

Institut für Maschinelle Sprachverarbeitung  
Universität Stuttgart  
Pfaffenwaldring 5B  
70569 Stuttgart

MASTER THESIS  
**Comparison of Object Detection Methods  
for Abstract and Concrete Concepts**

Katrin Schmidt

Studiengang: M.Sc. Computational Linguistics

Prüfer\*innen: Apl. Prof. Dr. Sabine Schulte im Walde  
Prof. Dr. Sebastian Padó

Betreuer\*innen: Apl. Prof. Dr. Sabine Schulte im Walde  
Dr. Diego Frassinelli

Beginn der Arbeit: 15.02.2023

Ende der Arbeit: 15.08.2023

# Contents

<b>1</b>	<b>Introduction</b>	<b>5</b>
<b>2</b>	<b>Abstractness vs. Concreteness</b>	<b>8</b>
2.1	General Distinction . . . . .	8
2.2	Theory of Cognition . . . . .	11
2.3	Concreteness Ratings . . . . .	13
2.4	Text-Based Computational Approaches . . . . .	14
2.5	Multi-Modal Computational Approaches . . . . .	15
<b>3</b>	<b>Object Detection</b>	<b>16</b>
3.1	One-Stage vs. Two-Stage vs. Multi-Stage Detectors . . . . .	18
3.2	Applicability of CNNs for Object Detection . . . . .	19
3.3	Technical Background of CNNs . . . . .	20
3.4	Object Detection Model Architecture . . . . .	22
3.5	Comparison of Toolboxes . . . . .	24
3.6	MMDetection . . . . .	25
3.7	Tools . . . . .	27
3.7.1	SSD . . . . .	29
3.7.2	YOLOX . . . . .	30
3.7.3	Faster R-CNN . . . . .	31
3.7.4	RetinaNet . . . . .	32
3.7.5	Double-Head R-CNN . . . . .	34
3.7.6	Deformable DETR . . . . .	35
3.7.7	Cascade R-CNN . . . . .	36

<b>4 Experiments</b>	<b>37</b>
4.1 Dataset . . . . .	37
4.2 Model Set-up . . . . .	39
4.3 Image Inference . . . . .	41
4.4 Results (Experiments) . . . . .	42
<b>5 Evaluation and Interpretation of Results</b>	<b>49</b>
5.1 Evaluation Set-up . . . . .	49
5.2 Results (Evaluation) . . . . .	54
5.2.1 Subtask A . . . . .	55
5.2.2 Subtask B . . . . .	62
5.3 Discussion . . . . .	69
5.3.1 Abstractness vs. Concreteness . . . . .	70
5.3.2 Model Comparison . . . . .	73
5.3.3 Top rated labels . . . . .	75
<b>6 Conclusion</b>	<b>75</b>
<b>Appendix</b>	<b>78</b>
<b>References</b>	<b>83</b>

## **Erklärung (Statement of Authorship)**

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst habe und dabei keine andere als die angegebene Literatur verwendet habe. Alle Zitate und sinngemäßen Entlehnungen sind als solche unter genauer Angabe der Quelle gekennzeichnet. Die eingereichte Arbeit ist weder vollständig noch in wesentlichen Teilen Gegenstand eines anderen Prüfungsverfahrens gewesen. Sie ist weder vollständig noch in Teilen bereits veröffentlicht. Die beigefügte elektronische Version stimmt mit dem Druckexemplar überein.<sup>1</sup>

(Katrin Schmidt)

---

<sup>1</sup>Non-binding translation for convenience: This thesis is the result of my own independent work, and any material from work of others which is used either verbatim or indirectly in the text is credited to the author including details about the exact source in the text. This work has not been part of any other previous examination, neither completely nor in parts. It has neither completely nor partially been published before. The submitted electronic version is identical to this print version.



## Abstract

In human communication, there are concrete concepts which are perceivable with one of the human senses (e.g. *table* or *apple*) and abstract concepts (e.g. *freedom* or *love*). The latter can not be perceived with the senses and relies more on subjective concepts. It is a challenging task to translate abstract concepts to automatic systems in computer-human interaction. This study examines characteristics of the visual representation of abstractness and concreteness by applying seven different object detection tools on images depicting abstract and concrete concepts. Further, the seven tools are compared with respect to model performance and their relation to abstractness and concreteness. In order to achieve this, image inference with a toolbox named MMDetection is performed on an image dataset with 500/500 of the most extreme abstract/concrete concepts. Further, the counts of detected objects and their distribution across abstractness, concreteness and models are analysed. Following, the model results are evaluated with the help of human annotators.

The study finds that significantly more objects are detected for concreteness. Further, abstractness yields better results if only those cases are included that have a high prediction confidence. While RetinaNet performs well overall, Deformable DETR is particularly suitable for abstract data. On concrete data, both RetinaNet and Cascade R-CNN perform well. Overall, these findings are a step towards the characteristics of abstract and concrete concepts, suggesting that concrete concepts tend to occur with more objects per image than abstract ones. When considering the number of object counts per image as *context*, this argues that abstractness tends to occur with less context than concreteness and vice versa.

# 1 Introduction

In the course of the evolution of deep learning methods, convolutional neural networks (CNNs) have raised great interest for both computer vision tasks as well as natural language processing (NLP) tasks. As a consequence of the attention on both areas, a new research branch evolved that involves the interaction between computer vision and NLP. However, when it comes to computer-human interactions there are also limitations, especially for communication. One of these limitations is the differentiation between *abstract* concepts and *concrete* concepts. The latter can be perceived with at least one of the human senses (see, hear, touch, smell, or taste) like *table*, *banana* or *cat*, whereas abstract concepts rely more on subjective experiences and are not perceivable, like *trust*, *love* or *freedom* (Brysbaert et al., 2014). Technology in general has shown the trend of trying to approximate human-like behavior. Yet, it is an open question how to translate subjective, abstract values and concepts to an automatic system that has no relation to human feelings and experiences.

In the long term it is an aim to translate these aspects to a computer, either in form of some inherently abstract features or some external features such as context. One use case for the translation of abstractness to an automatic system is automatic caption generation, a process where objects in an image are identified in order to further process the output linguistically. This involves both the task of *object detection* as well as the task of generating grammatically meaningful words, phrases or sentences. Object detection is a subdomain of computer vision and refers to the task of identifying entities of objects within images (Zou et al., 2023). An automatic human-like approach for caption generation can support visually impaired people in understanding images, automatically label a large image dataset, serve as a pre-selection for editorial work or improve sentiment analysis tasks. In this thesis, various state-of-the-art tools from the field of object detection are applied to a dataset consisting of images, each denoting an abstract or concrete concept. The

purpose of the study can be divided in three parts.

**(1) Abstractness/concreteness.** One aim is to address the question about the characteristics of abstractness and concreteness. In order to describe the properties of these two concepts, following questions are asked: **1a)** Is there a difference between the number of detected objects for abstractness and concreteness? **1b)** Does the performance of the object detection tools vary for abstract and concrete concepts? **1c)** What is the distribution of confidence scores per abstractness and concreteness?

**(2) Model performance.** Further, the performance of the individual models is compared in general and in relation to abstractness and concreteness. For this purpose, the following questions are examined: **2a)** Which model detects the most and the fewest objects? **2b)** Which tool performs best on abstract data, which on concrete data and which performs best overall? **2c)** How does the model performance change across different confidence score settings?

**(3) Top rated labels.** Finally, the top rated labels and their counts are examined considering different settings. **3a)** Which labels are associated with an annotators agreement of 50/50? **3b)** Which labels are associated with an annotators agreement 100%? **3c)** Which labels are associated with majority vote?

By answering these questions, this work aims to learn more about characteristics of concrete and abstract concepts, provide a comparison of the performance of object detection methods and show reproducibility of the tools. Apart from that, an image dataset with more in-depth labels is generated. These goals are accomplished as follows.

First of all, section 2 provides foundational aspects for the differentiation of abstractness and concreteness, such as an introduction into basic terms, followed by a general distinction and the development of theories from the perspective of cognitive science. Further, existing computational approaches towards abstractness and concreteness are summarized.

Section 3 focuses on object detection and begins with an introduction into the

three different types of detectors as well as an overview of convolutional neural networks (CNNs). The section explains the importance of CNNs for object detection and provides technical background. Also, the typical structure of object detection tools is described. The subsequent comparison of toolboxes results in the decision to use MMDetection for this study. Finally, the seven tools that are used in the experiments and their functionality are explained.

The next section presents the experiments that are conducted in this work. After the pre-processing and construction of the dataset are described, the section elaborates on the model set-up and presents the different weights and parameter settings of the seven tools that are used. Furthermore, code-specific adjustments and the handling of the output are explained. The section concludes with an overview of the results of the experiment, considering the object count for both abstractness and concreteness, as well as per model, per image, and per concept. The distribution of the occurrences of confidence scores is also shown.

This is followed by section 5, which deals with the evaluation and interpretation of the results. With the help of human annotators via *Amazon Mechanical Turk* (AMT), the object labels are judged with a binary label (yes or no) and the refinement of the bounding boxes is rated on a scale between 1 and 4. Precision is computed from the binary judgement and a comparison is made between abstractness and concreteness on the one hand and between the seven models on the other hand. The ratings on the scale between 1 and 4 are also analysed and sorted by performance for abstractness, concreteness and model. The discussion concludes that abstractness yields better results under certain settings and that significantly more objects are detected for concreteness. Further, RetinaNet shows good performance and while Deformable DETR performs well on abstract data, both RetinaNet and Cascade R-CNN perform well on concrete data. Also, object labels with the highest ratings are analysed.

Finally, the conclusion summarizes these findings and suggests explanations for

the described results. By doing so, the characteristics of the visual representation of abstractness and concreteness are addressed. With reference to the model comparison, these suggestions are further justified.

## 2 Abstractness vs. Concreteness

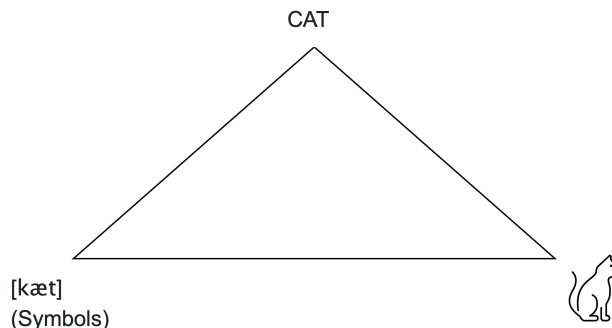
In this section, essential terms from cognitive linguistics are explained in order to enable the discussion about the differences between abstractness and concreteness. Subsequently, a general differentiation between abstractness and concreteness is approached, followed by an overview of cognition theory and related work of computational approaches.

### 2.1 General Distinction

For this discussion, the term *concept* is very central. A concept in cognitive science is considered to represent a certain category, having prototypical properties (Goguen, 2005). Furthermore, a hierarchy of concepts is assumed, where *basic level concepts* occur in the middle of this hierarchy and are short, easy to learn and share the most associated information (Goguen, 2005). Interestingly, from the perspective of cognitive science, metaphors often appear as families that share *sensory-motor* patterns (Goguen, 2005). These patterns can be broken down to the words *sensory*, which refers to the five human senses and *motor*, referring to the motor system, which is responsible for physical movement (Fischer and Zwaan, 2008). Psycholinguistics assumes that language processing involves the simulation of actions and experiences which is associated with the sensory-motor system (Fischer and Zwaan, 2008). A related concept is the term *imageability*. It refers to a measure for the degree of difficulty with that sensory-motor experiences are aroused (Dellantonio et al., 2014).

When it comes to the differentiation between abstractness and concreteness, also the term *mental representation* is important. In order to approach this term from

a cognitive psychology’s view it is helpful to look at the triangle of reference. In Figure 1, the mental representation is represented by the capitalized word CAT.



**Figure 1:** Triangle of reference, inspired by Meibauer et al. (2015; p. 172).

The triangle, inspired by Meibauer et al. (2015; p. 172), shows the linguistic expression at the bottom left, referring to the object in the real world mediated through the mental representation which represents the conceptual knowledge of the object. This includes stored conceptual information like *a cat meows*, *a cat has four legs* or *a cat is a pet*. In other words, a mental representation can be thought of “[...] as an encoding of some information, which an individual can construct, retain in memory, access, and use in various ways” (Smith, 1998; p. 391).

In the following, an approach to differentiate between abstractness and concreteness will be presented. A general approach towards the distinction of concreteness and abstractness is to define concreteness with regards to the degree of perceptiveness of the referent that a concept refers to (Brysbaert et al., 2014). Put another way, it can be seen as a measure of how easily an expression can be mentally represented via imageability. Building on this, extreme abstract expression can be placed on the non-perceptual end of the scale, whereas concrete expressions are located on the perceptual side (Brysbaert et al., 2014). Please note that such an expression can be a word, a phrase or a sentence. However, this thesis focuses on single-word expressions, also referred to as concepts. In the following, examples for concrete concepts are presented: *banana*, *cup*, *drawer*, *pencil* (Pecher et al., 2011; p. 7) and

*bucket* (Wiemer-Hastings and Xu, 2005; p. 720). Examples for abstract concepts are *democracy, power, freedom, majority, republic* (Pecher et al., 2011; p. 4), *emotion, personality traits, actions* (Pecher, 2018; p. 502) and *difference* (Wiemer-Hastings and Xu, 2005; p. 720). It is intuitive that a concrete concept like *banana* refers to an object that can be perceived with at least one of the human senses (see, hear, touch, smell, or taste) (Brysbaert et al., 2014). Therefore, it seems reasonable to assume that the mental representation of concrete concepts bases on sensory-motor experiences of the percipient. Contrary to that, an abstract concept like *democracy* is not perceivable by one of these senses. Here, no senses and therefore less sensory-motor experiences seem to be involved.

Wiemer-Hastings and Xu (2005) argue that a definition of abstractness which bases only on the absence of perceptive attributes is not very expressive. Even if studies suggest a correlation between imageability and concreteness, this seems to be mostly the case for extremely abstract vs. extremely concrete concepts. A further measure for characterization is the availability of linguistic context proposed by Schwanenflugel and Shoben (1983), suggesting that the comprehension of abstract concepts is more dependent on context than it is for concrete concepts. Apart from that, Wiemer-Hastings and Xu (2005) state that the context of abstract concepts varies more and tends to be more complex than the context of concrete concepts. In Wiemer-Hastings and Xu (2005) they find that abstract concepts are characterized by attributes related to subjective perception but have less attributes that are inherent to the concepts. Overall, there is no clear differentiation between abstractness and concreteness but an approximation based on the assumption that the differences in cognitive processes and imageability exist on a scale. Paivio (1971) highlights the assumption that there is no absolute absence or presence of perceptual attributes in mental representations of abstract or concrete concepts, suggesting them to gradually vary between the two extremes.

## 2.2 Theory of Cognition

Within this section, established theories from the field of cognitive science regarding abstractness and concreteness are described. In cognition theory one of the most influential works regarding the differentiation between abstractness and concreteness is *Imagery and Language* from Paivio (1971). According to Paivio (1971), the main difference can be pinpointed to the reference of the concept: while concrete concepts refer to *events* or *objects* that are more likely to evoke an image, the referents of abstract concepts trigger these mental images less. The performance in tasks involving learning and memory is shown to be superior when concrete concepts are used compared to abstract concepts. Paivio (1971) attempts to explain this phenomena with the *dual-coding theory* which states that cognitive processes involves two different systems, a verbal system and an imagery system: while concrete concepts can be stored in memory both verbally and nonverbally (via the imagery system), abstract concepts are rather stored only in the verbal system of the memory. The dual-code theory is also applicable on sentences or phrases, not only words. In the examples

- (1) The boy hit the girl.
- (2) The theory has predictive value. (Paivio, 1971; p. 21)

the first, rather concrete sentence involves human senses and is therefore considered to be both stored in the verbal and non-verbal system. The second, more abstract sentence does not really involve any senses and is therefore considered to be stored solely in its verbal form. As a result, the concrete sentence, which is associated with a higher degree of imagery, is assumed to be better stored in memory. The *Symbol Grounding Problem* proposed by Harnad (1990) discusses the question of how meaning can be obtained by symbols which have no inherent meaning. In other words, how does the linking from arbitrary symbols (words, phrases or sentences) to real life events or objects work. Based on the assumption that meaning is not intrinsic to symbols, Harnad (1990) proposes a *bottom-up* rather than a *top-*



*down* approach which states that symbols obtain their meaning from the ground-up, that is, through the linking of sensory-motor experiences to symbols. In order to acquire the understanding of a symbol, a human needs to be able to perceive real-life events and objects. This approach was further developed by Barsalou (1999), among others, and has become established in cognitive science as the *Grounded Cognition Framework* (Pecher et al., 2011), also known as *Sensory-Motor Grounding* (Pecher et al., 2011) or *Embodied Cognition* (George and Johnson, 1980). In this theory, the ability of perceptual simulation plays an important role in processing mental representations. Concepts are considered as "[...] simulations of sensory-motor experiences which share processing mechanism with perception and action" (Pecher, 2018; p. 501). In terms of concrete concepts, there are many studies nowadays that support this theory (Pecher, 2018).

Even though Harnad (1990) already provides an intuition on abstract concepts, stating that they seem to be rather symbolic instead of being grounded in sensory-motor experiences, most of the literature only considers concrete concepts for the grounded cognition framework (Pecher et al., 2011). The *Conceptual Metaphor Theory*, first introduced by George and Johnson (1980), is an attempt to address the sensory-motor grounding of abstract concepts. Since many abstract concepts have a link to concrete concepts, the theory claims that concrete concepts are partially used in order the mentally represent the abstract concept (Pecher et al., 2011). Considering the concrete concept *high* in the example

- (3) Susan has a high position in the department, (Pecher, 2018; p. 502 f.)

the concrete concept acts as a metaphor and the abstract concept *power* refers to the vertical position *high*, meaning that Susan has a powerful position in the department. The theory suggests that the structure of the mentally represented concrete concept specifies the structure of the abstract concept and therefore partially grounds in the mental representation of the concrete meaning. However, Pecher (2018) mentions some points that challenge the grounded cognition framework. First

of all, both the concrete as well as the abstract concept needs to have some structure since the metaphorical meaning is produced by mapping concrete structure to abstract structure. Assuming the abstract concept has no structure at all, a mapping is not possible. Apart from that, Pecher (2018) argues that many of the abstract concepts have a structure (e.g. *vertical position*) but due to its simplicity it seems to lack providing a good explanation for the high semantic variety of abstract concepts with the same structure. Furthermore, there is literature that challenges the grounded cognition theory even for concrete concepts. Among this literature Tucker and Ellis (1998) present the handling alignment effect which suggests that additional visual input such as the alignment of an object’s handle can influence the perception of participants. When participants perceive an object, motor actions to grasp the objects are not automatically activated, furthermore, an overload of stimuli of the motor system does not affect the memory for these objects (Pecher, 2018). In conclusion, it can be said that many findings support the grounded cognition theory, suggesting at least some involvement of the sensory-motor system in the understanding of concepts, even if some other findings point out that there might be also other factors involved. However, the cognitive mechanisms behind abstract concepts still are not sufficiently examined.

## 2.3 Concreteness Ratings

A well-known corpus that provides concrete and abstract concepts along with ratings on a scale between 1 and 5 can be found in the study of Brysbaert et al. (2014). The concreteness ratings which are publicly available are widely used in studies that investigate the properties of concreteness and abstractness as well as their relationship with various aspects of psychology and language processing, such as memory, reading, and language acquisition (Brysbaert et al., 2014).

In order to estimate the degree of concreteness of a concept, Brysbaert et al. (2014) set up a study to determine the concreteness of circa 40,000 English ex-

<i>Abstract (language based)</i>			<i>Concrete (experience based)</i>	
<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>
<i>N = I do not know this word well enough to give a rating.</i>				

**Figure 2:** Concreteness rating on a scale between 1 and 5 (Brysbaert et al., 2014; p. 906).

pressions (37,058 one-word and 2,896 two-word expressions). They used data from multiple sources such as the SUBTLEX-US corpus, English Lexicon Project, and the British Lexicon Project. For the experiment, they made use of internet crowdsourcing and asked the participants to rate each expression on a scale from 1 (abstract) to 5 (concrete), which means the ratings are based on subjective judgment (see Figure 2). In the instructions they asked the participants to consider the concreteness ratings with regard to the five human senses – while concrete words are perceptual by one of these senses, the degree of sense-based experience decreases with increasing abstractness (Brysbaert et al., 2014).

## 2.4 Text-Based Computational Approaches

There are several approaches using computational models when it comes to the text-based examination of abstract and concrete concepts. A lot of work is done on the automatic corpus-based analysis of context and co-occurrences. It was shown that the presence of context affects the automatic prediction of abstract vs. concrete concepts (Köper and im Walde, 2017). In order to provide a more detailed description of the context of abstract/concrete target words in terms of distributional patterns, semantics and entropy, Naumann et al. (2018) investigate 16,620 nouns, verbs and adjectives, labeled with a concreteness rating according to Brysbaert et al. (2014). Regarding the distributional pattern they find that the context of concrete target words are mostly concrete nouns, abstract verbs and abstract adjectives, whereas the context of abstract target words always are abstract. The investigation of the semantic variety shows a broader context of abstract verbs and adjectives compared

to the concrete ones, while both abstract as well as concrete nouns are more alike. Finally, both for abstract and concrete target words, concrete context shows lower entropy than abstract context.

Tater et al. (2022) takes a step further towards a fine-grained analysis of context, that is, the selectional preference of abstract/concrete target words. Selectional preference refers to the semantic requirements of a target word or phrase on its arguments. Based on some semantic features, Tater et al. (2022) conduct a study to predict the concreteness of target nouns and verbs – looking at several different experiment settings, the performance on target nouns is better than for verbs. Even though co-occurrence features show a better performance than the semantic fine-grained features for binary classification, regression has an improved performance of the selectional preferences for direct objects, suggesting some influence on the prediction of the degree of concreteness.

In general, established methods for the analysis of semantic, syntactic or the distributional relationship between context and target words are *Latent Semantic Analysis* (LSA), *Hyperspace Analogue to Language* (HAL) and *Distributional Semantic Models* (DSMs) (Schulte im Walde and Frassinelli, 2022). Common measures include distributional similarity and clustering methods (Schulte im Walde and Frassinelli, 2022). In summary, studies on the context of abstract and concrete targets have in common that their results show a higher variety in context for abstract than for concrete targets, whereas the context of concrete targets is rather compact and shows higher associations (Schulte im Walde and Frassinelli, 2022).

## 2.5 Multi-Modal Computational Approaches

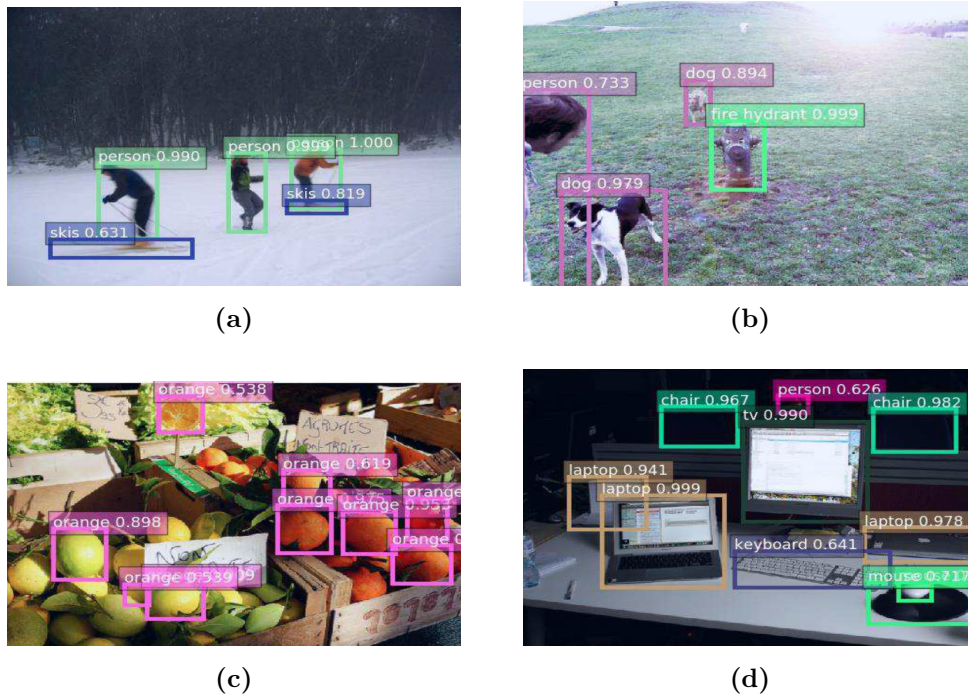
Furthermore, there exist multi-modal computational approaches towards the investigation of abstractness. Bhaskar et al. (2017) investigate the prediction of concrete/abstract nouns with the use of multi-modal models, involving a combination of textual and visual information vs. solely textual or solely visual information. They

refer to the grounding theory which assumes some involvement of perceivable stimuli (e.g. visual input) in cognition during the process of language understanding. This suggests that providing additional input such as images results in increased prediction performance. Although all three variants perform well, up to 96.45% accuracy, they find no significant differences among the model settings. An interesting takeaway from this is that models are able to perform on purely visual input information with an accuracy of 92.79%. Apart from text-based or multi-modal approaches there is no approach that focuses only on image-based methods for the automatic analysis of abstract vs. concrete concepts yet.

### 3 Object Detection

This section elaborates on object detection and provides an overview of the development from traditional detectors to current state-of-the-art methods, alongside the technical background of the network architecture. Object detection is the task of identifying all instances of objects belonging to a certain class within images (Zou et al., 2023; Amit et al., 2020). An example for this can be found in Figure 3 where several classes are outlined by *bounding boxes* which are used to mark the location of an object instance by framing the borders of the instance (Goodfellow et al., 2016). The objects detected in the example include the classes *person*, *skis*, *dog*, *fire hydrant*, *orange*, *laptop*, *chair*, *keyboard* and *mouse*.

The information about the location of an object is only one type of the so-called *pose information* which is used to give a spatial estimation of the object within the image – other possible poses could be a segmentation mask, where a value is assigned to each pixel, or a scale information, depicting the object’s scale (Amit et al., 2020). Object detection models are typically trained on large image datasets in order to build a model for certain object classes (Amit et al., 2020).



**Figure 3:** Examples for object detection on the MS-COCO dataset (Zhou et al., 2018; p. 535).

There are two core epochs when it comes to object detection methods: before 2014 they mostly rely on handcrafted features of predefined classes which have to be searched for in images (Zou et al., 2023). According to Zou et al. (2023), the deep learning methods dominate the era of object detection after 2014. The revolution of techniques is mostly associated with AlexNet, a convolutional neural network for computer vision tasks. Since then, there are significant improvements achieved in terms of computational speed, detection accuracy and model architecture (Zou et al., 2023; Gu et al., 2018). For this reason, this thesis focuses on deep learning methods.

Apart from that, it is important to mention that the common measure for the performance of object detection tools is *Average Precision* (AP), which can be understood as “[...] the average detection precision under different recalls” (Zou et al.,

2023; p. 6). However, this study compares model performance in the experiments section with precision, since the set-up does not allow for counting *False Negatives*, which are necessary for the computation of recall.

### 3.1 One-Stage vs. Two-Stage vs. Multi-Stage Detectors

When it comes to modern methods, there are single-stage detectors (e.g. YOLO and SSD), two-stage detectors (e.g. R-CNN and Fast R-CNN) and multi-stage detectors (e.g. Cascade R-CNN) (Zou et al., 2023). The different types of detectors are presented in the following.

The common architecture of a two-stage detector uses two stages, where in the first stage proposals for regions in the image are generated that are likely to contain an object, also called *Regions of Interest* (RoI) and the second stage, where each proposed region is classified and the bounding boxes are adapted to fit the object more accurately (Carranza-García et al., 2020). As opposed to this, the classifier in single-stage detectors performs both the prediction of the bounding boxes as well as the classification of the objects within one single pass (Carranza-García et al., 2020).

Carranza-García et al. (2020) highlight the fact that while one-stage methods have a higher speed and lower computational costs compared to two-stage approaches, they suffer from lower accuracy. Two-stage detectors on the other hand have higher accuracy but they are not as fast as one-stage methods due to the additional processing step. Apart from that, Cai and Vasconcelos (2018) present a multi-stage detection framework, where each stage performs a specific task in the object detection process. The processing of the input is organized in a sequential manner such that the output of each stage is the input for the following stage. This sequential processing results into improved accuracy compared to single- and two-staged methods and higher flexibility when it comes to tasks other than object detection. However, the downside is that the computational cost is higher and the

speed is lower.

In general, it can be said that the focus in object detection has shifted from two stage detectors to single stage detectors due to its applicability to real-time tasks (Cai and Vasconcelos, 2018). However, multi-stage detectors like Cascade R-CNN are still of interest, especially for domains where accuracy and robustness are prioritized over speed.

## 3.2 Applicability of CNNs for Object Detection

The neural network that is most often used for object detection is the *Convolutional Neural Network* (CNN). This deep learning architecture is particularly suitable for this task since it is able to hierarchically learn feature representations such that both local and global features can be learned (Farabet et al., 2012). As LeCun et al. (2010; p. 253) argue, a network for computer vision tasks “[...] should have multiple trainable stages stacked on top of each other, one for each level in the feature hierarchy”. By doing so, the lower layers can learn simpler structures and with each layer the complexity of the learned structure increases. With this hierarchical feature learning method, a CNN is able to detect objects at different scales and positions in the image (Gu et al., 2018; Farabet et al., 2012; Amit et al., 2020). A further advantage of CNN’s are the sparse interactions due to the convolution – given an image that consists of millions of pixels, a CNN can learn sparse features that consist of only hundreds of pixels (Goodfellow et al., 2016). As Goodfellow et al. (2016) argue, this also results in a lower number of overall operations. Further, the parameter sharing reduces the need of parameter storage and enables the model to be shift-invariant, meaning that if an image is shifted by one pixel to the right, the output still is the same. This is especially important for images where objects can occur only slightly different from each other.



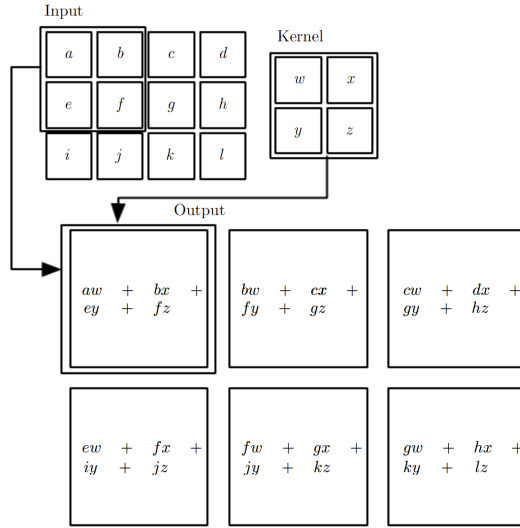
### 3.3 Technical Background of CNNs

In the subsequent part, technical background of CNNs is provided. CNN's can be implemented in a variety of ways but should always consist of three basic components: the *convolutional layer*, the *pooling layer* and the *fully-connected layers* (Gu et al., 2018). The convolutional layer has the task of learning input feature representations, the pooling layer aims to achieve shift-invariance such that small differences in the input data do not affect the output and the fully-connected layers use the prior learned representations and make informed, global decisions about the objects in the image, also taking into account the relationships between the learned representations (Gu et al., 2018). The latter is different from the local decisions in the convolutional layer. In general, the functionality of a CNN can be summarized as a hierarchical stacking of several layers where the input layer is the real-valued tensor of the image (Amit et al., 2020).

Convolution refers to the mathematical operation that maps two functions onto a third function (Goodfellow et al., 2016). The convolution in image processing is typically referred to as 2D-convolution and is formulated as

$$(1) \quad S(i, j) = (I * K)(i, j) = \sum_m \sum_n I(m, n)K(i - m, j - n) ,$$

where  $K$  is a function, also referred to as *kernel* or *filter*,  $I$  is the input image tensor together with its indices  $m, n$  and  $S$  is the output of the convolution at the according indices  $i, j$  (Goodfellow et al., 2016). The asterisk refers to the element-wise functioning convolution operator. A visual representation of the operation is shown in Figure 4. Here the kernel is represented as a  $2 \times 2$  matrix which moves over the  $3 \times 4$  input with a *stride* of 1. This means that the kernel slides by exactly one input value from one grid of input values  $(a, b, c, d)$  to the next grid of input values  $(b, c, f, g)$ . This representation of the convolution shows that the dimension of the output matrix is reduced to  $2 \times 3$  compared to the  $3 \times 4$  input matrix. For each



**Figure 4:** Visual representation of a 2D-convolution (Goodfellow et al., 2016; p. 334).

convolutional layer, different kernels can be used in order to extract several features (Gu et al., 2018). Note that both the kernel as well as the parameter of the fully connected layers are trainable parameters (Amit et al., 2020). After an element-wise non-linear operation is applied to the output of the linear convolution, the result is often referred to as *feature map* (Gu et al., 2018). One single feature  $z$  at the position  $i, j$  in the  $l$ -th layer belonging to the  $k$ -th feature map can be considered as the following linear operation:

$$(2) \quad z_{i,j,k}^l = \mathbf{w}_k^{lT} \mathbf{x}_{i,j}^l + b_k^l .$$

After applying the non-linear activation function to the single feature value  $z_{i,j,k}^l$ , the output is

$$(3) \quad a_{i,j,k}^l = a(z_{i,j,k}^l) ,$$

where  $a(\cdot)$  represents an appropriate activation function such as sigmoid, tanh or

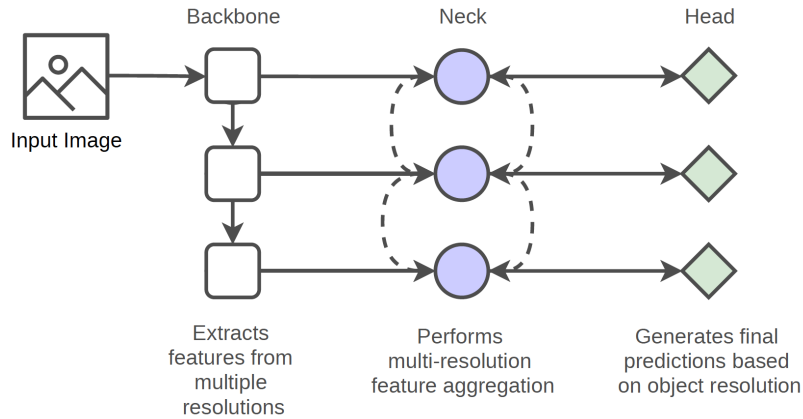
ReLU (Gu et al., 2018). The stage where non-linear activation is performed is sometimes referred to as the *detector stage* (Goodfellow et al., 2016). Next, the pooling function is applied to each feature map in order to reduce its the dimension, resulting in a trade-off between a less detailed spatial resolution and shift-invariance (Gu et al., 2018). The aim is to only keep the most important information in order to have a feature representation which is sparse but adequate enough. The functionality of the pooling layer involves replacing the output at a specific location by a summarized value (Goodfellow et al., 2016). An established pooling function is *max pooling*, where the maximal value per grid of outputs is used for further computations (Goodfellow et al., 2016).

When it comes to deep neural networks, it is important to note that while many networks tend to get deeper, there is still a trade-off between increasing depth resulting in more accurate feature representations on the one hand and increasing complexity possibly resulting in overfitting and a higher computational cost on the other hand (Gu et al., 2018).

### 3.4 Object Detection Model Architecture

Just like CNNs, object detection models can be implemented in several ways, however, they share a common structure which is presented below. An object detection network typically consists of a three-parted architecture including a backbone, a neck and a head (Kateb et al., 2021). As illustrated in Figure 5, the input image is fed to the backbone which extracts features from the image at multiple scales. These different features are aggregated and fused by the neck and subsequently conveyed to the head. Here, the classification of the objects at different levels of abstraction is performed. The single components will be now considered in more detail.

More precisely, the backbone is not only responsible to extract the features but also to keep additional information about the localization and structure of the objects (Kateb et al., 2021). It can be considered as a robust model (e.g. ResNet,



**Figure 5:** General concept of an object detection model (Kateb et al., 2021; p. 4).

ResNeXt, VGG, etc) where the classification layers are removed, basically being responsible for the generation of the feature maps (Kateb et al., 2021; Chen et al., 2019a).

The neck has the task to refine and fuse the feature maps from different resolutions and functions as an intermediary between backbone and head (Kateb et al., 2021). Kateb et al. (2021) state that usually, the neck is implemented as a *Feature Pyramid Network* (FPN) since FPN’s perform well on multi-scale feature fusion tasks. Their functionality is able to both capture fine-grained as well as coarse-grained information of varying sizes and fuses them to meaningful feature maps. For further details on the functionality of an FPN, please refer to section 3.7.

Based on the fused feature maps from the neck, the head finally predicts the confidence scores of the classes and bounding box information. Since several tasks are involved, like performing regression for bounding box refinements and classification of objects, many object detection models make use of several heads.

### 3.5 Comparison of Toolboxes

The upcoming section gives an overview of existing toolboxes for object detection tools and compares them in terms of their performance, the number of tools they offer and other aspects like speed and size. It is convenient to make use of a toolbox where existing models are already implemented when several models are needed. Common frameworks in the area of object detection methods are MMDetection (Chen et al., 2019a), Detectron2 built by Facebook AI Research (FAIR) (Wu et al., 2019), SimpleDet (Chen et al., 2019b) and maskrcnn-benchmark (Massa and Girshick, 2018).

They all differ in terms of quantity and quality of provided tools. While maskrcnn-benchmark provides only very few tools, their performance set the benchmark for competing frameworks in 2018 (Chen et al., 2019a). SimpleDet already offers more models but can not compete with other frameworks when it comes to performance. A comparison between Detectron2 and MMDetection suggests that even if Detectron2 offers more models than SimpleDet, it still can not keep up with the number of tools implemented in MMDetection. Furthermore, there are no regular updates for Detectron2, which means that it is not state-of-the-art anymore. The last update in the model zoo is from July 2021<sup>1</sup>. Since then, there was much progress in computer vision and object detection. In contrast to that, MMDetection is consistently receiving updates.

A comparison of speed, memory and performance of the different frameworks on Mask R-CNN and RetinaNet can be found in Table 1. Overall it becomes clear that MMDetection and maskrcnn-benchmark are similar in terms of training speed, inference time, memory and performance and that both perform better than the other frameworks. Given the above argumentation, this thesis makes use of MMDetection.

---

<sup>1</sup>[https://github.com/facebookresearch/detectron2/blob/main/MODEL\\_ZOO.md](https://github.com/facebookresearch/detectron2/blob/main/MODEL_ZOO.md), Detectron2 model zoo on github (last access: July 05, 2023)

<b>Toolbox</b>	<b>model</b>	<b>Train</b>	<b>Inf</b>	<b>Mem</b>	<b>AP<sub>box</sub></b>	<b>AP<sub>mask</sub></b>
MMDetection	Mask RCNN	0.430	10.8	3.8	37.4	34.3
maskrcnn-b.	Mask RCNN	0.436	12.1	3.3	37.8	34.2
Detectron	Mask RCNN	0.744	8.1	8.8	37.8	34.1
SimpleDet	Mask RCNN	0.646	8.8	6.7	37.1	33.7
MMDetection	RetinaNet	0.285	13.1	3.4	35.8	-
maskrcnn-b.	RetinaNet	0.275	11.1	2.7	36.0	-
Detectron	RetinaNet	0.552	8.3	6.9	35.4	-
SimpleDet	RetinaNet	0.565	11.6	5.1	35.6	-

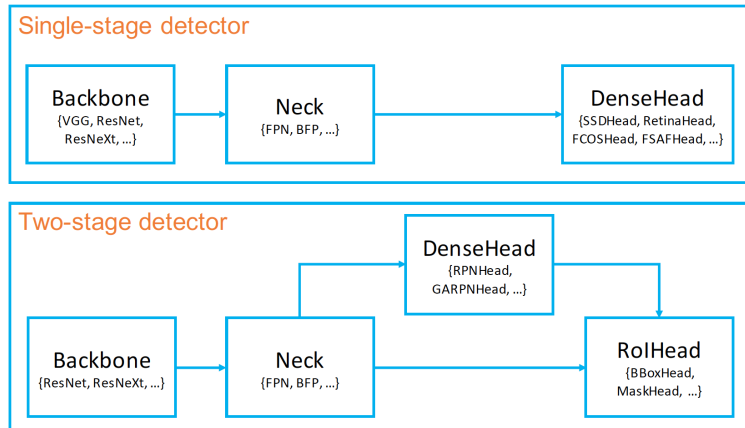
**Table 1:** Comparison of toolboxes in 2018, inspired by Chen et al. (2019a; p. 6). Training speed (Train) is given in iter/s, inference (Inf) in fps (frames per second) and memory (Mem) in GB.

### 3.6 MMDetection

Since the last section argues for the use of MMDetection in this study, this toolbox is now presented according to Chen et al. (2019a). MMDetection is an open-source project contributed by researchers and engineers from various colleges and companies. Originally, the framework began 2018 with the implementation of a team that won the COCO challenge. The codebase is continually being updated since then. The toolbox currently offers besides from algorithms for instance segmentation also more than 43 tools for object detection. Overall, they provide 53 algorithms, 450+ pre-trained models and 6 different datasets<sup>2</sup>.

---

<sup>2</sup>[https://github.com/open-mmlab/mmdetection/blob/main/docs/en/model\\_zoo.md](https://github.com/open-mmlab/mmdetection/blob/main/docs/en/model_zoo.md), MMDetection model zoo on github (last access: July 05, 2023)



**Figure 6:** Model architecture of single-stage and two-stage detectors in MMDetection (Chen et al., 2019a; p. 4).

The codebase is characterized by modular design, efficient training and a high variety of provided tools, including state-of-the art tools. As part of the OpenMMLab, a computer vision algorithm system implemented with PyTorch, MMDetection also uses PyTorch. The OpenMMLab is maintained by the Multimedia Laboratory of the Department of Information Engineering (MMLab) at The Chinese University of Hong Kong and is considered as one of the most sophisticated and influential institutions in the field of AI with a focus on computer vision and deep learning<sup>3</sup>.

As illustrated in Figure 6, the implementation of the single- and two-stage detectors in MMDetection is based on the common architecture containing a backbone, neck and head. One specificity of the architecture is that the head is implemented as a *DenseHead* which can be considered as an extension of the standard head (Chen et al., 2019a). The *DenseHead* enables dense connections between feature maps across different scales such that multi-scale feature fusion tasks can be performed (Lin et al., 2017b). The two-stage detectors in MMDetection contain additional head

---

<sup>3</sup><https://mmlab.ie.cuhk.edu.hk>, Multimedia Laboratory, The Chinese University of Hong Kong (last access: July 05, 2023)

components, the *RoIExtractor* and the *RoIHead*. The *RoIExtractor* is responsible for the extraction of features from feature maps on the RoI-level, meaning that it aligns extracted features with RoIs (Chen et al., 2019a). Furthermore, the *RoIHead* processes the RoI features and performs classification, regression or segmentation. In summary, it can be said that two-stage detectors divide the task of RoI-proposal generation from the classification task, while single-stage detectors predict bounding boxes and class probabilities directly from the feature maps.

### 3.7 Tools

This section first provides a general overview of the evolution of object detection tools and then elaborates on the tools that are used in the experiments. As already mentioned in the beginning of section 3, the early object detection methods are characterized by handcrafted features. After the breakthrough of AlexNet, the detectors diverged into single-stage and two-stage detectors. The following provides a general overview of the evolution of the tools according to Zou et al. (2023).

In 2014, the first two-stage detector *R-CNN* (Region-Based Convolutional Neural Network) was published (Girshick et al., 2014). The main steps of the functionality of R-CNNs can be summarized by three steps. First, RoI proposals are extracted. Following, the proposals are rescaled such that all proposals share a fixed size and can be fed into a CNN model for feature extraction. Finally, classification is performed for each RoI which results in the prediction of a class label as well as the absence or presence of an object per RoI. While the accuracy is good, R-CNN suffers from high computational costs and high training and inference time. Especially the redundancy due to the high overlap of RoI proposals and the rescaling of each RoI proposal are responsible for the slow speed.

Over the time, new models evolved based on R-CNN with the attempt to solve the speed issue. After R-CNN was published, *SPPNet* (Spatial Pyramid Pooling Network) was proposed by He et al. (2014). In SPPNet, the need for rescaling the



size of RoI proposals is abolished, resulting in a much faster detection speed while maintaining the same accuracy. The downside of SPPNet is that only the top layers are considered for fine-tuning and all other layers are ignored.

After that, Girshick (2015) were responsible for another detection speed-up by introducing *Fast R-CNN*. This is followed by *Faster R-CNN* (Ren et al., 2015) which introduced the *Region Proposal Network* (RPN), enabling the generation of RoI proposals with a CNN model. However, the redundancy arising from the overlapping RoI proposals was still not solved.

In 2017, Lin et al. (2017a) came up with the idea of an FPN in order to address the issue of detecting different objects at various scales or sizes within an image. Previous methods used single-scale feature representations. As opposed to this, the FPN provides a set of multi-scale feature maps (Lin et al., 2017b): it extends a regular network by a top-down approach with lateral connections such that on each level of the pyramid, objects of varying scales can be detected. The feature pyramid that is generated from the input image consists of feature maps from various resolutions, where higher-level feature maps contain more coarse-grained information and lower-level feature maps capture rather fine-grained information.

During this time there was also a great development in single-stage detectors. The first one was *YOLO* (You Only Look Once), proposed by Redmon et al. (2016). This method finally tackled the problem of redundancy by performing both classification of the object label and bounding box regression within one single pass through the CNN. The authors move away from the two-parted view of having a separate stage of the generation of RoI proposals and another stage for classification and regression. In contrast to that, they implement the idea that the image is processed only once by the network. This results in a significantly higher detection speed compared to two-stage detectors but also worse performance in accuracy. Based on YOLO, R. Joseph developed further variants (YOLOv2 and YOLOv3) which led to further advancements. The second single-stage tool, *SSD* (Single-Shot Detector) (Liu et al.,

2016), introduced the ability to detect objects of multiple sizes and at multiple scales. Detection is not only performed on top layers of the network – different layers of the network process objects of different scales. Liu et al. (2016) achieved higher accuracy with SSD than YOLO was able to perform. While YOLO struggles with the detection of small objects, SSD performs better on them. However, accuracy still was not able to beat those of the two-stage detectors.

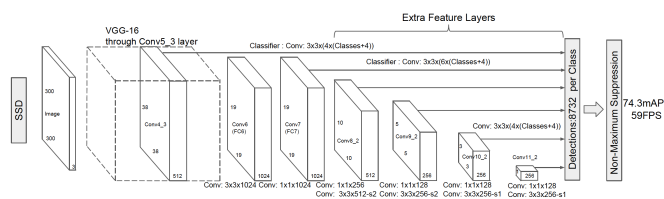
This accuracy issue was addressed by Lin et al. (2017b) with *RetinaNet*. They found that the class imbalance between foreground and background is a problem for former detectors, in other words, the number of positive object examples (in the foreground) are overbalanced by negative object examples (in the background). Their approach included the introduction of a novel loss function, the *focal loss* which redistributes the weights towards complex examples. This leads to a shift in focus towards difficult objects (in the foreground) and away from easy objects (in the background).

In summary, it can be said that R-CNN and its variants, YOLO, SSD and RetinaNet became established models in object detection and formed the basis for many advancements and developments of variants. Since then, also attempts in the development of multi-stage detectors have been made. The following is a more detailed overview of the functionality of the tools that are examined in this thesis. These tools are particularly interesting since they are state-of-the-art methods and contributed to the development of the model architecture in object detection.

### 3.7.1 SSD

SSD, a single-stage detector, is known for its simple architecture which enables fast and efficient detection of multi-scale objects (Chen et al., 2019a). The following summarizes the model architecture of an SSD according to Liu et al. (2016). First, the input image is fed to a basic CNN (in this example a VGG-16 network, functioning as the model’s backbone) which extracts bounding boxes and class probabilities

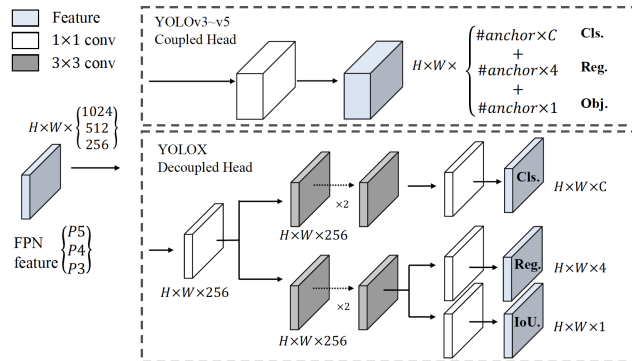
from it. As illustrated in Figure 7, there are additional feature layers stacked on the base network, aiming for an extraction of features at multiple scales. For each feature map, default boxes with varying aspect ratios are defined. These aim to achieve scalability and are comparable to the anchor boxes in Faster R-CNN. The model is trained by predicting and adjusting the class labels and bounding boxes. The loss function bases both on the confidence loss as well as the localization loss. *Non-maximum suppression* as a last step discards irrelevant features and keeps relevant ones.



**Figure 7:** Model architecture of SSD (Liu et al., 2016; p. 4).

### 3.7.2 YOLOX

The single-stage detection tool YOLOX is a further development of the YOLO series. Ge et al. (2021) use YOLOv3-SPP as backbone, which is a combination of an SPP layer and DarkNet53. The authors attempt to solve the issue of the simultaneous performance of classification and regression by applying a decoupled head. While YOLOv3-v5 still use a coupled head, as displayed in Figure 8, YOLOX separates the regression and classification tasks. Another specificity of YOLOX is that no anchors are used. An anchor-free approach reduces the number of parameters and simplifies the detection process. Overall it can be said that YOLOX is a modern implementation of the YOLO versions with a high detection speed, while maintaining efficiency.

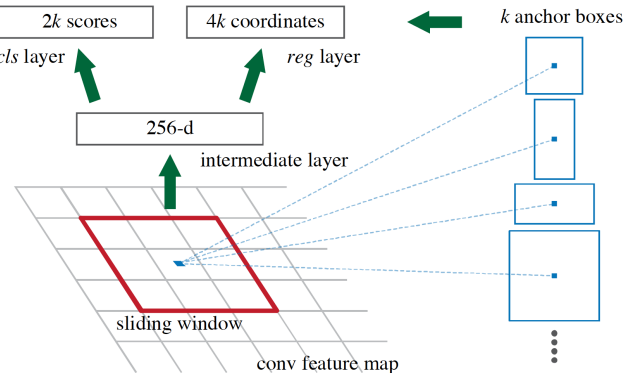


**Figure 8:** Model architecture of YOLOX (Ge et al., 2021; p. 3).

### 3.7.3 Faster R-CNN

The main contribution of the two-stage detector Faster R-CNN is the introduction of an RPN. Ren et al. (2015) significantly improve the speed of the detection process by implementing the RPN into the algorithm. Figure 9 shows the functionality of an RPN. The input image is fed to the network which processes it and outputs regions together with object proposals and an objectness score. This score measures the likelihood that a region proposal contains a certain object from the foreground rather than the background. The region proposals are generated by sliding a network over the feature map which is the output of the last shared layer between the RPN and the object detection network. Like SSD and RetinaNet, this approach also makes use of a set of anchor-boxes of various scales and aspect ratios. For each sliding location, a set of nine different anchor-boxes is used with three different scales and three different aspect ratios. Finally, the output of the RPN is used by the detection network in order to perform classification and box regression. Ren et al. (2015) use the implementation of a basic Fast R-CNN as detection network.

In summary, the RPN enables a significant detection speed-up compared to Fast-RCNN, SPPNet and R-CNN.



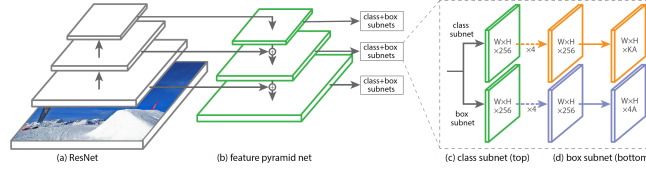
**Figure 9:** Visualization of a *Region Proposal Network* (RPN) (Ren et al., 2015; p. 3).

### 3.7.4 RetinaNet

RetinaNet, a single-stage detector from Lin et al. (2017b) is especially known for introducing a new loss function in order to address the extreme foreground-background class imbalance. The backbone of RetinaNet is an FPN built on top of *ResNet* (Residual Network). The layers of ResNet are considered as residual functions which have reference to the input layers (He et al., 2016). They address the question if increasing depth of a network per definition results in a better network. He et al. (2016) suggest that this is not the case and construct a deep network not only by simply increasing the number of layers but also letting the layers fit a mapping. For more information on the functionality of ResNet, refer to He et al. (2016).

As indicated in Figure 10, the FPN extends a CNN by a top-down approach on each level of the pyramid, producing multi-scale feature maps. Like in SSD, there are anchor boxes in RetinaNet. They are implemented at different locations and levels in the feature pyramid at three aspect ratios:  $\{1:2, 1:1, 2:1\}$ . In addition to the backbone, Lin et al. (2017b) make use of two subnetworks, with each performing a specific task. On the one hand, the classification subnet predicts the class probabilities of object candidates at each level of the feature pyramid for each anchor box. On the other hand, the box regression subnet aims to refine the anchor boxes

which potentially contain objects. Note that the subnets are small *Fully Convolutional Networks* (FCNs) at each pyramid level, which divide an image into regions and assigns a class to each pixel.



**Figure 10:** Model architecture of RetinaNet (Lin et al., 2017b; p. 2984).

For training, the focal loss is computed in order to put less focus on the training of objects that are easily predictable but more focus on hard examples. The focal loss is built on the *cross entropy* (CE) loss which is defined for a binary classification scenario as

$$(4) \quad \text{CE}(p, y) = \begin{cases} -\log(p) & \text{if } y = 1 \\ -\log(1 - p) & \text{otherwise,} \end{cases}$$

where  $y \in -1, 1$  denotes the ground truth label and  $p \in [0, 1]$  describes the class probability of the model for  $y = 1$ . For better readability, Lin et al. (2017b) define

$$(5) \quad p_t = \begin{cases} p & \text{if } y = 1 \\ 1 - p & \text{otherwise} \end{cases}$$

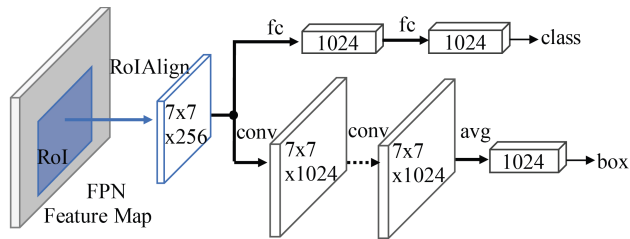
and achieve the binary cross-entropy in the form of  $\text{CE}(p_t) = -\log(p_t)$ . Now, a *modulating factor*  $(1 - p_t)^\gamma$  is incorporated into the equation together with a *focusing parameter*  $\gamma \geq 0$  which is tunable. Furthermore, the focal loss is  $\alpha$ -balanced, meaning that it balances the influence of easy/hard cases. With these modifications, the focal loss in RetinaNet is defined as

$$(6) \quad \mathbf{FL}(p_t) = -\alpha_t(1 - p_t)^\gamma \cdot \log(p_t) .$$

Depending on the prediction of the model, the modulating factor reduces or increases the focus on the training of a given example. In cases where  $p_t$  is small on a misclassified data point, the modulating factor is close to 1 and does not impact the loss. If  $p_t$  approaches 1, meaning the example was classified well, the modulating factor is near 0 and the loss has a low weight. In essence, the design of RetinaNet balances the high detection speed characteristic of single-stage detectors with the strong accuracy performance often associated with two-stage detectors.

### 3.7.5 Double-Head R-CNN

Wu et al. (2020) implemented an extension of the two-stage detector Faster-RCNN, called Double-Head R-CNN. They make use of the FPN as a backbone which outputs region proposals and object features at different levels in the feature pyramid (see Figure 11). The features are extracted using *RoIAlign*, a feature extraction method that is more precise than other methods like max pooling which suffer from the mapping of real-valued coordinates to integers (He et al., 2017). Following, the resulting feature maps are transformed by the heads for further processing. As the name indicates, the architecture involves two heads: the *fully connected head* (fc-head) and the *convolution head* (conv-head). The former consists of two fully connected layers which are responsible for the prediction of the class probabilities. The latter consists of three different components and aims to refine the bounding box coordinates. Both heads are trained jointly using an RPN. The double-head approach separates the tasks of classification and localization by using specific heads that perform well on the according tasks. Therefore, Double-Head R-CNN gains accuracy compared to other baselines with a single head.



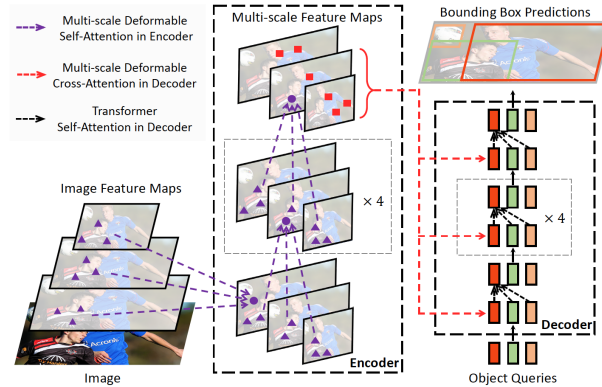
**Figure 11:** Model architecture of Double-Head R-CNN (Wu et al., 2020; p. 10186).

### 3.7.6 Deformable DETR

In 2020, Carion et al. (2020) introduced the first transformer based object detection tool named DETR (DEtection TRansformer). The architecture is a combination of a CNN and encoder-decoders from transformers. They aimed for an end-to-end object detection algorithm where no need for anchors, rules or NMS is. However, DETR suffers from a low detection speed and low accuracy. This is why Zhu et al. (2020) implemented an extension of DETR, Deformable DETR. First of all, an input image is fed to the backbone in order to extract feature maps, as displayed in Figure 12. Following, the encoder processes the extracted feature maps and outputs object specific keys and queries which again are processed by the decoder. The decoder uses cross-attention and self-attention modules in order to perform prediction.

Especially the slow detection speed issue in DETR can be ascribed to the *transformer attention module* which includes every possible location in its attention computation. This is achieved by introducing a more flexible attention computation method, the *deformable attention module*. Here, the module only considers a limited number of locations around a certain point. Further, Zhu et al. (2020) extend the module to a multi-scale deformable attention module and apply it to the encoder and decoder. The modifications lead to a higher efficiency and a speed-up in the detection process compared to DETR.



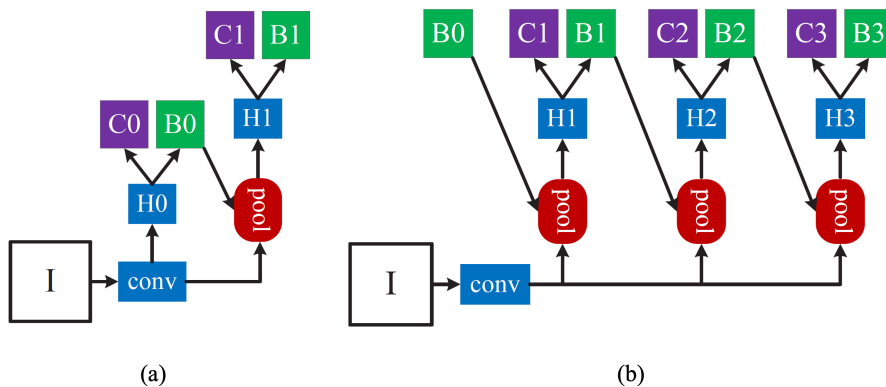


**Figure 12:** Model architecture of Deformable DETR (Zhu et al., 2020; p. 2).

### 3.7.7 Cascade R-CNN

Cai and Vasconcelos (2018) developed the multi-stage detector Cascade R-CNN based on the idea of Faster R-CNN. Figure 13 (b) shows the network architecture of Cascade R-CNN. It aims to tackle the regression issue of Faster R-CNN, as displayed in Figure 13 (a). Here, a single regressor is responsible for the bounding box regression at all levels of quality.  $H0$  refers to a proposal sub-network that generates object proposals  $C$  (classification score) and  $B$  (bounding box) from image  $I$ . These proposals are further processed by a second sub-network  $H1$  which outputs the final  $C$  and  $B$ . In contrast to that, Cascade R-CNN introduces a cascade of task-specific regressors. The cascaded regression approach involves a *resampling mechanism*. Instead of relying on the low-quality output of the RPN, which makes it difficult to post-process proposals for high-quality predictions, Cascade R-CNN resamples the example distribution in order to successively keep positive examples and filter out negative examples. In other words, resampling enables to train the model on increasingly difficult examples and to keep its performance constant even with increasing detection quality. While in the first detection stage all examples are considered in training, the subsequent stages are left with increasingly hard examples. The cascaded architecture makes the algorithm especially suitable for tasks that require

high-performance object detection.



**Figure 13:** Model architecture of (a) Faster R-CNN and (b) Cascade R-CNN. Faster R-CNN makes use of only one regressor, while Cascade R-CNN uses a cascade of task-specific regressors (Cai and Vasconcelos, 2018; p. 6157).

Overall it can be said that each tool has its strength and weaknesses and are applicable for different tasks. It seems like there is not the perfect tool for each application, rather it is important to keep specific design-choices in mind when choosing a tool for a particular task.

## 4 Experiments

The previous sections present a detailed background on object detection and the distinction between abstractness and concreteness. The following presents the experiments that are conducted.

### 4.1 Dataset

In order to construct an appropriate image dataset for the experiment, additional information on the abstractness and concreteness of images is needed. Therefore, the 1000 concepts from Schulte im Walde and Frassinelli (2022) are used, of which

500 are associated with concreteness and 500 with abstractness. The concepts are all nouns and are rated by human annotators as being very concrete or very abstract, at the extremes of the concreteness scale according to Brysbaert et al. (2014).

The images are downloaded automatically using the Bing API from Frassinelli (2023). The download process included a similarity check to avoid obvious duplicates using a cosine similarity measure for every pair of images per concept. Images with a similarity above a threshold of 0.95 are discarded.

The images come with labels on the image-level, denoting the concept that was searched for with Bing. There is no further information on the object labels, number of objects per image or position of the objects. For each image-level label there exist 25 images. After the concepts from Schulte im Walde and Frassinelli (2022) are assigned to the labels of the images, this results in an intersection of 500 (abstract) and 498 (concrete) nouns/labels. The two missing concrete concepts are removed from the image dataset due to explicit content. Further, another automatic check for duplicates is executed, based on the comparison of pixels. For the abstract images, 4 duplicates are found and replaced by additional images with the same label that are provided by Diego (2023).

Finally, the dataset consists of  $500 \times 25 = 12,500$  abstract labeled images and  $498 \times 25 = 12,450$  concrete labeled images, resulting in 24,950 labeled images in total. Additional to the image-level label, the images also have a binary label denoting the abstractness or concreteness  $\{1,0\}$ . Even if there is a small difference between the numbers of abstract and concrete data points, no images or concepts are removed from the abstract data. This decision bases on the aim of the thesis to construct a reusable dataset and to not loose information for further applications. However, since the dataset is applied to pre-trained models, the dataset is not considered as being unbalanced. For the evaluation part, the same number of abstract and concrete data-points is used in order to ensure balanced ratings.

## 4.2 Model Set-up

For this thesis, seven object detection tools are compared using the MMDetection framework. As single-stage detectors SSD (2016), RetinaNet (2017) and YOLOX (2021) are chosen. The two-stage detectors used for comparison are Faster R-CNN (2015), Double-Head R-CNN (2020) and a two-stage variant of Deformable DETR (2020). Cascade R-CNN (2018) is the only multi-stage detector considered in this thesis.

MMDetection provides models trained and tested on the MS-COCO 2017 dataset which is known to be challenging and popular in the field of object detection (Chen et al., 2019a). These results are made publicly available. For most of the models, they provide several weight files with different parameter settings to choose from. The weight files which achieved the highest AP on object detection tasks are chosen. Please note that the criterion of choosing those weights with the highest scores is time-dependent and may lead to deviations from the scores which are the highest in the period from March to June 2023. This is due to the fact that the models are constantly being updated and newer, faster and more accurate versions of weight files are provided. An overview of all seven tools and their parameters such as AP, backbone, dataset on which the model was trained on, epochs and the size of the weight files are presented in Table 2.

From the data in Table 2, it is apparent that the lowest performing tool is the single-stage detector SSD while the highest is the single-stage detector YOLOX. The weights of all tools are trained on the COCO dataset. While Cascade R-CNN is trained with the lowest number of epochs (20), YOLOX is trained with the highest number of epochs (300). The size of the weight files range from 138 MB (SSD) to 487 MB (Cascade R-CNN).

Looking at the backbones, it becomes clear that three tools make use of the

---

<sup>4</sup>[https://github.com/open-mmlab/mmdetection/blob/main/docs/en/model\\_zoo.md](https://github.com/open-mmlab/mmdetection/blob/main/docs/en/model_zoo.md), MMDetection model zoo on github (last access: July 05, 2023)

Model	AP <sub>box</sub>	Backbone	Dataset	Epochs	Weights
SSD	29.5	VVG16	COCO	120	138
RetinaNet	41.6	X-101-64×4d-FPN	COCO	36	367
YOLOX	50.9	YOLOX-x	COCO	300	379
Faster R-CNN	43.1	X-101-64x4d-FPN	COCO	36	381
D-H R-CNN	40.0	R-50-FPN	COCO	12	181
Deform. DETR	46.8	R-50	COCO	50	158
Cascade R-CNN	44.5	X-101-64x4d-FPN	COCO	20	487

**Table 2:** Overview of the tools and their performance on MS-COCO according to the model zoo<sup>4</sup> of MMDetection. Also, parameters of the training that are used for inference are specified. Note that weight-file size (Weights) is given in MB.

same backbone, X-101-64x4d-FPN. This refers to the backbone *ResNeXt*, which is an extension of ResNet used in RetinaNet (Xie et al., 2017).

The name of the backbone consists of an X, indicating that it is a variant of ResNeXt as well as the number 101, referring to the number of layers in the network and 64x4d-FPN. Xie et al. (2017) introduce the concept of *cardinality* which can be considered as a measure of the size of the set of transformations. This dimension exists in addition to factors like depth and width. The 64 refers to a design choice regarding the cardinality and 4d indicates that they consist of 4 channels. For more information, refer to Xie et al. (2017).

The backbone VGG16 relies on the architectures of VGG (Visual Geometry Group), proposed by Simonyan and Zisserman (2015). It is a simple and classic CNN with a uniform design. In this case, it consists of 16 layers. Apart from that, the backbone YOLOX-x bases on the CSPDarknet backbone with some modifications like a higher factor for the depth and the width of the network.

Both R-50-FPN as well as R-50 base on the backbone ResNet. However, they are different variants of ResNet. While R-50 refers to the ResNet model consisting of

50 layers, R-50-FPN has an FPN built on top of the 50-layers-deep ResNet model. This ensures that multi-scale features can be found.

### 4.3 Image Inference

This study compares the seven tools with respect to model performance and analyses quantitative and qualitative properties of detected objects in relation to abstractness and concreteness. In order to achieve this, image inference with MMDetection is performed.

#### Function Arguments

MMDetection takes three arguments for building a model: a configuration file of the specific tool, an according checkpoint file containing the pre-trained weights and the specification of device (cpu or cuda). The latter is set to cpu. The config file contains various settings and hyperparameters that additionally define design choices of the model. There is one metafile.yml for each tool which provides information on test results of these settings.

#### Code Adjustments

The implementation for this experiment includes several adjustments. When iterating over the images, a threshold for the confidence score can be entered. That means that during the object detection process for one image, only objects are kept that are above a certain threshold of confidence. Across all images and tools, a threshold of 0.1 is chosen in order to keep every possibly interesting object but keep the size of the output feasible at the same time. Furthermore, the choice of class labels for image inference is part of the self-implemented image-iterator. Considering the fact that the tools are trained on the MS-COCO 2017 dataset, the same classes are also used for the inference task. They are from Lin et al. (2014) and include 80 labels. A list of all class labels can be found in the appendix (see Table 16). The implementation specifies one more time that it only allows for images that end with jpg in order to standardize the image quality and therefore the quality of image inference.

## Output

Apart from that, the model output is handled in two separate ways: on the one hand, a txt file is generated keeping all inferred information and on the other hand, the coordinates of the bounding boxes are used together with the original image to draw the bounding boxes on a copy of the image and store it. In the txt file, the following tab-separated information is kept:

1. concept
2. image name
3. label
4. bounding box coordinate x1
5. bounding box coordinate y1
6. bounding box coordinate x2
7. bounding box coordinate y2
8. confidence score
9. concreteness (0 or 1)

## 4.4 Results (Experiments)

When it comes to the comparison of model performance for abstract and concrete concepts, there are several aspects to analyse. These aspects are elaborated in the following.

**I. Objects per abstractness/concreteness.** First of all, the analysis is approached from a very general point of view. A general approach is to check how many objects are found in total per abstractness/concreteness. There are 852,858 objects found for abstract images and 963,581 objects for concrete images, resulting in a total number of 1,816,439 objects. Accordingly, 110,723 more objects are detected for concrete concepts which are 12.983% more than for abstract concepts.

**II. Objects per abstractness/concreteness and image.** A more detailed approach is the analysis of the number of objects found per image, both for abstract

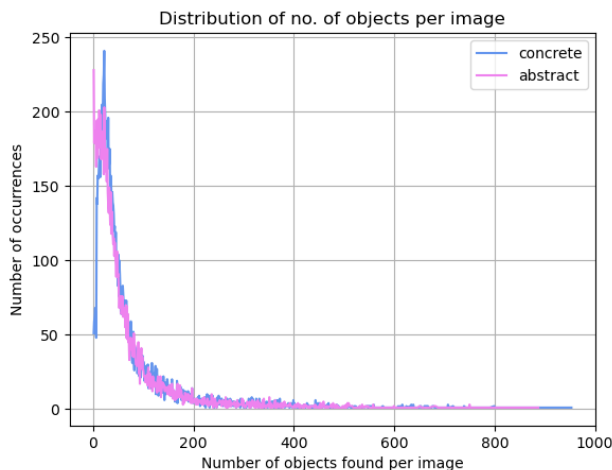
and concrete concepts. The large-scale distribution ranging from 0 to 1000 (detected objects) can be found in Figure 14 where both the abstract (pink) and concrete (blue) observations are recorded. It is clear from the Figure that the most interesting differences are at the extremes of the distribution, especially between 0 and 50. A more fine-grained analysis can be found in Figure 15. While abstract concepts show a high peak for low numbers of detected objects, having their maximum between 0 and 50, the opposite is the case for concrete concepts, starting with their local minimum of occurrences. The values of the extreme occurrences are summarized in Table 3 and the values for mean, median and standard deviation are illustrated in Table 4.

	Concrete	Abstract
Max	953	887
Min	1	1

**Table 3:** Maximal and minimal number of objects found per image for abstractness and concreteness.

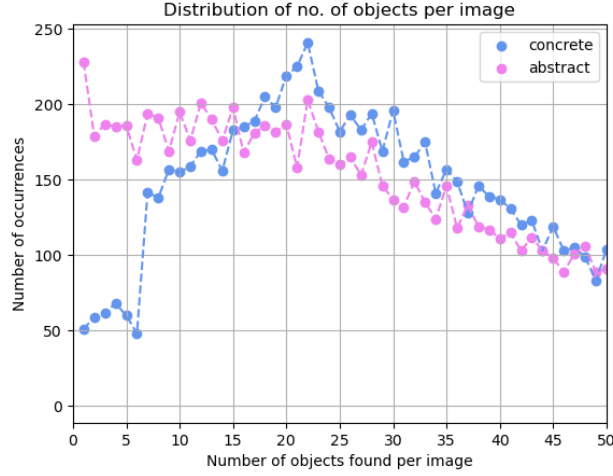
	Mean	Median	SD
Concrete	334.5	309.0	216.9
Abstract	306.2	288.0	195.7

**Table 4:** Statistical analysis on the distribution of detected objects per image for abstractness and concreteness.



**Figure 14:** Number of objects found per image for abstractness (pink) and concreteness (blue), ranging from 0 to 1000 objects per image (large scale).





**Figure 15:** Number of objects found per image for abstractness (pink) and concreteness (blue), ranging from 0 to 50 objects per image (small scale). Please note that the points indicate that the distribution is discrete and not continuous.

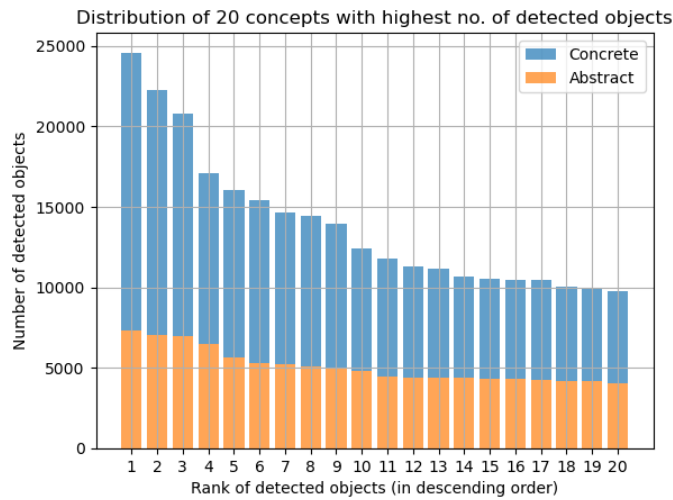
**III. Objects per abstractness/concreteness and concept.** In addition to the distribution of objects per image, the distribution of objects per concept is examined. Since there are 1000 concepts in total, it is particularly interesting to elaborate on the top 20 concepts in terms of highest and lowest object counts. A comparison for abstract and concrete concepts are displayed in Figure 16 for the highest 20 object counts per concept and Figure 17 for the 20 lowest object counts. Please refer to Tables 18 and 17 in the appendix for a list of the according 40/40 concepts. Moreover, the maximum and minimum object counts per concept are summarized in Table 5. Further, the mean, median and standard deviation are measured to gain insight into the overall distribution, which is too large to plot meaningfully. They are summarized in Table 6.

	Concrete	Abstract
Max	17297	7284
Min	176	356

**Table 5:** Maximal and minimal number of objects found per concept for abstractness and concreteness.

	Mean	Median	SD
Concrete	1935.0	1413.5	1825.0
Abstract	1705.7	1387.0	1069.7

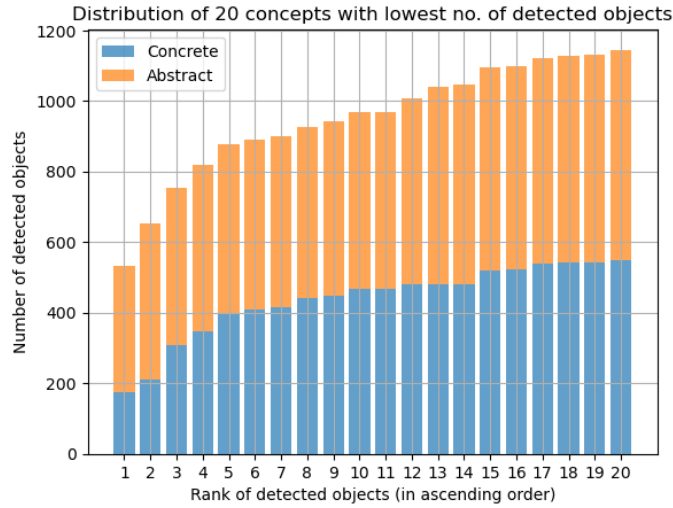
**Table 6:** Statistical analysis on the distribution of detected objects per concept for abstractness and concreteness.



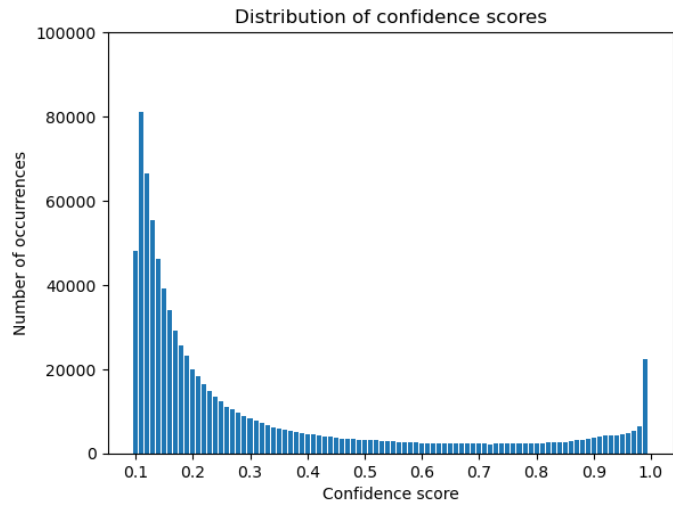
**Figure 16:** Ranking of the 20 concepts with the highest number of detected objects for abstractness (orange) and concreteness (blue).

**IV. Confidence scores.** When it comes to object detection, the distribution of the confidence scores is an important aspect. The Figures 18 and 19 give an overview of how the confidence score of detected objects is spread across abstract and concrete concepts. The highest number of occurrences can be observed for the confidence score of 0.1, both for abstract (81,221 occurrences) and concrete (90,553 occurrences). The minimum number of occurrences for abstract concepts is 0.72 with 2,218 occurrences and for concrete 0.74 with 2,560. Generally, a trend for both categories can be observed, that is, the accumulation of high occurrences in the range

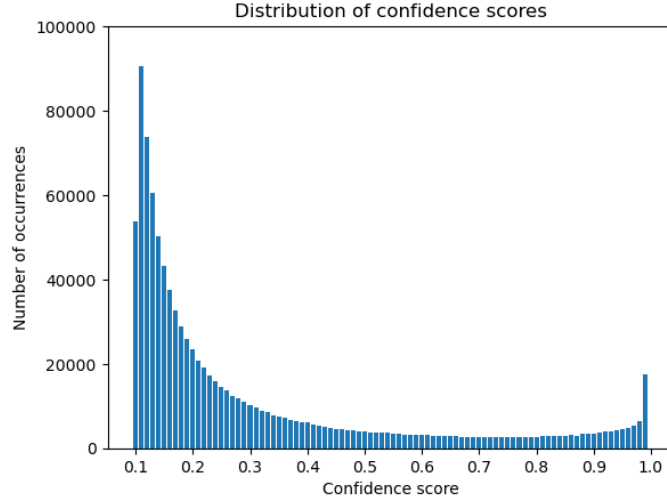
of  $[0.1, 0.3]$  and for a score of 0.99. In the range of  $0.3 < x < 0.99$ , the occurrences are similarly low for both distributions with a tendency to decrease until a minimum point of approximately 0.7 and slightly increase from there again.



**Figure 17:** Ranking of the 20 concepts with lowest number of detected objects for abstractness (orange) and concreteness (blue).



**Figure 18:** Distribution of confidence scores (abstract).

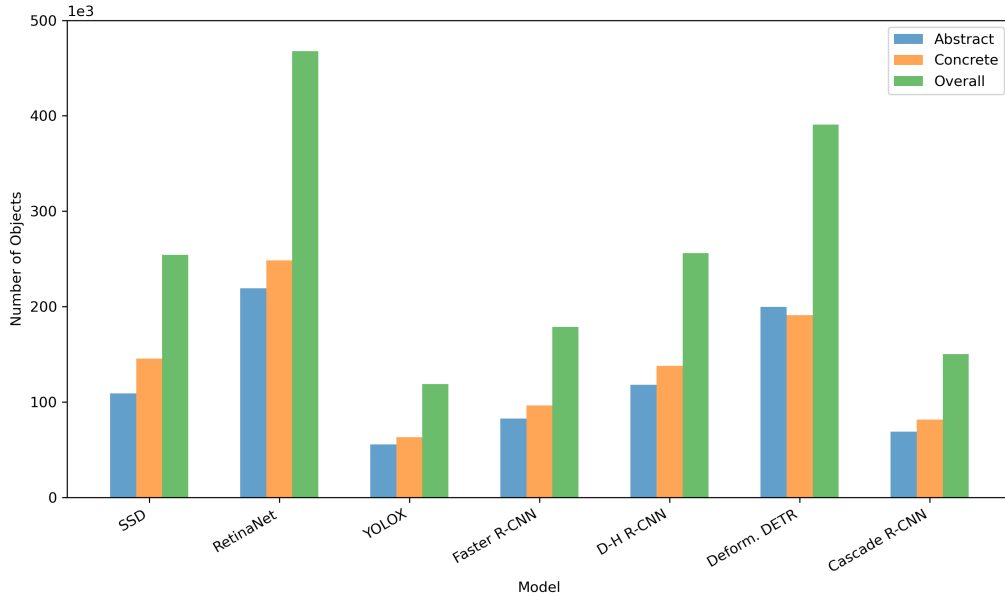


**Figure 19:** Distribution of confidence scores (concrete).

**V. Objects per abstractness/concreteness and model.** Furthermore, it is interesting to additionally consider the distribution of detected objects per model. The question of how many objects are found per model is answered in Table 7, which summarizes the numbers and Figure 20, a visualization of the Table. For all factors, that is, *i*) the number of objects found for abstract concepts, *ii*) the number of objects found for concrete concepts and *iii*) the total number of objects found, RetinaNet produces the highest numbers. For the same factors, YOLOX reports the lowest numbers. From this follows that both extreme variants belong to the category of single-stage detectors. While for almost all models the object counts associated with abstractness are lower than for concreteness, this is different for Deformable DETR. Here, the counts for abstract concepts are higher than for concrete ones. The order of overall counts is in descending order: RetinaNet, Deformable DETR, Double-Head R-CNN, SSD, Faster R-CNN, Cascade R-CNN and YOLOX.

Model	Abstract	Concrete	Total	%
SSD	108,863	145,507	254,370	14.03
RetinaNet	219,135	248,533	467,668	25.76
YOLOX	55,687	63,151	118,838	6.55
Faster R-CNN	82,476	96,231	178,707	9.84
D-H R-CNN	118,093	137,851	255,944	14.10
Deform. DETR	199,819	190,898	390,717	21.48
Cascade R-CNN	68,785	81,410	150,195	8.25

**Table 7:** Overview of the number of objects found per tool for abstract and concrete concepts as well as the total count. The %-column assigns the relative proportion of the object counts with respect to the total of 1,816,439 detected objects.



**Figure 20:** Comparison of the models and their object counts for abstractness and concreteness as well as the total value. The number of objects is given in thousands (1e3).

## 5 Evaluation and Interpretation of Results

Within this section, the set-up of the evaluation of the experiments is presented and the results are discussed. As evaluation method for the output of the different models, manual evaluation through Amazon Mechanical Turk (AMT) is chosen. AMT is a crowd-sourcing platform operated by Amazon which allows requesters to outsource tasks like data annotation and the participation in surveys<sup>5</sup>. A requester can upload one *batch* or several batches which means the collection of multiple single tasks of the same type. In the context of AMT, these single tasks are called *HITs* (Human Intelligence Task). The HITs for this thesis consist of two subtasks. The first task is to decide if the provided label depicts the object surrounded by the bounding box. In the second task, the workers are asked to judge on a scale between 1 (Good) and 4 (Bad) how well the object is surrounded by the bounding box.

### 5.1 Evaluation Set-up

There are several aspects to consider for the evaluation of the data, especially when only a subset can be evaluated. The following elaborates on the selection of a representative subset from the entire dataset and other settings like the number of annotators, concluding with the batch layout used in AMT.

**Number of concepts, images, annotators.** In order to obtain a representative evaluation of the tools and keep it feasible at the same time, it is important to choose a representative and reasonable subset of the model output. Therefore, 20 abstract and 20 concrete concepts are used per tool. Since the total number of detected objects per image varies between 1 and 953/887 (concrete/abstract), we decide to include three different objects per image to increase variety.

---

<sup>5</sup><https://www.mturk.com>, AMT (last access: August 1, 2023)

The number of annotators per image is set to nine<sup>6</sup>. These set-up choices result in a number of 7,560 images being annotated. The 840 images are split into 56 batches consisting of 15 images from the dataset and one additional image functioning as a randomized check for annotation quality. The goal was to have a total of 7,560 judgements for the model output and 8,064 judgements including the check in the end. Please find the summary of the calculations in equations (7)-(10):

$$(7) \quad 40 \text{ unique concepts} \times 7 \text{ tools} = 280 \text{ concepts in total,}$$

$$(8) \quad 280 \text{ concepts} \times 3 \text{ objects per image} = 840 \text{ images in total,}$$

$$(9) \quad 9 \text{ judgements} \times 840 \text{ images} = 7,560 \text{ original data-points,}$$

$$(10) \quad 9 \text{ judgements} \times 56 \text{ batches} \times 16 \text{ images} = 8,064 \text{ AMT data-points.}$$

**Criteria for choice of images.** The next step is the choice of the images for evaluation according to some criteria. Due to the variation in the distribution of detected objects per image and confidence scores, there are two options for the choice of data. Option 1 is to use the same images across tools but a varying set-up for the parameters *number of objects per image* and *confidence scores of the according objects*. This option ensures that the same images are compared, however, the option is unbalanced in terms of data-points (e.g. resulting in 2 objects for one image but 3 objects for another image) and less comparable regarding the confidence scores. In contrast to that, option 2 is to use different images for evaluation but to keep the same set-up for objects per image and confidence scores. In the end, option 2 is selected for the following reasons. First of all, keeping the set-up the same for the number of objects per image results in a more balanced dataset. Furthermore, the variability of the two parameters would need to be controlled in some way which probably results in a skewed and uninformative dataset.

---

<sup>6</sup>Please note that the number of annotators was also set to 10, 11 and 12 for test-cases, but never less than 9. However, the actual number of approved data-points per batch was always around 144 ( $9 \times 16$ ).

**Condition for confidence scores.** The choice of the confidence score for each object is based on the distribution of the confidence scores in Figures 18 and 19. As already mentioned, scores in the range  $[0.1, 0.3]$  and  $0.3 < x < 0.99$  occur most often. Based on this, we decide to choose one object with a score between  $[0.1, 0.2]$ , one object with a score between  $[0.2, 0.3]$  and one object with a score  $\geq 0.9$ , which we choose instead of  $\geq 0.99$  since it is more comparable to the range of the other conditions and even 0.9 still is a very high number. Please note that in the following, the notation for the ranges is  $[0.1, 0.2]$  and  $[0.2, 0.3]$  for simplicity, but actually refers to the ranges  $0.1 \leq x \leq 0.2$  and  $0.2 < x \leq 0.3$ .

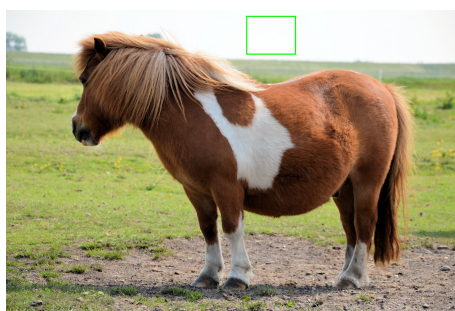
**Generation of the evaluation dataset.** The subset of the original dataset for the evaluation includes only those images where the conditions regarding the confidence scores meet. Those instances are extracted by a script and the original images without bounding boxes are stored. Second, the coordinates of the bounding boxes are used to automatically draw bounding boxes on the according images. For each bounding box a new image is used, resulting in 3 images per concept. Following this, a csv file is generated containing the image names, the respective labels and a mark on whether it is abstract or concrete. Note that the image naming follows a pattern which can be interesting for further analysis of the data, where *h* (high) denotes that the detected object has a confidence score  $\geq 0.9$ , *m* (middle) between  $[0.2, 0.3]$  and *l* (low)  $[0.1, 0.2]$ : `tool_concept_label_h/m/l_image_name.jpg`. Since concepts and labels are not unique, this ensures that the images can be tracked back to their confidence scores. As a result, each concept is represented by three image name entries in the csv following this example for the concept *supremacy*:

- fasterRCNN\_supremacy\_person\_h\_Image\_22.jpg
- fasterRCNN\_supremacy\_bed\_m\_Image\_22.jpg
- fasterRCNN\_supremacy\_book\_l\_Image\_22.jpg

After the generation of the csv, the file is shuffled in order to ensure the randomization of the distribution of concepts per tool. Subsequently, the large file with 840



lines is split into 56 csv files containing 15 lines. In addition to that, we choose 10 images from the original BING2023 dataset, where one object clearly can be determined and draw one bounding box in the same color as the automatically produced bounding boxes, but outside of the object (refer to Figure 21 for an example). For each of the 56 batches, one of these images is included and shuffled again. The 10 created random images function as a random check to ensure annotators quality. Finally, a header with variables from the AMT evaluation file is added to each of the batches.



**Figure 21:** Example for a random image check with an obviously wrong bounding box.

**Batch layout.** As mentioned before, one batch in AMT contains 16 HITs. For each HIT there is a reward of 0.06\$. The layout for one batch consists of some instructions as displayed in Figure 22, followed by the object label in the caption together with the according image and the two subtasks (see Figure 23). The instructions highlight the fact that the images are clickable and zoomable. First, workers are asked in subtask A to look at the image and to decide if the word presented in the caption depicts the object in the green box. Further, they are asked to provide a better word in case they choose option 2 (No). In Subtask B, a scale between 1 (Good) and 4 (Bad) is provided. Here, the workers are asked to rate how well the green box captures the according object in the image. The scale is between 1 and 4 to enforce a decision and avoid an uninformative accumulation of choices in the middle.

## Instructions

Below you will see an image containing a green box together with a word.  
The **green box** should surround the object. The **word** should describe the object within the box.

There are two subtasks:


- A. Please determine if the word depicts the object within the green box. You have the options **1 (Yes)** and **2 (No)**.  
**If you choose the option 2 (No), please provide a new word that depicts the object better.**
- B. Please determine how well the green box captures the described object on a scale between **1 (Good)** and **4 (Bad)**.

### Notes:

- We included one check in order to ensure annotation quality.
- There is one green box in every image. If you can't see it, please use the zoom functions.
- **Zoom in:** One click on the image.
- **Zoom out:** Doubleclick on the image.

**Figure 22:** Instructions that are given in AMT.

**Object: "bus"**



**Subtask A**

Does the word "bus" depict the object in the green box?

1 (Yes)  2 (No)

If you choose 2 (No), please provide here **only one** better word:

\_\_\_\_\_

**Subtask B**

Rate how well the green box captures the object "bus".

1 (Good)  2  3  4 (Bad)

**Submit**

**Figure 23:** Example for the batch layout in AMT with the label *bus*.

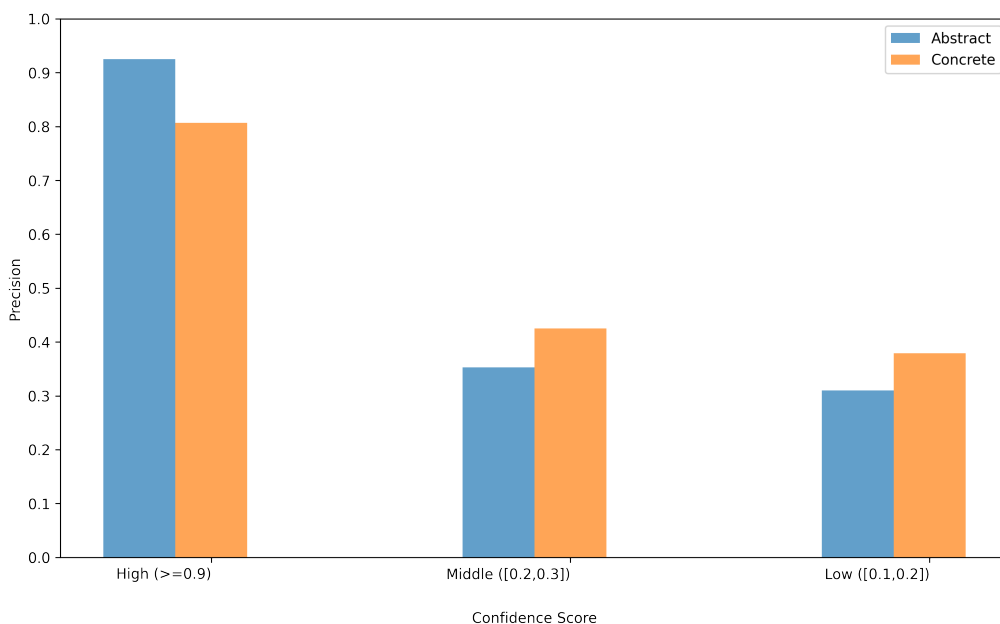
**Approval and rejection of workers.** The purpose of rejecting some workers is to ensure high annotation quality. The rejection primarily relies on a worker’s answer for the random check. If a worker chooses option 1 (Yes), the workerID is removed for every HIT in the batch. Furthermore, some image are manually checked and obviously wrong cases are removed. Some workers are removed also if they never enter anything in the text field where they are asked to provide a better label (only if 2 (No) is chosen). However, for every batch there are approximately 144 judgements in the end, resulting in a total of 7,756 approved judgements. The manually created random image checks account for 621 approved judgements. This results in a total of 7,135 judgements for 810 images which are used for the analysis of the results.

## 5.2 Results (Evaluation)

In the succeeding section, detailed insight is provided into the results of the evaluation via AMT, starting with a general overview, followed by an investigation of the results from subtask A and B. The judgements of the random checks are not included in the analysis of the annotations. This guarantees that only model output is evaluated. Also note that in subtask A some judgements have the same count for 1 (Yes) and 2 (No), resulting in indecisive data points. These cases are not included in the calculation of Precision. However, to gain some information out of these judgements, they are considered separately. Besides, there are many judgements with an agreement of 100% which are also examined in isolation. While there are 22 images with indecisive agreement and 432 images with 100% agreement, 356 images are judged based on their majority vote. The data used for the calculation of precision involved 788 images which are composed by the 432 100%-agreement-images and the 356 majority-vote-images.

Furthermore, careful consideration should be given to the fact that the data-points are associated with different confidence scores. It is questionable to which degree data-points with a probability between 0.1 and 0.3 can be seen as *True*

*Positives*, especially for the lower bound. However, it is also to be discussed if only data-points containing an object with a score of  $\geq 0.9$  count as *True Positives*, since scores around 0.2 and 0.3 also depend on an informed decision of the model. In order to account for this consideration, the data-points with differing scores are observed both mutually and separately from each other in the analysis. Figure 24 illustrates the distribution of abstract and concrete precision scores for every of the three probability ranges. From Figure 24 it becomes apparent that precision for high confidence scores is significantly higher than for the lower scores.



**Figure 24:** Comparison of precision for abstractness and concreteness across three settings of confidence scores (high, middle and low).

### 5.2.1 Subtask A

**Precision across all confidence scores.** Various aspects are investigated for subtask A. In Tables 8 and 9, precision across all confidence scores is provided, with Table 9 making an additional distinction for abstract and concrete data points. Also refer to Figure 25 for a visualization of the latter. For the cases where all scores are in-

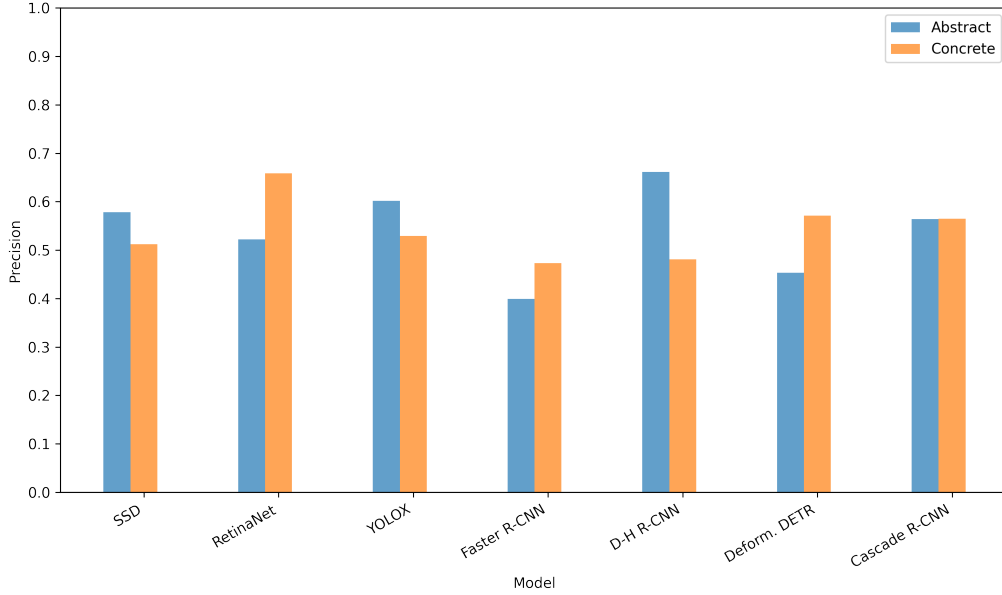
cluded, YOLOX achieves the highest performance across all models with a precision of 0.604, followed by RetinaNet, Double Head R-CNN and Cascade R-CNN, SSD and Deformable DETR with precision values between 0.577 and 0.473. The lowest precision score is found for Faster R-CNN with 0.439. When additionally considering the factor abstractness/concreteness, Double Head R-CNN notably outperforms the other models with a precision of 0.661 for abstract concepts, while RetinaNet achieves the highest score for concrete concepts with 0.658. Both for abstract and concrete concepts Faster R-CNN has the lowest scores with 0.400 (abstract) and 0.473 (concrete).

Model	Precision
YOLOX	<b>0.604</b>
RetinaNet	0.577
D-H R-CNN	0.569
Cascade R-CNN	0.557
SSD	0.542
Deformable DETR	0.473
Faster R-CNN	0.439

**Table 8:** Precision per model across all three confidence score settings, in descending order.

Model	Abstract	Concrete
SSD	0.578	0.512
RetinaNet	0.522	<b>0.658</b>
YOLOX	0.602	0.530
Faster R-CNN	0.400	0.473
D-H R-CNN	<b>0.661</b>	0.481
Deformable DETR	0.453	0.571
Cascade R-CNN	0.564	0.565

**Table 9:** Precision per model for abstractness and concreteness across all three confidence score settings.



**Figure 25:** Comparison of precision per model for abstractness and concreteness across all three confidence score settings.

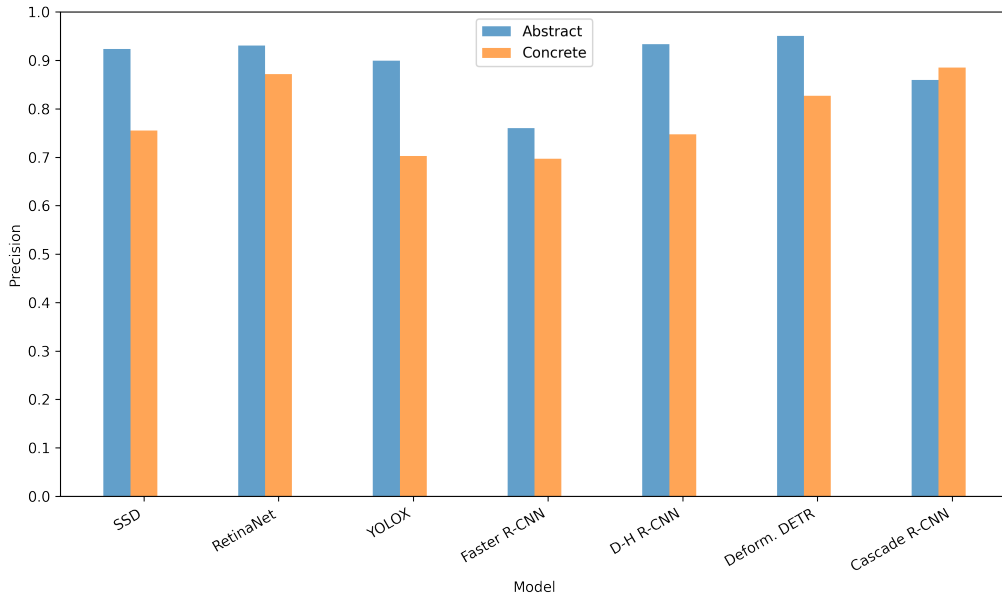
**Precision for high confidence scores.** As mentioned before, it is to be discussed if objects with a probability between 0.1 and 0.3 can really be considered as *True Positives*. Therefore, a separate analysis for reliable cases is shown in Table 10 with precision scores across all concepts and Table 11, taking the distinction of abstractness/concreteness into account. Also refer to the bar plot of the data in Figure 26. For the high confidence objects, RetinaNet achieves the best model performance with a precision of 0.900, this time followed by Deformable DETR, Cascade R-CNN, SSD, Double Head R-CNN, YOLOX and again Faster R-CNN with the worst performance (0.729). In addition, the analysis of the performance on abstract and concrete concepts reveals that Deformable DETR demonstrates the best precision with 0.951 for abstract concepts and Cascade R-CNN outperforms the other models for concrete concepts with a score of 0.885. Again, Faster R-CNN performs worst on both abstract (0.760) and concrete concepts (0.697).

Model	Precision
RetinaNet	<b>0.900</b>
Deformable DETR	0.892
Cascade R-CNN	0.873
SSD	0.841
D-H R-CNN	0.840
YOLOX	0.793
Faster R-CNN	0.729

**Table 10:** Precision per model for high confidence scores, in descending order.

Model	Abstract	Concrete
SSD	0.924	0.755
RetinaNet	0.930	0.872
YOLOX	0.899	0.703
Faster R-CNN	0.760	0.697
D-H R-CNN	0.933	0.747
Deformable DETR	<b>0.951</b>	0.827
Cascade R-CNN	0.860	<b>0.885</b>

**Table 11:** Precision per model for abstractness and concreteness for high confidence scores.



**Figure 26:** Comparison of precision per model for abstractness and concreteness considering only high confidence scores.

In summary, these findings indicate that model performance is higher for abstract

rather than concrete concepts, when the setting excludes low confidence scores. In contrast to that, it is apparent from Figure 24 that the precision for concrete concepts is higher than for abstract concepts in the [0.1,0.2] setting (0.379 concrete vs. 0.310 abstract) and the [0.2,0.3] setting (0.425 concrete vs. 0.352 abstract). However, this difference almost dissolves when all three ranges of confidence scores are considered (0.542 concrete vs. 0.532 abstract). Please be aware that these numbers don't have to add up to 1 since they are no opponents.

**50/50 agreement.** The analysis of the subset with an indecisive agreement gives some interesting insights into the distribution of the labels. Apparently, workers find it rather difficult to choose an option when it comes to the labels presented in Table 12. Workers reach an agreement of 50/50 for the label *person* six times. Furthermore, this happened two times for the labels *book*, *bear* and *tie*. The workers also achieve a balanced number of judgements once for *car*, *frisbee*, *cup* and seven more labels which can be found in the Table 12. Out of 22 cases, 13 cases are associated with concreteness and 9 cases are abstract. Please find the distribution of indecisive cases for abstractness and concreteness across models in Table 19 in the appendix.

Count	Label
6	person
2	book, bear, tie
1	car, frisbee, cup, bird, pizza, couch, tv, broccoli, donut, dining table

**Table 12:** Labels and their counts for cases with a 50/50 agreement.

**100% agreement.** Considering only the cases where workers achieve an agreement of 100% leads to an overview of the cases that seem to be very easy and clear for the workers. There is a total of 432 cases. All cases that receive a 100% for the 1 (Yes) rating add up to 258 and are summarized in Table 13. From this Table it is evident



that the label *person* is judged most often as suitable for the according object with an agreement of 100% (152 counts). The subsequent labels are *bird* (11 counts), *tie* (10 counts), *book* (9 counts), *cake*, *car* (6 counts) and more. As the rows in the Table approach lower counts, the number of labels increases. A comparison of the distribution of counts across abstractness/concreteness shows that 136 cases which are judged true with an agreement of 100% are associated with abstractness and 122 with concreteness. The model counts per abstractness/concreteness are illustrated in the appendix in Table 20 and are across all categories, also including the 100% annotations for (2) *No*. All model counts are between 62 and 68 except from YOLOX with only 44 counts.

Count	Label
152	person
11	bird
10	tie
9	book
6	cake, car
5	motorcycle
3	orange, refrigerator, pizza, toothbrush, airplane, bottle
2	cell phone, bus, chair, oven, spoon, knife, horse, cup, broccoli, handbag, truck, cat, toilet
1	train, donut, backpack, laptop, dining table, boat, bowl, bench, umbrella, clock, sink, bicycle, sheep, sandwich, couch

**Table 13:** Labels and their counts for cases with a 100% agreement on category 1.

**Majority agreement (without 100).** Finally, the cases are analysed that are evaluated with the category 1 (Yes) according to majority vote. These are 165 cases out of 356 majority ratings, where 91 are associated with concreteness and 74 with abstractness. Again, the label *person* is the one with the highest occurrences (68 counts), followed by *tie*, *potted plant*, *dining table* (8 counts), *boat* (7 counts), *chair* (6 counts) and more. Please see Table 14 for the numbers of all 1 (Yes) counts and their corresponding label. Among the distribution of majority votes across models, YOLOX has the highest counts with 67, while Cascade R-CNN has the lowest counts with 42. Additional information on the model counts per abstractness/concreteness can be found in Table 21 in the appendix.

Count	Label
68	person
8	tie, potted plant, dining table
7	boat
6	chair
5	bird, sports ball, book
4	truck
3	handbag, bed, car, cup
2	donut, tv, couch, airplane, carrot, bench, horse, vase
1	apple, orange, fork, sandwich, keyboard, wine glass, umbrella, broccoli, bowl, sheep, pizza, clock, suitcase

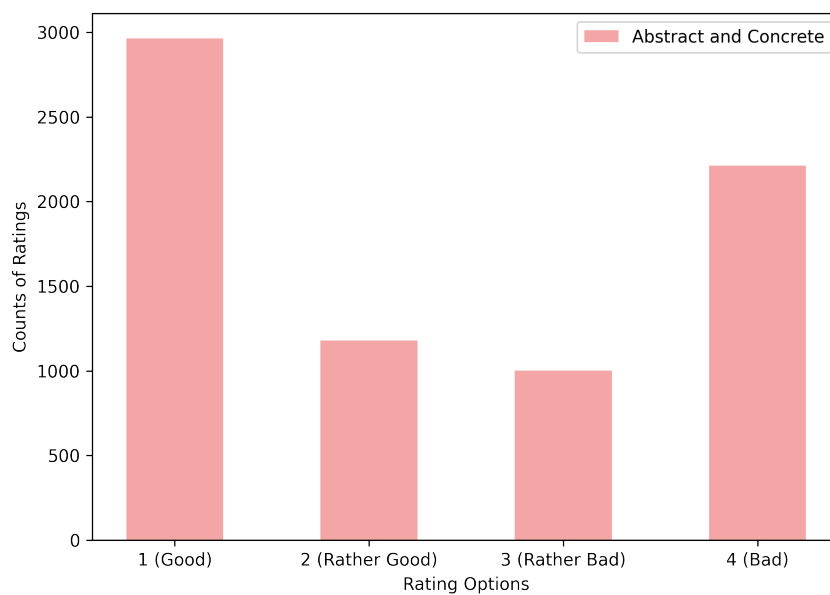
**Table 14:** Labels and their counts for cases with a majority vote for category 1.

### 5.2.2 Subtask B

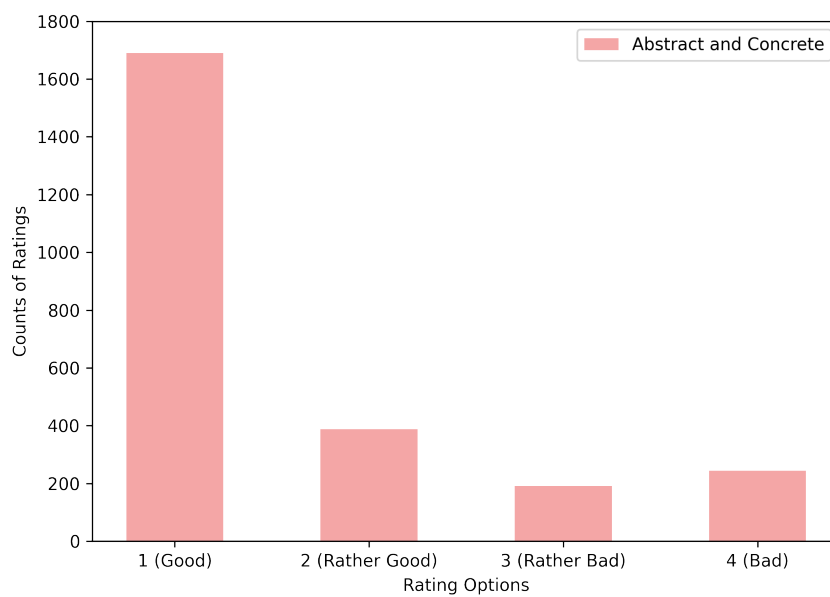
In Subtask B, workers are asked to rate on a scale between 1 (Good) and 4 (Bad) how well the bounding box captures the described object.

**Overall scale across all confidence scores.** For a more comprehensive view of the distribution of ratings, the data is analysed separately with respect to confidence scores, like in subtask A. On the one hand, Figure 27 displays the rating distribution across all confidence scores. While the ratings for 1 (Good) have the highest counts with 2,964, the values for 2 (Rather Good) and 3 (Rather Bad) decrease by more than half compared to the highest category (1,179 and 1,001 counts). For 4 (Bad), the counts increase again up to 2,212. Thus, the most counts are spread across the extremes with a tendency towards 1 (Good).

**Overall scale for high confidence scores.** On the other hand, ratings only for high confidence scores are depicted in Figure 28. This distribution looks different from the one in Figure 27. Again, category 1 (Good) receives the most counts with a value of 1,690. However, all other counts are significantly lower. 2 (Rather Good) is the second highest rating category with 387 counts, followed by 4 (Bad) with 244 counts. The lowest counts are recorded for 3 (Rather Bad) (191 counts). In contrast to the distribution including all three ranges of confidence scores, where the highest counts are distributed across both extreme categories, for the distribution considering only the high scores, a clear tendency towards the first category can be observed.

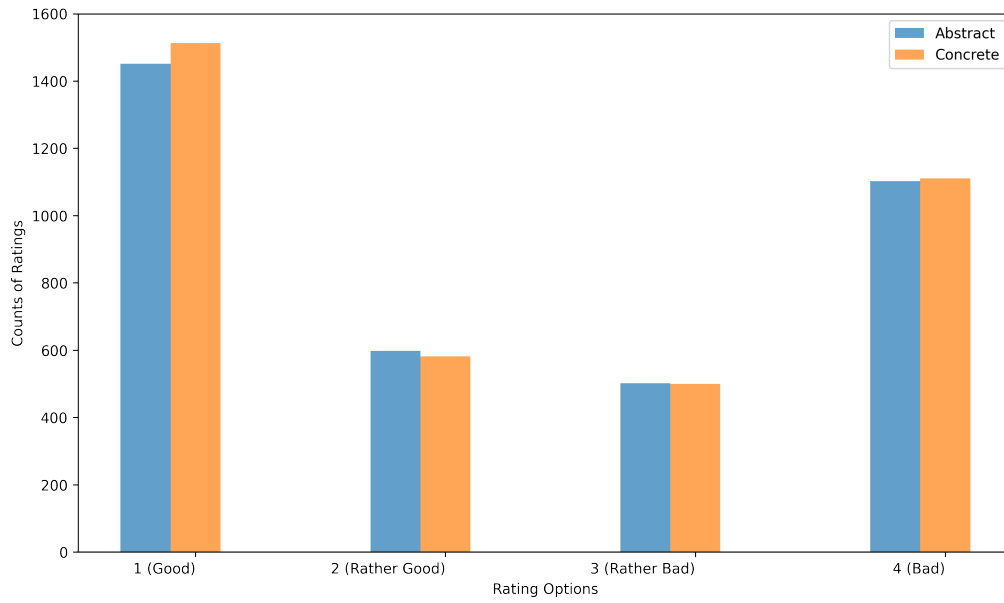


**Figure 27:** Distribution of the ratings on a scale between 1 and 4 for the accuracy of the bounding boxes (across all three confidence score settings).

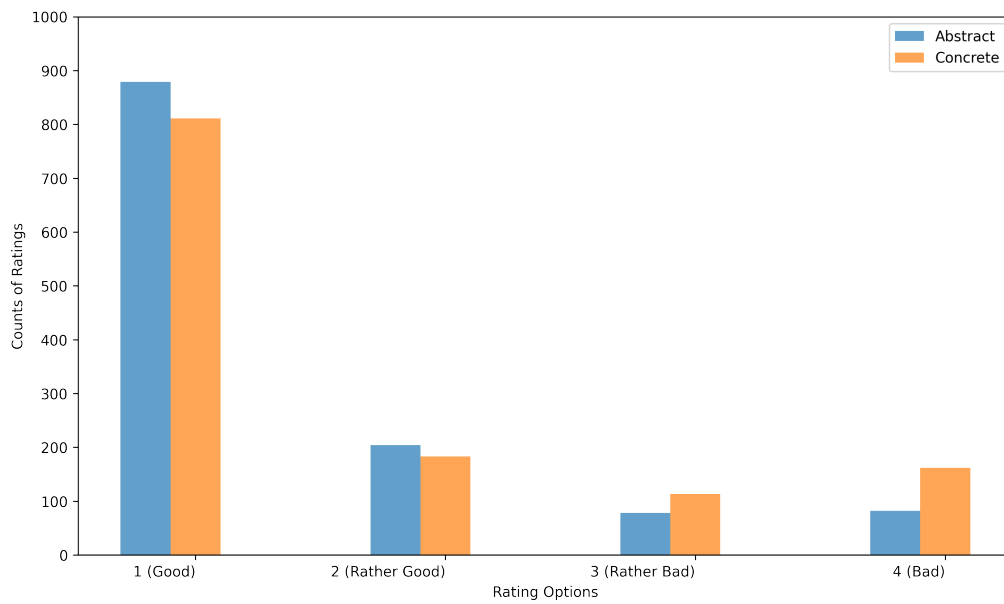


**Figure 28:** Distribution of the ratings on a scale between 1 and 4 for the accuracy of the bounding boxes considering only high confidence scores.

**Scale per abstractness/concreteness.** By adding the variables *abstractness/concreteness* per rating category, Figures 29 and 30 are obtained. A similar trend like in the plots for the overall scale can be observed. While for all confidence scores the counts are distributed across categories 1 and 4, the most counts for high confidence scores are clearly found for category 1. Splitting the counts for abstract and concrete concepts reveals that both variables generally follow this trend. Considering more fine-grained variations, it becomes apparent that small differences arise. While for category 1 in Figure 29 concrete concepts receive 62 more counts than abstract ones, in Figure 30, the same category receives 68 more counts for abstract concepts than for concrete concepts. When including only high confidence scores in the analysis, there are overall stronger differences between abstract and concrete than for the other setting. The differences between abstract and concrete counts on the scale from 1 to 4 for the setting in Figure 29 are 62, 17, 3, 8 in that order. The differences between abstract and concrete counts for the setting in Figure 30 are 68, 21, 35, 80. Particularly the last difference is interesting, since it is the highest. While bounding boxes for objects for abstract concepts only receive 82 ratings for category 4 (Bad), those for concrete ones receive 162. When the setting includes all three ranges of confidence scores, there are 1,451 counts for 1 (Good) associated with abstractness and 1,513 counts associated with concreteness. For the setting that excludes lower scores, this is the other way around: abstract concepts perform better (879 counts) compared to concrete concepts (811 counts). Overall, concrete concepts perform worse than abstract concepts in the setting for high scores, whereas abstract concepts perform worse than concrete ones in the other setting.



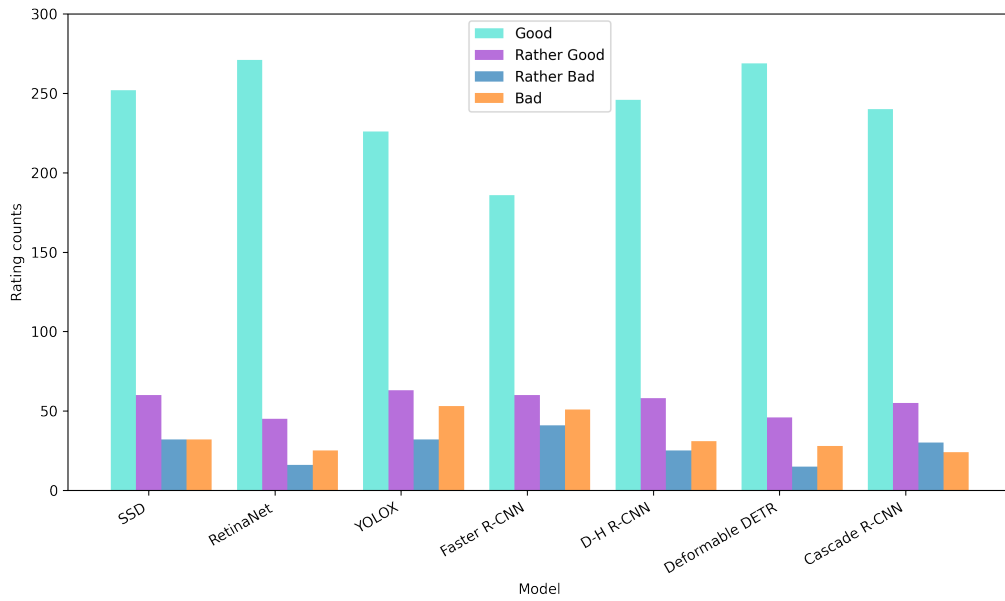
**Figure 29:** Ratings between 1 and 4 per abstractness and concreteness across all three confidence score settings.



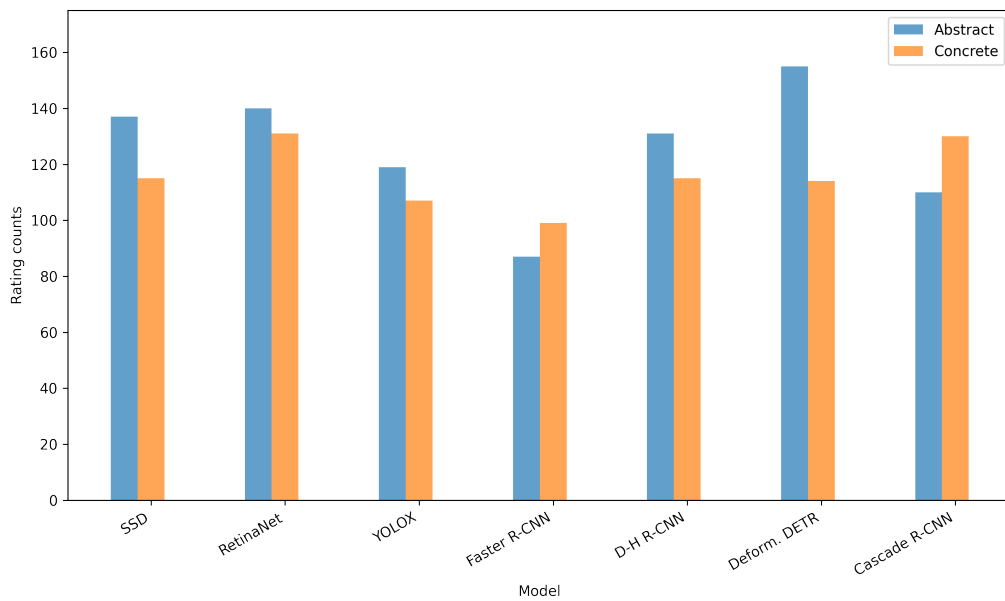
**Figure 30:** Ratings between 1 and 4 per abstractness and concreteness considering only high confidence scores.

**Scale per model.** An overview of model performance for high confidence scores is provided in Figure 31. As evident from the previous analysis, the inclusion of all confidence scores in the analysis leads to a blurry distribution such that differences converge. The differences become more apparent when considering only performance on high confidence scores. Therefore, this section which is already comparing seven tools, focuses on the exclusive setting. Refer to Figure 34 in the appendix for further details on the rating counts per model across all confidence scores. In Figure 31, almost all tools follow the trend of having category 1 (Good) with the highest counts, followed by category 2 at a large distance, some further small decrease for category 3 and finally a small increase for category 4 (Bad). Only SSD and Cascade R-CNN differ with regards to the order of category 3 and 4. While those categories are equal to each other for SSD, category 4 further decreases compared to category 3. To summarize, category 1 is the highest rated category across all models and category 3 the lowest, except for SSD, since category 3 and 4 share the lowest rank and Cascade R-CNN. In addition to that, the highest score for category 1 is achieved by RetinaNet with 271 counts, followed by Deformable DETR with 269 counts and SSD with 252 counts. In decreasing order the models Double-Head R-CNN (246), Cascade R-CNN (240), YOLOX (226) and Faster R-CNN (186) follow.

**Extremes per model and abstractness/concreteness.** Examining the differentiation of abstractness/concreteness in detail for the extreme categories, it stands out that for category 1, abstractness counts are almost always higher, except for Faster R-CNN and Cascade R-CNN. This phenomenon is visually represented in Figure 32). In contrast to that, counts for category 4 are always higher for concreteness, as depicted in Figure 33.

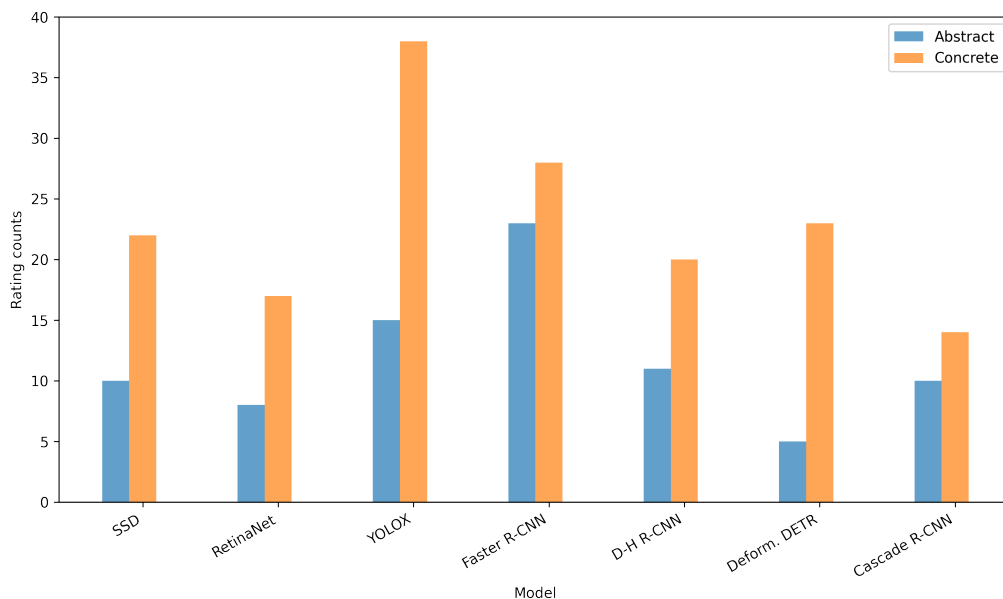


**Figure 31:** Comparison of ratings per model considering only high confidence scores.



**Figure 32:** Distribution of ratings for category 1 per model for high confidence scores.





**Figure 33:** Distribution of ratings for category 4 per model for high confidence scores.

**Top 15 labels.** It is particularly interesting to sort the ratings for category 1 which displays the ideal position of a bounding box. Table 15 provides an overview of the top 15 rated labels. The highest rated label by far is *person* with 1,425 counts for category 1. Top 2 is *tie* with 113 counts. The ranking of these two labels are exactly in line with the first two labels out of the 1 (Yes) label counts for majority agreement in subtask A, depicted in Table 14 and the 50/50 agreement in Table 12. Apart from the ranking, the following overlap between the labels with the top 15 ratings for the bounding boxes and the label counts for subtask A is found:

- 50/50 agreement: person, tie, bird, book, car, dining table (see Table 12)
- 100% agreement: person, tie, bird, book, car, dining table, chair, cake, truck, cell phone, motorcycle, handbag, clock (see Table 13)
- Majority agreement: person, tie, bird, book, car, dining table, chair, cake, truck, potted plant, handbag, clock, sports ball (see Table 14)

The unique set of overlaps is composed of the following 15 labels: person, tie, bird, book, car, dining table, chair, cake, truck, cell phone, motorcycle, handbag, clock, potted plant, sports ball.

<b>Label</b>	<b>1 (Good)</b>	<b>2</b>	<b>3</b>	<b>4 (Bad)</b>
person	1425	448	268	263
tie	113	80	71	136
bird	112	52	45	111
book	93	41	32	36
car	64	20	15	30
dining table	53	34	50	76
chair	49	37	39	100
cake	46	21	13	33
truck	45	5	9	10
cell phone	44	20	26	138
potted plant	41	20	13	10
motorcycle	39	8	8	11
handbag	38	23	22	66
clock	38	15	32	104
sports ball	36	18	20	48

**Table 15:** Top 15 labels and their counts for the categories 1, 2, 3 and 4.

### 5.3 Discussion

In the upcoming section, the results from section 4.4 and section 5.2 are discussed in relation to various aspects. First, characteristics of abstractness and concreteness are derived from the tendencies they show in the results. Following this, the performance of the models is compared in general and in relation to abstractness/concreteness. Finally, the top rated labels are addressed.

### 5.3.1 Abstractness vs. Concreteness

**Object counts.** Examining the distribution of objects based on their abstractness and concreteness revealed that while 852,858 objects are found in abstract images, 963,581 objects are detected in concrete images which are 110,723 more objects. Accordingly, 12.983% more objects are found for concrete images compared to abstract images. This finding suggests that images of concrete concepts contain notably more objects. **Suggestion 1:** A possible explanation for this could be that existing theories about the linguistic context of abstract/concrete concepts are applicable to their visual representation: Wiemer-Hastings and Xu (2005) stated that the context of abstract concepts tends to be more complex. Considering that the COCO classes involve only 80 categories, it seems reasonable that less complex objects, which are easier to recognize and distinguish, are more effectively detected by an algorithm. In contrast to that, abstract concepts might tend to be presented in a more visually challenging way. Following this logic, the number of objects per image could be understood as *context*. As Figure 15 illustrates, abstract concepts show a high peak for low numbers of detected objects per image compared to concrete ones. This goes along with the hypothesis that visual complexity in a 80 class setting leads to many occurrences of images with a low object count. **Suggestion 2:** Another possible explanation is that abstract concepts simply tend to have less visual context, regardless of their complexity, while concrete concepts rather appear with more context.

**Statistical analysis.** Interestingly, the statistical analysis of the object counts per image and the object counts per concepts assigns a higher mean, median and standard deviation for concrete concepts compared to abstract one (see Tables 4 and 6). This pattern is also reflected in the top 20 maximum and minimum values of detected objects which are both allocated to concrete concepts (see Figures 17 and 16). These observations indicate a higher variance for the object counts in case of concreteness. The reason for this might be that the 25 images per concept are not of the same quality and that the selection of images requires a more controlled setting.

However, this could also suggest that some concrete concepts can vary more in how they are visually represented. If the variety of object counts per concept or image is understood as the variety of context, this is not equivalent to the finding of Schulte im Walde and Frassinelli (2022), that abstract concepts occur with a higher contextual variety. In this case, the properties of visually represented abstract concepts and linguistic abstract concepts would differ.

**Precision and bounding boxes.** From the comparison of precision for abstractness vs. concreteness, no significant differences emerge for the setting that includes all confidence scores: model performance has a precision of 0.532 on abstract concepts and 0.542 on concrete concepts. If only high confidence scores are considered, a more obvious difference arises: while model performance on abstract concepts achieves a precision of 0.925, it yields on concrete concepts a precision of 0.807. Further, sub-task B records lower values for category 1 for abstract data (1,451) across all three ranges of confidence scores compared to concrete data (1,513). In contrast to that, the high confidence score setting is the other way around: bounding boxes associated with abstractness have a higher rating for 1 (Good) (879) than for concrete (811). In this set-up, also the lowest counts for 4 (Bad) are assigned to abstract images. Considering that high confidence scores represent the actual meaning of *True Positives*, it can be argued that models perform overall better on abstract concepts than on concrete ones. These findings challenge **suggestion 1**, the hypothesis that abstractness raises a higher complexity of context, at least if one assumes that complex objects rather have worse precision than simple ones. However, the lower precision related to concreteness can also be due to the higher amount of context, as suggested in **suggestion 2**. Assuming that a higher accumulation of objects on an image leads to a harder differentiation between the background and the object which is to be predicted, slightly worse performance on highly contextual concepts can be expected. This goes along with **suggestion 2**.

**Confidence scores.** Across all models, most of the detected objects have a confidence score in the range [0.1, 0.3] and 0.99 (Figures 18 and 19). This seems to be

reasonable and demonstrates high accuracy of the models. The models assign several potential labels to many object candidates and therefore predict many objects with a low probability. Only few objects are found in the middle which is an advantageous way to exclude candidates that are indecisive and thus do not provide informative content. The prediction of an object with a confidence of 50% represents a rather uninformed decision. Considering that the classes include 80 different labels, it is not as uninformed as for a binary scenario, however, it is preferred to have a clear decision. The high peak at 0.99 indicates that the models aim to avoid *False Positives* and favor informed decisions. Overall, this behavior can be well explained. It is especially striking that the abstract distribution has a count of 22,474 for a confidence score of 0.99 whereas the concrete distribution shows a count of 17,644. Together with the overall lower object counts for abstractness it seems that the model performance on abstract concepts is better than on concrete concepts. These findings are consistent with the distribution of precision across confidence scores in Figure 24. Abstract objects with a confidence of  $\geq 0.9$  have a precision of 0.925, whereas concrete objects have a precision of 0.807. At the same time, abstract precision values for middle and low confidence scores are worse. In conclusion, this means that models show an overall better performance for abstractness when the confidence scores are high, but worse performance for low to middle scores. In contrast to that, prediction on concrete data shows a more consistent performance.

**Agreement.** In section 5.2.1, several variants of annotator agreement are considered. For indecisive agreement (50/50), 13 cases are associated with concreteness and 9 cases with abstractness. On the basis of the limited data available, no reliable statement can be made about a tendency towards abstractness or concreteness. Moreover, the 100% agreement cases contain 224 abstract cases and 208 concrete cases. Among these, 136 abstract and 122 concrete cases are annotated with category 1 (Yes). The number of data-points is still small, however, a more obvious difference arises. Apparently, abstract cases lead to a higher agreement among annotators. This is consistent with the findings that the overall performance on concrete images

is worse. Besides, majority agreement for 1 (Yes) is higher for concrete cases. This leads to the assumption that there is a higher disagreement among annotators for concreteness and that these cases tend to be less clear than for abstractness.

### 5.3.2 Model Comparison

**RetinaNet and Deformable DETR: best overall.** When comparing model performance, it is interesting to take the distribution of object counts per model into account. Most objects are found from RetinaNet, followed by Deformable DETR, Double-Head R-CNN, SSD, Faster R-CNN, Cascade R-CNN and YOLOX. There is an overlap between the two models with the highest precision and the object count: for the high confidence score setting, RetinaNet gains best precision with 0.900, followed by Deformable DETR with 0.892. For the rating of the bounding boxes in subtask B, RetinaNet achieves the most counts for category 1 (Good), closely followed by Deformable DETR. Accordingly, tasks that do not involve the separation of abstractness/concreteness are well accomplished by these two models.

**Deformable DETR: best for abstract.** If additionally the variable of abstractness is taken into account, it is notable that Deformable DETR performs best for abstract concepts. Again, an overlap can be found: Deformable DETR is the only model with a higher object count for abstractness than for concreteness. Furthermore, Deformable DETR records both the highest counts for category 1 (Good) for abstract concepts and the lowest counts for category 4 (Bad) for abstract concepts. This indicates that Deformable DETR does not only perform overall well on images, but is particularly suitable for tasks involving abstract concept images. However, it is not clear why especially Deformable DETR is the tool performing well on abstractness. One possible explanation is that this is related to the encoder/decoder architecture.

**Cascade R-CNN or RetinaNet: which is best for concrete?** The highest precision on concrete concepts is recorded for Cascade R-CNN with 0.885, closely

followed by RetinaNet with 0.872. It should be noted that Cascade R-CNN shows the second lowest objects counts compared to other tools. Of course, the number of predictions and precision values are not directly proportional and a high prediction number does not necessarily influence a model’s performance. However, if the task is to examine the context of objects in images or quantify features of objects, it is preferable to have a higher number of detected objects available. When it comes to bounding box judgements for concrete concepts, RetinaNet and Cascade R-CNN perform nearly similarly: RetinaNet has 131 ratings for 1 (Good), whereas Cascade R-CNN has 130 ratings. If **suggestion 2** is followed, it is reasonable that both tools perform well on concrete concepts which occur with a higher number of objects per image. On the one hand, RetinaNet is known to address the extreme foreground-background class imbalance by introducing a new loss function (Lin et al., 2017b). On the other hand, Cascade R-CNN is able to perform well even in challenging situations: the resampling mechanism enables high quality detection by keeping increasingly hard examples in the training progress (Cai and Vasconcelos, 2018).

**YOLOX: robust but with flaws.** Moving forward to the precision values across overall confidence scores, YOLOX performs best when abstractness and concreteness are not considered. This suggests that YOLOX has the most consistent performance across the low, middle and high confidence scores. It can be argued that this is an indication of robustness, since it performs satisfying across different scenarios. Nevertheless, YOLOX has the second lowest counts for category 1 (Good) but the highest counts for 4 (Bad) in subtask B. From this follows, that the performance on the bounding boxes of YOLOX is not as good as its performance for classification. Also, YOLOX achieves the lowest object counts out of the seven tools.

**Faster R-CNN: worst performance.** Finally, it is conspicuous that Faster R-CNN performs worst in terms of precision across all settings. When it comes to the judgment of the bounding boxes, Faster R-CNN also has the lowest count for category 1. For the number of objects that are detected in total, Faster R-CNN is in fifth place.

**SSD and Double-Head R-CNN: satisfying performance.** SSD and Double-Head R-CNN do not show any major abnormalities and seem to perform satisfying. Double-Head R-CNN only stands out for precision for abstractness in the setting where all confidence scores are included (Table 9). However, this setting is rather too detailed and specific for an overall examination of abstractness and concreteness and can be interesting for further examinations on overall model performance or robustness of models.

### 5.3.3 Top rated labels

The main focus of this thesis is not on an in-depth analysis of the labels, however, there are some interesting results that are worth looking at. The label associated with the highest ratings for category 1 (Yes) in subtask A and for category 1 (Good) in subtask B is *person*. The gap to the second best rated label is especially remarkable for the refinement of bounding boxes: *person* with a count of 1425 ratings for category 1 is followed by *tie* with a count of 113 ratings. Further, *person* is also by far the highest counted label in subtask A for category 1 (Yes) with an agreement of 100% (152 vs. 11 for the second place) as well as with majority vote (68 vs. 8 for the second place). This suggests that models perform particularly well on people. For the indecisive cases, *person* is also the top count, however, a count of six is not really expressive. Please also be aware that the data used for evaluation is only a subset of the original dataset and some labels/concepts might not even occur in the evaluation.

## 6 Conclusion

In summary, this thesis compares seven different object detection tools and examines their performance on abstract and concrete concepts, depicted as images. Further, the goal was to gain insights into characteristics of abstract and concrete concepts



and their visual representation. In order to do so, an experiment is conducted on a dataset consisting of 24,950 images using the toolbox MMDetection. The data is evaluated both automatically in terms of object counts, as well as manually by human annotators from AMT. It is found that for concrete concepts 110,723 more objects are detected than for abstract concepts. Two potential explanations are suggested for this disparity. Suggestion 1 assumes that higher complexity of context in abstract images leads to less model predictions, where context is understood as number of objects per image. On the other hand, suggestion 2 proposes that abstract concepts simply occur with less context than concrete concepts. Taking additionally into account that precision values and bounding box judgements are better for abstract concepts in the high confidence score setting, suggestion 1 is challenged. Moreover, these findings support the second hypothesis that concreteness is associated with more context than abstractness. However, this is based on the assumption that crowded context leads to worse performance due to a difficult background/foreground distinction.

A comparison of the tools reveals that it is more recommendable to use RetinaNet, Deformable DETR or Cascade R-CNN when high accuracy is chosen over robustness. For tasks that involve low probability predictions, a rather robust tool like YOLOX or Double-Head R-CNN can be useful. It is also remarkable that both RetinaNet and Cascade R-CNN achieve the best results on concrete data, especially since RetinaNet is well-known for its ability to address foreground/background imbalances. This supports suggestion 2. On abstract data, Deformable DETR performs best. It remains unclear, why Deformable DETR is especially suitable for abstractness, however, this can be related to the transformer based architecture.

Further, the distribution of cases with 100% agreement and majority vote agreement across abstractness and concreteness suggests that disagreement among annotators is more prevalent in concrete cases, whereas abstract cases seem to be more clear for the workers.

Finally, it is still to be discussed which confidence scores are considered as *True Positives* and if it is useful to stick to a specific setting which includes or excludes certain scores. On the one hand, factors like robustness and distributional patterns can be observed, when several scores are included. However, this also skews the dataset regarding actual *True Positives* and leads to the need of isolated analyses.

The results of this study extend to the domain of computer vision by providing information on the performance of object detection tools, as well as to psycholinguistics and natural language processing with regards to insights into characteristics of abstract and concrete concepts. Also, this thesis raises the question what the visual context of abstract and concrete concepts looks like. Looking ahead, it would be interesting to have a more in-depth analysis of the definition of context in images and to examine the question if the linguistic context theories for abstractness/concreteness can be applied to the visual representation of the concepts. One approach might be to not only count the number of objects per image but also to measure the average distance between objects per image (or per concept). Further, entropy is an interesting measure for the degree of variety and can be used to investigate the questions of how strongly the context labels differ per abstractness/concreteness. This again can be compared with linguistic studies about entropy of context. Future investigations might also include more than the 80 classes from COCO. Regarding the labels, a more detailed analysis of the synsets from WordNet for each label per abstractness/concreteness sounds promising. By doing so, the characteristics of abstract and concrete image data can be estimated better in order to yield further improvement of the model performance. Finally it would be interesting to have a study that examines the performance of different transformers vs. a baseline on abstract concepts.

## Appendix

---

airplane	cat	kite	snowboard
apple	cell phone	knife	spoon
backpack	chair	laptop	sports ball
banana	clock	microwave	stop sign
baseball bat	couch	motorcycle	suitcase
baseball glove	cow	mouse	surfboard
bear	cup	orange	teddy bear
bed	dining table	oven	tennis racket
bench	dog	parking meter	tie
bicycle	donut	person	toaster
bird	elephant	pizza	toilet
boat	fire hydrant	potted plant	toothbrush
book	fork	refrigerator	traffic light
bottle	frisbee	remote	train
bowl	giraffe	sandwich	truck
broccoli	hair drier	scissors	tv
bus	handbag	sheep	umbrella
cake	horse	sink	vase
car	hot dog	skateboard	wine glass
carrot	keyboard	skis	zebra

---

**Table 16:** List of all 80 COCO classes (Lin et al., 2014).

Concept	Concrete	Abstract
1	grass	significance
2	tree	validity
3	ocean	certainty
4	iceberg	characterization
5	eagle	possibility
6	forehead	eligibility
7	rhino	severity
8	hemp	aptitude
9	keyboard	mindset
10	owl	coefficient
11	horse	competence
12	bird	likelihood
13	nostril	tendency
14	gorilla	pragmatism
15	nun	perseverance
16	portrait	standardization
17	scissors	ambiguity
18	zebra	discernment
19	diploma	competency
20	hummingbird	adversity

**Table 17:** Ranking of the 20 concepts with the lowest number of detected objects for abstractness (orange) and concreteness (blue).

Concept	Concrete	Abstract
1	bookstore	colonialism
2	restaurant	austerity
3	canteen	tradition
4	motorway	extremism
5	nightclub	conformity
6	rooftop	enlightenment
7	theater	assumption
8	cafe	martyrdom
9	airport	activism
10	kitchen	appropriation
11	hotel	nationalism
12	patio	dominion
13	platter	accountancy
14	volleyball	succession
15	office	independence
16	vegetable	anarchism
17	sushi	epitome
18	salad	fruition
19	cathedral	socialization
20	desk	credence

**Table 18:** Ranking of the 20 concepts with the highest number of detected objects for abstractness (orange) and concreteness (blue).

Model	Abstract	Concrete
SSD	0	2
RetinaNet	1	2
YOLOX	3	3
Faster R-CNN	1	1
D-H R-CNN	1	0
Deform. DETR	0	3
Cascade R-CNN	3	2

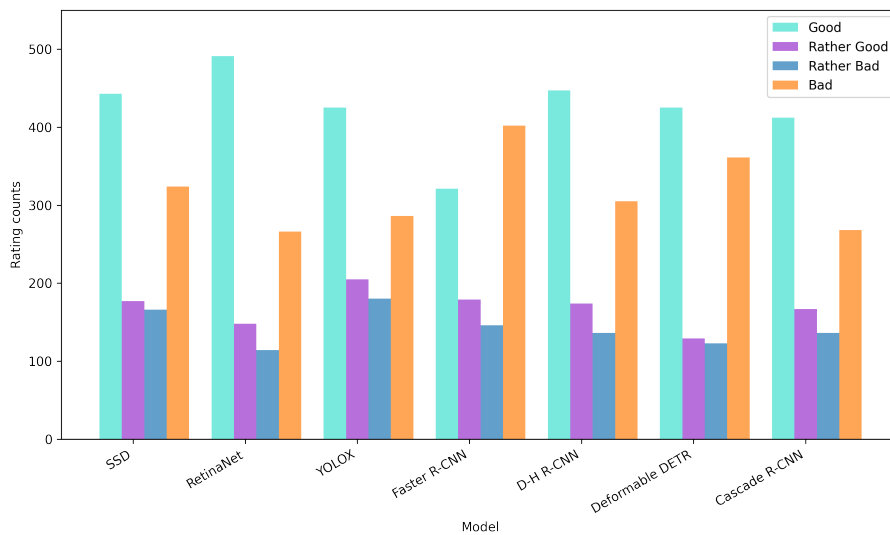
**Table 19:** Counts per model and abstractness/concreteness for cases with a 50/50 agreement.

Model	Abstract	Concrete
SSD	34	28
RetinaNet	35	33
YOLOX	25	19
Faster R-CNN	32	31
D-H R-CNN	32	32
Deform. DETR	37	30
Cascade R-CNN	29	35

**Table 20:** Counts per model and abstractness/concreteness for cases with a 100% agreement.

Model	Abstract	Concrete
SSD	26	30
RetinaNet	21	22
YOLOX	30	37
Faster R-CNN	24	27
D-H R-CNN	26	26
Deform. DETR	22	23
Cascade R-CNN	22	20

**Table 21:** Counts per model and abstractness/concreteness for cases with majority vote agreement.



**Figure 34:** Comparison of ratings per model across all three confidence score settings.

## References

- Yali Amit, Pedro Felzenszwalb, and Ross Girshick. 2020. Object detection. *Computer Vision: A Reference Guide*.
- Lawrence Barsalou. 1999. Perceptual symbol systems. *Behavioral and brain sciences*.
- Sai Abishek Bhaskar, Maximilian Köper, Sabine Schulte Im Walde, and Diego Frassinelli. 2017. Exploring multi-modal text+image models to distinguish between abstract and concrete nouns. In *Proceedings of the IWCS workshop on Foundations of Situated and Multimodal Communication*.
- Marc Brysbaert, Amy Beth Warriner, and Victor Kuperman. 2014. Concreteness ratings for 40 thousand generally known English word lemmas. *Behavior research methods*.
- Zhaowei Cai and Nuno Vasconcelos. 2018. Cascade R-CNN: Delving into high quality object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.
- Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. 2020. End-to-end object detection with transformers. In *European conference on computer vision*. Springer.
- Manuel Carranza-García, Jesús Torres-Mateo, Pedro Lara-Benítez, and Jorge García-Gutiérrez. 2020. On the performance of one-stage and two-stage object detectors in autonomous vehicles using camera data. *Remote Sensing*.
- Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. 2019a. MMDetection: Open MMLab Detection Toolbox and Benchmark. *arXiv preprint arXiv:1906.07155*.



- Yuntao Chen, Chenxia Han, Yanghao Li, Zehao Huang, Yi Jiang, Naiyan Wang, and Zhaoxiang Zhang. 2019b. Simpledet: A simple and versatile distributed framework for object detection and instance recognition. *Journal of Machine Learning Research*.
- Sara Dellantonio, Remo Job, and Claudio Mulatti. 2014. Imageability: now you see it again (albeit in a different form). *Frontiers in psychology*.
- Clement Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. 2012. Learning hierarchical features for scene labeling. *IEEE transactions on pattern analysis and machine intelligence*.
- Martin Fischer and Rolf Zwaan. 2008. Embodied language: A review of the role of the motor system in language comprehension. *Quarterly journal of experimental psychology*.
- Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. 2021. YOLOX: Exceeding YOLO series in 2021. *arXiv preprint arXiv:2107.08430*.
- Lakoff George and Mark Johnson. 1980. Metaphors we live by.
- Ross Girshick. 2015. Fast R-CNN. In *Proceedings of the IEEE international conference on computer vision*.
- Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.
- Joseph Goguen. 2005. What is a concept? In *International Conference on Conceptual Structures*. Springer.
- Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep learning*. MIT press.

- Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, et al. 2018. Recent advances in convolutional neural networks. *Pattern recognition*.
- Stevan Harnad. 1990. The symbol grounding problem. *Physica D: Nonlinear Phenomena*.
- Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. 2017. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*.
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2014. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *European Conference on Computer Vision*. Springer.
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.
- Faris Kateb, Muhammad Mostafa Monowar, Abdul Hamid, Abu Quwsar Ohi, and Muhammad Firoz Mridha. 2021. Fruitdet: Attentive feature aggregation for real-time fruit detection in orchards. *Agronomy*.
- Maximilian Köper and Sabine Schulte im Walde. 2017. Improving verb metaphor detection by propagating abstractness to words, phrases and individual senses. In *Proceedings of the 1st Workshop on Sense, Concept and Entity Representations and their Applications*.
- Yann LeCun, Koray Kavukcuoglu, and Clément Farabet. 2010. Convolutional networks and applications in vision. In *Proceedings of 2010 IEEE international symposium on circuits and systems*. IEEE.
- Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. 2017a. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

- Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017b. Focal loss for dense object detection. *IEEE International Conference on Computer Vision ICCV*.
- Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6–12, 2014, Proceedings, Part V 13*. Springer.
- Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander Berg. 2016. SSD: Single shot multibox detector. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*. Springer.
- Francisco Massa and Ross Girshick. 2018. maskrcnn-benchmark: Fast, modular reference implementation of Instance Segmentation and Object Detection algorithms in PyTorch. <https://github.com/facebookresearch/maskrcnn-benchmark>. Last access: July 09, 2023.
- Jörg Meibauer, Ulrike Demske, Jochen Geilfuß-Wolfgang, Jürgen Pafel, Karl Heinz Ramers, Monika Rothweiler, and Markus Steinbach. 2015. *Einführung in die germanistische Linguistik*. Springer-Verlag.
- Daniela Naumann, Diego Frassinelli, and Sabine Schulte im Walde. 2018. Quantitative semantic variation in the contexts of concrete and abstract words. In *Seventh Joint Conference on Lexical and Computational Semantics (SEM 2018)*.
- Allan Paivio. 1971. Imagery and language. In *Imagery*. Elsevier.
- Diane Pecher. 2018. Curb your embodiment. *Topics in Cognitive Science*.
- Diane Pecher, Inge Boot, and Saskia Van Dantzig. 2011. Abstract concepts: Sensory-motor grounding, metaphors, and beyond. In *Psychology of learning and motivation*, volume 54. Elsevier.

- Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.
- Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster R-CNN: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*.
- Sabine Schulte im Walde and Diego Frassinelli. 2022. Distributional measures of semantic abstraction. *Frontiers in artificial intelligence*.
- Paula J Schwanenflugel and Edward J Shoben. 1983. Differential context effects in the comprehension of abstract and concrete verbal materials. *Journal of Experimental Psychology: Learning, Memory, and Cognition*.
- K Simonyan and A Zisserman. 2015. Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations (ICLR 2015)*. Computational and Biological Learning Society.
- Eliot R Smith. 1998. Mental representation. *The handbook of social psychology*.
- Tarun Tater, Diego Frassinelli, and Sabine Schulte im Walde. 2022. Concreteness vs. Abstractness: A Selectional Preference Perspective. In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing: Student Research Workshop*.
- Mike Tucker and Rob Ellis. 1998. On the relations between seen objects and components of potential actions. *Journal of Experimental Psychology: Human perception and performance*.
- Katja Wiemer-Hastings and Xu Xu. 2005. Content differences for abstract and concrete concepts. *Cognitive science*.

- Yue Wu, Yinpeng Chen, Lu Yuan, Zicheng Liu, Lijuan Wang, Hongzhi Li, and Yun Fu. 2020. Rethinking classification and localization for object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*.
- Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. 2019. Detectron2. <https://github.com/facebookresearch/detectron2>. Last access: July 09, 2023.
- Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. 2017. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.
- Peng Zhou, Bingbing Ni, Cong Geng, Jianguo Hu, and Yi Xu. 2018. Scale-transferrable object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.
- Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. 2020. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv e-prints*.
- Zhengxia Zou, Keyan Chen, Zhenwei Shi, Yuhong Guo, and Jieping Ye. 2023. Object detection in 20 years: A survey. *Proceedings of the IEEE*.