the actual object detection training is started. The training status is documented using a start and end flag on the file system on the server. During the training the mAP (mean average precision) of the model is monitored as it provides a comprehensive evaluation of the model's object detection performance across different classes and varying levels of object detection confidence thresholds. mAP takes into account both precision and recall, providing a single metric that quantifies the model's accuracy in detecting and localizing objects. Monitoring mAP has helped with identifying areas of improvement during the training and development process. If the general suggestions about good image quality, sufficient data (photographs from different angles of each object plus images portraying objects in pairs) and good contrast between the foreground objects and background are followed, a mAP of 100% can be easily reached withing the 6000 iterations. (Fig. 69)

During the training the weights are intermediately saved every 1000 iterations and at the end of the training the best weights are identified and will be

used in the following for recognition. When the training is finished, the object detector can be already used in a live setting. The end user just needs to take a picture of the already arranged scene, preferably again using the same white canvas as a background, and submit it via the scene2model-helper-app to the server. This will trigger the execution of the object detection and relation prediction module, which takes about 60 to 70 seconds.

Once the recognition is successfully finished, a banner, showing the recognition id for importing the scene into ADOxx, will appear on the screen. After dismissing the overlay, the initial image of the scene will show and the recognized objects will be marked with a bounding box, a class label and confidence score. Objects having a confidence score of less than 50% will be ignored for the relation detection, as these are more likely to be incorrect or have low certainty. Setting a confidence threshold ensures that only detections with higher confidence levels are considered, improving the overall precision and reliability of the prediction system.

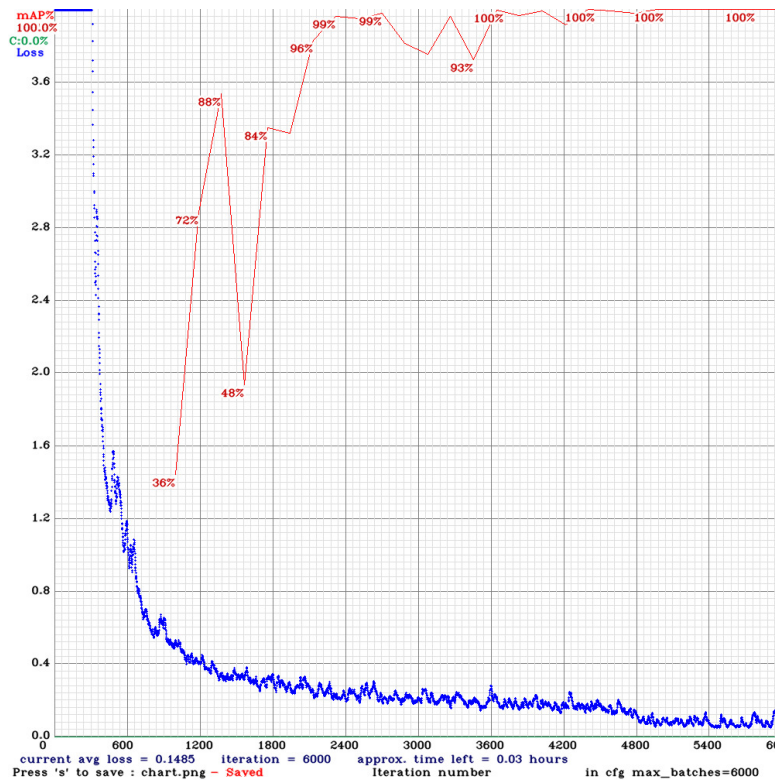


Figure 69: Example: Object detection training chart

5.6 Relation prediction using logical tensor networks

The relationship prediction module, implemented in the scope of this research, consists of two parts: (1) a relation identification software, that is responsible for determining whether a relationship persists between the different pairs of objects given some visual and spatial cues and (2) a prediction module, which picks the most appropriate out of all possible relations or denies that the two objects are connected, based on the suggestions of the identification module and a set of logical constraints.

The relation identification module takes as input the list of objects and their relative coordinates, returned from the object detection, and calculates for every possible object pair the ratio of the area of the intersection rectangle to the area of each object and to the joint area. If there is an overlap of over 1%, the object pair is considered as a potential relation and will be tested in the following by the prediction module, if it fulfills the logical constraints of the domain. We are also assigning close by but not overlapping object pairs with a high probability to be related. The criterion for this is again the overlap but this time between the enlarged by 20% bounding boxes of the objects in question.

We regard the proximity of objects as a strong indicator for a high likelihood for the objects to be related in some way. When two objects are closely located to each other in an image, it is likely that they are connected due to the principles of spatial coherence, semantic relationships, and visual grouping. The closeness of objects suggests that they share a common context or belong to the same cluster, what is also referred to as spatial coherence. Objects that are physically close often have functional or semantic connections, such as a knife next to a plate or a person holding an umbrella. Moreover, the human visual system tends to group objects that are close together based on their visual similarity, leading to the perception of connectedness. Additionally, occlusion, where one object partially hides another, further supports the notion of a connection between closely located objects. Considering these factors, the likelihood of two objects being connected increases when they are closely located in an image. The other way round, it is the workshop participants, that arrange the objects in the scene, so unsurprisingly based on the same rationale, they are also more likely and advised to place related objects near each other.

In the next step, the result of the relationship identification module, which consists of a list of object pairs, that are highly likely to be connected, to-

gether with the object detection output is passed to the relation prediction software to process. The relationship recognition module determines whether two objects can be connected based on a predefined set of facts and rules (a knowledge base), that apply to the domain, and if more than one relation is viable - which is the most suitable one.

The domain knowledge base is defined by the business experts and built using logical tensor networks (LTN), which is a neurosymbolic framework that supports querying, learning and reasoning with tensor-based representations of real world concepts. The knowledge base typically consists of constants (the object entities), predicates (the object classes and the possible relations) and logical expressions (statements or formulas in predicate logic, which consist of variables or the elements of interest, logical operators such as AND, OR, and NOT, that combine or modify the variables, and quantifiers like "for all" or "there exists", used for specifying the scope of the variables.)

Logical tensor networks (LTN) aim to bridge the gap between symbolic reasoning and deep learning by integrating logical reasoning into neural network architectures. LTN represent logical statements using tensors, where each tensor element corresponds to a truth value of a logical atom or a higher-level logical concept. By employing tensor operations, LTN can perform logical operations such as conjunction, disjunction, and implication.

LTN enable the learning and inference of logical rules by optimizing the neural network parameters based on training data and logical constraints. They can handle uncertain or incomplete information by assigning continuous truth values to logical propositions, allowing for probabilistic reasoning. LTN also provide mechanisms for logical constraint satisfaction, where the model is trained to satisfy logical formulas or perform logical entailment.

One of the key benefits of LTN is their ability to combine logical reasoning with the expressive power of deep neural networks, enabling the integration of logical constraints and domain knowledge into neural models. This makes LTN suitable for tasks like ours that require logical reasoning, such as natural language understanding, knowledge representation, and decision-making.

In the relation prediction module we take advantage of the LTN's capability to learn language symbols (predicates, functions, constants) using the satisfaction of a knowledge base as an objective for training. As already mentioned, a domain knowledge base consists of facts and logical rules or constraints that encode domain-specific expertise, and in the scope of the Scene2Model framework always contains following elements:

-
- LTN constants - the object recognized on an image, are represented as learnable LTN constants or embeddings, which are vector representations, that capture their specific semantic and contextual information. These embeddings encode the features and characteristics of the objects in question in a continuous vector space.
 - Unary LTN predicates - the membership of an object into a certain object class is denoted as a unary LTN predicate (one input value - the object embedding or the learnable LTN constant), and is grounded in a mapping that assigns a value between 0 and 1. Predicates in LTN can be neural networks or any other function that achieves such a mapping. We have decided to use a Multilayer Perceptron (MLP) for the LTN predicates, as MLP are capable of learning complex mappings between input features and output labels, enabling the LTN predicate to capture intricate relationships and patterns in the data. Additionally, MLP provide a well-established and widely used framework for training and optimization. With backpropagation and gradient-based optimization techniques, MLP can learn the parameters of the LTN predicate efficiently, enabling the network to adapt and improve its performance over time.
 - Binary LTN predicates - following the same logic, the relationships are expressed as binary LTN predicates with MLP as a mapping function. The relationship predicates have two input parameters - a pair of objects (their learnable LTN constants).
 - LTN variables - the sequence of all objects, contained in an recognition image (their learnable LTN constants) is stored into LTN variables. Logical tensor variables are symbolic placeholders within a logical tensor network that represent logical concepts and are used to encode and reason about the truth values and relationships between logical statements in a continuous tensor space. If we want to reason about all objects in a picture, we need to create a variable, containing all recognized entities.
 - LTN formulas - the known facts and logical constraints about the domain are represented as logical expressions, consisting of LTN variables or the elements of interest, connected by logical operators such as AND, OR, and NOT, and with a scope, specified by quantifiers like "for all"
-

or "there exists". LTN formulas provide a way to capture and process logical relationships within a continuous space.

To better illustrate how a domain knowledge base is built using LTN constructs, we will go through a minimalistic example. Imagine we have trained the object detection module to recognize following classes: business man, desk, office building and car. In the following a design thinking workshop has taken place and the domain experts have arranged following scene and submitted it for recognition. The object detection module has recognized the four objects on the image and has assigned them with a bounding box, tightly covering the object, a class label and a confidence score. (Fig. 70) The object detection result is passed to the relation identification module, which calculates the area of intersection between the bounding boxes (original and enlarged by 20%) of every possible pair of objects. The relation identification software has found 5 potential relationship candidates: [office_building, business_man], [office_building, desk], [office_building, car], [business_man, desk] and [desk, car]. (Fig. 71)

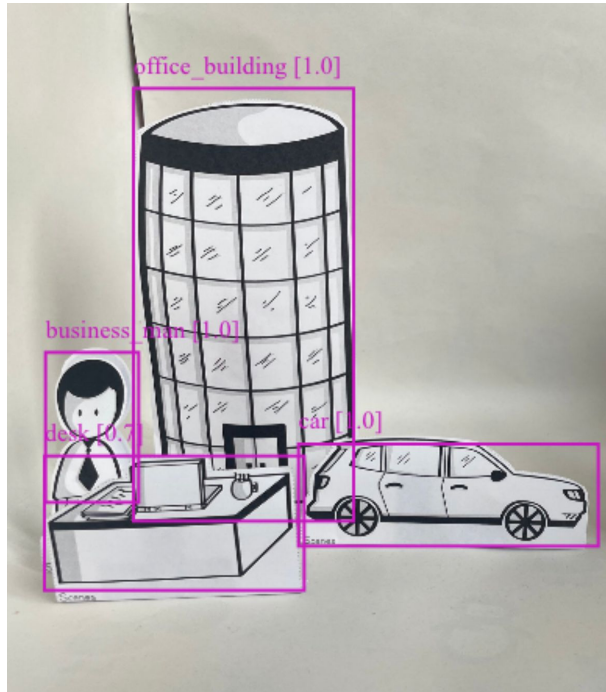


Figure 70: Example: Object detection result

```

+++ Relation detected. [1: office_building, 2: business_man]
- Original - Joint intersection ratio: 1%
- Enlarged - Joint intersection ratio: 1%
+++ Relation detected. [1: office_building, 2: desk]
- Original - Joint intersection ratio: 9%
- Enlarged - Joint intersection ratio: 34%
+++ Relation detected. [1: office_building, 2: car]
- Original - Joint intersection ratio: 4%
- Enlarged - Joint intersection ratio: 7%
+++ Relation detected. [1: business_man, 2: desk]
- Original - Joint intersection ratio: 10%
- Enlarged - Joint intersection ratio: 22%
-- Relation NOT detected. [1: business_man, 2: car]
- Original - Joint intersection ratio: 0%
- Enlarged - Joint intersection ratio: 0%
+++ Relation detected. [1: desk, 2: car]
- Original - Joint intersection ratio: 1%
- Enlarged - Joint intersection ratio: 4%

```

Figure 71: Example: Relation identification result

Next, the object detection result, containing all objects with their labels, bounding boxes and confidence scores and the relation identification result, consisting of the relationship suggestions are passed to the relationship prediction module. At the preparation phase of the workshop, the business experts have been asked general questions about the domain with the aim to create a formal representation of this area in life by outlining what object classes there are, what the existing relationships are and to identify any logical constraints and rules, that apply. The outcome in natural language:

```

There are 4 object classes: business_man, desk, office_building, car
There are 5 possible relationships: workplace, use_for_work, drive, part_of, parked
There are 4 abstract classes: _person, _furniture, _building, _vehicle

business_man is a _person
desk is _furniture item
office_building is a _building
car is a _vehicle

Furniture is usually part of buildings.
People use furniture for work.
People can drive cars.
Cars are usually parked in front of buildings.
Office buildings are the workplace of people.
Any other combination of objects is not foreseen.

```

Figure 72: Example: Domain knowledge in natural language

Out of all gathered information the domain knowledge base can be built using logical tensor network constructs as follows.

First the four objects, that have been recognized on the image, are initialized as LTN constant, holding random vectors in continuous space, that contain values between 0 and 1, also called learnable variables or embeddings.

```
1 ### LTN constants - the recognized objects
2 obj_0_office_building = ltn.Constant(np.random.uniform(low=0.0,high=1.0,size=embedding_size),trainable=True)
3 obj_1_business_man = ltn.Constant(np.random.uniform(low=0.0,high=1.0,size=embedding_size),trainable=True)
4 obj_2_desk = ltn.Constant(np.random.uniform(low=0.0,high=1.0,size=embedding_size),trainable=True)
5 obj_3_car = ltn.Constant(np.random.uniform(low=0.0,high=1.0,size=embedding_size),trainable=True)
```

Then the object classes and abstract classes are introduced as unary LTN predicates with MLP as a mapping function.

```
7 ### LTN unary predicates - the object classes
8 Office_building = ltn.Predicate.MLP([embedding_size])
9 Business_man = ltn.Predicate.MLP([embedding_size])
10 Desk = ltn.Predicate.MLP([embedding_size])
11 Car = ltn.Predicate.MLP([embedding_size])
12
13 _Person = ltn.Predicate.MLP([embedding_size])
14 _Furniture = ltn.Predicate.MLP([embedding_size])
15 _Building = ltn.Predicate.MLP([embedding_size])
16 _Vehicle = ltn.Predicate.MLP([embedding_size])
```

The known relationship types plus a general abstract relationship are defined by binary LTN predicates again using MLP for mapping the input embeddings into a single value between zero and one:

```
18 ### LTN binary predicates - the possible relationships
19 Workplace = ltn.Predicate.MLP([embedding_size,embedding_size])
20 Use_for_work = ltn.Predicate.MLP([embedding_size,embedding_size])
21 Drive = ltn.Predicate.MLP([embedding_size,embedding_size])
22 Part_of = ltn.Predicate.MLP([embedding_size,embedding_size])
23 Parked = ltn.Predicate.MLP([embedding_size,embedding_size])
24
25 _Relation = ltn.Predicate.MLP([embedding_size,embedding_size])
```

The last element, that should be created, before starting with the actual knowledge base definition, are the variables, containing all objects instances, so that we could reason over all found objects. (We need at least two variables to be able to reason over the binary predicates, but the exact number depends on the complexity of the logical rules.)

```
27 ### LTN variables - two variables containing all objects
28 p = ltn.Variable.from_constants("p", [obj_0_office_building, obj_1_business_man, obj_2_desk, obj_3_car])
29 q = ltn.Variable.from_constants("q", [obj_0_office_building, obj_1_business_man, obj_2_desk, obj_3_car])
```

Now that all necessary LTN constructs are defined, the knowledge base, consisting of the known facts and logical constraints can be built. First, we can introduce the knowledge, coming from the object detection result (object 0 is an office building, object 1 is a business man, object 2 is a desk, object 3 is a car). Additionally, it is also clear, that the object classes are mutually

exclusive, which means that if an object is classified as a car, then it is not a desk, not a business man and not an office building.

```
31 ### LTN formulas
32 axioms = [
33     Office_building(obj_0_office_building),
34     Not(Business_man(obj_0_office_building),
35     Not(Desk(obj_0_office_building),
36     Not(Car(obj_0_office_building),
37
38     Not(Office_building(obj_1_business_man)),
39     Business_man(obj_1_business_man),
40     Not(Desk(obj_1_business_man)),
41     Not(Car(obj_1_business_man)),
42
43     Not(Office_building(obj_2_desk)),
44     Not(Business_man(obj_2_desk),
45     Desk(obj_2_desk),
46     Not(Car(obj_2_desk)),
47
48     Not(Office_building(obj_3_car)),
49     Not(Business_man(obj_3_car),
50     Not(Desk(obj_3_car)),
51     Car(obj_3_car),
```

What is also available as information are the relationship suggestions, consisting of object pairs, that are potentially connected, based on their mutual positioning. The type of relationship is still not known, therefore they will be denoted with the abstract relationship type. The abstract relation is symmetrical or bi-directional, meaning that if object A is in a relationship with object B, then it is also true, that object B is in a relationship with object A.

```
53     ## relationship suggestions
54     _Relation([obj_0_office_building,obj_1_business_man]),
55     _Relation([obj_0_office_building,obj_2_desk]),
56     _Relation([obj_0_office_building,obj_3_car]),
57     _Relation([obj_1_business_man,obj_2_desk]),
58     _Relation([obj_2_desk,obj_3_car]),
59
60     ## relation symmetry
61     Forall([p,q], Implies(_Relation([p,q]),_Relation([q,p]))),
```

From the interview with the business experts with the aim of gathering knowledge about the domain it has gotten clear that except for the concrete object classes, there are also following general classes: person, vehicle, building, furniture. Logical rules sometimes apply to a group of object classes, which has led to identifying the mentioned abstract classes. Every concrete object class is part of exactly one abstract class and abstract classes are mutually exclusive.

```

63  ## ∀p Office_building(p) → _Building(p) ∧ ¬_Person(p) ∧ ¬_Furniture(p) ∧ ¬_Vehicle(p)
64  Forall(p,
65      Implies(
66          Office_building(p),
67          And(And(_Building(p), Not(_Person)), And(Not(_Furniture), Not(_Vehicle)))
68      ),
69  ),
70  ## ∀p Business_man(p) → _Person(p) ∧ ¬_Building(p) ∧ ¬_Furniture(p) ∧ ¬_Vehicle(p)
71  Forall(p,
72      Implies(
73          Business_man(p),
74          And(And(_Person(p), Not(_Building)), And(Not(_Furniture), Not(_Vehicle)))
75      ),
76  ),
77  ## ∀p Desk(p) → _Furniture(p) ∧ ¬_Building(p) ∧ ¬_Person(p) ∧ ¬_Vehicle(p)
78  Forall(p,
79      Implies(
80          Desk(p),
81          And(And(_Furniture(p), Not(_Building)), And(Not(_Person), Not(_Vehicle)))
82      ),
83  ),
84  ## ∀p Car(p) → _Vehicle(p) ∧ ¬_Building(p) ∧ ¬_Person(p) ∧ ¬_Furniture(p)
85  Forall(p,
86      Implies(
87          Car(p),
88          And(And(_Vehicle(p), Not(_Building)), And(Not(_Person), Not(_Furniture)))
89      ),
90  ),

```

The last component of the knowledge base are the logical constraints. For completeness they always come in pairs: a positive statement (e.g. people use furniture for work) and the negated one (if object one is not a person or object two is not a furniture item, then object one does not use object two for work), to ensure that there are no unknown states. (To not over flood the research paper with code, only one expression is shown as an example. The other ones are also added to the knowledge base.) Here is a good place to note that logical tensor networks embrace fuzzy operators, which are mathematical operations that handle uncertainty or vagueness in logical reasoning. Unlike traditional crisp logic, which assigns binary truth values of either true or false, fuzzy logic allows for the representation of partial truth values, where propositions can have degrees of truthfulness ranging between 0 and 1. The key word "usually" requires exactly this type of representation.

```

93  ## logical constraints
94
95  ### furniture is usually part of buildings.
96  ## ∀p,q _Relation(p,q) ∧ _Furniture(p) ∧ _Building(q) → Part_of(p,q)
97  Forall([p,q],
98      Implies(
99          And(
100              _Relation([p,q]),
101              And(_Furniture(p), _Building(q))
102          ),
103          Part_of([p,q])
104      )
105  ),
106
107  ## ∀p,q _Relation(p,q) ∧ (¬_Furniture(p) ∨ ¬_Building(q)) → ¬Part_of(p,q)
108  Forall([p,q],
109      Implies(
110          And(
111              _Relation([p,q]),
112              Or(Not(_Furniture(p)), Or(_Not(Building(q))))
113          ),
114          Not(Part_of([p,q]))
115      )
116  ),

```

Finally, once all logical axioms are added to the knowledge base, the resulting logical tensor network can be trained for maximizing its satisfaction rate. This includes adjusting all trainable variables and mapping functions for a fixed number of 250 iterations to usually reach a satisfaction level of over 99%. When the training is finished, we could apply the relation predicates on the set of object pairs, suggested by the relationship identification module, and get a truth value between 0 and 1, denoting how likely the relation is. Similar to the object detection logic, we are ignoring values under 0.5 as these are more likely to be incorrect or have low certainty. Voila. The relation prediction module has assigned each object pair with the most suitable relationship and has determined its direction. The existence of a connection between one object pair has also been denied (car and desk).

```

Checking [0 - 1]
- TYPE: business_man - 0.99 obj2: 1
- TYPE: office_building - 0.99 obj1: 0
- ABSTRACT TYPE: _human - 1.00 obj2: 1
- ABSTRACT TYPE: _building - 0.99 obj1: 0
- RELATION: workplace - 0.98 obj1: 1, obj2: 0
{'relation': 'workplace', 'from': 1, 'to': 0}
Checking [0 - 2]
- TYPE: desk - 0.99 obj2: 2
- TYPE: office_building - 0.99 obj1: 0
- ABSTRACT TYPE: _building - 0.99 obj1: 0
- ABSTRACT TYPE: _furniture - 0.99 obj2: 2
- RELATION: part_of - 0.99 obj1: 2, obj2: 0
{'relation': 'part_of', 'from': 2, 'to': 0}
Checking [0 - 3]
- TYPE: car - 0.99 obj2: 3
- TYPE: office_building - 0.99 obj1: 0
- ABSTRACT TYPE: _building - 0.99 obj1: 0
- ABSTRACT TYPE: _vehicle - 1.00 obj2: 3
- RELATION: parked - 0.97 obj1: 3, obj2: 0
{'relation': 'parked', 'from': 3, 'to': 0}
Checking [1 - 2]
- TYPE: desk - 0.99 obj2: 2
- TYPE: business_man - 0.99 obj1: 1
- ABSTRACT TYPE: _human - 1.00 obj1: 1
- ABSTRACT TYPE: _furniture - 0.99 obj2: 2
- RELATION: use_for_work - 0.97 obj1: 1, obj2: 2
{'relation': 'use_for_work', 'from': 1, 'to': 2}
Checking [2 - 3]
- TYPE: car - 0.99 obj2: 3
- TYPE: desk - 0.99 obj1: 2
- ABSTRACT TYPE: _vehicle - 1.00 obj2: 3
- ABSTRACT TYPE: _furniture - 0.99 obj1: 2
()

```

The logical constraints do not allow a relationship between a car and a desk.

Figure 73: Example: Relation prediction result:

- relation: workplace [0.98], direction: office building $\rightarrow$ business man
- relation: part of [0.99], direction: desk $\rightarrow$ office building
- relation: parked [0.97], direction: car $\rightarrow$ office building
- relation: use for work [0.97], direction: business man $\rightarrow$ desk
- no relation between car and desk

5.7 Modelling language creation

In order to digitalize the physical scene, emerged in a design thinking setting, into a conceptual model a domain-specific modelling language has to be developed, to cover the particular area of life, its individual concepts and their interconnections. The meta-modelling tool of choice is ADOxx. For a functioning modelling language, we need to first define our classes and abstract classes in the class hierarchy in the ADOxx development toolkit. In our minimalistic example, we need a class definition for the concrete classes [office building, desk, business man, car] and for the abstract classes [building, furniture, person, vehicle]. The best option would be to dynamically create the classes from the recognition result. It is unfortunately not possible to programmatically add class definitions in ADOxx, which is why we took an alternative approach, which consists in defining a set of reusable abstract classes (e.g. person, device, building, sign etc.), based on the suggested design thinking toolkit - Scenes by SAP AppHaus [24], with a type attribute, denoting the specific object class. Furthermore, per design each concrete class should have a different graphical representation in the modelling tool, which means that one needs to first provide an image for every specific class, part of the domain of choice, in a special sync folder. The sync folder is further divided into n-folders, named after the previously mentioned abstract classes. The images of the specific classes have to be placed under the according abstract class folder. For the ease of our end users, we have prepared icons for all elements from the Scenes toolkit. It should be noted, that if the end users do not use the same name for the object classes as we do, the automatic assignment of an image would not work. (e.g. if we have added an image for the object class "car" and the end user refers to this object as "automobile") We follow the same strategy for defining the existing relationships in the domain. There is one general relation ("has relation") with a type attribute, that holds the concrete relationship type. Except for the classes and relations, we at least needed to create one new model type ("scene"), that builds upon the newly defined classes and relation. Additionally, to enable the representation of connections between different scenes (e.g. to construct a road map of intermediate steps) we have made available one more model type ("storyboard").

5.8 Conceptual model generation

Since all the modelling constructs for importing our physical scene as a digital model into ADOxx are available, there is only the logic for importing the recognition result into ADOxx left. We have implemented a script in AdoScript, that takes as input a recognition id, and makes a REST HTTP call to the automated training and recognition server to get the complete information about the scene, submitted for recognition. (Fig. 75) The script loops through the objects, found by the object detection module, while also iterating the folders corresponding to the available abstract classes in the special sync folder, to map the object to an abstract class and a specific class respectively. If the mapping is successful, an instance of the found abstract class is created with type attribute equal to the specific object class, provided as `class_name` in the recognition result, and with the graphical representation supplied for the specific class in the sync folder for each object. The object instances are placed on the modelling canvas based on the relative coordinates, contained in the returned result from the server. Finally, the predicted relationships with their assigned type and direction are imported as the general relation "has relation" to connect the corresponding object instances.

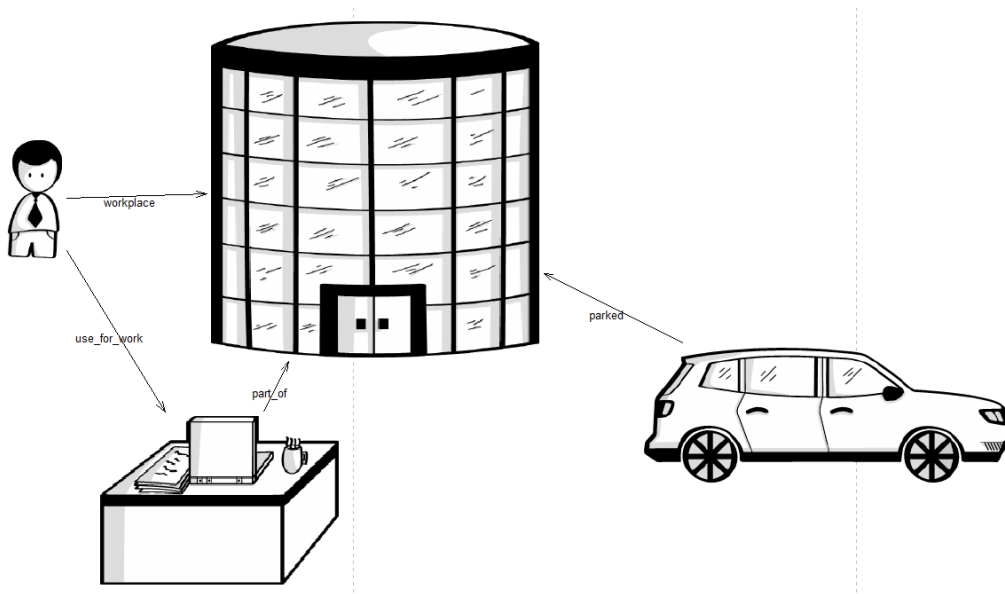


Figure 74: Example: Imported conceptual model in ADOxx

```

1 {
2   "img_path": "/home/neli/Desktop/help_resources/recognition_images/recognition_img_20230526_231614.jpg",
3   "img_width": 1170,
4   "img_height": 1331,
5   "objects": [
6     {
7       "id": 0,
8       "class_id": 27,
9       "class_name": "office_building",
10      "relative_coordinates": {
11        "center_x": 0.389216,
12        "center_y": 0.437512,
13        "width": 0.361752,
14        "height": 0.624923
15      },
16      "confidence": 1.0
17    },
18    {
19      "id": 1,
20      "class_id": 26,
21      "class_name": "business_man",
22      "relative_coordinates": {
23        "center_x": 0.140022,
24        "center_y": 0.614859,
25        "width": 0.153102,
26        "height": 0.217009
27      },
28      "confidence": 1.0
29    },
30    {
31      "id": 2,
32      "class_id": 19,
33      "class_name": "desk",
34      "relative_coordinates": {
35        "center_x": 0.275298,
36        "center_y": 0.752864,
37        "width": 0.427005,
38        "height": 0.194189
39      },
40      "confidence": 0.65
41    },
42    {
43      "id": 3,
44      "class_id": 15,
45      "class_name": "car",
46      "relative_coordinates": {
47        "center_x": 0.725158,
48        "center_y": 0.712194,
49        "width": 0.492867,
50        "height": 0.14765
51      },
52      "confidence": 0.99
53    }
54  ],
55  "relations": [
56    {
57      "relation": "workplace",
58      "from": 1,
59      "to": 0
60    },
61    {
62      "relation": "part_of",
63      "from": 2,
64      "to": 0
65    },
66    {
67      "relation": "parked",
68      "from": 3,
69      "to": 0
70    },
71    {
72      "relation": "use_for_work",
73      "from": 1,
74      "to": 2
75    }
76  ],
77  "success": true,
78  "recognition_id": "20230526_231614"
79 }

```

Figure 75: Example: Recognition result

6 Conclusions

6.1 Thesis Results

The main challenge in driving business innovation and process optimization stems from the fact, that individuals possessing domain expertise and making strategic decisions do not overlap with those capable of translating this knowledge into an abstract and structured format necessary for the practical execution. This disparity inspired the development of our Scene2Model framework. The Scene2Model framework enables the extraction and formalization of domain knowledge, sourcing directly from the business experts, by providing all necessary tools for facilitating a design thinking workshop, during which they are encouraged to create a physical representation of their innovation plan or a scene, which is subsequently converted into a digital conceptual model using object detection and relation identification and prediction software. The resulting digital model is executable and can be used for simulating the outcome of improvement tactics without needing to allocate any valuable resources.

The Scene2Model (S2M) framework, created in the scope of this thesis, is represented both in the physical world and in the digital reality. The physical aspect of the framework involves in person design thinking workshops and tangible artifacts that represent different concepts from the problem domain. The bigger contribution of the S2M framework lays in the digital space and consists of five components: an automated server for object detection training and recognition, a mobile application as a graphical interface and for dataset preparation, a visual relationship identification and prediction module, a domain-specific modeling language, and an extension tool for importing and transforming the recognized scene components into a universal model.

6.2 Open Topics

As already noted, this master thesis has been conducted with the generous support and guidance of the OMiLAB collaboration network. (Section 2.2) During our sparring sessions with the aim of getting meaningful peer feedback, we have come across some weaknesses and suboptimal design decisions in the Scene2Model framework.

First of all, contrary to the current implementation, it is desirable that users can share trainings and not only see and use their own for recognition. A possible solution would be provide a list of all available trainings in the mobile application and the end user can then select, which one should be used for recognition. The drawback of this approach is, that the user should either download all training images for all trainings from the server to review and select the most suitable training for his concrete case, which might be very resource intensive and would probably explode very fast, or the user has no other choice but to make a blind decision, without knowing what training data stand behind it.

Second of all, it would be advantageous if one could be provided with a graphical user interface for defining the possible relationships and the logical constraints, that apply to them. In the current setup it is the digital engineering team, which facilitates the design thinking workshops, that manually transforms the business expertise from natural language to a formal knowledge representation. Additionally, it would be best, if the end user input could be formalized into first order logic, which is then automatically translated into logical tensor network constructs. This approach requires two automation steps, but would eliminate a lot of manual effort and error opportunities.

Lastly, the fully automated creation of the modelling language in ADOxx would be very useful. Currently the only manual step is the provision of the graphical representations for each object class. One possible solution would be, if the recognition server would crop the identified object from the recognition image and send it together with the recognition result, when called from ADOxx. That way, before importing the actual model the image could be saved in the according sync folder with the correct name, which also removes the human factor, having to ensure that the same names are used for the same object classes when labeling the objects on the training images as the pre-defined ones from the Scenes design thinking toolkit.

Appendices

A User manual

Define your domain:

1. Identify the domain participants or the different object classes.
2. Establish the relevant relations, that can connect object pairs. This includes both actions (e.g. drive or ride) and associations (e.g. is-a, related-to, depends-on) between objects.
3. Construct a knowledge base, by writing down all known facts and limitations about the domain (e.g. "a cat can not drive a car", "every train stops at a train station and nowhere else") and translate them into first-order logic.
4. Select an image representation for every object class from the Scenes design thinking toolkit (download it from <https://apphaus.sap.com/resource/scenes>) or use a custom source and print down the chosen symbols. Tip: pick simple black and white icons, because this would make the recognition easier and thus robuster.

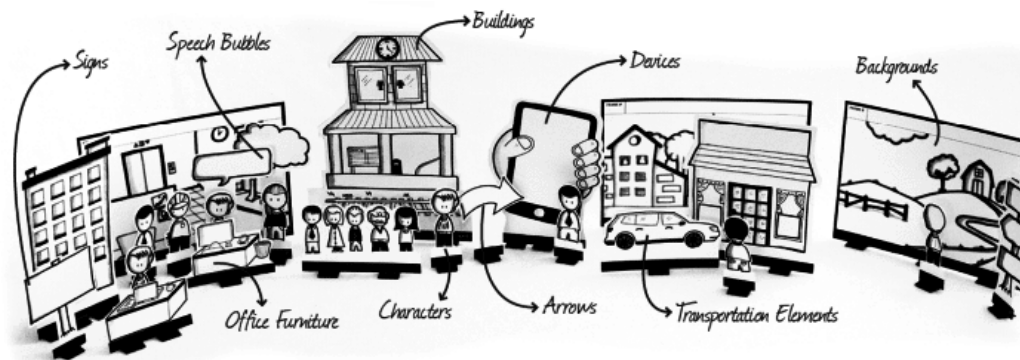


Figure 76: Scenes - the design thinking toolkit by SAP AppHaus [24]

Train your objects:

1. Install the scene2model-helper-app on you mobile phone or access the web application via your browser under <https://scene2model.web.app/>
2. Create a new training. Choose training name and type as shown below.



Figure 77: Mobile application screenshots - part 1

3. Take 2-3 images of each figure from different angles on a contrasting canvas. Tip: use a white canvas, if you have decided on using black-and-white icons, as this will create the perfect contrast.
4. Optional: Take pictures of pairs of object in interaction. This would help with the recognition of objects, that are only partially visible. (e.g. a person behind a bike - only the upper body of the person would be visible)
5. Upload the prepared images in the scene2-model-helper-app. Alternatively, you could do steps 2. and 3. on the go and capture the objects directly from the app.

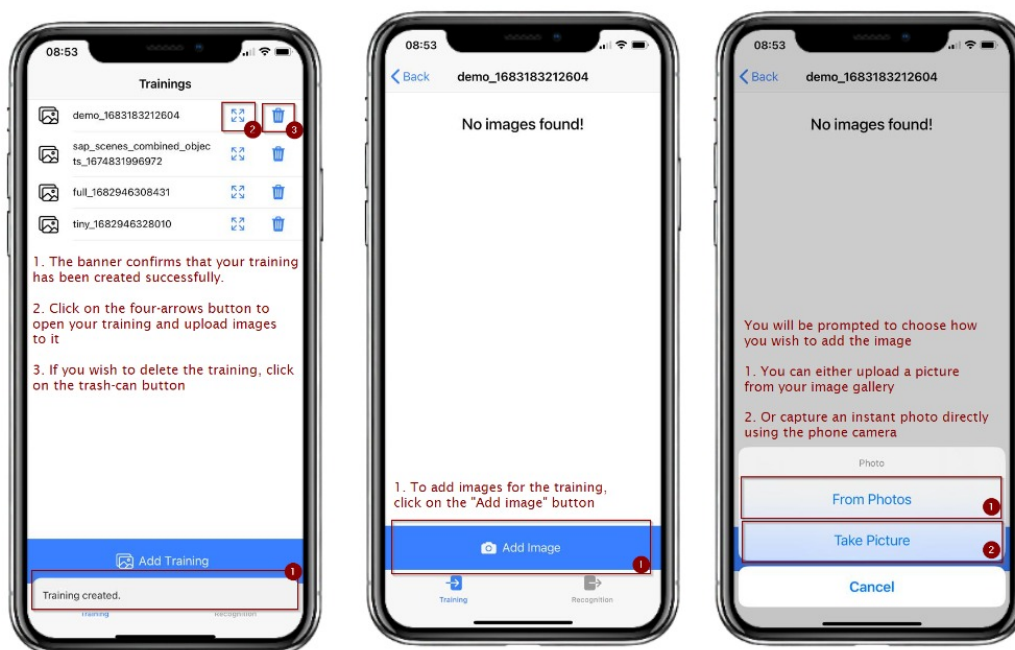


Figure 78: Mobile application screenshots - part 2

6. Open one image and label the contained in it objects. Align the bounding box for each object tightly around it and give it a class label. Always use the same class label for objects having the same object class. When you are done, submit the labels for the image.
7. Repeat steps 6 until all images have label assignments.
8. Start the training.
9. Wait for the training to finish. This might take between 1 hour and 12 hours depending on the training type you have chosen (Yolov4 or the tiny version - Yolov4-tiny). Once done, the training progress info will contain a training end time.

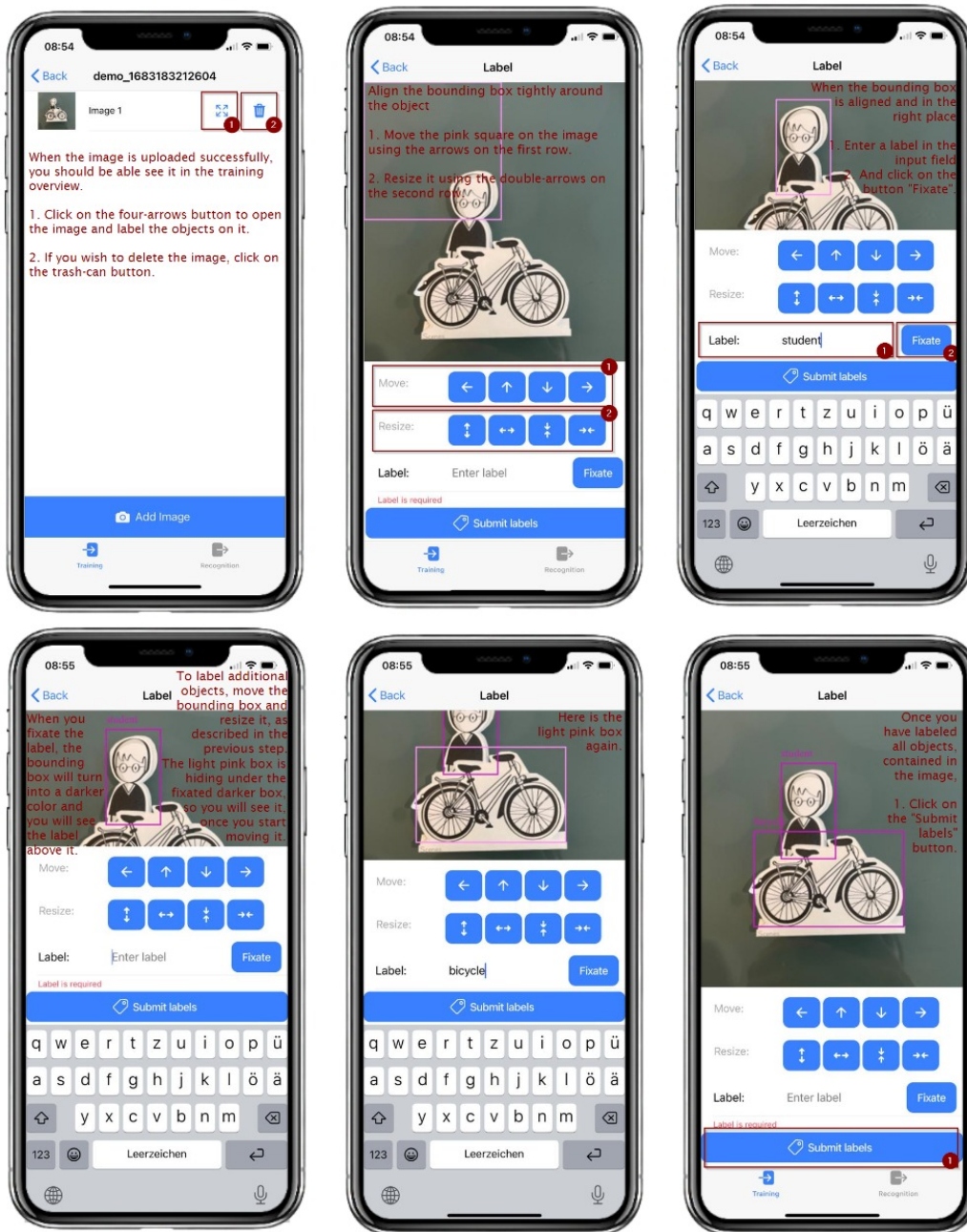


Figure 79: Mobile application screenshots - part 3

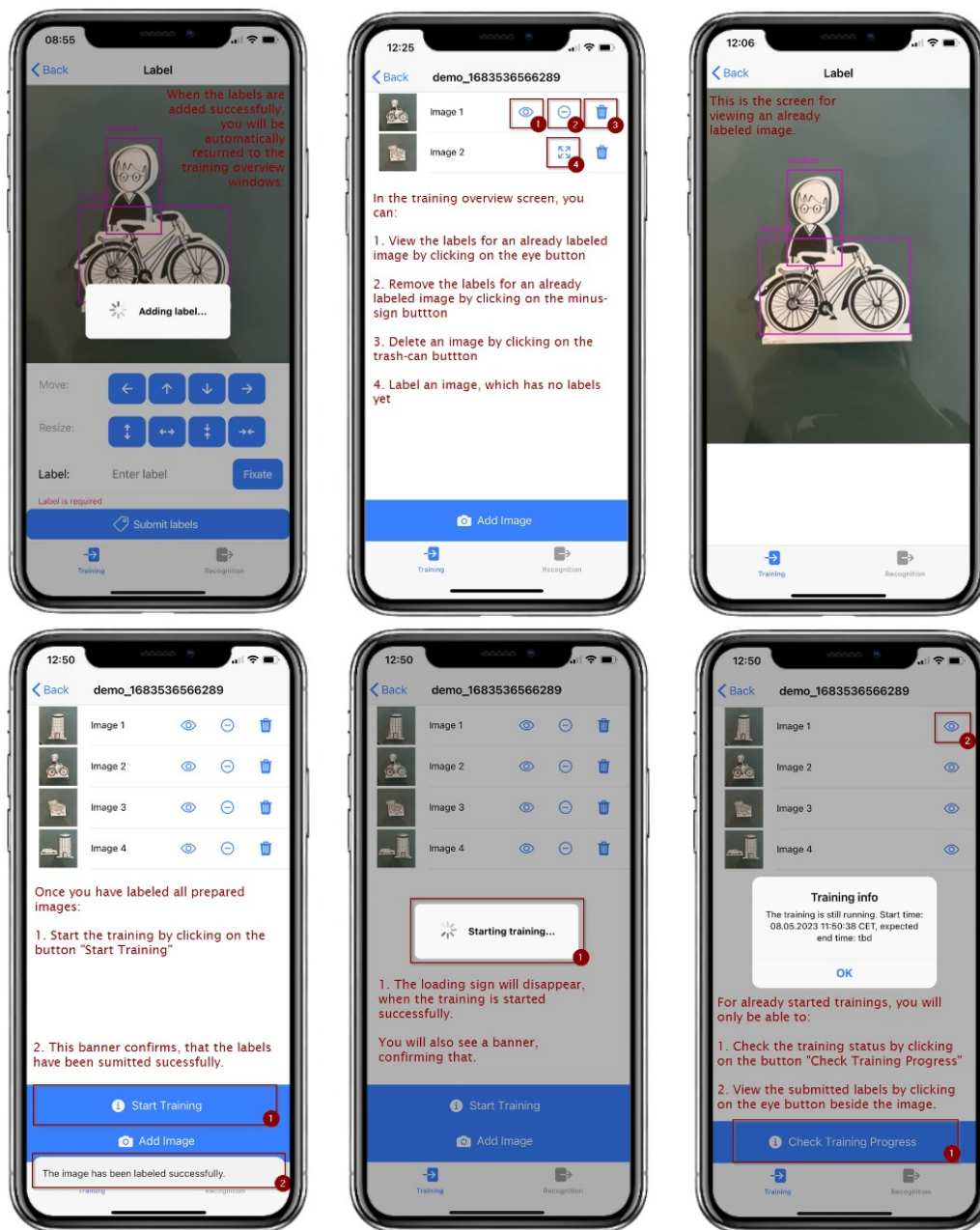


Figure 80: Mobile application screenshots - part 4

Apply the framework in a live design thinking workshop setting:

1. Use the printed paper figures to arrange a scene, that depicts your innovation ideas as an abstract and executable plan of action. Place related objects in a way that they have a visible overlap. The recognition of relations depends on that. Tip: Use the same white canvas, which you used for training the objects, as a background.
2. Make a picture of the complete configuration and upload it to the mobile application. Alternatively, you can capture the scene directly from the app.
3. Start the recognition. This step will take around a minute. Do not close the application during that time, because this will interrupt the recognition and you might need to re-upload the image.

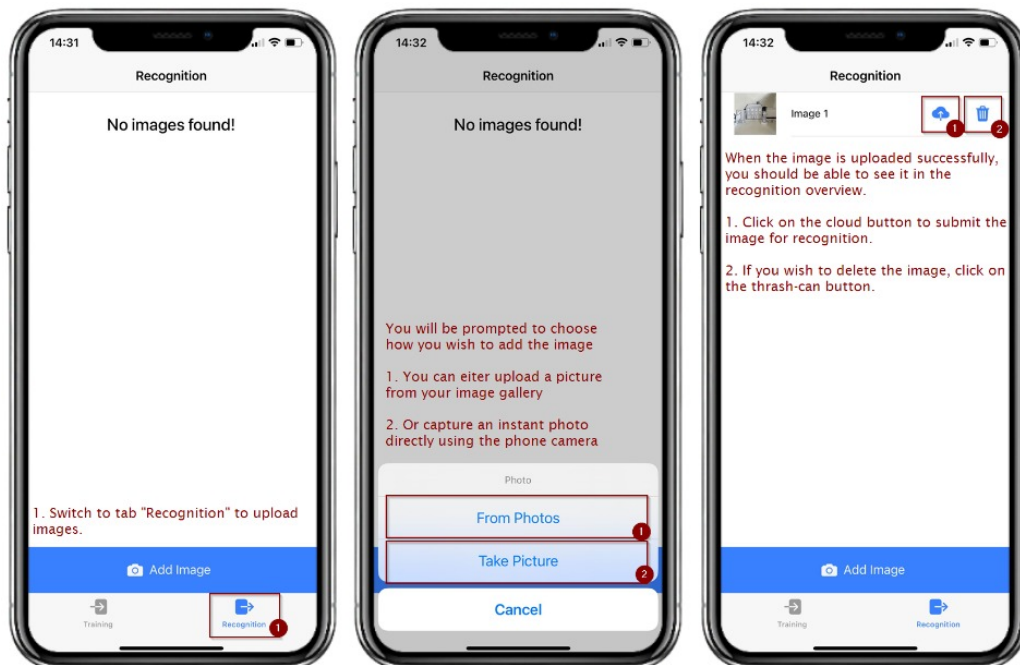


Figure 81: Mobile application screenshots - part 5

4. Open the image to view the result from the recognition. You will see the detected objects, all assigned with a bounding box, a label and a confidence score, as well as an id for the recognition.

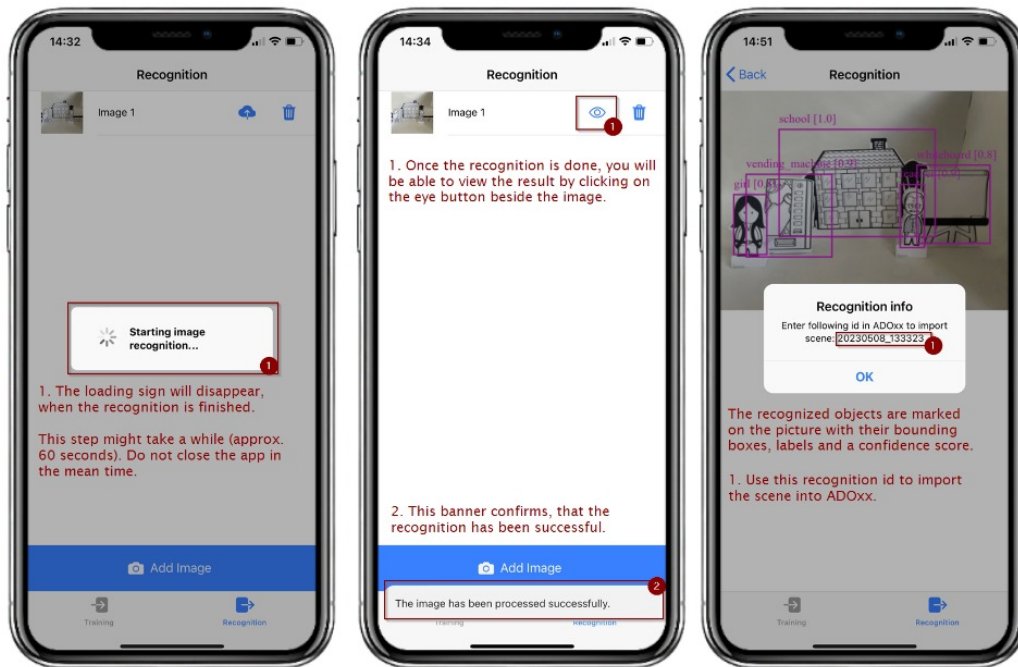


Figure 82: Mobile application screenshots - part 6

5. Install Scene2Model modelling toolkit by following the deployment instructions under https://code.omilab.org/scene2model/development/scene2model-package/-/blob/main/4_Modelling%20tool/README.md#deployment
6. Add your own figure figure library for modelling by following the guide https://code.omilab.org/scene2model/development/scene2model-package/-/blob/main/4_Modelling%20tool/README.md#import-figure-library
7. Open the Scene2Model modelling toolkit. Create a new model from type "Scene".
8. Select "Image Recognition" - "Import from recognition" from the navigation menu and enter the recognition id from the previous step.

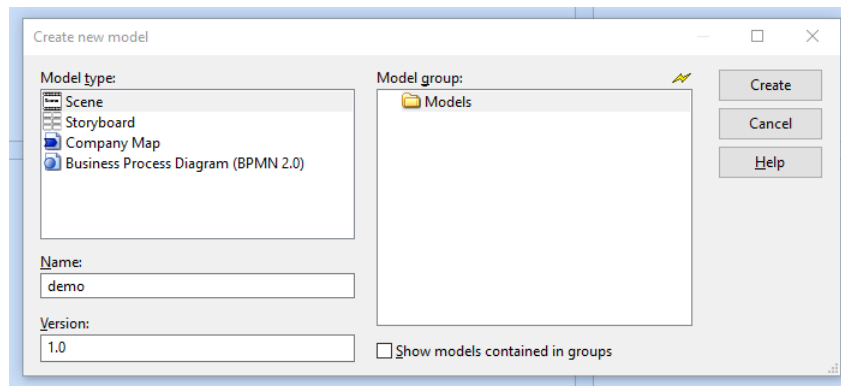


Figure 83: Create a model from type "Scene" in the ADOxx modelling toolkit

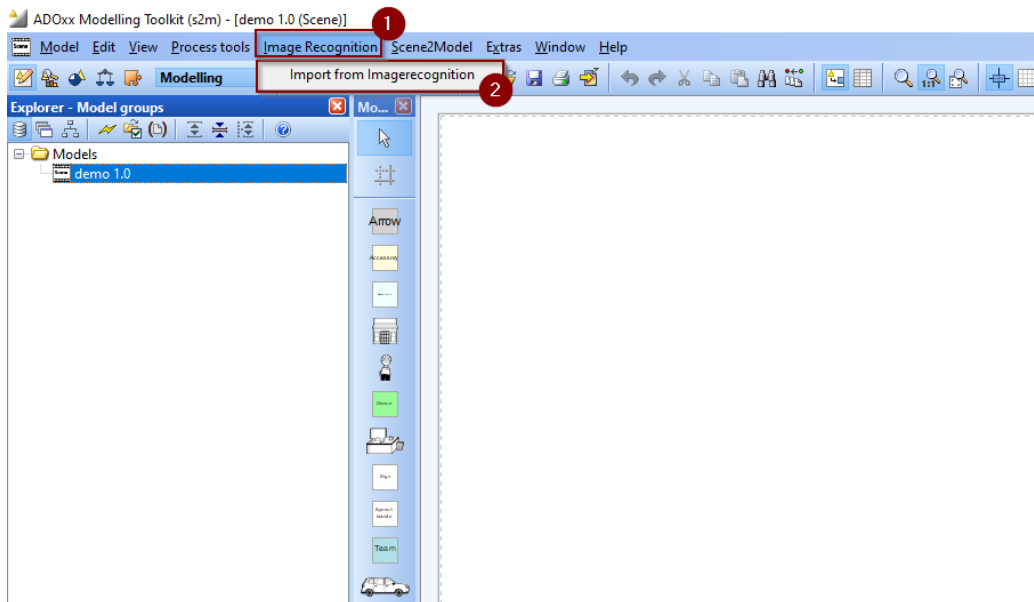


Figure 84: Import a scene in the ADOxx modelling toolkit

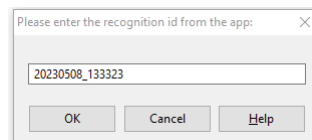


Figure 85: Enter the recognition id from the mobile application

-
9. By double-clicking on the imported objects and relations, you can review if they were recognized and imported correctly.

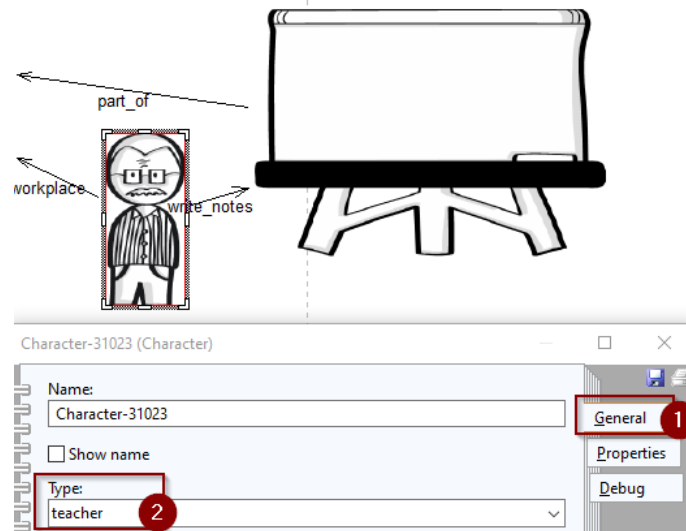


Figure 86: An imported object

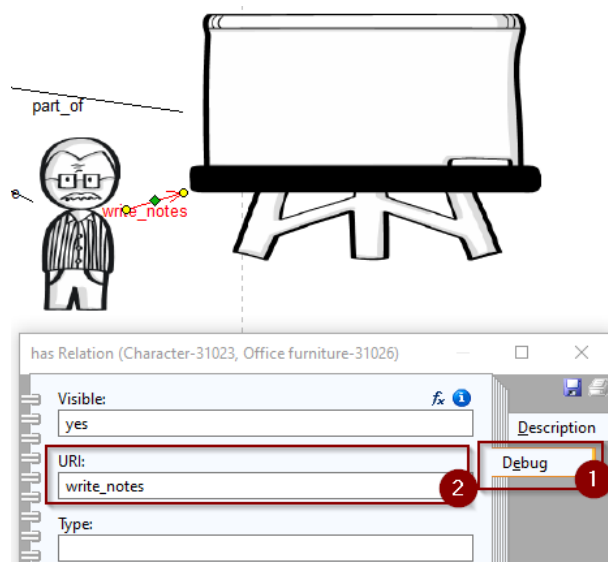


Figure 87: An imported relation

-
10. Voila. The physical scene has been transformed to a digital domain-specific conceptual model.

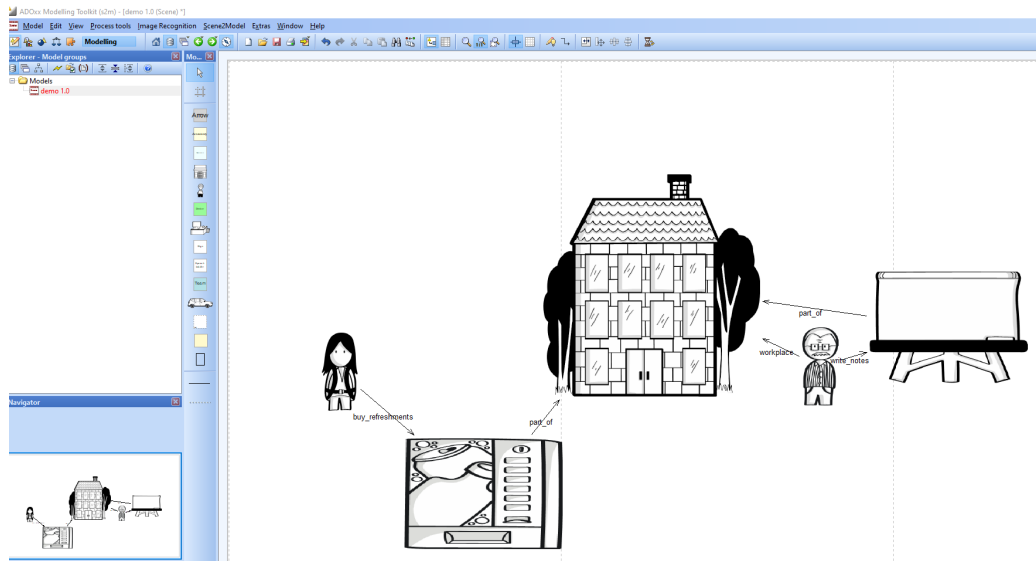


Figure 88: The domain-specific conceptual model, created by digitalizing a physical design thinking scene

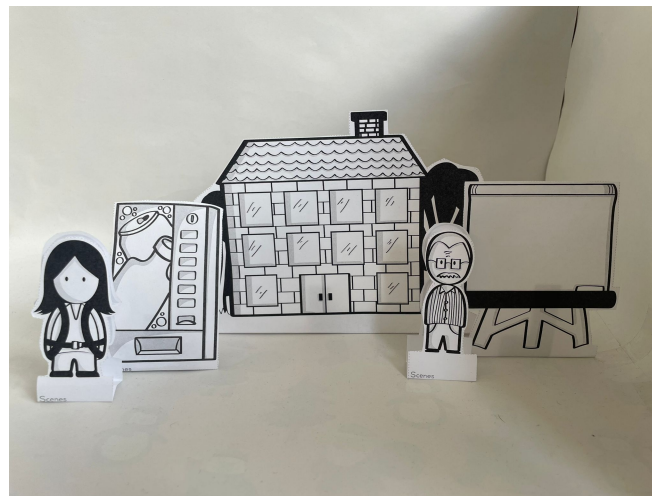


Figure 89: The physical design thinking scene, from which the above model was created

B Source code

The source code for the scene2model-helper-app mobile application can be found under:

- https://gitlab.dke.univie.ac.at/nnservice/scene2model/-/tree/main/2_mobile_application

The code base of the automated training and recognition server has been published in following repository:

- https://gitlab.dke.univie.ac.at/nnservice/scene2model/-/tree/main/1_training_and_recognition

The AdoScript script, containing the logic for generating the domain-specific conceptual model from the recognition result, provided by the automated training and recognition server, can be reviewed under:

- https://gitlab.dke.univie.ac.at/nnservice/scene2model/-/tree/main/2_mobile_applicationhttps://gitlab.dke.univie.ac.at/nnservice/scene2model/-/tree/main/3_import_to_modelling_tool

C Statutory Declaration

I declare that I have developed and written the enclosed master thesis completely by myself, and have not used sources or means without declaration in the text. Any thoughts from others or literal quotations are clearly marked. The master thesis was not used in the same or in a similar version to achieve an academic grading or is being published elsewhere.

D Abstract

D.1 Abstract in English

The continuous need for driving business innovation and process optimization requires for manners to close the gap between expert knowledge and the machines or individuals that will be performing the actual work. To address this challenge, we present our Scene2Model framework that enables the extraction and formalization of domain knowledge directly from business experts by motivating them to create a physical representation of their innovation ideas in a guided design thinking workshop. The emerged physical scene is then transformed into a digital conceptual model using object detection and relation prediction tools. The output of the Scene2Model framework is a universal digital model (an abstract generic representation that captures the fundamental concepts and relationships applicable across various domains or industries), which is machine processable and can be used for checking the feasibility of otherwise resource-intensive and thus high-cost solutions.

D.2 Abstract in German

Der fortwährende Bedarf an unternehmerischer Innovation und Prozessoptimierung erfordert Möglichkeiten, die Lücke zwischen Expertenwissen und den Maschinen oder Individuen, die die tatsächliche Arbeit ausführen, zu überbrücken. Um diese Herausforderung anzugehen, präsentieren wir unser Scene2Model-Framework, das die Extraktion und Formalisierung von Fachwissen direkt von Unternehmensexperten ermöglicht, indem sie dazu motiviert werden, ihre Innovationsideen in einem geführten Design Thinking Workshop physisch darzustellen. Das entstandene physische Szenario wird dann mithilfe von Objekterkennungs- und Beziehungsvorhersage-Tools in ein digitales konzeptionelles Modell umgewandelt. Das Ergebnis des Scene2Model-Frameworks ist ein universelles digitales Modell (eine abstrakte generische Darstellung, die die grundlegenden Konzepte und Beziehungen erfasst, die in verschiedenen Bereichen oder Industrien anwendbar sind), das maschinenverarbeitbar ist und zur Überprüfung der Machbarkeit ansonsten ressourcenintensiver und somit kostenintensiver Lösungen verwendet werden kann.

References

- [1] D. Bork, R. Buchmann, D. Karagiannis, M. Lee, and E.-T. Miron, “An open platform for modeling method conceptualization: The omilab digital ecosystem,” *Communications of the Association for Information Systems*, vol. 44, pp. 673–697, 05 2019.
- [2] D. Karagiannis, R. A. Buchmann, X. Boucher, S. Cavalieri, A. Florea, D. Kiritsis, and M. Lee, “Omilab: A smart innovation environment for digital engineers,” pp. 273–282, 2020.
- [3] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “SSD: Single shot MultiBox detector,” pp. 21–37, 2016. [Online]. Available: https://doi.org/10.1007%2F978-3-319-46448-0_2
- [4] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, “Focal loss for dense object detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 2, pp. 318–327, 2020.
- [5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, 2015, cite arxiv:1506.02640. [Online]. Available: <http://arxiv.org/abs/1506.02640>
- [6] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” *CoRR*, vol. abs/1311.2524, 2013. [Online]. Available: <http://arxiv.org/abs/1311.2524>
- [7] K. He, X. Zhang, S. Ren, and J. Sun, “Spatial pyramid pooling in deep convolutional networks for visual recognition,” *CoRR*, vol. abs/1406.4729, 2014. [Online]. Available: <http://arxiv.org/abs/1406.4729>
- [8] R. B. Girshick, “Fast R-CNN,” vol. abs/1504.08083, 2015. [Online]. Available: <http://arxiv.org/abs/1504.08083>
- [9] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks,” 2015. [Online]. Available: <https://arxiv.org/abs/1506.01497>
- [10] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask r-cnn,” 2017. [Online]. Available: <https://arxiv.org/abs/1703.06870>

-
- [11] M. A. Sadeghi and A. Farhadi, “Recognition using visual phrases,” pp. 1745–1752, 2011.
 - [12] C. Lu, R. Krishna, M. S. Bernstein, and L. Fei-Fei, “Visual relationship detection with language priors,” *CoRR*, vol. abs/1608.00187, 2016. [Online]. Available: <http://arxiv.org/abs/1608.00187>
 - [13] B. Dai, Y. Zhang, and D. Lin, “Detecting visual relationships with deep relational networks,” *CoRR*, vol. abs/1704.03114, 2017. [Online]. Available: <http://arxiv.org/abs/1704.03114>
 - [14] K. Liang, Y. Guo, H. Chang, and X. Chen, “Visual relationship detection with deep structural ranking,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, Apr. 2018. [Online]. Available: <https://ojs.aaai.org/index.php/AAAI/article/view/12274>
 - [15] H. Zhang, Z. Kyaw, S.-F. Chang, and T.-S. Chua, “Visual translation embedding network for visual relation detection,” 2017.
 - [16] R. Yu, A. Li, V. I. Morariu, and L. S. Davis, “Visual relationship detection with internal and external linguistic knowledge distillation,” *CoRR*, vol. abs/1707.09423, 2017. [Online]. Available: <http://arxiv.org/abs/1707.09423>
 - [17] I. Donadello and L. Serafini, “Compensating supervision incompleteness with prior knowledge in semantic image interpretation,” pp. 1–8, 2019.
 - [18] C. Muck and S. Palkovits-Rauter, *Conceptualizing Design Thinking Artefacts: The Scene2Model Storyboard Approach*. Cham: Springer International Publishing, 2022, pp. 567–587. [Online]. Available: https://doi.org/10.1007/978-3-030-93547-4_25
 - [19] E.-T. Miron, C. Muck, and D. Karagiannis, “Transforming haptic storyboards into diagrammatic models: The scene2model tool,” in *Hawaii International Conference on System Sciences*, 2019.
 - [20] A. Bochkovskiy, C.-Y. Wang, and H.-y. Liao, “Yolov4: Optimal speed and accuracy of object detection,” 04 2020.

-
- [21] E. Evans, *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Boston: Addison-Wesley, 2003. [Online]. Available: <https://www.safaribooksonline.com/library/view/domain-driven-design-tackling/0321125215/>
- [22] M. Walch, T. Morita, D. Karagiannis, and T. Yamaguchi, “A knowledge-based conceptual modelling approach to bridge design thinking and intelligent environments,” in *Knowledge Science, Engineering and Management*, 2019.
- [23] F. Schierlinger-Brandmayr, B. Moesl, P. Url, W. Vorraber, and S. Vössner, *A Modeling Tool for Exploring Business Ecosystems in a (Pre-)conceptual Phase*. Cham: Springer International Publishing, 2022, pp. 315–338. [Online]. Available: https://doi.org/10.1007/978-3-030-93547-4_14
- [24] S. AppHaus, “Scenes: Every great experience starts with a great story.” [Online]. Available: <https://apphaus.sap.com/resource/scenes>
- [25] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, pp. 436–444, 2015.
- [26] Z.-Q. Zhao, P. Zheng, S.-t. Xu, and X. Wu, “Object detection with deep learning: A review,” 2018, cite arxiv:1807.05511. [Online]. Available: <http://arxiv.org/abs/1807.05511>
- [27] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *nature*, vol. 323, no. 6088, pp. 533–536, 1986.
- [28] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *nature*, vol. 521, no. 7553, p. 436, 2015.
- [29] K. O’Shea and R. Nash, “An introduction to convolutional neural networks.” *CoRR*, vol. abs/1511.08458, 2015. [Online]. Available: <http://dblp.uni-trier.de/db/journals/corr/corr1511.html#OSheaN15>
- [30] S. B. Kotsiantis, “Supervised machine learning: A review of classification techniques,” *Informatica*, vol. 31, no. 3, pp. 249–268, 2007. [Online]. Available: <http://>
-

//www.informatica.si/PDF/31-3/11_Kotsiantis%20-%20Supervised%20Machine%20Learning%20-%20A%20Review%20of...pdf

- [31] H. B. Barlow, “Unsupervised learning: introduction,” in *Unsupervised Learning: Foundations of Neural Computation*, G. E. Hinton and T. J. Sejnowski, Eds. Bradford Company Scituate, MA, USA, 1999, pp. 1–17. [Online]. Available: <http://books.google.com/books?id=yj04Y0lje4cC>
- [32] O. Chapelle, B. Schölkopf, and A. Zien, Eds., *Semi-Supervised Learning*. The MIT Press, 2006. [Online]. Available: <http://dblp.uni-trier.de/db/books/collections/CSZ2006.html>
- [33] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy, “Speed/accuracy trade-offs for modern convolutional object detectors,” *CoRR*, vol. abs/1611.10012, 2016. [Online]. Available: <http://arxiv.org/abs/1611.10012>
- [34] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, “Microsoft coco: Common objects in context,” 2014, cite arxiv:1405.0312Comment: 1) updated annotation pipeline description and figures; 2) added new section describing datasets splits; 3) updated author list. [Online]. Available: <http://arxiv.org/abs/1405.0312>
- [35] P. Henderson and V. Ferrari, “End-to-end training of object class detectors for mean average precision,” *CoRR*, vol. abs/1607.03476, 2016. [Online]. Available: <http://arxiv.org/abs/1607.03476>
- [36] S. Goswami, “False detection (positives and negatives) in object detection,” 2020.
- [37] A. Benali Amjoud and M. Amrouch, “Convolutional neural networks backbones for object detection,” in *Image and Signal Processing*, A. El Moataz, D. Mammass, A. Mansouri, and F. Nouboud, Eds. Cham: Springer International Publishing, 2020, pp. 282–289.
- [38] A. Krizhevsky, I. Sutskever, and G. Hinton, “Imagenet classification with deep convolutional neural networks,” *Neural Information Processing Systems*, vol. 25, 01 2012.

-
- [39] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *CoRR*, vol. abs/1409.1556, 2014. [Online]. Available: <http://arxiv.org/abs/1409.1556>
- [40] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” 2014.
- [41] J. Redmon, “Darknet: Open source neural networks in c,” <http://pjreddie.com/darknet/>, 2013–2016.
- [42] O. Corcoll Andreu, “Semantic image cropping,” 07 2021.
- [43] A. Shrivastava, A. Gupta, and R. Girshick, “Training region-based object detectors with online hard example mining,” 2016. [Online]. Available: <https://arxiv.org/abs/1604.03540>
- [44] M. Gao, Y. Du, Y. Yang, and J. Zhang, “Adaptive anchor box mechanism to improve the accuracy in the object detection system,” *Multi-media Tools and Applications*, vol. 78, 10 2019.
- [45] J. Hui, “Ssd object detection: Single shot multibox detector for real-time processing,” 2018. [Online]. Available: <https://jonathan-hui.medium.com/9bd8deac0e06>
- [46] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *CoRR*, vol. abs/1512.03385, 2015.
- [47] T. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, “Feature pyramid networks for object detection,” pp. 936–944, jul 2017. [Online]. Available: <https://doi.ieeecomputersociety.org/10.1109/CVPR.2017.106>
- [48] J. Redmon and A. F. Ali, “Yolov3: An incremental improvement,” 2018. [Online]. Available: <https://arxiv.org/abs/1804.02767>
- [49] J. Redmon and A. Farhadi, “Yolo9000: Better, faster, stronger,” 2016. [Online]. Available: <https://arxiv.org/abs/1612.08242>
- [50] J. Uijlings, K. Sande, T. Gevers, and A. Smeulders, “Selective search for object recognition,” *International Journal of Computer Vision*, vol. 104, pp. 154–171, 09 2013.
-

-
- [51] H. D. P. Felzenszwalb, Pedro F, “Efficient graph-based image segmentation,” *International Journal of Computer Vision*, vol. 59, pp. 1573–1405, 09 2004.
- [52] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” vol. 25, 2012. [Online]. Available: <https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf>
- [53] C. Szegedy, S. Reed, D. Erhan, D. Anguelov, and S. Ioffe, “Scalable, high-quality object detection,” 2015.
- [54] P. F. Felzenszwalb, R. B. Girshick, and D. McAllester, “Discriminatively trained deformable part models, release 4,” <http://people.cs.uchicago.edu/~pff/latent-release4/>.
- [55] R. B. Girshick, F. N. Iandola, T. Darrell, and J. Malik, “Deformable part models are convolutional neural networks,” *CoRR*, vol. abs/1409.5403, 2014. [Online]. Available: <http://arxiv.org/abs/1409.5403>
- [56] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” 2015.
- [57] J. D. Lafferty, A. McCallum, and F. C. N. Pereira, “Conditional random fields: Probabilistic models for segmenting and labeling sequence data,” p. 282–289, 2001.
- [58] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L. Li, D. A. Shamma, M. S. Bernstein, and L. Fei-Fei, “Visual genome: Connecting language and vision using crowdsourced dense image annotations,” *CoRR*, vol. abs/1602.07332, 2016. [Online]. Available: <http://arxiv.org/abs/1602.07332>
- [59] S. Badreddine, A. S. d’Avila Garcez, L. Serafini, and M. Spranger, “Logic tensor networks,” *CoRR*, vol. abs/2012.13635, 2020. [Online]. Available: <https://arxiv.org/abs/2012.13635>
- [60] S. Baier, Y. Ma, and V. Tresp, “Improving visual relationship detection using semantic modeling of scene descriptions,” *CoRR*, vol. abs/1809.00204, 2018. [Online]. Available: <http://arxiv.org/abs/1809.00204>
-

-
- [61] X. Liang, L. Lee, and E. P. Xing, “Deep variation-structured reinforcement learning for visual relationship and attribute detection,” 2017.
- [62] B. Zhuang, L. Liu, C. Shen, and I. D. Reid, “Towards context-aware interaction recognition,” *CoRR*, vol. abs/1703.06246, 2017. [Online]. Available: <http://arxiv.org/abs/1703.06246>
- [63] D. Karagiannis, “Agile modeling method engineering,” pp. 5–10, 10 2015.
- [64] OMG. (2011) Unified modeling language™ (uml®). [Online]. Available: <http://www.omg.org/spec/UML/index.htm>
- [65] E. Dickhaut, A. Janson, and J. M. Leimeister, “It takes two to tango: Design thinking and design patterns for better system development,” in *Design Thinking for Software Engineering*, ser. Progress in IS, J. Hehn, D. Mendez, W. Brenner, and M. Broy, Eds. Cham, Switzerland: Springer, jan 2022 2022, pp. 201–211. [Online]. Available: https://pubs.wi-kassel.de/wp-content/uploads/2022/02/JML_860.pdf
- [66] T. Brown, “Design thinking,” *Harvard Business Review*, vol. 86, no. 6, pp. 84–92, jun 2008. [Online]. Available: <https://hbr.org/2008/06/design-thinking>
- [67] M. Gardner and S. Dorling, “Artificial neural networks (the multilayer perceptron)—A review of applications in the atmospheric sciences,” *Atmospheric Environment*, vol. 32, no. 14-15, pp. 2627–2636, 1998.
- [68] CapacitorJs, “Capacitor: Cross-platform native runtime for web apps.” [Online]. Available: <https://capacitorjs.com/docs>
- [69] ADOxx, “Adoxx: Develop your own modelling toolkit.” [Online]. Available: https://www.adoxx.org/live/hardware_software-requirements-15
- [70] Ionic, “Ionic: Open-source ui toolkit to create your mobile apps.” [Online]. Available: <https://ionicframework.com/docs/>
- [71] J. Lee, H. Kim, J. Park, I. Shin, and S. Son, “Pride and prejudice in progressive web apps: Abusing native app-like features in web applications,” in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS ’18. New York,
-

-
- NY, USA: Association for Computing Machinery, 2018, p. 1731–1746.
[Online]. Available: <https://doi.org/10.1145/3243734.3243867>
- [72] Google, “Firebase.” [Online]. Available: <https://firebase.google.com/docs>
- [73] Ionic, “Progressive web apps in angular.” [Online]. Available: <https://ionicframework.com/docs/angular/pwa>
- [74] Z. Jiang, L. Zhao, S. Li, and Y. Jia, “Real-time object detection method based on improved yolov4-tiny,” 2020.