DEPARTMENT OF PHILOSOPHY,
LINGUISTICS AND THEORY OF SCIENCE

EMERGENCE OF REFERRING EXPRESSIONS THROUGH LANGUAGE GAMES

Dominik Künkele

Master's Thesis:	30 credits
Programme:	Master's Programme in Language Technology
Level:	Advanced level
Semester and year:	Spring, 2023
Supervisor:	Simon Dobnik
Examiner:	Asad Sayeed
Keywords:	referring expressions, language games, artificial 3-d dataset

Abstract

Contents

1 Introduction

Language is a complex system that enables humans to communicate and convey meaning. However, the mere existence of words and grammatical structures is not enough to guarantee effective communication. Without grounding, language remains detached from the physical world, making it difficult for individuals to comprehend and interpret linguistic expressions accurately. Grounding bridges the gap between abstract linguistic representations and concrete experiences, providing a shared context and reference point for language users (Roy, 2002; Bisk et al., 2020; Bender & Koller, 2020). By linking language to sensory perceptions, physical experiences, and the situational context, grounding enhances comprehension, facilitates communication. However, one challenge lies in how arbitrary symbols in language can be connected to real-world objects and concepts and acquire meaning, described as the symbol grounding problem by Harnad (1990). Artificial agents can be taught grounding in different ways. One approach is to show the agent an image of the scene as visual input together with a textual description of what is to be grounded in this scene. This can be for example a description of the whole scene, certain visible objects or also actions that are happening (Ahrens et al., 2022; Ilinykh et al., 2022; Lu et al., 2017; Mitchell et al., 2013). By doing this, the agents can learn to identify the same descriptions and symbols for similar visual input. In other approaches, the agents can learn to ground language in visual input through interaction and dialogue, either with humans or other agents. Thus, agents might develop certain strategies to ground language efficiently (Dobnik & Storckenfeldt, 2018; Dobnik & Silfversparre, 2021; Zarrieß & Schlangen, 2019).

A central part in many of the ways to learn to ground are referring expressions. Referring expressions are used to point out a particular subset of entities from a set of similar entities or the surrounding context. By that, they can take many forms as noun phrases (e.g. 'the large table', 'the second chair from the right'), proper nouns (e.g. 'Francesca', 'the Big Ben') or complete descriptions (e.g. 'the men that are wearing glasses'). When referring to objects, they often include inherent attributes of these objects as their color or shape. Furthermore, spatial terms play a vital role to refer to and disambiguate objects (Regier, 1996; Dobnik & Kelleher, 2013; Dobnik & Silfversparre, 2021; Ghanimifard & Dobnik, 2017; Ramisa et al., 2015).

One problem that arises is the question of ambiguity in language and referring expressions. Natural language is often used in an underspecified way. Interlocutors rely on pragmatic and semantic context to communicate, particular interlocutors share a common ground which is used to disambiguate referring expressions. In case a common ground is not shared, referring expressions might be misunderstood, because of their underspecification.

Furthermore, the real world is characterized by its continuous nature. On the other side, natural language is a set of discrete symbols. In the process of mapping this continuum to the discrete symbols, information is lost, since the whole context of the real world can't be captured by the natural language. By this, symbols get ambiguous and refer to multiple concepts in the world. One example are the shades of color (Zaslavsky et al., 2018). Discrete English terms like 'red', 'green' or 'blue' might refer to an infinite number of colors in the real world.

To deal with this, humans rely on communicative protocols to disambiguate referring expressions. Dale & Reiter (1995), for instance, describe an incremental algorithm how referring expressions are generated. The goal of the algorithm is to construct referring expressions incrementally, taking into account the salience of different properties and abstracting away from differences in lexical choice. It works by gradually building up a referring expression based on the properties of the object being referred to. Hereby, it considers the most salient or distinguishing properties first and then adds additional properties as needed. This helps to create concise, effective and unambiguous referring expressions.

On the other side, humans as well as machines need to be able to comprehend referring expressions. Often

this task is defined as identifying a unique target object from a set of distractors in a visual scene (Mao et al., 2016). Hereby, many different architectures for models are studied that try to solve this task (Qiao et al., 2020). In some approaches the models for example embed both the referring expression and bounding boxes around objects in the visual scene in the same embedding space to align them. In other approaches, models don't rely on bounding boxes, but extract features from the whole visual scene and combine them with the referring expression in one step. An extension of referring expression comprehension are visual question and answering tasks (VQA), in which a model is presented with a question and a visual scene that should be used to answer the question (e.g. (Antol et al., 2015)). In these tasks, the model often needs to extract multiple referring expressions at the same time and ground them in the visual scene.

Meaning of symbols and referring expression emerge through dialogue between interlocutors (Wittgenstein, 1953; Clark & Wilkes-Gibbs, 1986). Moreover, Wittgenstein (1953) introduces the concept of *language games*: small parts of conversation between interlocutors. In these language games, their context defines which meaning emerges for the referring expressions. This reasoning was taken up in training artificial agents to learn to refer to entities. Hereby, agents need to communicate arbitrary symbols to each other in multiple turns to solve a given task. By solving the task, they assign meaning to these symbols and an artificial language emerges. This setup allows to control the architecture of the agents, the information the agents receive, how and what they are able to communicate and which goals they are trained on. By this, their behavior and the emergence of referring expressions can be studied precisely and consistently. In early research, agents and their communication were studied in the context of robotics and rule-based systems (Steels & Loetzsch, 2009; Roy, 2002; Kirby, 2002; Kirby et al., 2008), while in current research, agents are based on deep neural networks (Lazaridou et al., 2017; Baroni, 2020; Baroni et al., 2022; Kottur et al., 2017) Furthermore, a focus lies on the nature of the emerged language. Agents can for instance learn a compositional language that allows them to combine already learned symbols to create new meaning, when presented with a changed environment and unseen situations (Kharitonov & Baroni, 2020; Lazaridou et al., 2018; Gupta et al., 2020). Additionally, emergent languages can be learned to encode meaning in efficient ways (Chaabouni et al., 2019; Zaslavsky et al., 2018).

1.1 Research Questions

In this thesis, a deeper look is taken into how agents can ground their emerged language in visual input. The focus hereby lies on referring expressions and how agents are able to generate and understand them. Multiple experiments are designed in a way that agents need to use referring expressions in their messages that are communicated. These messages are analyzed with respect to the visual input to answer the two following questions:

1. What are the limits of the agents' architectures and input representations on learning successfully grounding referring expressions through language games?
2. To what degree do emergent referring expressions align with referring expressions in a natural language such as English and what constraints can be imposed on the environment and the agents themselves that languages align?

1.2 Contribution

This thesis aims to add three contributions to the field of research. First, new artificial visual datasets are created, consisting of images, depicting objects and their attribute and spatial relations. Many existing datasets that are used to study referring expressions use real photos taking by humans. This adds a lot of inherent bias to the dataset, since these photos for instance often focus on similar objects and actions. Additionally, they require external knowledge about the world and the functions of objects which is not

present in the image. Furthermore, information about the objects and their relations in the image are not present in a structured way. The datasets that are created in this thesis aim to reduce this bias and provide all information about the scene and objects in the image. In fact, the artificial creation of the dataset allows controlling precisely for the bias in each scene which can be utilized by agents in a language game.

Secondly, the thesis evaluates these datasets, by training separate models to generate and understand referring expressions, describing objects in the images. By doing this, it is tested if first the models are able to extract useful visual information that doesn't rely on bias and latent patterns in the textual information in the dataset and secondly shows the impact of different levels of ambiguity in the datasets on the performance of generating and understanding of referring expressions.

Lastly, the thesis brings the separate tasks of generating and understanding of referring expressions together into one single task. This process resembles the learning of referring expressions in natural language (Clark & Wilkes-Gibbs, 1986). In language games, one agent needs to extract visual information from an image and generate a referring expression that is sent to the second agent. The second agent on the other hand needs to understand this referring expressions and combine it with its visual input. Only if both of these subtasks succeed, a new artificial language can emerge and the overall task can be solved. The emerging language is then analyzed to understand, how the artificial referring expressions are built up and compare to natural language.

1.3 Scope

The focus of this thesis is the study of referring expressions. Referring expressions can hereby be based on inherent attributes of objects, as well as their spatial relations towards other objects. In the present study, only the relations of the inherent attributes are studied. Spatial relations add another level of complexity and may be studied in future work. Furthermore, the emerged language will be interpreted by comparing it to referring expressions in natural language. This analysis focuses on the emerged meaning of symbols and if they align with natural language. A deeper study of its compositionality or complexity is out of scope.

2 Background and Related Work

2.1 Grounding

Symbols of a symbolic system such as words in a natural language are typically processed and manipulated in a purely formal manner, without direct reference to their real-world concepts. The question arises, how these symbols can acquire meaning and be connected to the real world without any external interpreter, in other words how the symbols can be grounded in the real world. This is known as the symbol grounding problem (Harnad, 1990). Hereby, Harnad (1990) describes that a high-order *symbolic representation* needs to be grounded bottom-up in two non-symbolic representations. *Iconic representations* refer to analogue representations of sensory input like for instance vision in the retina. These representations are mapped to *categorical representations* which are filtered clusters of the iconic representation. These categories are assigned so-called elementary symbols, in which symbolic representation are grounded.

In computational models this problem is tried to be solved in multiple ways. Recent large language models (LLMs) such as BERT (Devlin et al., 2019) or GPT-3 (Brown et al., 2020) are often trained on big amounts of text corpora produce impressive results on many tasks in Natural Language Processing. Meaning of symbols emerges through relating it to its surrounding context of other symbols it appears in. In this way, their only way of learning, how to generate language as well as solve these tasks is to find patterns of how words and sentences are used in these large corpora. This approach is however criticized by many researchers, since the meaning of symbols is still not connected to the real world and only relies on other symbols in an infinite recursion.

In (Bender & Koller, 2020), the authors argue that meaning is bound to a communicative intent. These are the purposes of why humans are using language. The intent is always embedded in a broader context that exists outside the language itself. This context includes the real world, but also the background knowledge of the speaker as well as interlocutors or reason for what the speaker is saying. By grounding the intent in this context, it becomes meaningful. Only with this step, the intent can be interpreted by the listeners. This grounding is missing when models only train on abstract textual representations of the world (Regier, 1996; Landau et al., 1998).

Bisk et al. (2020) argues that a multimodal approach, including for instance perception as well as social context, is needed to learn meaning in a broader context. One added modality to ground language is often visual input. Hereby, the model needs to learn how to associate linguistic concepts in text corpora with features, extracted from visual input. For instance, a model can learn to associate the noun ”dog” with an animal seen in an image or associate the action of ”jumping over” with the animal being above an object. Roy (2002) for example studies how an artificial agent can ground spatial relations with a visual scene of geometric shapes. Hereby, the agent builds up hierarchical knowledge that is used to produce correct description in a rule-based manner. In more recent research, complete natural language utterances as well as the visual representations are mapped into an embedding space. By aligning both representations, natural language can be connected to a second modality. Other research explores how to ground smaller parts of the utterances, as phrases and words and combine them compositional (Larsson, 2018; Kollar et al., 2010). This might allow artificial machines to learn grounded concepts more general.

Furthermore, artificial agents can learn to ground language through dialogue, either with human tutors or with other artificial agents. Skocaj et al. (2011) for example teach a robot different objects and concepts. By asking questions and participating in the dialogue with a human tutor, the robot builds up and verifies beliefs, and connects them to language and its visual input. Lauria et al. (2001) follow a similar approach where a robot is taught to follow instructions to generate a map of its physical surroundings. By incorporating dialogue and deep-level reasoning questions the learning gains are enhanced. How artificial agents can ground language in dialogue between each other will be studied in this thesis. Section 2.3 gives an insight

in the state of the research.

2.2 Referring expressions

A specific case of grounding linguistic information in visual input are referring expressions. While a general linguistic expression can be grounded in multiple regions of a visual scene, referring expressions are expected to refer to a unique region (Sánchez et al., 2022). Given for example a visual scene with multiple pens lying on a table, the word 'pen' can be grounded in all visible objects. A referring expression like 'the red pen on the left' in contrast would only refer to one specific object. Referring expressions are however not limited to refer to one single entity, but might also refer to a group of objects as for instance with 'the green pens'. Still they refer to a specific subset of entities and concepts.

Hereby, Krahmer & van Deemter (2012) classify referring expressions into four categories: *One-place predicates* refer to one single entity in the visual scene without. By this, the entity is identifiable without using another entity as basis. Often, the referring expressions includes intrinsic attributes of an object or absolute locations in the scene as in 'the red pen on the left'. *Relational* referring expressions refer to entities via other entities in the scene. Accordingly, a relating between the target entity and distractors is described. This includes for example spatial relations like 'the pen behind the cup' or relations based on the entities' properties like 'the largest pen'. Referring expressions can also identify *sets* of entities. Instead of uniquely referring to on single pen, the referring expression 'the green pens' refers to all objects that are pens and that are green. Finally, *gradable* referring expressions involve gradable properties as opposed to absolute values. For example, given the referring expression 'the tall building', 'tall' is a gradable property because it can be measured on a scale. The success of the referring expression depends on how well it captures the intended level of height compared to other buildings. By this, vagueness is introduced and referring expressions depend on context, either present in the visual scene or pragmatic context.

One of the main challenges of referring expressions lies in ambiguity. Ambiguity is introduced in two forms. First, there is an inherent difference in natural language and the real world it is grounded it, also described in the symbol grounding problem. The nature of the world is continuous, while natural language being a formal symbol system uses discrete symbols. In especially, natural language utilizes a very limited set symbols to represent the world. When trying to map continuous concepts to this limited set of discrete symbols, information nesecarilly gets lost. Apparent examples are the color systems in human languages. While color grades exist on an infinite spectrum in the real world, languages combine certain color grades into categories (Zaslavsky et al., 2018). The word 'red' for example refers to many different shades of red. Even more, the shades of color it actually refers to depends also on the pragmatic context and the people uttering it (Monroe et al., 2017). The same also applies to spatial references. References like 'near to' or 'left to' might refer to objects that are touching each other, but in other contexts also are kilometers away from each other.

The other form of ambiguity is introduced through under-specification of the speakers (e.g. Dobnik & Silfversparre (2021)). When interlocutors interact in a dialogue, they share some common knowledge. The first interlocutor might presuppose that the second interlocutor knows a certain fact. Based on this, they might utter a referring expression that together with this shared fact uniquely identifies an entity. However, the referring expression on its own is under-specified and might refer to multiple entities. Given for example the following situation: Two people are in a room with their beverages standing on the table and Person A requests his beverage from Person B. A referring expression like 'the beverage close to me' would uniquely identify on of the beverages. Despite that, Person A might more likely use 'the beverage' to refer to theirs. On its own, the referring expression is under-specified and can refer to both of the present beverages. However, both persons share the fact that the targeted beverage belongs to person A, which disambiguates the referring expression in this context. In another context, the shared fact would be that Person A wants to try the beverage of Person B. Again, this fact would disambiguate the referring expression, but now refer

to the second beverage. The referring expression is the same in both contexts, but refers to two different entities. This creates a challenge for computational models, that need to understand and generate referring expressions if they are only presented with the text, but not with the shared context.

The research of referring expressions can be split into two fields: referring expression generation (REG) and referring expression comprehension (REC). In **referring expression generation**, computational models are trained to generate a referring expression that uniquely identifies an entity given some perceptual, often visual input. The research goes back into the 70s, where Winograd (1972) developed an algorithm to generate referring expressions step by step, taking into account the context and the information available at each stage of the generation process. A similar approach is described in (Dale & Reiter, 1995). Their incremental algorithm is based on the salience of the properties of the present target object and distractor objects. The most salient properties are considered first, and added to the referring expression until the referring expression is unambiguous. In other words, the object is described unambiguously using the lowest number of words.

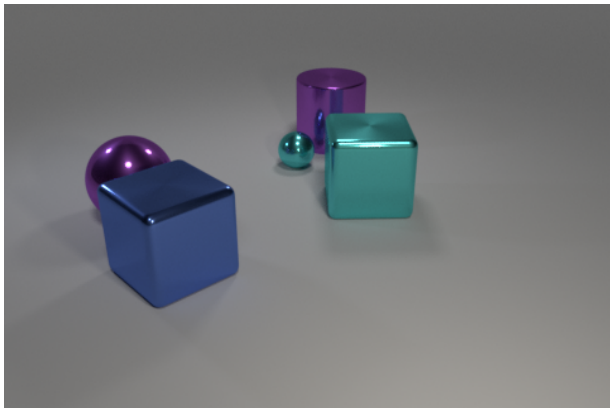


Figure 1: The target object in this scene is the *small turquoise sphere*. Using the incremental GRE-algorithm, the target object can be uniquely referred to as *turquoise sphere*

Given for example the scene in Figure 1. Objects in this scene have three attributes that we consider in the following order of salience: a shape, a color and a size. This order defines, which attributes can be left out, while still identifying the object uniquely. The target object in the given scene is the *small turquoise sphere*. In the first step, the attribute with the highest salience, the shape, is selected from the target object. This produces the referring expression *sphere*. In the next step, it is checked if there are also distractors, described by this referring expression. In our case, both cubes and the cylinder are already disambiguated, but the purple sphere can still be described by the produced referring expression. Therefore, the algorithm also adds the next lower salient attribute, the color to the referring expression, which produces the *turquoise sphere*. With this, also the last remaining distractor is as well disambiguated, and the referring expression describes the target object uniquely. The size, the least salient property is therefore not necessary in the referring expression. This algorithm will be utilized in this thesis and will be referred to as *the incremental GRE-algorithm*.

Additional to generating an efficient disambiguating referring expressions, a challenge also lies in processing the visual input. This especially applies to spatial and geometric positions and relations of objects in a scene, where models struggle to extract geometric information from visual features (Kelleher & Dobnik, 2017). More complex scenes also involve multiple perspectives, where the models are tasked to produce referring expressions that depend on the spatial perspective of the speaker and the listener (Ahrens et al., 2022; Lee et al., 2022). In (Dobnik & Silfversparre, 2021), visual dialogue is analyzed where speakers have different views on cups that are placed on a table. When speakers refer to a certain cup, they need to take the perspective of the listener to disambiguate the referring expression; for instance in the referring expression 'the cup on the left', *left* is dependent on the speaker's and listeners' spatial position in relation to the cup.

Current research focuses hereby on generating relational referring expressions, which add more complexity to the task (Ghanimifard & Dobnik, 2017, 2019; Ramisa et al., 2015; Liu et al., 2023). Models need to first extract visual features of multiple objects from the scene. Of these extracted objects, a meaningful landmark object needs to be found which helps to refer to the target object. Finally, they need to find proper relations between them to describe the target object. Relations can be mostly based on the geometric and spatial relation of two objects (e.g. above/below), while the function of objects play a stronger role in other relations (e.g. over/under) (Coventry et al., 2005; Dobnik & Kelleher, 2013).

Referring generation tasks can be hard to evaluate, since often there are many possible referring expressions that are possible and uniquely identify the target object (Mao et al., 2016). Instead, the task can be turned around: A model needs to identify an object when presented with a referring expression. This is called **referring expressions comprehension**. Many challenges remain similar to the REG task, since the model needs to extract disambiguating visual features from the scenes as well. However, the goal is typically to select the best matching region of a given choice in a visual scene (Qiao et al., 2020). They describe seven different approaches how the challenge is approached in the current research. In *joint embedding* approaches, both visual input and the textual referring expressions are embedded into the same embedding space. By doing this, the model is able to find similarities and combine them. In some cases, an attention mechanism is applied on top. This allows the model to focus on important parts of each representation for the specific task. *Modular models* are split up in multiple submodels that split up the complete task and solve them step by step. This helps to reduce the complexity of a task, as well as making more transparent, where models fail and succeed. *Graph-based models* try to make use of the structure in a visual scene when multiple objects are present and the model is tasked to disambiguate them. In a first step, the visual features of the image are transferred into a graph that captures objects' properties and their relations. This graph is then combined with the referring expression to extract, which of the nodes is represented by it. To give a bigger focus on syntactic and semantic structure in a referring expression (which might get underrepresented by sentence-level representations of e.g. an LSTM), some models utilize *external parsers*. These make use classical methods as for example dependency parsers, to parse the referring expressions. *Weakly supervised* models are used when there is a lack of large datasets that annotate the relation between a referring expression and the corresponding regions in the image. Instead, they can operate on datasets that only include the visual scene with a matching referring expression. This is done by selecting a matching region in the visual scene and then reproducing the input referring expression from that region. During testing, the model only extracts the region from a given referring expression. While most of the described approaches grounding the referring expression in multiple steps, by first identifying possible regions around objects in the visual scene and then combining each proposal with the representation of the referring expression, *one-stage approaches* combine them into one step. The image is processed as a whole and combined with the embedded referring expression. The model then predicts a bounding box around the referred entity. Finally, *Visual-Language pre-training* tries to jointly learn from visual and textual representations, instead of separate visual and textual processors in the other approaches. Models like ViLBERT (Lu et al., 2019) learn joint representations of language and visual input. These can then be fine-tuned on the specific referring expression comprehension task.

Closely related to the REC task is visual question and answering (VQA) (Ahrens et al., 2022; Antol et al., 2015; Ilinykh et al., 2022; Xu & Saenko, 2016). In these tasks, a model is given a question about a visual scene. The goal is to find an answer in the scene and generate an answer. Central to this is to extract referring expressions from the question and ground them in the visual scene. In (Johnson et al., 2017b), a model is for example shown an image similar to Figure 1 and asked the question 'Is there a blue cube with the same size as the purple cylinder?'. To answer this, the model needs to understand two referring expressions and try to ground them in the image. In some cases the model can't ground a referring expression, when no object in the scene matches the referring expression. In the paper, it is solved by using a modular design of the model, where in a first step the question is translated into a sequence of instructions that defines, how the model is supposed to extract information from the image. In the first part, the referring expression is

parsed, while in the second part the model grounds it in the visual input.

2.3 Language Games

In this thesis, referring expression generation and referring expression comprehension are studied at the same time. This is done by using language games between two artificial agents. The term 'language games' was first introduced by Wittgenstein (1953). Following his argumentation, meaning of words and language is not absolute, but is situated in human dialogue. Hereby, two interlocutors converse in language games, that define how the utterances are meant and interpreted. The words' meaning is not just based on the context or surrounding words, but more on the situation, including pragmatic and social context. The rules in each of these situations differ and therefore the meanings and semantics of words and sentences may differ from situation to situation. This is summarized as a language game. These can be any small parts of conversations, for instance between a teacher explaining a new concept to a student, or between two people, discussing a specific topic. An interjection 'Water!' may be a warning, an answer to a question, a request or something else, depending on the context. This applies as well to the ambiguous referring expressions, as in the example about the beverages described in section 2.2. Which referring expression Person A uses to ask for the beverage, depends on the assumed shared knowledge. With other interlocutors or situations, different referring expressions are uttered.

This reasoning was taken up when trying to train artificial entities to produce a language. One of the original games is the signaling game, proposed by Lewis (1969). These games are composed of a sender and a receiver. The sender has access to the state of the world, while the receiver does not. The sender can choose from a fixed set of signals to communicate information to the receiver. The receiver, upon receiving the signal, must take an action based on that information. The goal of both sender and receiver is for the receiver to take the correct action in every state, as both players have a common interest in achieving the same outcomes. Both sender and receiver are rewarded in the same way if the proposed action was correct. Hence, both agents need to invent a language together, fit to the conditions of the game.

The study of the emergence of an artificial language serves multiple goals. First, the study of the emergence of artificial language can help to shed light into why and how natural languages evolved (Bartlett & Kazakov, 2005; Kirby, 2002; Kirby et al., 2008; Noukhovitch et al., 2021). This approach allows a deep study of the constraints and restrictions that are necessary so that symbols start to be connected with meaning and get grounded in the real world. Insights in this field might give indications for in which circumstances natural language evolves. On the other hand, research analyzes how the emerged language looks and how the agents represent information in messages and symbols (Baroni et al., 2022; Lazaridou et al., 2017; Chaabouni et al., 2022; Kottur et al., 2017). Some studies observe different properties of the emerged languages, such as compositionality, complexity or efficiency, and study how the agents can be biased towards producing a language that holds certain properties. Eventually, this can produce languages that agents can use in practical applications instead of relying on the predefined unflexible protocols that are currently used.

In language games it is central that the means of communication are very restricted. Typically, the agents are given a set of arbitrary symbols, a vocabulary, that the agents can use to convey information. These symbols initially don't have any meaning, but by using and interpreting them repeatedly to solve the task the agents learn to use certain of those symbols, to encode specific information. By this, a new artificial language emerges between these agents. The emergence of a language however is highly dependent on the constraints posed on (1) the agents' architectures, (2) the way of communication, (3) the perceptual input for the agents and (4) the task they need to solve (Baroni et al., 2022).

In (Steels & Loetzsch, 2009) two robots are trained by using language games. The authors explore how spatial language, more specifically language that focuses on spatial positions and movements, emerges when the agents are situated in the real world and can perceive each other. Two robots equipped with a camera

are moving freely in a space until they see each other as well as the target object of the game. The target object is moved by the researchers, and that movement is registered by both robots. The robots now need to communicate the movement to each other using language games. Hereby, the experiments show that when agents are embodied and have a specific view on the surrounding world the capability of aligning their own perspective with the other's is central for the success of communication and the emergence of a language.

In more recent work, deep neural models are utilized for the agents. Instead of being embodied, agents only receive visual input, without any perception that the other agent is present. In (Lazaridou et al., 2017), two agents play a discrimination game. The same set of images is shown to both agents. While they are unordered for the receiver, the sender always sees the target image as its first image. The task is now for the sender to communicate the target image in a way that the receiver can discriminate it from the distractors and then point towards it. Hereby, two of the constraints mentioned above are studied at once. First, the way of communication is constrained in the way that the sender can only communicate one symbol per message, which results in the agents clustering images into categories that are described with a symbol. Secondly, two different sender architectures are compared: the 'agnostic' sender that encodes each image separately and then produces the message, and the 'informed' sender that uses convolutions to combine both images for better discrimination. While both architectures are able to solve the task with an emerged language, the emerged language of the informed sender converges faster and makes use of more symbols than the agnostic sender.

While the previously described examples always study prosocial agents (agents that have a common goal), Cao et al. (2018) examine self-interested agents. Hereby, multiple agents need to distribute a set of items between them. Each item has a utility score for each agent that is only visible to themselves. The goal for each agent is to gain the highest utility score by choosing the correct items. To do that, the agents first negotiate with arbitrary symbols in a 'linguistic' channel. At any time, any agent can choose to end the negotiation by proposing how to distribute the items, using predefined symbols in the 'proposal channel'. These can be accepted or rejected by the other agents. Their experiments show that in this self-interested setup, the agents utilize the proposal channel well to solve the task, but no language emerges in the linguistic channel. However, in a similar setup with prosocial agents, they start to utilize the linguistic channel and create an artificial language.

2.3.1 Setup of language games

In this thesis, the language games are studied with prosocial non-embodied agents that are deep neural networks. They can exchange multiple symbols to communicate after being presented with images. This is done using the *EGG* framework that allows a high configuration of the agents' architectures, of the ways of communication, and of the technical constraints (Kharitonov et al., 2019).

Using the framework, how the messages of the sender are produced can be controlled. For instance the sender can send single symbols or use different kinds of RNNs to produce a sequence as a message. The same applies to the receiver for parsing the message.

Furthermore, a central problem in language games is how to train the sender. Usually every layer in the receiver model is differentiable and therefore standard backpropagation methods can be used to update the weights in the receiver. However, the sender's messages are discrete symbols and the loss coming from the receiver can therefore not be simply passed on to the sender. A classic solution to this problem is to use REINFORCE algorithms (Williams, 1992) that can handle discrete symbols. More recently, Gumbel-Softmax relaxation (Jang et al., 2017) is used to turn the discrete categorical distribution into a continuous probability distribution. Doing this, receiver and sender act as a whole deep neural model that can be completely backpropagated with standard methods. The methods are discussed deeper in section 3.5. When setting up the language game with the EGG, both REINFORCE and Gumbel-Softmax relaxation can be

used.

In classical machine learning problems, the models are trained using a train dataset for multiple epochs. To verify the performance, the model is verified on unseen test samples, which allows to test the ability to generalize its learned knowledge. Opposed to that, the main goal of language games is not to solve a specific task, but instead the success of communication with a useful emerged language. Consequently, the agents are trained continuously, and the testing doesn't validate the model on unseen samples, but instead is used to extract the exchanged messages, namely the emerged language. In other words, during training, the agents are shown randomly drawn samples from the dataset for each game. This is done repeatedly for k 'games' so that the agents improve and ground their knowledge in the used symbols. During the testing, the agents are frozen and again randomly drawn samples are shown. The messages that the agents are exchanging now represent the language that emerged after k language games which can be analyzed.

Section 3.4 explains the framework in more detail.

2.3.2 Properties of the emerged language

When agents develop an artificial language, they can learn to use it in different ways. Several studies research the properties of emerged languages, in particular if they share properties with human languages. One important property is the capability of forming new meanings by combining multiple symbols in a message, so-called compositionality (Chaabouni et al., 2020; Gupta et al., 2020; Kharitonov & Baroni, 2020; Lazaridou et al., 2018). Lazaridou et al. (2018) construct a referential game where agents are presented with symbolic input instead of visual input in form of images. More specifically, each of the objects, target object and distractors, are represented as one-hot vectors where each dimension corresponds to an attribute of the object. This setup should bias the agents to rely directly on these given attributes in the messages to communicate the target object, instead of first having to extract every possible attribute, for instance from visual features. Hence, this helps them to develop a compositional language. A compositional language should help the agents to communicate objects with unseen combination of attributes during training. Therefore, to analyze the degree of compositionality in the emerged language, the authors present the agents with new combinations of objects and measure the accuracy. Even if the accuracy drops lower, the fact that it is still above the random baseline shows that the agents are able to generalize to a certain degree. Furthermore, they utilize new combinations of symbols in their messages. However, Kharitonov & Baroni (2020) show that compositionality is not necessarily tied to the ability to generalize. In their experiments, non-compositional languages that emerge are able to generalize better than their compositional counterparts.

One essential property of human languages is *Zip's Law of Abbreviation*, in other words that more frequent words tend to be shorter. In (Chaabouni et al., 2019), the researchers analyze if this law also applies to emerged languages. They find that without any pressure to produce shorter strings the agents produce a language that follows the inverse rule: the most frequent inputs are associated with longer messages. Only when long messages are penalized, a language which follows similar rules as human languages emerges.

Finally, a strong focus in the study of emerged languages lies in interpretability (Dessì et al., 2021; Lazaridou et al., 2017; Chaabouni et al., 2021). While the agents ground the symbols in their input and certain symbols correspond to certain categories, they do not necessarily align with human categories and words. However, agents can be biased to use similar categories, as shown in (Lazaridou et al., 2017). This is done by infusing human labels during training so that the agents ground the visual input not only in their artificial language, but also in human language. More specifically, their experimental setup is a referential game, where the sender needs to describe the target image in a set of distractors to the receiver. Alternating with describing the target image to the receiver, the sender is trained to classify its input with human labels. The results show that this setup doesn't have any negative effect on the communicative success, but instead it only biases the agents to align their vocabulary with human descriptions. This makes the emerged language much easier to

interpret for humans.

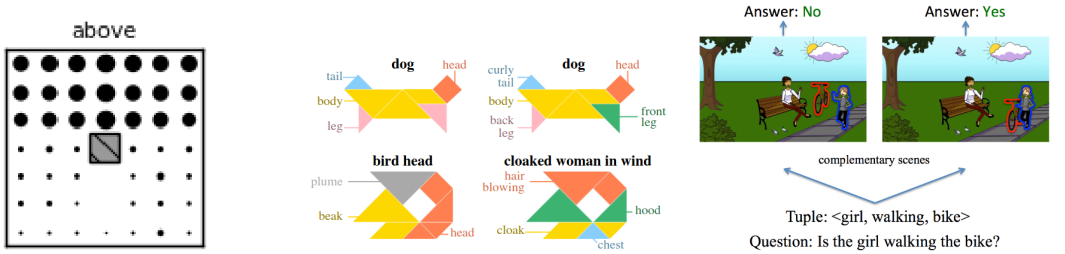
2.4 Artificial dataset

Language is heavily influenced by the perception and the environment (Hofstadter & Sander, 2013; Dobnik & Åstbom, 2017; Ji et al., 2022). This is for instance very apparent in color naming systems across different languages (Zaslavsky et al., 2018). Which and how many words are used to refer to a specific color is dependent on the perceptual and social context around the speakers. Different studies show that while all languages tend to share similar basic color categories, some languages' systems are based on different underlying principles. Color names in the language of the Berinmo people from Papua New Guinea for example reference natural objects and hence color categories are different from non-referential English color names (Davidoff, 2001; Steels & Belpaeme, 2005). Further, this fact can be applied to more abstract constructs like metaphors, which often reference perceived events in the world (Lakoff & Johnson, 1980). Perceiving that cats always land on the ground on their feet is for example the origin of the metaphor 'as nimble as a cat'.

This influence on the language by the perception of the world is not only limited to human language learning and evolution, but it as well applies to machines that are trained to produce language. Many models in different disciplines of Natural Language Processing are trained on datasets such as the MS COCO dataset (Lin et al., 2014) or the Flickr30k dataset (Young et al., 2014), which are sourced from uploaded photos in social networks. While these datasets often contain a large amount of samples, they can contain structural biases. The photos usually feature and center on specific objects, people or animals. This is not necessarily a faithful representation of human perception of the world, in which these entities are for instance partially hidden, shown from the back or appear at the edges of the field of view. The datasets might therefore give a euphemized view on the world. Furthermore, the datasets can contain underlying patterns in the semantic information of the photos (Hendricks et al., 2018; Johnson et al., 2017a; Hudson & Manning, 2019). For instance, when dogs are photographed, they usually appear running on a meadow. An image captioning model might learn to predict the caption 'a dog running on a meadow' for all images containing a dog, independently if the dog is actually running on a meadow or sitting in the living room, just because of the higher frequency in the dataset. In other words, the model might utilize these underlying structural biases rather than relying on the extracted visual information from the image.

Different approaches are taken to mitigate this problem. While some research focuses on improving the models (e.g. Hendricks et al. (2018); Lu et al. (2017)), others create artificial datasets. Using artificial datasets, one can control exactly which structural patterns occur. Furthermore, all ground truths as for example spatial positions of objects are known and can be used for training models. Figure 2 shows several examples of different artificial datasets. Real world photos often contain a lot of distracting visual information that poses challenges for models to identify the relevant information. In (Ghanimifard & Dobnik, 2017) and (Ji et al., 2022), the authors for example create abstract visual scenes that only contain specific concepts, like for instance spatial positions. With no distracting concepts present, researchers can focus only on the effects of the concept that is relevant for the respective study. Other datasets consist for example of comic-like scenes (Zitnick et al., 2013; Zhang et al., 2016). These scenes, while still being abstract also contain functional objects. This adds more complexity for predicting spatial relations. More recently, datasets are created using 3D environments (Johnson et al., 2017a; Lee et al., 2022; Ahrens et al., 2022). Scenes created in this way resemble photos taken in the real world and models can more easily transfer the learned knowledge to real world photos. Extending this reasoning, several datasets include embodied perspectives. Images are for example taken from the perspective of a person in the scene (Dobnik et al., 2015) or from the perspective of an embodied robot navigating a 3D environment (Hill et al., 2021).

This thesis utilizes an artificial dataset for two reasons. First, all ground truth information about the scenes can be extracted during their generation. This includes spatial positions of entities placed in the scene,



(a) Abstract spatial templates (Ghanimifard & Dobnik, 2017)

(b) Abstract tangram shapes (Ji et al., 2022)

(c) comic-like scenes (Zhang et al., 2016)

(d) 3D-generated scenes (Lee et al., 2022)

(e) 3D-generated scenes including perspective (Dobnik et al., 2015)

(f) embodied robot in a 3D environment (Hill et al., 2021)

Figure 2: Example images from different artificial datasets

their properties and relations to other entities. Having all information present, several different tasks for the models can be created using the same generated images. More importantly, the usage of an artificial dataset allows for perfect control over the bias present in the images, in particular which entities are present, and which properties and attributes they have. As detailed before, perceptual input shapes the language that describes and interacts with it. When agents in a language game describe and refer to entities present in a scene, the emerged language is expected to utilize these biases by making use of the controlled attributes in the scenes.

3 Methodology and Frameworks

3.1 CLEVR framework

The basis for my datasets is the Visual Question Answering (VQA) framework CLEVR (Johnson et al., 2017a). This framework provides code to generate configurable VQA datasets that are split in two parts: a collection of images, and a set of questions and answers that refer to and describe each image. Many of the existing VQA datasets come with two problems. First, they include many biases, such as biases in the base images and biases in the linguistic properties of the questions and answers. A relatively high number of images of dogs in a dataset, might for instance bias a classifier model towards classifying dogs most of the time. On the other hand, repeating patterns in the questions and answers might also be exploited by a model, without extracting the needed information from the image. For these reasons the CLEVR framework aims to reduce the biases as much as possible in both images and questions and answers as well as precisely control the remaining biases to explore their limits. Secondly, datasets may come with only a limited amount of annotations and information about the state in an image. The CLEVR dataset uses artificially rendered 3D-scenes. By doing so, all information about for instance the location of objects or their relations to each other can be later used in training models or analyzing their results. Furthermore, it allows making predictions about the effects of varying contexts and information present in the scene.

For this thesis, the CLEVR framework will be extended to have more control over the generation of the images. With this extension, several datasets are created. The extensions are described in Chapter 4. Hereby, only the images and their ground truth properties are interesting for the present study of referring expressions. The questions and answers won't be used. In the following section the image generation of the original CLEVR framework is described. The visual part contains images of 3D-generated scenes depicting different kinds of simple objects. Each of these objects is made up of a different combination of attributes, such as *shape*, *color*, *size* and *material*. The possible values of these attributes are listed in Table 1. Three to ten objects are placed in random locations into the scene and assigned with random attributes. To enhance realism and reduce ambiguity, objects are placed in a way so that they do not intersect and have a certain distance from each other. Furthermore, it is made sure that every object is almost completely visible. The positions of the light and the camera are slightly jittered for each image to add noise and reduce recurring patterns. Since the objects are part of 3D scenes, they may look different in each image, because of different lighting and shadows, distances to the camera and rotations. This noise approximates the real world and natural environment, and makes it harder for models to learn compared to relatively noise-free projections on a 2D plane. Figure 3 shows an example of a generated image in the CLEVR dataset.

shape	**color**	**size**	**material**
cube	gray	small	rubber
sphere	red	large	metal
cylinder	blue		
	green		
	brown		
	purple		
	cyan		
	yellow		

Table 1: Attributes of objects in the CLEVR dataset

Furthermore, the dataset contains information about each scene. This includes all selected attributes for each object as well as the exact position of the centers of all the objects, both 3D-coordinates in the 3D scene and 2D-coordinates in the final rendered image. In addition, simple spatial relations (in front of, behind, left, right) between the objects are calculated and stored. These are simply based on the 3D-coordinates of

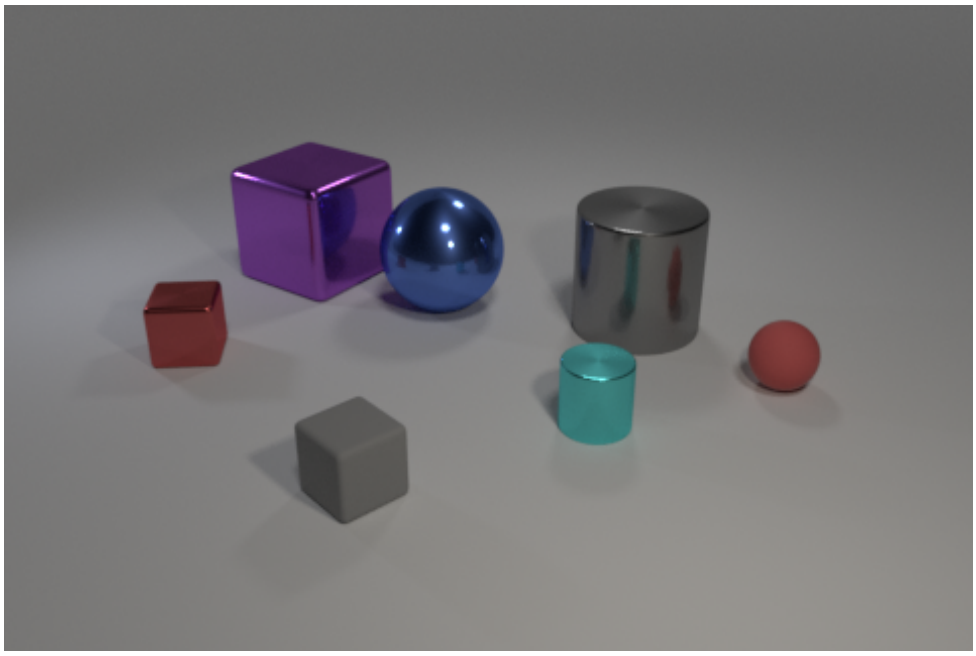


Figure 3: Example of a generated image in the CLEVR dataset

the objects in relation to the position of the camera.

3.2 Feature extractors

In computer vision tasks, machines need to analyze images and extract information from them. To do this, machines often rely on feature extractors. Features are important parts or patterns in an image, which can have different levels of abstractions. They can for example be low-level features, as geometric information about lines and edges in an image, or also very abstract information about whole objects. Traditional approaches involve extracting key points and descriptors from an image and using them to represent the image (Harris & Stephens, 1988; Lowe, 1999; Bay et al., 2006). More recently, convolutional neural networks (CNN) became popular due to their ability to learn complex features automatically from raw image data. These are also used in this thesis as a first layer to extract important information from the image. Hereby, there are currently two different architectures.

First, the VGG19 (Simonyan & Zisserman, 2015) is used which is an architecture based on many convolutional layers. Using 16 to 19 convolutional layers with small convolution filters helps the model to solve localization and classification tasks on the training dataset, but also enables it to generalize onto other datasets. After the convolutional layers, the data is passed first through an average pooling layer which outputs $512 \times 7 \times 7$ dimensions. Next follow three linear layers with *ReLU* non-linearities in between. After flattening the input, these classification layers output 4069, 4069 and 1000 dimensions respectively.

The second architecture is ResNet-101 (He et al., 2016). This architecture tries to overcome the degradation of very deep networks, where the accuracy rapidly drops after it gets saturated. This is done using residual blocks. A residual block consists of two or three convolutional layers and a residual connection, also known as a shortcut connection. The residual connection allows the input to be added directly to the output of the block, allowing the network to learn the residual function with respect to the input. This approach enables the network to better preserve information from earlier layers and avoid the problem of information loss that can occur in very deep networks. There are four blocks that output $256 \times 56 \times 56$, $512 \times 28 \times 28$, $1024 \times 14 \times 14$ and $2048 \times 1 \times 1$ dimensions. A following average pooling layer outputs $2048 \times 1 \times 1$ dimensions as well. The final linear layer reduces the flattened data to 1000 dimensions, corresponding to the ImageNet classes.

Both architectures are available pretrained on an image classification task on the ImageNet dataset (Deng et al., 2009). In this thesis, the implementations and weights available for PyTorch are used.[1,2] In both cases, images that should be passed through the feature extractors need to have the same structure and shape as the images that were used in the pretraining. Hence, all images passed to them in the experiments in this thesis are preprocessed in the following way. As described in (He et al., 2016; Simonyan & Zisserman, 2015), the images are resized to 256 pixels for the shorter side, then cropped around the center to a square of 224×224 pixels. Finally, the RGB channels are normalized, by subtracting the means (0,485; 0,456; 0,406) and dividing the result by the standard deviation (0,229; 0,224; 0,225) for each channel respectively.

Since the task in this research is very different from a classification, it likely learned representations that are not directly transferrable to other tasks. The pretrained knowledge in both models might not be directly transferrable to new tasks as studied by Yosinski et al. (2014). In this thesis, the task differs for multiple reasons. First, the used images for the pretraining and the present study are part of a different domain. Even though images of the CLEVR framework are generated to resemble the real world, they still only include abstract geometric objects while ImageNet contains real photos of persons, animals and objects. Furthermore, the original task for the pretraining is a classification task, whilst this thesis is interested in generating and understanding referring expressions. For this reason, multiple different adaptions of these architectures are usually used. Table 2 lists the different adaptions for both VGG19 and ResNet-101, where later layers are omitted. Preliminary results showed that early layers like *ResNet-1* to *ResNet-3* or *VGG-0* and *VGG-avg* provided better results. This corresponds to the hypothesis that later layers include too much task specific knowledge from pretraining. There was no visible difference between *VGG* and *ResNet*, so *ResNet* is used in all experiments due to its more efficient architecture. However, *ResNet-1* and *ResNet-2* contain a high number of dimensions, that slows the training and raises the amount of needed memory on the GPU. Some experiments couldn't be performed using these layers, so all experiments are run, using *ResNet-3*.

	description	**output dimensions**
VGG-0	contains only the convolutional layers	$512 \times 7 \times 7$
VGG-avg	contains an additional average pooling layer	$512 \times 7 \times 7$
VGG-cls1	contains an additional one classification layer, including its non-linearity	4069
VGG-cls2	contains another additional classification layer, including its non-linearity	4069
VGG-cls3	the original VGG19 architecture	1000
ResNet-1	contains one residual block	$256 \times 56 \times 56$
ResNet-2	contains two residual blocks	$512 \times 28 \times 28$
ResNet-3	contains three residual blocks	$1024 \times 14 \times 14$
ResNet-4	contains four residual blocks	$2048 \times 1 \times 1$
ResNet-avg	contains an additional average pooling layer	$2048 \times 1 \times 1$
ResNet-cls	the original ResNet-101 architecture	1000

Table 2: Different adaptions of VGG19 and ResNet-101 used in this research

3.3 Image processing

In the experiments different agent architectures are compared. To make these architectures comparable, the agents' models are composed of submodules that solve a specific subtask. Submodules can for example

[1] https://pytorch.org/hub/pytorch_vision_resnet/
[2] https://pytorch.org/hub/pytorch_vision_vgg/

be image encoder modules or text parsing modules. Hereby, different ways of solving the subtask are manifested in multiple exchangeable submodules. For instance, one image encoder module uses CNNs while another image encoder module uses a traditional approach. These can simply be exchanged in an agent's model without needing to adapt the whole agent. Consequently, the experiments stay comparable, since the differences are only added or removed submodules.

The feature extractors described in section 3.2 are not enough to be solely used to encode the images used in this thesis. Even though final layers that contain very task and domain specific knowledge can be removed, the previous layers still don't contain any information about the new domain and task. For this reason they need to be extended with further layers that are not frozen and can learn and store information about the new domain by connecting the pretrained general knowledge with the specific task. These additional layers on top of the feature extractors that make up the *image encoder* submodule, however, are always trained from scratch for each of the experiments and tasks. In this thesis, only this submodule is used to encode images for all the conducted experiments, except otherwise specified.

Johnson et al. (2017b) describe an architecture that was used for training baseline models on the CLEVR dataset. This architecture will be used as the submodule. Hereby, the image is first passed through a frozen feature extractor. Two convolutional networks with subsequent *ReLU* non-linearities condense the important information from the output of the feature extractor. The convolutional layers reduce the channels to 128 channels, using a kernel size of 3 and a stride and padding of 1. Afterwards, a 2-dimensional max pooling is applied with a kernel size and stride of 2. Finally, the resulting matrix is flattened and passed through a linear layer to reduce it to an encoding size. This vector represents the encoded image with its extracted features. Table 3 shows the layers and their output dimensions when using ResNet-3 as the feature extractor and an image embedding size e_i.

	layer	**output dimensions**
	Input image	$3 \times 224 \times 224$
1	ResNet-3	$1024 \times 14 \times 14$
2	Conv(3×3, $\rightarrow 128$)	$128 \times 14 \times 14$
3	ReLU	$128 \times 14 \times 14$
4	Conv(3×3, $\rightarrow 128$)	$128 \times 14 \times 14$
5	ReLU	$128 \times 14 \times 14$
6	MaxPool(2×2, stride 2)	$128 \times 7 \times 7$
7	Flatten	6272
8	Linear Layer($\rightarrow e_i$)	e_i

Table 3: Image encoder with ResNet-3 and an image embedding size e_i

In some experiments, the focus of the model should be pointed to certain regions in the image. This is done by providing the model additionally to the original image of the scene with a masked version of the scene. In particular, the masked version only contains black and white regions. All pixels in the relevant region are white, while all remaining pixels outside this region are black. This is expected to help the model to learn which are the important parts in the original image. However, a challenge lies in combining both original image and masked image. When extracting features from an image using the previously described submodule, the output of the final linear layer doesn't directly correspond to pixels and regions in the image, but only contains the important extracted features sequentially. In other words it looses the 2D geometrical structure of the image. The model will have problems to align these extracted features with the masked image, still having the 2D geometrical structure.

For that reason, another submodule for masked images is used, the *masked image encoder*. In this submodule, the original image and the masked image are combined at an earlier stage, when both have the same

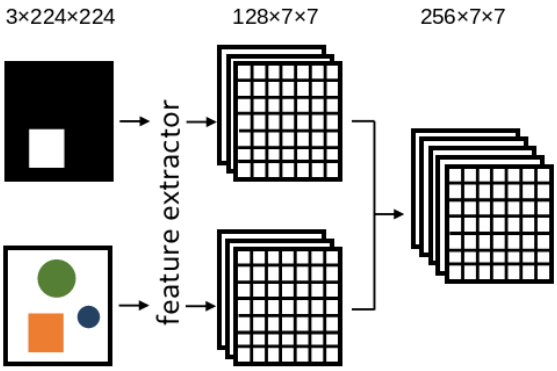


Figure 4: Combination of the original image and its masked version

structure (see Figure 4). In particular, both images are processed in exactly the same way, by passing it through a feature extractor and then through the above described extending layers until layer 6, the max pool layer. Even though the masked image doesn't contain similar features to the images the feature extractors are pretrained on, this method still provided better results in the experiments. After this, the images are concatenated along the channels, which results in a matrix with the dimensions $256 \times 7 \times 7$ (still including the geometrical structure with 7×7 grids). Finally, this matrix is flattened and reduced to the image embedding size e_i. Doing this, the model can directly align a region in the original image with a region in the masked image.

3.4 EGG framework

The goal of this thesis is to run and compare different setups of language games systematically. To do this, all experiments rely on the ***E**mergence of lan**G**uage in **G**ames* (EGG) framework (Kharitonov et al., 2019) which is implemented in PyTorch. This framework allows the implementation of language games in code, where agents are neural models that communicate with each other. It consists of a heavily configurable core that controls the generating and parsing of the message, the calculation of the loss and the rules, for how the weights of all neural models are trained. The configuration includes for example:

- a choice between single symbol and sequence messages with varying RNNs,
- an easy switch between different loss functions,
- or a choice between two optimization functions (Gumbel-Softmax relaxation and REINFORCE algorithms) to learn neural models containing discrete symbols.

Furthermore, runs of games can be saved to analyze the used messages of the agents and how they vary over the duration of the learning.

The EGG framework is set up in three levels, shown in . Part of the lowest level are the *agents* themselves. The agents are neural models that need to be implemented from scratch and define how the agents process their input and in case of the receiver combine it with the message. The second level consists of *wrappers* that take care of generating and parsing the message. The sender wrapper uses the output of the sender agent, to produce a message. The receiver wrapper on the other hand parses the message received by the sender and passes the result as an additional input to the receiver agent. The third level, the *game*, links all described parts together. It provides the agents with the input and passes the message from the sender to the receiver. Furthermore, it uses the output of the receiver and calculates the loss, which is then the basis for the adaption of the weights for both wrappers and agents.

For the language games which are run in this thesis, the sender will always produce a sequence of symbols

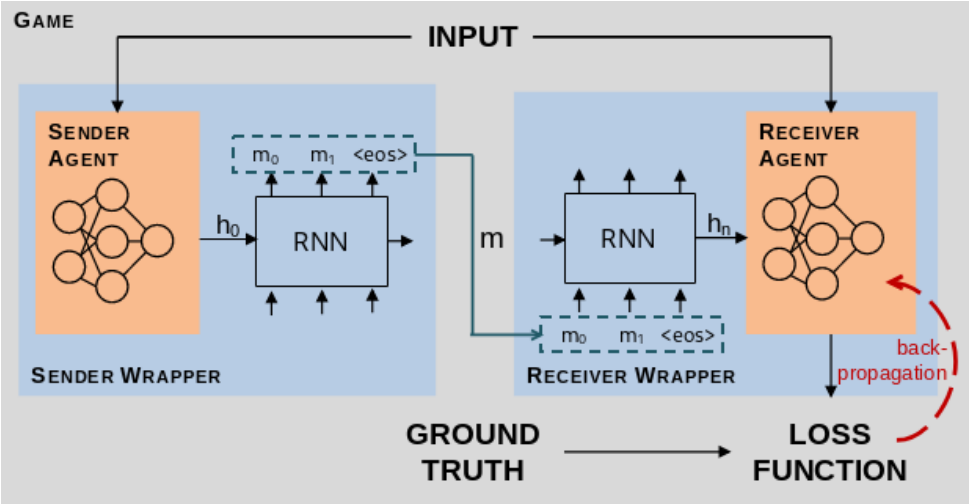


Figure 5: The EGG framework is made up of three levels: the game (gray), the wrappers (blue) and the agents (orange)

as a message, which the receiver will parse. Gumbel-Softmax relaxation is applied to produce discrete symbols. This is done in the default method of the EGG framework using two LSTMs, an encoder LSTM in the sender wrapper and a decoder LSTM in the receiver wrapper. The output of the sender agent is used as the initial hidden state for the encoder LSTM. This LSTM is then producing symbols until it generates an end-of-sequence symbol. This sequence is then passed to the receiver wrapper with its decoder LSTM. Its hidden state is initialized randomly. The received message sequence is processed symbol by symbol. After each time, a symbol is processed by the LSTM, the resulting new hidden state is passed to the receiver agent as the parsed message. The receiver agent is combining it with its representation of the image input and is predicting an output. In other words the receiver agent produces as many outputs as symbols are present in the message. The *game* is then calculating a loss for each of these outputs separately. These losses are summed up to a total loss that is used to adapt the weights in both agents as well as in both LSTMs.

3.5 Optimization in language games

Training agents in a language game is not a straight forward task. In this thesis, the agents share a common goal, in particular optimizing the predictions of the receiver. The loss function of the whole game can therefore be defined as the loss of the receiver's predictions given its input and the sender's message. By this, the receiver can be trained relatively easily by backpropagating the game's loss through all layers in the model of the receiver. A problem however arises when passing on the loss to the sender. The generation of the message involves sampling from a categorical distribution and is therefore not differentiable. Consequently, the gradients of the loss function can't be estimated with standard backpropagation methods for the sender. To solve this problem two dominant methods are currently used: REINFORCE algorithms (Williams, 1992) and Gumbel-Softmax relaxation (Jang et al., 2017).

The traditional approach to solve problems with a discrete channel formulates the problem as a reinforcement learning task and relies on the REINFORCE algorithms. In reinforcement learning, the neural models are trained to optimize their policy given a reward. Given for example the referential game, described in (Lazaridou et al., 2017), where the sender is presented with two images and needs to communicate the target image to the receiver. The receiver is shown the same images and is tasked to point to the correct target image given the sender's message. The sender model and receiver model are following a policy with parameters θ respectively. The policy of the sender can be defined as $s(\theta_S(i_L, i_R, t)) \in V$ with θ_s being the parameters of the senders neural model, i_L and i_R the presented images, t the target image and V the vocabulary. The receiver's policy is dependent on the message by the sender and therefore directly

dependent on the sender's policy and can be defined like this: $r(i_L, i_R, s(\theta_S(i_L, i_R, t))) \in \{L, R\}$. The reward is given if the receiver's policy successfully produced the target image:

$$R = \begin{cases} 1, & \text{iff } r(i_L, i_R, s(\theta_S(i_L, i_R, t))) = t \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

The task for the agents to learn is to maximize the reward function R across all possible rewards (dependent on receiver's policy and target image) $\mathbb{E}_{\tilde{r}}[R(\tilde{r})]$. The problem hereby is that not all possible rewards are known. Without training, it can't be determined if a certain configuration of the agents, the policy, yields the correct target image. The rewards are in fact the outcome of the training. To solve this, Monte Carlo sampling is used to approximate the expected value $\mathbb{E}$. Monte Carlo sampling works on the fact that the mean of all drawn samples from an unknown probability distribution approximates its expected value. The more samples are drawn, the better the approximation. Using the approximation, the function becomes differentiable and can be used to calculate the gradients to update the policies. In this case, the approximation is based on the already determined rewards, in other words on the already played games. Early games are more or less trial and error, because the approximation is very bad. Only when many games are played, the approximation becomes better and the models can be updated properly. This however also means that the training can take very long until the models start to converge, and it suffers from high variance.

For this reason, more recently Gumbel-Softmax relaxation is used to represent the discrete message. Basically, this method is a way to turn a discrete categorical distribution into a continuous distribution that can be differentiated. By doing this, sender and receiver are completely differentiable and act as whole deep neural model where the loss can be backpropagated with standard methods. In particular the problem lies in sampling from the categorical distribution C over $|V|$ symbols, which can't be differentiated. Therefore, the method applies a reparameterization trick: random samples are instead drawn from the Gumbel distribution G (Equation 2) and summed with the logarithmic probabilities of each symbol (Equation 3).

$$g = -log(-log(u)) \qquad \text{with } u \sim Uniform(0, 1) \tag{2}$$

$$s_k = log(c_k) + g_k \qquad \text{for } k = 1, ..., |V| \tag{3}$$

To determine the symbol to use, usually the $argmax$ function would be applied on the resulting vector s. However, this function is again undifferentiable and is therefore approximated using the $softmax$ function:

$$y_k = \frac{exp(s_k/\tau)}{\sum_{i=1}^{|V|} exp(s_i/\tau)} \tag{4}$$

The temperature τ of the $softmax$ function determines how well it approximates the $argmax$ function. For $\tau \to 0$, the samples from the Gumbel-Softmax distribution are identical with the categorical distribution while for $\tau \to \infty$ the Gumbel-Softmax distribution becomes uniform.

The Gumbel-Softmax method offers stable gradient estimates and can be more efficient than the traditional REINFORCE algorithm, especially when dealing with large action spaces. This also applies to language games, where Havrylov & Titov (2017) demonstrated that Gumbel-Softmax relaxation is more effective. For that reason, the experiments in this thesis utilize Gumbel-Softmax relaxation.

3.6 Ethical considerations

In the field of natural language processing (NLP) ethical issues often play major roles. These can be part of the used datasets, the created models and their training as well as the application of the models. For

datasets, the role data privacy is increasing with the necessity of larger amounts of data (Klymenko et al., 2022). Furthermore, datasets often contain biases, based for instance on the authors of the collected natural language texts. Often they also contain biases such as overrepresentations and underrepresentations. Even though some of these biases are inherent to the data and not necessarily negative, much research is indicating that undetected and unaddressed biases in datasets might lead to negative consequences (Shah et al., 2020; Field et al., 2021; Bender et al., 2021). Training neural models can also lead to environmental issues, as large models need to process huge amounts of data and require a lot of energy (Bender et al., 2021). Finally, the application of trained models can create harm. This applies for example to easy accessible large language model (LLMs) that can be used to create information hazard (Weidinger et al., 2022).

The research in this thesis tries to reduce these risks. Looking at the datasets that are used in this thesis, all data is created artificially and contains therefore no personal information. Even further, the aim of the creation of these datasets is to reduce and study the remaining biases. It doesn't include any social information, but on the other hand consists only of abstract scenes. The choice of which attributes the objects are made up is inherently biased towards human cognition, but doesn't have a social impact. The models and agents are therefore trained, by including as few human biases as possible.

Looking at the environmental issues, the models used in this thesis consist of only few trained layers and the training is therefore short and doesn't require much energy. Larger models as the feature extractors are already pretrained and add no additional consumption.

Finally, the purpose of this thesis is to analyze the results and the emerged language and draw conclusions, how emerged languages can be grounded better in the environment. For that reason, the final models can't be used in any real world applications and produce potential harm. On the other hand, this work can on the long run mitigate harm as it provides a study of models, how they would behave on real data. This therefore contributes towards interpretability of AI.

4 Creation of the datasets

This research investigates how agents use referring expressions to discriminate objects seen in images based on their relations. For that reason, the original CLEVR framework offers too little control over how a new dataset is created. Especially, which objects and in which attributes they share with each other in each image can't be controlled. Following, we extended the framework for generating new datasets.[3] By this, the objects in the generated images are controlled to have different human-recognizable attributes, namely the *shape*, *size* and *color*. These attributes also correspond to referring expressions in natural language such as English. The *material* is always the same for all objects in a generated image. There were three main extensions to the framework:

First, objects in the scene were separated into three categories: one *target object*, objects in a *target group* and *distractor* objects. The target object is the main object in the scene and the models are trained to identify and communicate between each other. All other objects and their relations are based on this target object. The target group contains similar objects to the target object. These are objects that the agents need to discriminate the target object from. Finally, the distractors are objects that add noise to the scene and should make it more complex. They are expected to teach the agents more precise descriptions of the target object. The number of the objects in both groups can be controlled.

In a second step, when generating the images it is possible to define the relations between *target object*/*target group* and *target object*/*distractors*. The relation is defined as **how many** attributes of the target object are identical with the attributes of a single object in the target group and distractors respectively. For example the target object is a *small red cube*. If two attributes are shared between target object and target group, objects in the target group could include *small blue cube*, *big red cube* or *small red sphere*, but couldn't include another *small red cube* or a *small blue cylinder*. The number of shared attributes can also be set to a range to control how challenging the referring task is.

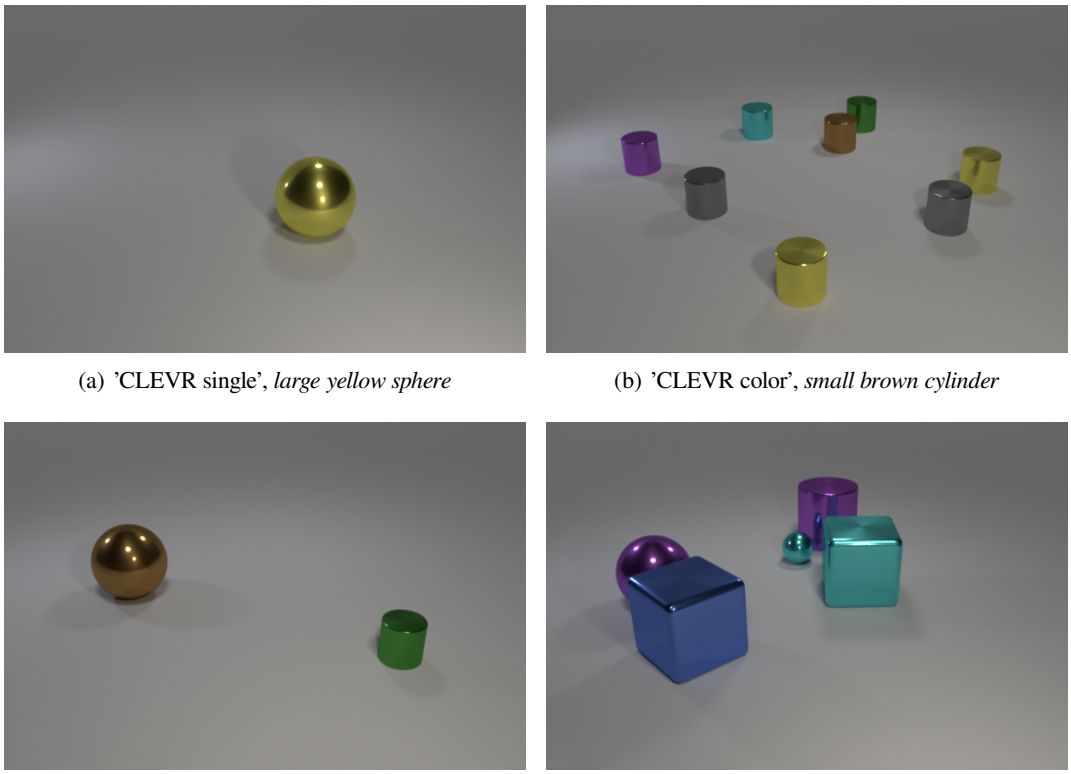
Lastly, it is also possible to define exactly **which** attributes should be shared between the target object and the groups. For example, it can be defined to have the same size for objects in the target group, but have different, randomly selected shapes and colors. This allows for a very controlled generation of relations between the objects in the scene. Figure 6(d) shows one generated image with this extended source code. Here, the target object is the large purple cylinder. The target group contains four objects that share zero to a maximum of two attributes. It is not controlled, which attributes are shared (they are selected randomly). The large purple cylinder shares the same color and size with the large purple sphere, the same size with both cubes and no attribute with the small turquoise sphere. There are no distractor objects.

For all generated datasets in the following sections, the general constraints and settings are as close as possible to the original CLEVR dataset. The size of the generated images is 480x320 pixels. 10.000 images are created for each of the datasets. Each image contains a maximum of 10 objects, that are not intersecting, have the same minimum distance between objects and are at least partially visible from the camera.

4.1 CLEVR single

The simplest new dataset is called 'CLEVR single'. This is a very simple dataset and has the purpose to simplify the problem the model needs to learn as much as possible. Each scene in the dataset contains only one single object, the target object. There are neither objects in the target nor in the distractor group. All attributes are assigned randomly to the target object. The differences across the whole dataset are the

[3] https://github.com/DominikKuenkele/MLT_Master-Thesis_clevr-dataset-gen



(a) 'CLEVR single', *large yellow sphere*

(b) 'CLEVR color', *small brown cylinder*

(c) 'Dale-2', *small green cylinder*

(d) 'Dale-5', *large purple cylinder*

Figure 6: Example images of each dataset, with the target object specified

locations and rotations of the objects. With this dataset, neural models can focus on only the features, as well as the locations of this single object. There are no objects that distract the model from extracting features from the target object. This helps to understand if the models are actually able to assign features or learn locations of these features in an image. Figure 6(a) shows an example with the only object being the *large yellow sphere*.

4.2 CLEVR color

The second dataset that is created is called 'CLEVR color'. The purpose of this dataset is to create scenes, where the target object is completely unique and as easily identifiable as possible. For this reason, there exist only two groups in the scene, the target object and distractors. The distractor group can contain in between 6 and 9 objects. To make the discrimination as simple as possible the target object and the objects in the target group share exactly two attributes. Furthermore, to simplify the relation between target object and distractors over the whole dataset, it is also controlled which attributes are shared. The distractors have always the same size and shape as the target object, but the color is different. The reason for choosing the color as the only discriminating attribute is that it is assumed that the color is easier to learn for neural models as opposed to for instance abstract shapes.

As seen in Figure 6(b), the *small brown cylinder* is unique. By this, it is possible to refer to the target object using the attributes with four different combinations: the *brown* object, the *brown cylinder*, the *small brown* object and the *small brown cylinder*. All attributes, apart from the color are not discriminating the target object from the distractors. Notice as well that this restriction doesn't apply to the distractors, where multiple objects with the same color are allowed. In other words, the choice of attributes is random for the distractors, and they may overlap.

4.3 CLEVR Dale datasets

The above described dataset is very restrictive in the relation between the objects, where only *one* attribute is used to disambiguate them. The number and the type of shared attributes are controlled exactly. In the real world, objects have overlapping attributes and hence objects can only be identified by an intersection of multiple attributes. In real situations, there is no restriction at all how objects or things relate to each other. Natural language emerged that can refer to distinct attributes of these objects to discriminate them from each other. This emergence of referring attributes and their combination is studied deeper in this work.

For this, we created a dataset that allows almost any relation between a target object and the distractors. However, the creation is inspired by incremental algorithm for the Generation of Referring Expressions (GRE) described in (Dale & Reiter, 1995) who observe that attributes in descriptions occur in certain order and are added incrementally in a certain hierarchy. This algorithm ensures that every scene contains a unique object in respect to its and the distractors' attributes. Using the algorithm, one can refer to an object using its attributes to discriminate it from all other objects as efficiently as possible. In other words, the object is described unambiguously using the lowest number of words. For the dataset that means that zero, one or two attributes can be shared between the target object and distractor objects. This ensures the uniqueness of the target object. On the other side, it is not controlled which attributes are shared. These are assigned randomly. There is again no control over the relations between distractors, which means that distractors can appear multiple times.

Two datasets following these rules are created. The Dale-2 dataset contains one target object and one distractor (see Figure 6(c)), while the Dale-5 dataset contains one target object and exactly four distractors. Consider Figure 6(d), with the target object being the *large purple cylinder*. The large purple sphere shares the size and color, the two cubes only share the size, and the small turquoise sphere doesn't share any attribute.

These two datasets allow a more realistic look in how models can acquire knowledge about attributes of objects. More specifically it helps to understand how models learn to discriminate objects from each other, since the model may only need to learn discriminative features of objects and not all features of the whole object.

[Dominik Künkele] probabilities of shared attributes for both Dale-2 and Dale-5

5 Grounding referring expressions

This chapter serves two purposes. First, the generated dataset from the precious section is validated. For this, models are trained to both generate referring expressions of the target object and understand existing referring expressions. A success of these experiments indicates that the target objects in the datasets are possible to refer to and the datasets can be used in more complex setups in language games.

Secondly, the experiments in this chapter provide the basis for the setup of the agents in the language games. Language games are very complex setups for machine learning models. The models need to solve multiple tasks at the same time in order to solve the overall problem. For instance, in a simple setup of a game two agents are involved. The first agent, the sender, is shown a scene with objects and needs to communicate one target object to the other agent, the receiver. The receiver is shown the same scene and needs to identify the target object with respect to the message of the sender. In this case, the sender first needs to learn to encode the scene, all objects and their attributes, as well as the information about the target object into its own game specific space. In a next step it needs to learn how to translate this encoding into a message that is sent to the receiver. The receiver then needs to learn to decode this message, after which it needs to learn how to combine the decoded message with its own encoding of the scene and objects. And finally it needs to learn how to identify the target object with this information. There are many points of possible failure to train the agents.

For this reason, we decided to divide the main problem and let the models learn simpler subtasks and increase the complexity step by step.[4] This will give a very detailed overview where the models struggle to learn and in which ways they can be improved. Mainly, the tasks are separated into language games with two agents and classical machine learning tasks without any communication, namely only one 'agent' that solves the task alone. With this division, we can analyze the learning of the encodings of the scenes separately from the learning of producing and decoding messages.

The final objective of this thesis is to find out, how agents can communicate about relations of objects based on their attributes, namely how to generate and understand referring expressions. Because of that, the first experiments focus on extracting information from images and combining them with structured knowledge about the objects. Here, we structured the experiments into three levels. In the first level, the models are trained to learn the position of objects in the image and attend to specific regions of the image, by understanding a referring expression of the target object. In the second level, the models are trained to differentiate objects in the scene from each other, again by understanding referring expressions. In the last level, the models are trained to generate referring expressions, more specifically the models learn to caption and describe objects in the image. These combined experiments should lay the basis for how to build up the agents in the language games.

For all experiments in this section, the batch size is 32 samples and the optimizer is *Adam* (Kingma & Ba, 2015). 8000 randomly selected samples are used for training, the remaining 2000 samples for testing. Other hyperparameters are specified for each experiment.

[4] https://github.com/DominikKuenkele/MLT_Master-Thesis

5.1 Object identification

Setup

In a first task a neural model is trained to discriminate multiple objects from each other. Hereby, the model is shown bounding boxes of all objects in the scene as well as a description of the target object. The model needs to combine all information and then point to the correct bounding box. This task forces the model to extract visual features from the objects in the shown inputs and connect them to a representation of human given attributes. Since the model is shown only bounding boxes around each object rather than the whole scene, geometric information about the objects' locations doesn't play a role for the success in this task.

In a first step, bounding boxes are extracted from each scene. Each bounding box is a square with a side length of 96 pixels around the center coordinates of each object. By doing this large objects in the front of the scene fill the bounding box completely, while smaller objects in the back of the scene contain some space around them. They also might contain noise in the form of parts of adjacent objects. Preliminary experiments indicated however that a more complex method for extracting the bounding boxes, depending on spatial position in the scene to reduce the noise didn't improve the results. Therefore, bounding boxes in this thesis rely on the former simpler approach. Since, each bounding box will be passed through on of the feature extractors, they are preprocessed as described in Section 3.2. The array of bounding boxes is padded with matrix of zeros to the maximum possible number of objects present in a scene across the dataset. For the 'Dale-2' dataset, this corresponds to a maximum of two bounding boxes, 5 bounding boxes for the 'Dale-5' dataset and 10 for the 'CLEVR color' dataset. For each sample in the dataset the bounding boxes are shuffled.

The attributes of the target object are encoded as one-hot encodings. There is a three-dimensional vector encoding the *shape*, an eight-dimensional vector encoding the *color* and a two-dimensional vector encoding the two different *sizes*. The values of each dimension of these vectors can either be zero or one, depending on the attributes of the target object. These three encodings of the attributes are then concatenated to a vector with 13 dimensions.

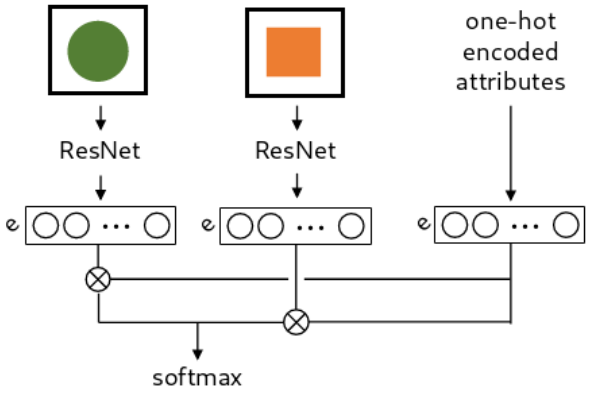


Figure 7: Simplified architecture of the object identification task

The model, the **object identifier** is split into three submodules, the *bounding box encoder*, the *attribute encoder* and the *identifier*. Each normalized bounding box of the sample is encoded with the *bounding box encoder*. Hereby, they are passed through *ResNet-avg*. The preceding layers *ResNet-1* to *ResNet-4* couldn't be compared due to memory restrictions on the GPU. The resulting vector is flattened and projected to the embedding dimension e via a linear layer. Correspondingly, the one-hot vector of the attributes is projected to the same embedding dimension with another linear layer, using the *attribute encoder*. In the *identifier* submodule, each bounding box is combined with the representation of the attributes. By bringing both vectors to the same embedding size, the dot product between each embedded bounding box and the

embedded attributes can be calculated. A high correlation between the attributes and the object should result in a high dot product, while a low correlation results in a lower dot product. The model can so learn to connect the object representation with the attributes. The dot products are concatenated and form a vector with as many dimensions as objects present. The $softmax$ function is applied over the resulting vector, which returns a probability distribution over all objects; the model points to the object with the highest probability.

The experiments are conducted with the following hyperparameters: a learning rate of 2×10^{-4} and 30 epochs. The number of epochs was chosen manually after identifying when the loss as well as the test accuracies didn't improve significantly. The loss is calculated using cross entropy. The following embedding dimension e are compared: 2, 10, 50, 100, 500, 1000.

The object identification task is trained on all datasets that include more than one object in each sample, namely the 'CLEVR color', 'Dale-2' and 'Dale-5' dataset. Random baselines for each dataset would yield accuracies of 10%, 50% and 20% respectively. The 'Dale' datasets are directly comparable to each other for the similar setup of their creation. Especially interesting is the effect of the increasing the number of distractors and the growing number of attributes that are needed to discriminate the objects.

Results

Table 4 lists the accuracies of the models' predictions after 30 epochs for all three different datasets and different embedding sizes e.

	Dale-2	**Dale-5**	**CLEVR Color**
e	**Accuracy**	**Accuracy**	**Accuracy**
2	92%	72%	40%
10	**100%**	94%	92%
50	99%	**95%**	**94%**
100	**100%**	**95%**	93%
500	**100%**	**95%**	93%
1000	**100%**	94%	**94%**

Table 4: Accuracy scores of the object identifier after 30 epochs: e are different embedding sizes

The first trend that is visible is that fewer distractor increase the accuracy of the model. The model achieves almost perfect accuracy in all configurations for the *Dale 2* dataset with only two objects. With five objects in the *Dale-5* dataset, the accuracy drops to 95% and with a maximum possible number of 10 objects in the *CLEVR color* dataset the model only achieves 94% in the best configuration. This is not very surprising, since the model has a higher chance of predicting a wrong object, given the random baselines of 50%, 20% and 10%. Furthermore, when more objects are shown to the model, more attributes are needed to discriminate the target object from the distractor, since it is more likely that the distractors share attributes with the target object. The task is getting therefore more challenging.

A second conclusion is that the model needs a certain embedding space, to represent the features of each image. Across all dataset, an embedding space of only 2 dimensions result in a much lower performance compared to the best configuration. Even though the model is still able to achieve an accuracy of 30% to 50% points over the random baseline, this shows that a bigger embedding size is beneficial to extract and represent the features of the objects. Between 50 and 1000 dimensions, the model performs the best and varies only by 1% point, which may be due to different random initialization of the weights in the model.

In conclusion, the model is able to learn to discriminate the objects based on the visual appearance. The

model can generate these high results, even with its relatively simple architecture. In this architecture, the model doesn't compare the objects directly to each other, but each object is only associated with the attribute encodings. A more complex architecture, in which the model is additionally tasked to discriminate the objects directly from each other might even improve the results.

5.2 Referring expression generation

Setup

Opposed to the previous experiment, where the focus lied on extracting visual features, the model is now tasked to generate referring expressions. This is done in two setups. The first setup uses bounding boxes as in the previous section as input, where the model needs to describe the target object of one of the bounding boxes. In the second setup, the model is presented with the complete image and therefore also needs to accommodate to geometric information of the scene.

The bounding boxes are extracted and normalized in the same way as for the object identification task. Since for the second setup, the images will also be passed through the feature extractors, they are normalized as well in like manner. The referring expressions for the target object are generated using the incremental GRE-algorithm, described in Section 2.2. By this, the model needs to describe the target object with respect to the distractor objects. There are some minor additions concerning the padding of the referring expression. As before, the referring expression is padded to a number of three tokens, corresponding to the maximum of three attributes. However, there are three different ways how the padding is applied. First, the referring expression are, as usual in captioning tasks padded at the end with a specified padding token. A problem could arise when the referring expression is not viewed as a natural language sentence, but as slots filled with tokens. More specifically, following the GRE-algorithm, the last token in the referring expression is always the shape. The second last token if existing describes the color, while the third last token if existing describes the size. As soon as this sequence is padded at the end, these slots disappear. A referring expression that only describes the shape, such as *cube* will be padded to *cube* <*pad*> <*pad*>, where the third last slot is filled up with the shape instead of the size. Since this task is not focussing on producing natural language with a correct grammar, but focuses instead on extracting attributes, having a slot structure could help the model to express the extracted attributes correctly. For this reason, the second method of padding the referring expression is prepending the referring expression with padding tokens. By this, the positions of the slots are preserved and if not specified just filled with a padding token. Each slot has always the same semantic value, e.g. the last slot always contains the shape of an object. This can be done since the referring expressions are not free text, but instead the structure and the possible content is given by the dataset. This method might help the model to learn the correct referring expressions. The last variation concerns the order of producing each token. When the referring expressions are prepended, the model would need first produce two padding tokens, before it finally can produce a much more meaningful token for the shape. This could be difficult to learn for a model, as the longer a sequence of tokens is, the more information about the beginning of the sequence gets lost. Even though a sequence of just three tokens may not be long enough for this factor to be a problem, we experimented to reverse the referring expression. Instead of producing for instance <*pad*> *green sphere* as correct in English, the model would now need to produce *sphere green* <*pad*>. Notice that the padding token is again at the end of the generation, but the order of slots as well as the amount of information in the referring expression are still preserved.

This task inherently involves learning human knowledge and natural language structure. Nonetheless, this helps to understand more detailed if and how the model discriminates objects. Can the model solve the task, or do specific attributes used by humans pose challenges to the model?

In the first setup, the **bounding box RE generator**, the model receives the bounding boxes as input (see Figure 8). The first bounding box is always the target object, while the remaining bounding boxes are shuffled. As in the previous experiment, each bounding box is embedded, using the same *bounding box encoder* submodule: first it is passed through *ResNet-avg* and afterwards projected to an image embedding dimension e with a linear layer. The *RE generator* submodule uses these embedding to produce a referring expression. Hereby, all encoded bounding boxes are concatenated and again compressed to the decoder output dimension $LSTM_o$ using another linear layer. This representation of all objects serves as the initial

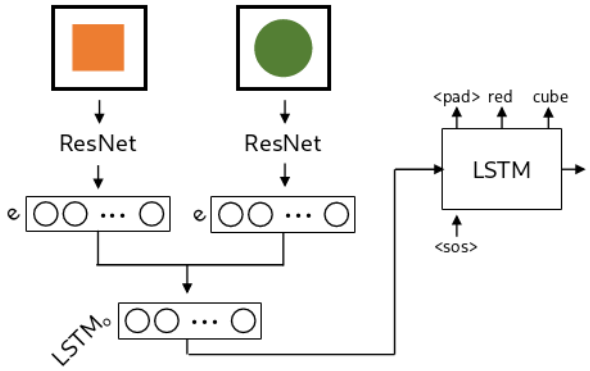


Figure 8: Simplified architecture of the bounding box RE generator

hidden state of an LSTM, which generates the referring expression. Tokens used in the LSTM are embedded with embedding dimension $LSTM_e$. During training, teacher forcing is applied by using embeddings of the ground truth tokens as the input sequence for the LSTM, instead of the output of the LSTM. The output of the LSTM is passed through a linear layer at each step to determine logits over the symbols of the vocabulary. During testing, the LSTM is always forced to generate three tokens, with an embedded start-of-sequence token as first input to the LSTM. Each token in the sequence is determined greedily, by selecting the highest logit in the output of each step in the LSTM.

In the second setup two different models are compared against each other. The first model, the **basic RE generator** acts as the baseline and only receives the complete image as the input. The image is encoded, using the *image encoder* submodule, described in Section 3.3, in particular it is passed through *ResNet-3* and subsequently processed by several additional convolution layers. The same *RE generator* submodule as for the bounding boxes is used to produce the referring expression. Hereby, since there is only one input scene, its encoding is directly reduced to the decoder output dimension $LSTM_o$. The LSTM is trained in the same way as above.

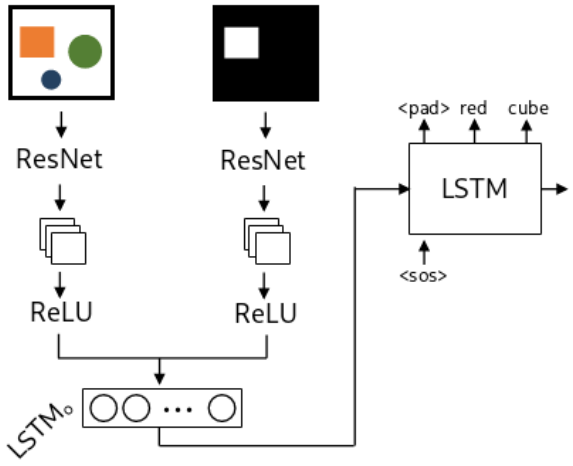


Figure 9: Simplified architecture of the masked RE generator

Using this approach, the model doesn't have any information about the target object and is therefore expected to produce referring expressions for a randomly selected object on the scene. Therefore, it is extended in a second step with attention to the target object, the **masked RE generator** (see Figure 9). A masked version of the image is created and passed to the model. For this, a fixed size squared area of 96 $\times$ 96 pixels around the center of the target object is filled in white, while the rest of the image is filled in black. Both original and masked image are processed together with the *masked image encoder* submodule described in Section

3.3. Accordingly, the same *RE generator* submodule is utilized here. This should point the model towards which object to describe and discriminate from the distractors.

For both setups, the same hyperparameters as in the previous experiments are used: a learning rate of 2×10^{-4} and 30 epochs. The loss is calculated using cross entropy. Table 5 shows which variables are compared for each model:

	e $[100, 500, 1000]$	$LSTM_o$ $[100, 500, 1000]$	$LSTM_e$ $[10, 15, 30]$
bounding box RE generator	$\times$	$\times$	$\times$
basic RE generator	-	$\times$	$\times$
masked RE generator	-	$\times$	$\times$

Table 5: Variables for each model where e is the embedding dimension, $LSTM_e$ the embedding dimensions for tokens in the LSTM and $LSTM_o$ the output dimension of the LSTM

As already discussed before, this task can be interpreted as a classification task rather than a natural language generation task. The main reason for this is that the model is tasked to assign specific attributes to the target object instead of producing free text with a large vocabulary. Following, we are interested in the classification mistakes the model makes. For this, the model's success is validated on accuracy, recall and precision scores. These are calculated in the following ways. The first measure is the **overall accuracy** if the model predicted every word in the referring expression correctly. This gives a hint, how the model fares in general and if it is able to predict any of the attributes. However, a 'false' prediction doesn't give much insight into why the model predicted a wrong referring expression.

It could be the case that the model predicted the correct shape, but wrong color. Even worse, the model could have predicted more attributes than necessary to uniquely identify the target object and didn't follow the rules of the GRE-algorithm. For instance, consider the scene in Figure 6(d). The correct referring expression is *cylinder*. If the model would predict *purple cylinder*, the accuracy determines it as false as generating the referring expression *large purple cylinder* as well *small green cube*. The first two descriptions identify the target object perfectly, but the model only didn't learn to exclude unnecessary attributes. To mitigate this, the accuracies for each class are included, as well as the macro average. This can give a better understanding of the errors the model makes. The same is done for **precision** and **recall**.

With the **non-target accuracy**, we identify if the model described another object, which is not the target image. This is basically an inverted accuracy score; the lower the score, the better the model fares. By this, it measures the false negative generated referring expressions. For this, referring generations are generated for all the non-target objects and distractors in the images using the GRE-algorithm. Importantly to notice is that the referring expressions may not uniquely identify a distractor, since multiple distractors with the same shape, color and size are allowed. If the generated description of the model describes an object that is not the target object, it gets assigned 100%. If not, independently of describing the target object, no object, or one of the objects insufficiently, it gets assigned 0%. Using this measure, we can get an overview if the model's problem lies in extracting and relating attributes or in understanding which of the presented objects is the target object.

The RE generator models are trained on both 'Dale' datasets. Again, each of these datasets increases the complexity of the description. While the referring expression for the *Dale-2* datasets are generally shorter, expressions of the *Dale-5* datasets need to be more specific and use more attributes. Additionally, the model needs to attend to many more locations in the image at the same time to find discriminating factors between those. Furthermore, the models are trained on the *CLEVR color* dataset. In this dataset, the focus lies mainly on only one discriminating attribute, the color. This might help the model in its prediction and will be tested

in the experiments.

Results

Table 6 shows the *overall accuracy*, *F1 scores* for each word and the *non-target accuracies* of the **bounding box RE generator** when trained on the *Dale-2*, *Dale-5* and *CLEVR color* datasets. As can be seen, the overall accuracies, in other words perfect matches of the generated referring expression depend very much on the dataset. With the *Dale-2* dataset, the model can achieve perfect matches in 99% of the cases in its best configuration. Also the *CLEVR color* dataset allows the model to predict the correct referring expression in 93% of the samples. Opposed to that the model can only generate perfect referring expressions in 69% of the samples of the *Dale-5* dataset. Looking at the *non-target accuracy*, one of the problems can be identified for the *CLEVR color* dataset. Summing the *non-target accuracy* and the *overall accuracy*, one gets the accuracy that any of the shown object was described independently if it was the target object. For the *CLEVR color* dataset, this score lies at 96% to 97% in the best configurations, which means that the model described a shown object for almost all the samples. In 3% to 4% of the cases, the model only picked the wrong object to describe. This looks different for the *Dale-5* dataset. Here, the model describes only in 71% of the cases one of the shown objects, 69% the target object and 2% a distractor. The model therefore struggles more with the correct generation of words than with choosing, which object to describe.

When looking at the different configurations, in especially different values for e, $LSTM_o$ and $LSTM_e$, it can be seen that the results are not very different for the *Dale-2* and *CLEVR color* datasets. However, bigger effects can be identified for the *Dale-5* dataset. Here, a low output dimension of the LSTM $LSTM_o$ tends to give lower scores. This especially enhanced, when also the embedding dimensions of the tokens $LSTM_e$ is low. Again, the image embedding size e doesn't seem to have a big effect, when over 50 dimensions. Indeed, the model achieves an accuracy of 69% with all three tested embedding sizes.

			Dale-2			**Dale-5**			**CLEVR color**		
e	$LSTM_o$	$LSTM_e$	**Acc.**	**F1**	**NT**	**Acc.**	**F1**	**NT**	**Acc.**	**F1**	**NT**
100	100	10	97	97,67	1	65	88,17	2	92	94,95	3
100	100	15	97	97,77	0	62	86,51	2	92	94,45	4
100	100	30	98	97,81	1	65	87,97	2	92	94,47	4
100	500	10	98	98,16	0	67	88,13	2	93	95,18	4
100	500	30	**99**	**98,57**	**0**	**69**	**89,53**	**2**	**93**	**95,17**	**3**
100	1000	10	98	98,28	0	68	88,23	2	**93**	**95,41**	**3**
500	1000	30	**99**	**98,85**	**0**	**69**	**89,15**	**2**	93	95,05	4
500	500	10	**99**	**98,71**	**0**	67	88,99	2	**93**	**95,4**	**3**
1000	100	10	97	97,72	0	59	85,43	2	93	95,04	4
1000	100	30	98	98,16	0	61	86,77	2	92	94,68	4
1000	1000	10	**99**	**98,95**	**0**	**69**	**89,21**	**2**	93	95,15	4
1000	1000	15	99	98,67	0	68	88,98	2	93	95,01	4

Table 6: Overall accuracies (Acc.), F1-Score (F1) and non-target accuracies (NT) in % of the bounding box RE generator after 30 epochs: e are different embedding sizes, $LSTM_o$ are different LSTM output sizes and $LSTM_e$ are different embedding sizes for the tokens in the LSTM.

Tables 7 and 8 give a more detailed insight in the results and especially what mistakes the model is making for both the *Dale-5* and *CLEVR color* datasets. They list *precision* and *recall* metrics for each token for the best configuration of the model with $e = 100$, $LSTM_o = 500$ and $LSTM_e = 30$. The tokens are grouped by attribute and also show the metrics averaged over each of the attributes. Since the overall results for the *Dale-2* dataset are already close to perfect, the focus for this analysis lies on the remaining two datasets. The metrics of the <*pad*> token indicate if the model produced the correct length of the referring expression,

in other words if it was able to determine which attributes are necessary to discriminate the target object from the distractors. For the *CLEVR color* dataset, the scores are perfect. This is not surprising, because all referring expressions for the *CLEVR color* dataset consist of exactly two attributes, shape and color, and the first generated token will always be the only <*pad*> token in the referring expression (corresponding to the unspecified size). The <*pad*> token is therefore easy to learn. For the *Dale-5* dataset, the model struggles more to predict the correct length of the referring expression.

When looking at the tokens for the shape, it can be seen that the model is able to identify it very well across all datasets. The model predicts the correct shape for all samples using the *CLEVR color* dataset, while both *precision* and *recall* lie around 98,3% when using the *Dale-5* dataset. Even though the score is almost perfect, the slight difference might stem from the fact that all distractors have the same shape in the first case, while distractors can be different in the second case. Consequently, the model is only exposed to one shape at a time for each sample, which might simplify its identification.

For the color attribute, the metrics drop significantly for both *Dale-5* and *CLEVR color* to an average of around 93%. Hereby, no meaningful difference can be seen across the datasets, but there are differences between the colors. Some colors are predicted with *precision* and *recall* around 95% to 96%, while others are only around 90%. However, these differences are not reproducible across multiple runs and configurations. The best and worst predicted colors vary and no conclusions can be drawn which colors are easier to predict for the model.

Finally, the size is the most difficult attribute to predict for the model. Apart from the *CLEVR color* dataset, where a size never needs to be predicted and also is never predicted, the metrics for the prediction of size tokens are the lowest across all tokens. They are the only mistakes, the model makes, when exposed to the *Dale-2* dataset and the average *precision* lies around 23% below the average of predictions of the color for the *Dale-5* dataset, while the average *recall* lies around 28,82% below. The reason why the *precision* is higher than the *recall* is the <*pad*> token, which is predicted very often instead of a token specifying the size. In fact, the opposite relationship is visible for the *precision* and *recall* for said token. The much higher absolute number of <*pad*> tokens leads to a smaller relative difference of %-points shown in the table. Again, no conclusion can be drawn if larger or smaller objects are easier to predict, since the results vary across runs and configurations.

		small	large	**size**	cube	cylinder	sphere	**shape**	<pad>
Dale-2	Precision	99,17	98,29	**98,73**	99,86	99,71	99,67	**99,75**	99,64
	Recall	97,54	94,26	**95,9**	100	99,56	99,67	**99,74**	99,77
Dale-5	Precision	69,65	69,21	**69,43**	98,19	98,32	98,39	**98,3**	82,22
	Recall	62,11	66,15	**64,13**	98,79	97,87	98,25	**98,3**	84,59
CLEVR color	Precision	-	-	**-**	100	100	100	**100**	100
	Recall	-	-	**-**	100	100	100	**100**	100

Table 7: Precision and Recall in % for <pad>, size and shape tokens with $e = 100$, $LSTM_o = 500$ and $LSTM_e = 30$. The columns **shape** and **size** show the average across all tokens of the respective attribute.

The approach, how the padding is produced and in which order the attributes are concatenated didn't have an effect on the described metrics. When the order was reversed and the padding appended, the model was converging slightly faster and reached the limit around two to three epochs earlier. The final peak stayed exactly the same and the effects were therefore not studied deeper.

In conclusion, it can be said that the model is able to extract discriminative features from the shown bounding boxes and produce referring expressions. However, the result highly depends on the amount of distractors

		blue	brown	cyan	gray	green	purple	red	yellow	**color**
Dale-2	Precision	94,51	98,77	97,59	98,68	98,89	98,8	97,47	100	**98,09**
	Recall	97,73	100	98,78	97,4	96,74	98,8	100	98,8	**98,53**
Dale-5	Precision	92,12	93,82	89,13	89,12	92,63	91,12	97,24	94,36	**92,44**
	Recall	92,12	89,78	94,91	94,51	95,71	92,42	89,34	94,85	**92,95**
CLEVR color	Precision	93,46	92,37	94,47	93,86	92,04	91,13	90,07	94,7	**92,76**
	Recall	92,75	92	95,98	89,92	94,12	91,13	94,23	91,91	**92,76**

Table 8: Precision and Recall in % for color tokens with $e = 100$, $LSTM_o = 500$ and $LSTM_e = 30$. The column **color** shows the average across all colors.

and the resulting need of more discriminative features to describe the target object. While the shape is easily identified, the model has bigger problems to identify the color and especially the size of the target object.

In the following paragraphs, the results of the **basic RE generator** as well as the **masked RE generator** are evaluated. Compared to the *bounding box RE generator*, the task differs substantially. Instead of bounding boxes, the model is shown the image of the whole scene. In addition to extracting features of the target object and distractors separately, the model is now tasked to extract these at once for all objects. Furthermore, it needs to learn which of the objects is the target object by combining the image with the masked image. The *basic RE generator* acts hereby as a baseline. Since there is no information about the target object present, the model is expected to extract the features of all objects, but not to be able to focus on the target object. Masking should help the model, to also solve the second task.

		Dale-2			**Dale-5**			**CLEVR colour**		
$LSTM\_o$	$LSTM\_e$	**Acc.**	**F1**	**NT**	**Acc.**	**F1**	**NT**	**Acc.**	**F1**	**NT**
100	30	38	41,16	40	14	48,91	34	15	36,98	34
1500	15	45	62,66	45	17	51,89	38	13	35,85	33
2000	10	45	60,72	44	16	50,77	37	13	34,96	36

Table 9: Overall accuracies (Acc.), F1-Score (F1) and non-target accuracies (NT) in % of the basic RE generator after 30 epochs for selected configurations: $LSTM_o$ are different LSTM output sizes and $LSTM_e$ are different embedding sizes for the tokens in the LSTM.

Table 9 shows some selected results for the *basic RE generator* after 30 epochs. The sum of the *overall accuracy* and *non-target accuracy* represents in how many cases any of the shown objects were described, independently if it is a distractor or the target object. As can be seen, the sum is always relatively high independently of the dataset. This shows verifies that the model is able to extract information about the objects in the image. However, it can't decide which object to describe. Most clearly, this is shown with the *Dale-2* dataset where the sum lies around 90%. In almost all samples, the model describes an object perfectly, but in 45% of the cases it is the target object, in 45% the distractor. Which object, the model chooses is pure chance. The same pattern is visible for the other datasets. Since there are more distractors in the scenes, the model is more likely to describe on of those, when deciding randomly, and the *non-target accuracy* is higher than the *overall accuracy*. However, another fact plays a role here. The incremental GRE-algorithm can only generate unique referring expressions, when the referred to object is unique in the scene. This applies to all target objects, but there is no such restriction on the distractors. One scene can include for instance multiple *large red spheres* as distractors. The algorithm then generates two identical referring expressions which can affect the *non-target accuracy*. Indeed, while the average number of distractors per scene in the *CLEVR color* dataset is 7,5, the average number of unique distractors lies at 2,7. As seen later, the model learns to correctly never predict a *size* and always predict the correct *shape*. Given this

fact, a random guess of a color would predict in $\frac{1}{8} * 2,7 = 33,9\%$ of the cases a unique distractor while in $\frac{1}{8} = 12,5\%$ of the cases it would predict the target object. This corresponds the *non-target accuracies* and *overall accuracies* the *CLEVR color* dataset. The high *non-target accuracy* is therefore misleading and doesn't represent that the model identified other objects in the scene.

		Dale-2			**Dale-5**			**CLEVR color**		
$LSTM_o$	$LSTM_e$	**Acc.**	**F1**	**NT**	**Acc.**	**F1**	**NT**	**Acc.**	**F1**	**NT**
100	10	81	76,16	9	21	57,07	23	16	43,69	31
100	15	78	72,06	11	17	53,29	25	26	50,87	30
100	30	83	78,76	7	19	54,91	24	**29**	**53,37**	**26**
500	10	83	79,78	6	27	62,05	16	17	44,93	32
500	15	83	79,75	7	28	63,78	16	21	47,08	31
500	30	84	83,03	5	30	64,36	17	21	40	31
1000	10	84	82,2	5	31	65,58	13	12	37,67	34
1000	15	85	82,62	5	31	64,28	16	16	43,27	32
1000	30	85	83,13	4	31	64,06	17	13	41,79	34
1500	10	84	81,27	4	33	67,5	14	12	39,04	34
1500	15	**85**	**84,11**	**4**	35	69,05	12	17	44,98	31
1500	30	84	80,93	4	36	68,97	15	14	42,99	33
2000	10	83	82,69	4	**36**	**69,11**	**13**	13	41,85	34
2000	15	84	81,95	4	34	66,94	14	14	38,79	33
2000	30	85	82,8	4	32	65,52	16	18	45,41	32
3000	10	82	79,83	4	34	67,61	12	12	38	34
3000	15	85	81,8	4	32	65,64	15	13	38,52	35
3000	30	83	80,6	3	30	63,5	16	12	38,45	34

Table 10: Overall accuracies (Acc.), F1-Score (F1) and non-target accuracies (NT) in % of the masked RE generator after 30 epochs: $LSTM_o$ are different LSTM output sizes and $LSTM_e$ are different embedding sizes for the tokens in the LSTM.

Table 10 shows the results for the *masked RE generator* after 30 epochs. A first look at the sum of the *overall accuracy* and the *non-target accuracy* shows the model is able to extract the attributes from the shown objects. For the *Dale-2* dataset, this number lies at 90% for the best configuration, at 51% for the *Dale-5* dataset and at 55% for the *CLEVR color* dataset. Those sums correspond more or less to the sums of the model without masking shown in Table 9. In other words, the masking doesn't hinder the model to extract and describe attributes in the image. However, the balance in between the two accuracy scores shifts towards the *overall accuracy*, meaning that the model is now able to identify the correct target object more often. While the model still achieves 85% *overall accuracy* in its best configuration, for the *Dale-2* dataset, the model only reaches 36% for the *Dale-5* dataset and 29% for the *CLEVR color* dataset. The scores for all datasets are all well above the *basic RE generator* baselines of 45%, 16% and 15% respectively, while in all cases the *non-target accuracies* drop lower. In other words, the masked image helps the model to focus on the correct object.

Comparing it to the less complex *bounding box RE generator*, the results are much less accurate. This applies to all datasets, even though it is most apparent for the *CLEVR color* dataset. Given the highly increased complexity of the task, the model fares well in comparison on the *Dale* datasets.

Moreover, the referring expression generated for the *Dale-2* dataset are also efficient in the sense that they follow the incremental GRE-algorithm and only use necessary discriminative attributes. Features of both target object and distractor are therefore extracted and associated with the vocabulary. Striking in these results is also the *non-target accuracy*, which is much higher than for the *bounding box RE generator*. For

the *Dale-2* dataset, it arrives at 11% in one configuration, for the *Dale-5* dataset, the model describes for at least 12% of the samples a distractor, for one configuration even in over 25% of the cases. The *non-target accuracy* of the *CLEVR color* is misleading, and is therefore not considered. This shows that the model still struggles to learn which of the objects in the scene is the target object, even though masking is applied.

Looking at the effects of the LSTM output size $LSTM_o$ and the LSTM embedding size $LSTM_e$, a difference between the *Dale* datasets and the *CLEVR color* dataset can be identified. A higher $LSTM_o$ tends to increase the performance of the model, when presented with the *Dale* datasets until a size of 1500 to 2000 dimensions. When it is further increased the model starts to perform worse. Furthermore, with a low $LSTM_o$, the model is more likely to identify and describe a distractor, while a higher $LSTM_o$ helps the model to focus on the target object. The $LSTM_e$ however doesn't seem to have a big consistent effect on the results. With an $LSTM_o$ of around 1500 to 2000, the model seems to fare slightly better with higher $LSTM_e$ of 15 or 30 dimensions. For the *CLEVR color* dataset, the results look different. Opposed to the former datasets, a lower $LSTM_o$ helps the model to describe the correct target object. On the other side, the *non-target accuracy* stays relatively constant independently of the variables at the same level as for the *basic RE generator*. This indicates that the model uses random guesses of the color when it is not able to extract the color of the target object. To confirm this, the model's performance on different tokens and attributes are analyzed.

		small	large	**size**	cube	cylinder	sphere	**shape**	<pad>
Dale-2	Precision	68,63	76	**72,31**	98,41	98,18	98,92	**98,5**	94,77
	Recall	30,17	34,55	**32,36**	98,7	98,33	98,47	**98,5**	98,64
Dale-5	Precision	49,47	57,3	**53,38**	87,09	83,26	84,78	**85,04**	64,99
	Recall	24,67	26,7	**25,69**	83,66	85,28	86,03	**84,99**	77,87
CLEVR color	Precision	-	-	**-**	100	100	99,7	**99,9**	100
	Recall	-	-	**-**	99,85	99,85	100	**99,9**	100

Table 11: Precision and Recall in % of <pad>, size and shape tokens in the best configuration for each dataset (marked in bold in Table 10). The columns **shape** and **size** show the average across all tokens of the respective attribute.

		blue	brown	cyan	gray	green	purple	red	yellow	**color**
Dale-2	Precision	88,89	85,9	81,82	85,53	79,76	84,52	87,76	87,36	**85,19**
	Recall	89,89	83,75	87,8	79,27	93,06	76,34	90,53	81,72	**85,3**
Dale-5	Precision	69,51	72,73	73,48	66,27	63,23	64,32	67,98	70,76	**68,53**
	Recall	72,43	71,29	73,48	60,77	74,6	74,87	71,13	73,57	**71,52**
CLEVR color	Precision	37,63	39,23	22,09	20,23	33,33	22,97	24,9	43,96	**30,54**
	Recall	29,44	42,68	27,48	22,22	25,44	26,53	26,23	37,14	**29,64**

Table 12: Precision and Recall in % of color tokens in the best configuration for each dataset (marked in bold in Table 10). The column **color** shows the average across all colors.

Tables 11 and 12 shows the precision and recall for each token. The metrics on the *<pad>* token show that the model always predicts the correct length of the referring expression when presented with the *CLEVR color* dataset. For the *Dale-2* dataset, the model is able to do this in almost all the cases, while the model struggles the most with the *Dale-5* dataset. While this was similar to the *bounding box RE generator*, the effect here is amplified. Furthermore, the easiest attribute to predict is the shape. Again, the metrics for the *Dale-2* and *CLEVR color* dataset are almost the same, but the model is much less able to predict the correct shape for images from the *Dale-5* dataset with only around 85%. Hereby, the precision is the highest for

predicting *cubes* and lowest for predicting *cylinders*, but this varies across runs.

The prediction of the color drops very much in comparison. While the metrics all lied above 90% for the *bounding box RE generator*, now none of datasets passes this limit. With the *Dale-2* dataset, the precision and recall lie around 85%, with the *Dale-5* dataset around 70% and around 30% with the *CLEVR color* dataset. The drop for the 'Dale' datasets is expected, because of the higher complexity of the task, but the model's performance on the *CLEVR color* dataset is much worse. Only every third prediction is correct. Hereby, a difference across the colors is visible. Some colors reach both precision and recall of around 40%, while others only of only around 20%. Again, which color is captured better differs from run to run. Interesting is also the margin between the precision and recall for the colors 'blue', 'cyan', 'green', 'purple' and 'yellow'. A low precision and high recall indicates that the model predicts the specific color more often than needed while the opposite indicates that it predicts the color less often than needed. In particular, 'blue', 'green' and 'yellow' are predicted too often and 'cyan' and 'purple' are predicted too few.

Finally, the size is again the most difficult attribute to predict. As the *bounding box RE generator*, the *masked RE generator* learns correctly to never predict a size for the *CLEVR color* dataset. Interestingly is the fact that the recall is low which shows that the model likely predicts a <*pad*> token instead. This might be due to the fact that the <*pad*> token is very frequent in the size slot across all referring expressions and the model amplified this fact. Between both sizes, 'large' objects are easier to predict for the model than 'small' objects. The difference lies for both *Dale* datasets at around 8%. However, the precision is still far above a random guess, when used on the *Dale-2* dataset. Using the *Dale-5* dataset, the precision only lies at around 50% to 57%.

In conclusion, when the model doesn't need to extract the objects from the scene it performs very well. In this case, it is able to achieve almost perfect results on the *Dale-2* dataset and very good results on the remaining two datasets. However, using complete scenes impairs the performance highly. The *overall accuracies* decrease and the *non-target accuracies* increase; the model has more difficulties to focus on the correct object. The most difficulties the model has with the *CLEVR color* dataset where only the color is a discriminative attribute. Looking at the attributes, the last slot (shape) is always the easiest to predict, while the first slot (size for the *Dale* dataset and color for the *CLEVR color* dataset) is the most difficult. This effect is again amplified when the model needs to extract the objects itself instead of being presented with bounding boxes.

There are two main explanations for these results. First, as already seen in the previous experiments, a bigger number of distractors confuses the model more where to focus on. In the *Dale-2* dataset, there are only two possibilities, while the four distractors in the *Dale-5* dataset and up to nine distractors in the *CLEVR color* give the model a bigger choice. Moreover, having only one attribute to discriminate the target object, poses the model with more difficulties. The second explanation lies in the incremental GRE-algorithm. For instance using the *Dale-2* dataset, when only two random objects are placed in a scene, a second (or third) attribute to discriminate the objects is only needed, when the shape is the same. Otherwise, the shape is enough, and the RE is only one word long. The probability for this lies at 66,6%. For shape and color being enough, the probability lies at 29,2% and that all three attributes are necessary is 4,1%. Opposed to that the probabilities with four distractors as in the *Dale-5* dataset are 19,8%, 47% and 33,2% respectively. With four distractors the algorithm is much more likely to produce longer referring expressions. These are harder to learn and generate for the model since it needs to take more extracted attributes into account to discriminate the target object from the distractors. This is verified by the lower precisions and recalls of the size and color tokens. Furthermore, the low recall of the size tokens shows that the model produces too short referring expressions too often.

5.3 Reference resolution

Setup

With the reference resolution task, the model is trained to understand referring expressions, by pointing towards the target object in the scene. Compared to the object identification task in Section 5.1, this task is more complex since it also includes identifying locations instead of choosing from given objects. This level should help to analyze, how the final task of the language game should look like, in especially what the receiver is tasked to predict. As described before, the sender should communicate an object in the image and the receiver needs to identify it. The challenge lies in how the receiver refers to the identified object. There are multiple possibilities, how it can be done. One of them could be to describe the target object with human language, using the attributes. The main goal however is to let the language of the agents emerge as natural as possible. Including human referring expressions into the task would bias also the emerged language towards attributes and words, used in natural language. For this reason, the final task of the receiver will be to 'point' to the target object. In this section, this is done in two ways. First, the models are tasked to predict the center coordinates of the target object (*coordinate reference resolver*). This requires a very high precision of the model in extracting spatial information. However, the exact location is not necessarily important to highlight, which object the model predicts to be the target object, and might challenge the model too much. Therefore, in a second approach, the model is tasked to predict a region around the target object (*attention reference resolver*). This reduces the necessary amount of spatial information and might be easier to learn for the models. With both approaches, the models receive few natural language information, but are still able to rely on all information present in the image to discriminate the objects.

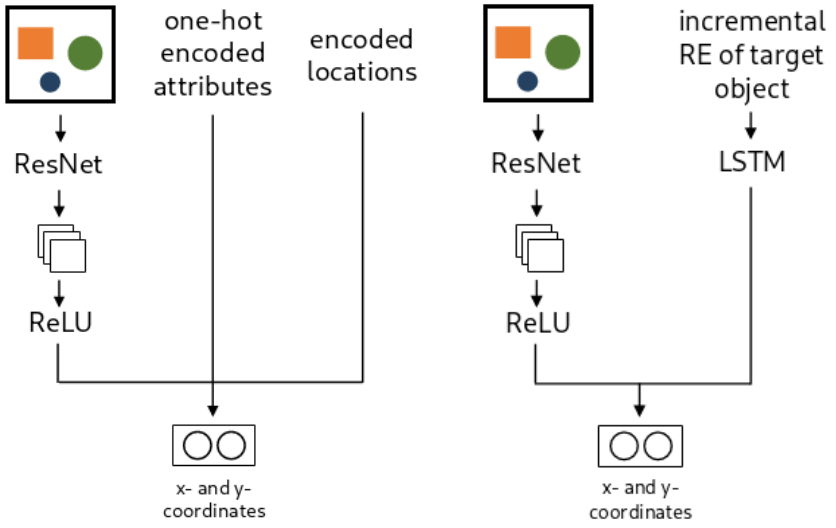
In the simplest setup, the **coordinate reference resolver**, the model receives only the image as an input and produces two numbers as an output, the predicted x- and y-coordinate of the target object. Again, the image is encoded, using the *image encoder* submodule, described in Section 3.3. The *coordinate predictor* submodule takes the encoder image to predict x- and y-coordinates of the target image, Hereby, the encoded image is flattened and passed through two linear layers with a *ReLU* non-linearity in between. These reduce the dimensions first to the coordinate predictor dimension c and finally to 2.

To determine the loss, the euclidean distance between the resulting predicted point on the image and the ground truth point are calculated. This distance is learned to be minimized. By doing that, the model learns to focus and attend on a specific part in the image, in a perfect model the center of the target objects.

With this simple setup, the model is theoretically able to very precisely focus on an object in the image. The problem arises as soon as multiple objects are present in the image. There is no information available for the model to understand which one of these objects is the actual target object, except for the final calculation of the loss. Since there is not necessarily a pattern for which object in the image is the correct target object over the whole dataset, the models will likely fail to generalize. Therefore, the models need to receive more information. Here, four different ways to encode and refer to the target object are tested.

In the first method, the target objects' attributes are encoded as **one-hot encodings**, as described in Section 5.1. In an extension to this method, also the **center coordinates of all objects** in the image are included. The center coordinates of all objects are simply extracted. Since there is no direct way of ordering objects, distributed in 2 dimensions into a list, these are shuffled, to just provide the model with all possible targets to choose from. Since there are varying numbers of objects in the image, this vector of variable length is padded to the maximum number of objects in the dataset. The padded locations consist of two zeros for both coordinates. In both methods, the encodings of the target object are concatenated with the encoded image and then passed to the coordinate predictor.

The third method encodes the attributes of the target object with human language using the **incremental**



(a) Model including one-hot encoded attributes and locations

(b) Model including incremental referring expression

Figure 10: Simplified architectures of the *coordinate reference resolvers*

GRE-algorithm. This opposes the idea described before, to share as few natural language information as possible with the model. Still, this approach can help to understand and analyze if the model was able to extract information about the objects and more specifically their attributes from the image. If the model is able to match parts of the image with human words it would show that the model learned this attribute. If the model in a next step can learn this for the whole dataset, this would mean that it could generalize over these attributes and assign them to certain regions in an image. This insight would help for succeeding models that make use of these learnings without human language.

For the experiments, each image is captioned with a referring expression of the target object using the described algorithm. To include it in the model, the referring expressions need to be padded to an equal length. In this case they are padded to a length of 3, which is the maximum number of attributes that can be used. For this, as standard practice in captioning tasks, the referring expressions are padded at the end with a specified padding token.

In the model, the referring expression is encoded, using the *GRE parser* submodule. Here, the learned embeddings of each token are parsed by the LSTM and its final hidden state is then used as a summary of the complete referring expression. Tokens are embedded with embedding dimensions $LSTM_e$ and the output of the LSTM has the size if $LSTM_o$. The vocabulary that is used for the descriptions is based on 14 symbols, including the padding token. Both the processed image and the final hidden state of the LSTM are flattened and concatenated, which is then passed through the coordinate predictor.

The fourth method, to encode the target object utilizes **masking** of the image. The image is masked in the same way as in Section 5.2. While even the one-hot encodings contain human language knowledge by translating human referring descriptions into a vector, the masking method will only point the model towards the target object without giving more information. It therefore can only rely on its own extracted visual features and the inherent human bias in the image when looking at masked images. Both original and masked image are processed as described in 3.3 and afterwards passed through the coordinate predictor.

For all setups, the same hyperparameters as in the previous experiments are used: a learning rate of 1×10^{-4} and 30 epochs. The loss is calculated using cross entropy. Table 13 shows the variables that are changed during the experiments for each of the models.

	e_i $[50, 100, 300, 500]$	c $[512, 1024, 2048]$	$LSTM_e$ $[15, 30]$	$LSTM_o$ $[1000, 1500]$
coordinate reference resolver	×	×	-	-
+ one-hot	×	×	-	-
+ one-hot + locations	×	×	-	-
+ incremental RE	×	×	×	×
+ masking	×	×	-	-

Table 13: Variables for each model where e_i is the image embedding dimension, c the dimensions of the coordinate predictor, $LSTM_e$ the embedding dimensions for tokens in the LSTM and $LSTM_o$ the output dimension of the LSTM

The test dataset is again evaluated on the euclidean distance of the predicted coordinates to the ground truth coordinates. This distance needs to be minimized. The mean of all calculated distances is calculated across the whole epoch, which results in a mean distance score per epoch. Since this score only takes the average of all predictions into account it doesn't show how every prediction fared individually. If for instance the prediction of one object is getting more precise with growing number of epochs, but the precision of another object gets worse, the mean distance will stay the same. It doesn't reflect this change. For that reason, we also introduced an accuracy score. For that we defined a fixed size circle with a radius of 20 pixels around the center of each object. If the model's prediction lies in this circle, it will be counted as a correct prediction, if it lies outside, it is a false prediction. These scores are averaged for the epoch and result in an accuracy score, where 100% means that all predictions were very close to the center coordinates and 0% means that no predictions were close to the center coordinates. This of course doesn't give a perfect representation since the size of the objects varies, but it will still show, how precise each individual prediction is. A high accuracy may indicate that the model could identify this specific object better.

In the second approach, the **attention reference resolver** predicts a region around the center of the target object. For this, the image is divided into 14×14 regions. The target area is located around the center of the target object, consisting of 3×3 regions. Hereby, the center of the target object is located in the central region. The model is then tasked to predict the matrix $A = (a_{ij})$, where:

$$a_{ij} = \begin{cases} 1, & \text{if region } i, j \text{ in target area} \\ 0, & \text{otherwise} \end{cases}$$

To do this, the model combines a representation of the image and a representation of the target object. More specifically, the image is encoded using the *image encoder* submodule, omitting the last *max pool* layer. By this, the resulting matrix has 128×14×14 dimension, corresponding to the 14regions. For the target object, the incremental RE is passed through an LSTM as described before. Both encodings are passed through two linear layers separately to project them to the projection dimension p, followed by the $tanh$ function. Following, the dot product is calculated between the projected image and the projected attributes which results in a 14×14 matrix. This emphasizes the correlation between the attributes and each region. A high dot product for a region indicates a high correlation between the attributes and the specific region, while a low dot product indicates the opposite. Training a model like this should therefore highlight the regions in the image that are described by the referring expression, namely the region around the target object. To calculate the loss, the $softmax$ function is applied over the prediction and compared to the ground truth matrix A using the *binary cross entropy*. Hereby, the models converged faster and were trained for 20 epochs with a learning rate of 2×10^{-4}. An $LSTM_e = 15$ and $LSTM_o = 1500$ showed the best results. Projection sizes $p = \{50, 100, 300, 500, 1000\}$ are compared.

The *attention reference resolver* is evaluated on the summed predicted probability for the regions in the target area. In particular, the predicted matrix, consisting of probabilities for each region is multiplied with the ground truth matrix A, consisting of only ones and zeros. The result is summed and returns the probability mass for the target area. If the model predicts the target area perfectly, the probabilities in the target area sum to 1. If the model for instance focuses on the wrong object, the probability mass in the target area is lower.

The reference resolution models are trained on the 'CLEVR single' as well as on both 'Dale' datasets. The 'CLEVR single' dataset should test the model if it can actually learn locations of an object. Since the model relies on the extracted features of ResNet, locational information of the objects in the image might not be present anymore. This is due the fact that CNNs compress information of the image summarizing small regions (convolutions) several times. While this extracts the key features of objects in an image and might capture relational position between these objects, absolute locations might get lost as shown by Kelleher & Dobnik (2017). Training on this dataset should make sure that the model can converge towards the correct pixels, utilizing these features. In a next step, the 'Dale' datasets provide the actual problem of discriminating objects from each other and afterwards pointing to the correct one. Here, the models should make use of the additional given referring expressions about the scene, as one-hot encodings of the attributes, descriptions using the incremental GRE-algorithm or the encoded locations. 'Dale-2' and 'Dale-5' provide two different difficulties for the model, where it needs to discriminate a target object from one or four distractors and then point towards it. Latter task is assumed to be significantly harder.

Results

Tables 14 to 18 show the results of the five adaptions of the reference resolver. Across the first three models (basic reference resolver, +one-hot, +one-hot+locations), the same patterns can be seen.

The **basic reference resolver** acts as a baseline for the other models. Since this model doesn't have any information about the target object, it shouldn't be able to point towards the correct target object. However, this model should learn underlying structures in the images that it can use to make better predictions. Indeed, the model converges on all datasets to a certain limit. Random guesses of the coordinates correspond to a mean distance of 95 pixels, given the image size of 224×224 pixels, which the model can beat for all datasets.

		CLEVR single		**Dale-2**		**Dale-5**		**CLEVR color**	
e_i	c	**loss**	**Acc.**	**loss**	**Acc.**	**loss**	**Acc.**	**loss**	**Acc.**
100	512	2,45	96%	38,16	35%	46,44	18%	51,8	12%
100	1024	2,18	94%	37,23	38%	47,38	18%	52,04	13%
100	2048	3,44	99%	36,8	40%	46,77	18%	51,29	13%
500	512	2,71	96%	37,1	37%	47,28	18%	51,01	14%
500	1024	2,61	97%	36,2	39%	48,6	17%	52,73	12%
500	2048	5,18	99%	36,09	40%	44,81	18%	51,8	13%

Table 14: Mean loss and accuracy of the basic reference resolver for different configurations of the model after 40 epochs: e_i is the image embedding dimension and c the coordinate predictor dimension

Table 14 shows the results. When trained on the 'CLEVR single' dataset, the model can minimize the mean loss consistently across all variations of the variables to a few pixels. In effect, this means, that the model is able to almost always predict the correct location of the center of the target object. Since no distractor is present, the model also doesn't need any description of the target object and only needs to discriminate the object from the background. This shows that the model is in general able to derive geometrical information from abstract feature, extracted by a feature extractor. Geometrical information

therefore doesn't get completely lost during this abstraction, but the model is able to point to a specific object, as long as only one object is part of the image.

However, the model has more difficulties when distractors are present in the image. For the 'Dale-2' dataset, the mean loss converges towards 35 to 39 pixels The model was able to extract some structural information from the image, but the predictions are not consistently on the target object. Looking at the accuracy, the best variations of the models are only able to point in 41% to 42% of the samples towards the correct object, which lies just below a random guess of 50%. Using the 'Dale-5' and the 'CLEVR color' datasets, the results are similar. While the mean loss stays consistently around 46 to 48 pixels and 50 to 53 pixels respectively, they are still far below the random baseline. Also the accuracies, lie a bit below the random guesses of 20% and 15% respectively. However, it is visible that more present distractors in the scene challenge the model more.

Looking at the image embedding size e_i and the coordinate predictor dimension c, it seems, they don't have any significant effect on the results. The results staying the same across all variations indicates that the increase compared to the random baseline is just due to learning an underlying structural pattern of how scenes are set up. This hypothesis will be discussed further below with examples from the predictions.

Interestingly, providing the model with the target object encoded as a **one-hot vector** and the **locations** of all objects doesn't improve the results. Tables 15 and 16 show the same results as for the *basic reference resolver*. It seems that the model is not able to combine the image representation with the representation of that target object or the locations of all objects and another representation is needed.

Encoding the attributes of the target object with the **incremental GRE-algorithm** finally improves the model's performance. Table 17 shows the results of selected configurations. Using the 'Dale-2' dataset, the model is able to predict the center coordinates of the target object for most of the images. The results, hereby, are dependent on the different variables. The best results are achieved with $e_i = 300$, $c = 2048$, $LSTM_o = 1500$ and $LSTM_e = 15$ and this configuration will be used in the succeeding analysis and the remaining language games experiments. With this configuration the mean distance of the model's predictions is around 14 pixels, and they lie in around 79,45% of the cases on the target object. This is far below the random guess of the center coordinates as well as better than a random choice between the objects. Comparing it to the other configurations, it becomes apparent that an image embedding size e_i between 100 and 300 dimensions provides the model with enough information from the image. Contrary to this, a relatively high decoder size of the LSTM $LSTM_o$ that decodes the referring expression is necessary. However, it seems to be dependent on e_i. A larger e_i requires a larger $LSTM_o$. Preliminary experiments have shown that lower sizes didn't provide any good results. Furthermore, lower embedding dimensions of the tokens $LSTM_e$ tend to give better results. An $LSTM_e$ around the vocabulary size of 14 gave the best results while for instance more dimensions with 30 always worsened the model.

		CLEVR single		**Dale-2**		**Dale-5**		**CLEVR color**	
e_i	c	**loss**	**Acc.**	**loss**	**Acc.**	**loss**	**Acc.**	**loss**	**Acc.**
100	512	2,95	98%	37,27	34%	47,05	18%	51,25	12%
100	1024	3,64	98%	37,19	37%	46,39	19%	50,25	14%
100	2048	6,15	99%	37,04	39%	45,77	18%	50,55	14%
500	512	3,29	98%	38,58	34%	47,25	18%	50,56	14%
500	1024	2,15	93%	37,18	40%	47,57	18%	51,75	14%
500	2048	2,86	98%	35,71	41%	46,62	18%	52,67	11%

Table 15: Mean loss and accuracy of the *reference resolver + one-hot* for different configurations of the model after 40 epochs: e_i is the image embedding dimension and c the coordinate predictor dimension

		CLEVR single		Dale-2		Dale-5		CLEVR color	
e_i	c	**loss**	**Acc.**	**loss**	**Acc.**	**loss**	**Acc.**	**loss**	**Acc.**
100	512	3,02	97%	36,88	37%	47,04	16%	51,36	12%
100	1024	2,48	95%	38,24	33%	46,91	15%	52,67	12%
100	2048	3,8	100%	37,89	32%	45,33	13%	49,12	11%
500	512	3,6	98%	35,56	42%	47,41	16%	51,86	12%
500	1024	2,51	98%	36,56	39%	45,88	17%	51,73	12%
500	2048	2,37	97%	37,33	35%	48,04	16%	53,68	11%

Table 16: Mean loss and accuracy of the *reference resolver* + *one-hot* + *locations* for different configurations of the model after 40 epochs: e_i is the image embedding dimension and c the coordinate predictor dimension

Using the 'Dale-5' and 'CLEVR color' datasets, providing the incremental referring expression to the model didn't help. The results are similar to the baseline with no information about the target object.

Finally, Table 18 shows the results, when the model is additionally provided with the **masked image**. Hereby, the model is able to predict correct locations using all datasets. All configurations achieve mean distances of a few pixels and accuracies of almost 100%. This shows that the model can make use of the masked image. However, this is also not surprising since the model doesn't need to use the original image, but can completely rely on the mask. Since the mask only includes the target object but no distractors, it corresponds to the 'CLEVR single' dataset which as already seen in the results before is an easy task to learn for the model.

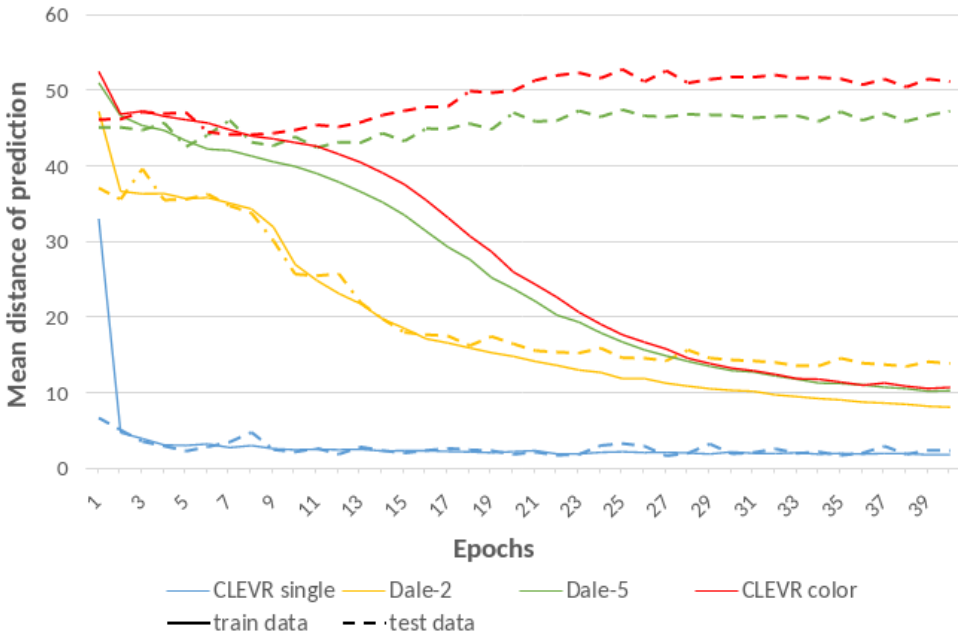


Figure 11: Mean loss of training and testing data with all datasets, using attributes encoded with the incremental GRE algorithm. The following variables are used for all models: $e_i = 300$, $c = 2048$, $LSTM_o = 1500$, $LSTM_e = 15$

Figure 11 gives more insight in why the models don't converge to a lower mean loss. It shows how the mean loss changes during the training after each epoch. Hereby, the figure shows the graph for the best performing configuration of the model that is presented with the incremental referring expression, using $e_i = 300$, $c = 2048$, $LSTM_o = 1500$ and $LSTM_e = 15$. The different colors correspond to the different datasets. Two lines are shown for each dataset. The solid line represents the loss when trained on

the training data. The dotted line shows the mean distance to the target object of the model's predictions when presented with the unseen test data. In particular, after each epoch, the model is frozen and tested on the test split. After this, the weights are unfrozen again and the training is continued. As can be seen, the training loss of the model always converges towards a low value. Unsurprisingly, the datasets with fewer distractors are faster to learn than ones with more. For both successful datasets, 'CLEVR single' and 'Dale-2', also the training loss converges towards the same value. However, the testing curves for the remaining two datasets don't match their corresponding training curve. Indeed, both curves stay at the same value until around epoch 9. Then the loss of the training starts to drop while the loss in the testing starts to rise. That indicates that until epoch 9, the model learns to general structural patterns in the images that apply to the complete dataset, both training and testing split. After this, the model begins to memorize the training samples. This knowledge is not transferrable to the test split and indeed even harms the test performance slightly. That behavior can be observed for all models and configurations that don't outperform the baseline. Preliminary experiments showed that applying dropout to help the model to generalize didn't increase the performance which shows that the architecture is not able to extract the necessary features from the images and the referring expression and combine them.

An interesting pattern appears when doing a qualitative analysis of the models' predictions. Here, the predicted coordinates compared to the ground truth coordinates are visualized (Figure 12), using the best performing model with $e_i = 300$, $c = 2048$, $LSTM_o = 1500$ and $LSTM_e = 15$. Figure 12(a) shows random examples of predictions for images in the train dataset of Dale-2. The green circle shows the ground truth center coordinates of the target object, while the red circle shows the prediction of the model. As can be seen, the predictions are very precise. Figure 13 combines the predictions and ground truths across all images in the train dataset. This shows general patterns of the models predictions over the complete dataset. Here, all predicted coordinates are placed as red circles into the image, while all ground truth coordinates are placed as green circles. For the predictions on the 'Dale-2' dataset, the resulting shape is a rhombus, which reflects that all objects are placed usually central into the scene (see Figure 13(a)). As expected the green and red rhombus align mostly in the same area for the train split of the 'Dale-2' dataset. Figures 12(c) and 13(c) show the same patterns for the 'Dale-5' dataset, while Figures 12(e) and 13(e) show it for the 'CLEVR color' dataset.

The results look very different when using the predictions for the test split. As discussed in the paragraphs above, the model performs well on the 'Dale-2' dataset which is also reflected in the qualitative analysis. The visualization of the test split resembles the train split very much. However, the other two datasets show differences. As can be seen in Figures 12(d) and 12(f), the three randomly selected predictions don't align with the ground truth coordinates. While for some images they lie on a distractor, on some images, the prediction is on the background. However, they tend to be in an area towards the target object, but are still quite imprecise. These findings align with the mean distance scores, described in the paragraphs before. However, it seems that the model's predictions are all towards the center of the image. This can be seen clearer in Figures 13(d) and 13(f). Again, the green circles form the shape of rhombus. In contrast, the predictions in red almost all cluster in the center of the image. They form roughly the shape of a smaller rhombus.

These results allow three conclusions. First, the underlying structure in all images is the placement of objects in the center of the scene. The models, learn to use the structure and are biased to predict coordinates towards the center. The reason for this is likely that the model can produce a relatively low loss, without relying on many extracted features of the objects. Since all objects are always located in the center of the image and never in its corners, a prediction of any coordinate in the center is on average closer to the target object than any random prediction or predictions of coordinates at the borders of the image. The model therefore learns only, where *any* object is likely located and can minimize the mean distance to a certain extent with this strategy.

Secondly, even though the models are biased towards the center of the image, the predictions are still often leaning towards the location where many objects lie. This can be seen for the 'CLEVR color' dataset, especially in left and central image in Figure 12(f). Again, by this strategy, the model can minimize the mean distance, since the probability is high that the target object lies in this cluster of objects.

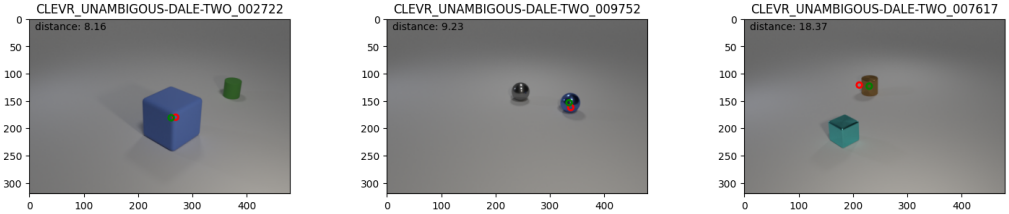
Thirdly, when few distractors are present in the scene, the model is able to combine the encoding of the referring expression with the encoding of the image. In particular, it is able to learn which of the objects in the image is described by the referring expression. As soon as more objects are present, the task gets to difficult and the model only relies on the general patterns in the dataset. Concluding, the model is able to extract, where objects are located in the image, but can't consistently make use of the referring expressions, to decide which of these objects is the target object.

				Dale-2		Dale-5		CLEVR color	
e_i	c	$LSTM_o$	$LSTM_e$	**loss**	**Acc.**	**loss**	**Acc.**	**loss**	**Acc.**
50	1024	1000	15	22,64	63,55%	48,23	14,95%	52,62	11%
50	2048	1500	15	21,35	65,9%	46,95	16,75%	52,2	11,35%
100	1024	1000	15	**16,08**	**75,5%**	47,62	16,5%	50,65	12,8%
100	1024	1000	30	20,69	69,5%	46,9	16,45%	51,23	12,2%
100	1024	1500	15	21,24	66,3%	48,78	15,55%	52,21	12,25%
100	2048	1000	15	19,46	70,1%	46,79	16,9%	50,24	13,3%
100	2048	1500	15	21,19	66,65%	46,79	16,65%	50,55	11,3%
300	1024	1000	15	20,63	68%	46,91	17,4%	50,39	11,75%
300	1024	1500	15	21,38	66,15%	46,36	15,55%	51,29	12,25%
300	2048	1000	15	21,12	66,95%	46,98	16,65%	51,4	12,85%
300	2048	1500	15	**14,01**	**79,45%**	47,41	16,9%	51,32	12,6%
300	2048	1500	30	22,09	64,75%	46,77	17,65%	49,75	13,4%
500	1024	1000	15	35,14	41,25%	46,41	16,6%	50,63	12,8%
500	2048	1500	15	36,52	41,95%	44,9	18,65%	50,02	13,4%

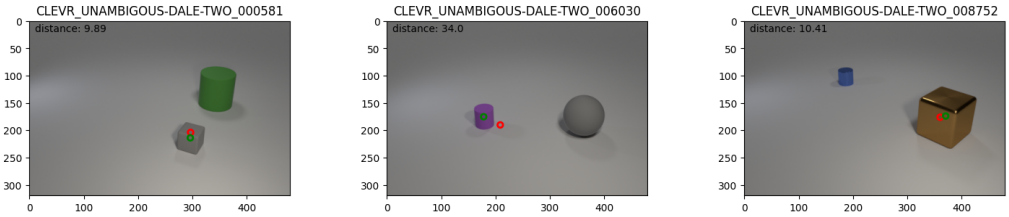
Table 17: Mean loss and accuracy of the *reference resolver + incremental RE* for different configurations of the model after 40 epochs: e_i is the image embedding dimension, c the coordinate predictor dimension, $LSTM_o$ the output dimension of the LSTM and $LSTM_e$ the embedding dimensions for tokens in the LSTM

		Dale-2		Dale-5		CLEVR color	
e_i	c	**loss**	**Acc.**	**loss**	**Acc.**	**loss**	**Acc.**
100	512	8,01	98%	6,77	100%	6,05	95%
100	1024	8,77	99%	3,61	98%	5,29	93%
100	2048	8,71	99%	4,28	96%	3,24	93%
500	512	14,27	99%	9,97	99%	5,9	94%
500	1024	8,6	99%	2,07	94%	13,03	95%
500	2048	2,67	95%	10,57	96%	4,24	98%

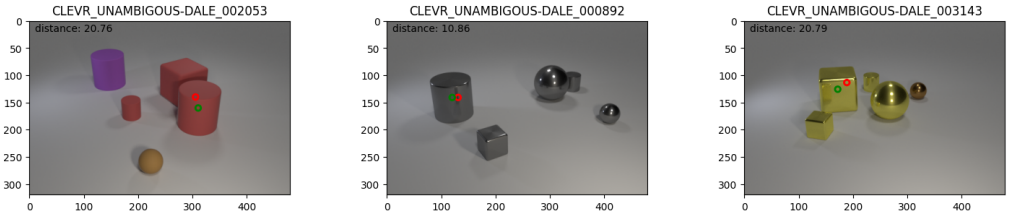
Table 18: Mean loss and accuracy of the *reference resolver + masking* for different configurations of the model after 40 epochs: e_i is the image embedding dimension and c the coordinate predictor dimension



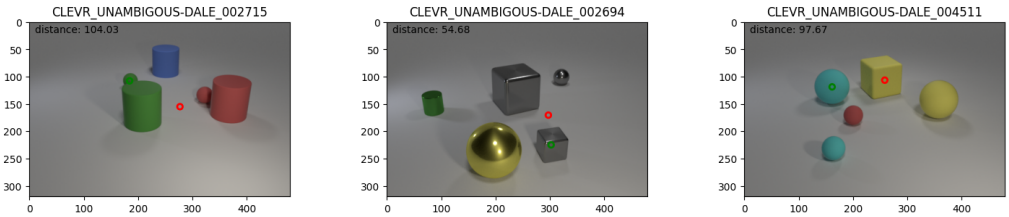
(a) 'Dale-2', train split



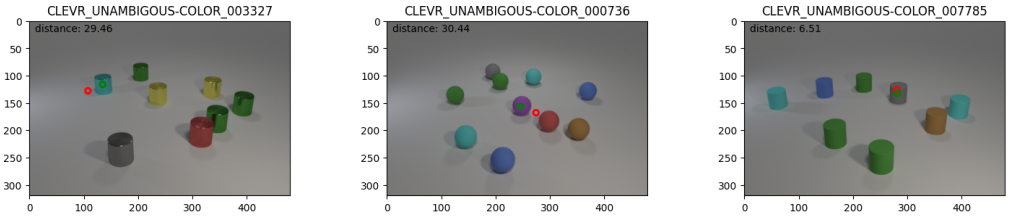
(b) 'Dale-2', test split



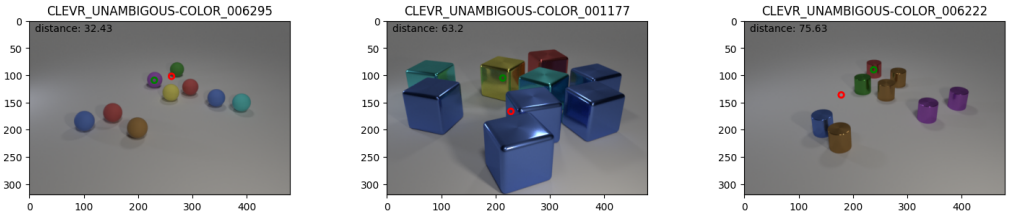
(c) 'Dale-5', train split



(d) 'Dale-5', test split

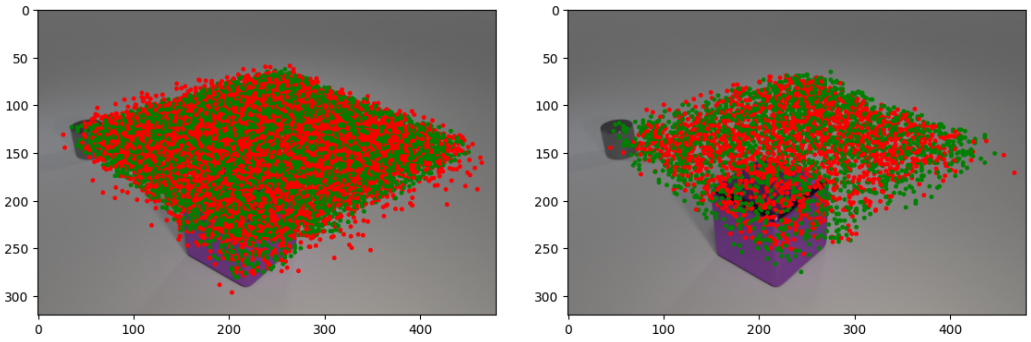


(e) 'CLEVR color', train split



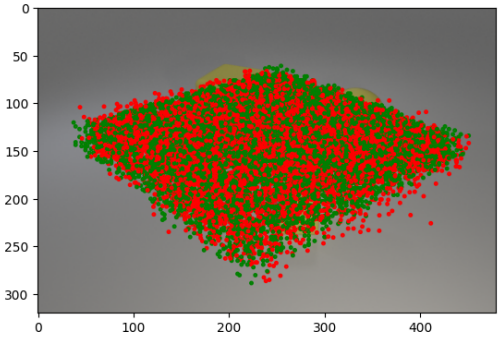
(f) 'CLEVR color', test split

Figure 12: Examples of the models' predictions on the 'Dale' and 'CLEVR color' datasets. The green circle represents the target object, the red object the model's prediction. Notice that the images are shown in the original size while the model processes them normalized. Hence, the distances don't correspond to the axes.

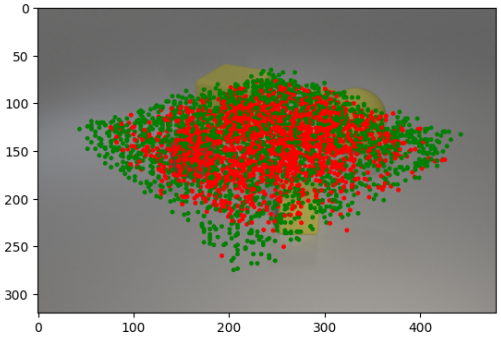


(a) 'Dale-2', train split

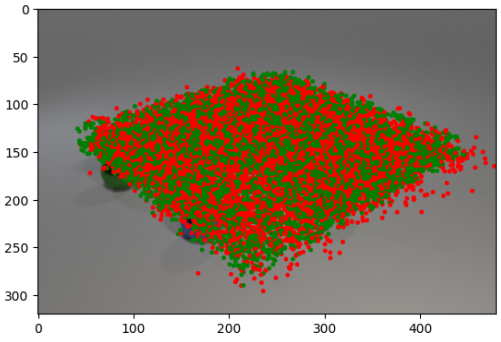
(b) 'Dale-2', test split



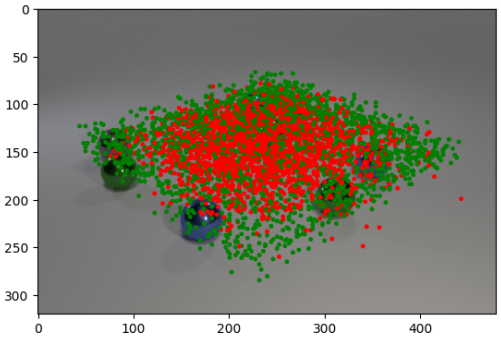
(c) 'Dale-5', train split



(d) 'Dale-5', test split



(e) 'CLEVR color', train split



(f) 'CLEVR color', test split

Figure 13: The images show all the model's predictions (red dots) and the ground truths (green dots) combined for the 'Dale' and 'CLEVR color' datasets

6 Grounding referring expressions in language games

The experiments that are executed using language games have a similar structure as the experiments in the previous chapters, since those provided the basis for the language games. However, every task is solved by two neural networks that communicate through a discrete bottleneck.

The language games in this research have an asymmetric setup. One agent, the sender is shown some information and needs to generate a message. This message is received by the second agent, the receiver. The receiver needs to parse this message and combine it with the same information that the sender was presented with. The receiver then makes a prediction, which is compared to the ground truth. The game is set up prosocial, which means that both agents receive the same loss based on the receiver's prediction. All weights of the agents are adapted in the same way.

The vocabulary that the sender can draw from to produce a message is made up of initially arbitrary symbols. The meaning of these symbols is created as soon as the sender uses them in one of the message, and the receiver is able to use it to solve the task successfully. Over time, specific meanings are reinforced and a language emerges. Hereby, the production of the messages is constrained in two ways. First, the size of the vocabulary is varied. A smaller size forces the agents to compress their information more which is a prerequisite for the emergence of a language. With more different words, they can express more information, but it is harder to learn. Different sizes are compared to analyze its effect on the emerged language. Secondly, the maximum possible length of a message is constrained. Again, a shorter message requires more compression, while the longer message allows for more information exchange.

As the experiments before, three different tasks are set up. For the *object identification* task, the agents play discrimination games similar to the setup in (Lazaridou et al., 2017). For the *RE generation* task, the receiver needs to generate a human language referring expression based on the emerged language referring expression by the sender. Since, it is possible that the agents simply learn to align these two languages, the agents are tasked to point towards the center coordinates of the target object in the final task, the *reference resolution* task. For all three setups, the sender uses the same setup and only the receiver is switched. By this, differences in the produced referring expression in the message by the sender are only dependent on the receiver's ability to solve the different tasks. In particular, the impact of the task on the emerged language can be analyzed in more detail.

As discussed in Section 2.3.1, language games are evaluated on the success of communication and the emergence of a language. The testing phase is not necessarily used to test the generalization capabilities of the agents, but rather to extract a sample of the emerged language. It is therefore not as important that the samples in the testing phase are unseen samples. Hence, for both training and testing a certain number of samples are randomly drawn from the complete dataset. In these experiments, the dataset consists of 10.000 scenes in total. To train the agents, 128.000 games are played (using 128.000 randomly drawn samples from the dataset) with a batch size of 32 samples. To retrieve the samples of the emerged language, after the agents are trained, they are frozen and play 2.048 games with randomly drawn samples from the dataset. This yields 2.048 interactions that will be used to analyze the emerged language.

Across all experiments, the same restriction on the vocabulary and the produced messages are compared. First, five different vocabulary sizes $|V|$ are tested: 2, 10, 16, 50 and 100 symbols. Notice that these vocabularies include the <*eos*> token which signals the end of the message. A vocabulary of two symbols therefore consists effectively of only one token that can be learned by the agents. The center of attention lies on the vocabularies with 16 and 50 symbols. The vocabulary with 16 symbols corresponds the 13 attributes the objects can have and align with human language referring expressions. However, a smaller vocabulary size increases the pressure to converge on meaning, which is my also 10 and 2 symbols are tested. On the other side, the agents might instead of a compositional language learn to assign one symbol with every

combination of attributes. This results in $2 * 3 * 8 = 42$ unique combinations, which corresponds to the vocabulary with 50 symbols. Furthermore, a vocabulary of 100 symbols is used to test the effect of a much bigger bottleneck.

Secondly, also different maximum message lengths are compared: 1, 2, 3, 4 and 6 symbols. Notice that this applies only to the maximum possible length; the agents can choose to use shorter messages. Again, the corresponding length to human referring expressions in English is 3 for the three attributes. However, the agents can be pressured to condense the information more precisely with shorter messages or given less restrictions with longer messages. Vocabulary size and message length play hereby an opposing role in the pressure on condensing information.

All agents are trained using the *Adam* optimizer (Kingma & Ba, 2015). LSTMs are used to produce and parse the message between the agents. Gumbel-Softmax relaxation is applied to train the agents with those discrete messages. Hereby, a temperature of $\tau = 1$ is used which showed the best results in preliminary experiments. All setups are additionally compared to a baseline. In the baseline the sender is "blinded", meaning that the sender is not shown any input but produces random messages. Effectively, the receiver needs to solve the task on their own, by exploiting underlying structures in the shown data and the target data. If a non-baseline model doesn't achieve a better performance than the baseline, the sender didn't communicate any meaningful messages and no language emerged.

6.1 Object identification

Setup

The object identification game is set up as a discrimination game. In a discrimination game, the agents are presented with two or more images, one of these being the target image. The sender needs to communicate this target image to the receiver by discriminating it from the other distractor images. The receiver then needs to decide based on the message, which of the images is the target image.

The discrimination games in this research have a very similar setup as described in (Lazaridou et al., 2017). The agents in this research resemble their *agnostic sender* as well as their *receiver*. One central difference is the production of the message. The main goal of their language games was the identification of the concept that the shown image was related to. Therefore, the sender communicated only single-symbol messages to the receiver which should describe the concept of the target image. Opposed to that, in this research, the agents are tasked to discriminate objects from each other based on their attributes. It is therefore assumed that the sender will communicate these discriminative attributes. For that reason, the sender is allowed to generate sequences as a message where for instance each symbol in the sequence might correspond to one attribute.

The data is prepared in the same way as for the single model object identification task in section 5.1. Bounding boxes of all objects in the scene are extracted and preprocessed to be used for the feature extractors. For datasets with a variable number of objects, the list bounding of bounding boxes is padded with a matrix of zeros to the maximum possible number of objects present in a scene across the dataset. Opposed to the experiments with one neural model, no additional information about the target object's attributes is provided. Instead, the sender agent is presented with the target object always as the first object, while the distractors are shuffled. The receiver is presented with completely shuffled bounding boxes.

Accordingly, also the models from the single model tasks serve as basis for the object identification task. The sender uses the architecture of the model described in section 5.2 apart from the final LSTM. Instead, it will use the LSTM with a hidden size h_s included in the *EGG* framework to produce a message. Based on the results from the single model object identifiers, the image embedding size e_s is fixed to 100.

The receiver instead uses the architecture of the single model object identifier, described in section 5.1. Similarly to the sender, the LSTM of the *EGG* framework with the hidden size h_e is used to encode the sender's message instead of the one-hot encoded attributes in the single model. The hidden state of the LSTM is projected to the embedding size $e_r = 500$ with a linear layer, based on the results in the previous experiments. As before the dot product between the embedded message and the encoded image is calculated to let the model point towards the bounding box that is described best by the message. Figure 14 shows, how the sender and the receiver of the discriminator are built up.

The experiments are conducted with a learning rate of 2×10^{-4}, the loss is calculated using cross entropy. The following values for the variables are compared:

- h_s: 100, 500, 1000
- h_r: 100, 500, 1000
- $|V|$: 2, 10, 16, 50, 100
- n: 1, 2, 3, 4, 6

The language game is run on the 'Dale-2', 'Dale-5' and 'CLEVR color' datasets. A random guess corresponds

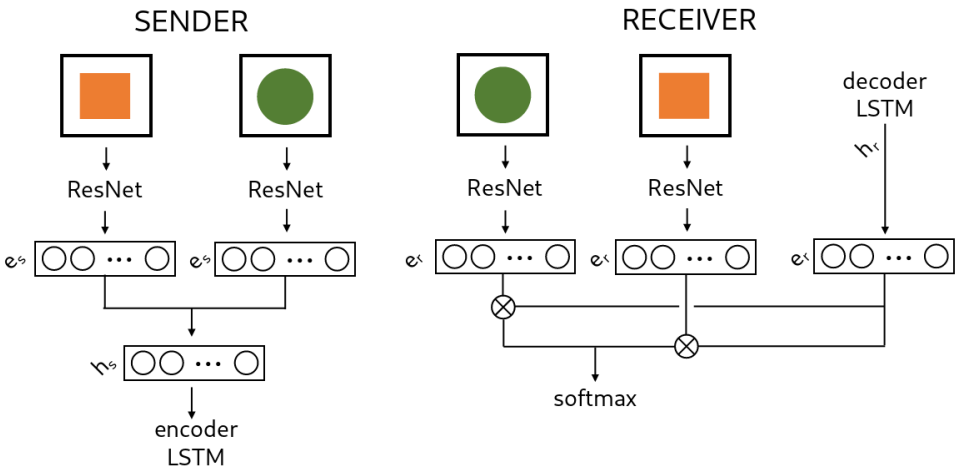


Figure 14: Sender and receiver architectures in the discrimination game

to 50% in the *Dale-2* dataset, 20% in the *Dale-5* dataset and 13% in the *CLEVR color* dataset.

Results

When looking at the results of the experiments, it can be seen that the agents are successfully discriminating the objects across all datasets. Tables 19 to 21 give an overview over the accuracy scores of selected configurations. For each dataset there are many configurations that outperform the baseline models by a high margin. However, there are still clear differences between the accuracy scores of each dataset.

[Dominik Künkele] baseline

[Dominik Künkele] learning curve

When the agents are tasked to discriminate between the two objects of the *Dale-2* dataset, it succeeds in almost all configurations with an accuracy of over 94%. Selected results are listed in Table 19. In many configurations the accuracy lies above 99%, while only 4 configurations have an accuracy of only 61% to 64%. An antecedent for a low accuracy seems to be a low vocabulary size $|V|$. All experiments with a vocabulary size of only two symbols (including the <eos> symbol) tend to give worse results. Three of the four results that don't surpass the baseline use a vocabulary size of 2. On the other side, only few configurations with $|V| = 2$ are among the overall best results. When the vocabulary size is increased, the results become better. Especially configurations with $|V| = 10$, $|V| = 16$, $|V| = 50$ and $|V| = 100$ consistently achieve high accuracies. While there are some few exceptions, the results show that the agents tend to perform better, when they can exchange messages with more than one symbol. Hereby, even games with an $n = 6$ perform as well as games with $n = 2$ as seen for example for $|V| = 10$. However, when only one symbol is allowed in the exchange, the accuracy still surpasses the baseline by a big margin and configurations with a high vocabulary of $|V| = 50$ and $|V| = 100$ tend to achieve higher accuracies. Finally looking at the hidden sizes, no direct effect can be identified and the differences in the results seem to be explained by vocabulary size and message length.

When the agents are presented with the *Dale-5* dataset, the results are less clear (see Table 20). Again, two groups can be identified: One group of configurations doesn't surpass the baseline of 35% accuracy, while the other group achieves accuracies in between 62% and 94%. For this dataset, the group of successful communication has a wider range of results compared to the *Dale-2* dataset. Only 16 configurations surpass 90% accuracy, while the majority lies between 80% and 90%. As before, for the hidden sizes, no pattern

				Dale-2
h_r	h_s	n	$\|V\|$	**Accuracy**
100	100	2	10	99,96%
100	1000	2	10	99,96%
1000	100	3	10	99,87%
500	100	3	100	99,83%
100	100	6	50	99,83%
100	500	3	16	99,7%
1000	100	6	16	99,61%
100	100	6	100	99,57%
100	500	1	100	99,52%
100	100	1	50	99,35%
500	100	1	2	99,26%
100	500	1	50	99,26%
500	500	1	100	99,26%
100	500	3	100	99,83%
500	1000	1	16	98%
100	500	1	10	96,61%
100	1000	1	100	99,44%
500	500	6	2	95,92%
100	1000	4	2	94,88%
100	1000	6	10	63,15%
1000	500	2	2	62,98%
-	-	-	-	62,72% (baseline)
100	1000	3	2	61,2%

Table 19: Selected results of the object identification games on the Dale-2 dataset with $e_s = 100$ and $e_r = 500$

is visible. However, there is now a clearer influence of the message length n and the vocabulary size $|V|$. First, 15 of the 16 configurations above 90% accuracy are allowed messages with more than 3 symbols. 8 of these configurations even have a message length $n = 6$. Although there are a few configurations with $n = 1$ that perform well, most of them have an accuracy of below 85%. Overall, a longer message tends to help the agents to communicate the correct object. Concerning the vocabulary size, a different pattern can be seen. A vocabulary size of $|V| = 10$, $|V| = 16$ and $|V| = 50$ gives the best results. All the configurations with results above 90% use these vocabulary sizes. While a large vocabulary size of $|V| = 100$ can still give good results, most of the time, the accuracy stays around 80% to 84%. When given a vocabulary size of only two symbols, the agents still perform far above the baseline. However, compared to the other sizes, the accuracy scores are at the lower end, mainly between 62% and 80%. As expected, when the message length is allowed to be longer, these configurations perform the best.

Finally, the results for the *CLEVR color* dataset show a similar structure (see Table 21). Two main groups can be identified: Configurations that perform as the baseline with an accuracy of around 23% and configurations that successfully communicate with accuracies of 40% to 73%. Compared to the *Dale-5* dataset, the highest accuracy is much lower. Furthermore, almost 70% of the configurations are unsuccessful, which is much bigger compared to the *Dale-5* dataset. Still, similar conclusions can be drawn. Different hidden sizes don't seem to have an influence. For this dataset, medium message lengths of $n = 2$, $n = 3$ and $n = 4$ provide the best results for the agents. However, the agents can still communicate successfully with $n = 1$ and $n = 6$, but the accuracy reaches maximum 64% and mainly stays around 50%. As before, the agents can

				Dale-5
h_r	h_s	n	$\|V\|$	**Accuracy**
1000	500	3	50	91,75%
100	100	6	10	91,02%
100	100	3	10	90,76%
100	100	4	16	89,76%
100	100	2	10	88,85%
500	1000	1	100	87,11%
1000	1000	6	50	85,07%
500	1000	1	10	84,46%
500	500	1	50	83,29%
100	100	1	10	80,86%
500	100	1	100	76,74%
500	500	4	2	77,78%
1000	1000	3	2	75,78%
100	100	3	2	74%
500	500	1	2	63,41%
500	1000	2	100	34,98%
500	500	3	16	34,77%
500	500	6	2	34,38%
1000	1000	4	2	35,33%
1000	1000	6	100	35,24%

Table 20: Selected results of the object identification games on the Dale-5 dataset with $e_s = 100$ and $e_r = 500$

utilize a vocabulary size of $|V| = 10$ and $|V| = 16$ the best. Some configurations with $|V| = 2$ and $|V| = 100$ achieve accuracies of up to 50%, but most lead to an unsuccessful communication. Interesting hereby is that a vocabulary size of $|V| = 2$ requires n to be bigger for better results, while the opposite is true for $|V| = 100$.

Overall these results show that across all datasets successful communication between the agents emerge. The sender is able to encode a description of the target object into the message which the receiver is able to decode. Unsurprisingly, this works the best for simplest dataset *Dale-2* with only one distractor. With an increasing number of distractors, fewer games are successful, and successful games perform worse. Interestingly, with all vocabulary sizes a language can emerge between the agents. While the vocabulary size $|V| = 2$ usually requires longer messages to succeed, there are some games that also achieve high accuracies with short messages. This fact suggests that the agents don't necessarily rely on the attributes shape, color or size, since they would require more symbols to be encoded. Instead, the agents likely utilize other patterns that are not used in human referring expressions. This will be further analyzed in section 7.

				CLEVR color
h_r	h_s	n	$\|V\|$	**Accuracy**
100	1000	3	10	73,65%
100	1000	3	16	73,57%
500	500	4	10	73,57%
100	500	3	16	70,75%
100	1000	4	50	69,31%
1000	500	1	16	62,85%
100	500	1	10	57,12%
1000	500	6	16	54,69%
100	100	4	2	52,95%
1000	500	1	100	52,82%
100	1000	1	100	52,34%
100	500	3	2	51,48%
500	1000	3	16	24,61%
100	1000	3	50	24,48%
1000	1000	4	100	24%
500	1000	1	2	23,96%
1000	100	6	10	23,87%
500	1000	6	50	23,87%
500	500	2	2	22,35%

Table 21: Selected results of the object identification games on the CLEVR color dataset with $e_s = 100$ and $e_r = 500$

6.2 Referring expression generation

Setup

In a next step, it is tested if the agents can learn to extract features of the objects together. For this the same sender as in the previous section is used. Based on the results, h_s is fixed to 500 and e_s to 100 However, the task and by that the receiver's architecture is adapted in two different ways. In the first setup, the **referring expression generator**, the receiver is tasked to describe the target object with natural language following the *incremental GRE-algorithm* based on the sender's message (see Figure 15). This follows the approach of the approach of the *RE generators* described in section 5.2. The whole scene is encoded using the *image encoder* submodule, projecting it to the image encoding dimension $e_{ri} = 100$. The sender's message is decoded with the hidden size $h_r = 500$ and $e_r = 100$, and concatenated with the encoded image. This is then reduced to $LSTM_o = 1500$ dimensions based on the results of the previous experiments, and used as the initial state of the captioning LSTM. Tokens are embedded with $LSTM_e = 15$ dimensions. Since the position of the padding as well as the order of the words in the target didn't have an effect on the results, the usual approach in natural generation task is used: padding tokens are appended and the referring expression is not reversed. During training, the ground truth caption is used as the input to the LSTM using teacher forcing. When presented with test data, the LSTM always produces three tokens, by using its own predicted words as the input for the next step. The loss is calculated using cross entropy.

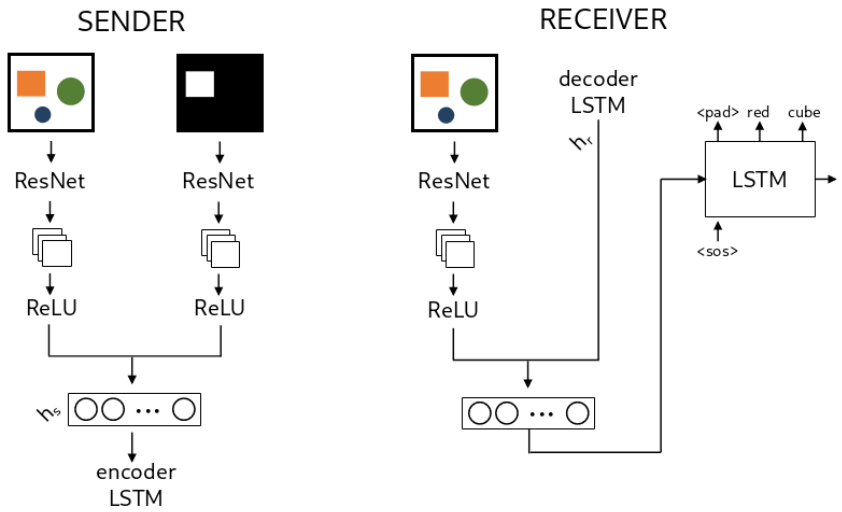


Figure 15: Simplified architecture of the caption generator game

However, this setup could lead to two problems: First, the task is quite complex for both agents to learn, since two language generation steps are involved. This might stop the agents from converging towards a useful language. Secondly, in case a language emerges, it could be aligned with the natural language of the target referring expressions. In other words, the sender might just learn to effectively repeat the target referring expressions without extracting the necessary features themselves. For that reason, in the second setup the receiver is tasked to predict only the attributes of the target object in the shape of a one-hot vector. By this, no natural language including its salience order is involved and the complexity of the complete game is reduced. The receiver uses the same way to encode and combine the complete scene and the sender's message with $e_{ri} = 100$, $h_r = 500$ and $e_r = 100$. This is then passed to a linear layer that produces a vector with 13 dimension, corresponding to the 2 sizes, 3 shapes, and 8 colors. The loss is then calculated using binary cross entropy.

[Dominik Künkele] figure

The experiments for both setups are conducted with a learning rate of 2×10^{-4}. As in the previous section, the following values for the variables are compared:

- $|V|$: 2, 10, 16, 50, 100
- n: 1, 2, 3, 4, 6

Since the agents are trained to describe the target object discriminatively based on the described GRE-algorithm, they are trained on the *Dale-2*, *Dale-5* and *CLEVR color* dataset. The *Dale-5* and the *CLEVR color* should be again much harder to learn, since there are more objects that the agents need to discriminate the target object from. The same metrics as in the section 5.2 are used to evaluate the results for the first setup. When predicting one-hot vectors, an overall **accuracy** is reported that reflects if all attributes were predicted correctly. Additionally, the accuracy for each attribute is calculated separately to show which attributes give bigger challenges for the models.

Results

			Dale-2			Dale-5		
$\|V\|$	h_s	h_r	**Acc.**	**word-by-word**	**length**	**Acc.**	**word-by-word**	**length**
10	10	10	22,9%	62,8%	1	7,1%	40%	1
13	10	10	22,8%	62,9%	0	7,3%	38,7%	1
20	10	10	24,6%	64%	1	6,7%	38,7%	1
100	10	10	24,4%	62%	1	7,8%	40%	1
100	100	100	21%	62%	1	6,5%	37,8%	1

Table 22: Results of the caption generator: $|V|$ are different vocabulary sizes and h hidden sizes.

The results of the caption generator game are summarized in Table 22. In general, it can be seen that the agents have much bigger problems, to solve the task together than a single neural network. The highest accuracy for descriptions, the agents manage to predict correctly is at 24,6% for images of the 'Dale-2' dataset. Compared to the (masked) accuracy of the single model with 72%, the agents predict 47,4% points less correct descriptions. A similar worse performance can be seen for the 'Dale-5' dataset. Here, the agents only manage to produce for 7,8% of the images correct descriptions with a vocabulary size of 100, 13,2% points less than the single neural model. The same effect can be seen for the word-by-word accuracy, which is much lower than the metric for the single neural model for both datasets.

When looking, how the different variables affect the performance, it can be seen that a bigger vocabulary size tends to help the agents. This is only visible for the 'Dale-2' dataset. With constant hidden sizes of 10, the agents score around 22,9% with only 10 and 13 available symbols. When this is increased to 20 and respectively 100 symbols, the agents can increase their accuracy to around 24,5%. However, the increase is relatively small. Interestingly, this effect only occurs, when the hidden sizes are small with only 10 dimensions. As soon as they are increased to 100 dimensions with a vocabulary size of 100 symbols, the accuracy drops to 21%.

Looking at the 'Dale-5' dataset, the increase is still there, when the vocabulary is increased to 100 symbols. Nonetheless, the difference is with 0,5% points even smaller and the reason may be due to other influences, such as the random initialization of the weights of the agents. This is confirmed, when looking at emerged languages. In all the setups, the same message is communicated for all samples, independently of the input image. This is also reflected in the length of the messages. For the setup with a vocabulary size of $|V| = 13$, no message is transferred, and the accuracy stays the same as in the other setups.

These results show that the agents are not at all able to encode meaning about the images and target objects in their messages. This is especially interesting, compared to single model caption generator in section 5.2. In these experiments, the model was able to converge towards correct captions and therefore able to extract the necessary information. This shows that a main challenge for the agents lies in grounding symbols in these extracted features.

6.3 Reference resolution

Setup

In the final experiments, the agents are tasked to produced and understand referring expressions only based on the implicit human knowledge present in the scenes, as for example the attributes of the objects. Opposed to the previous experiments, explicit human language information as human referring expressions or one-hot encoded attributes are avoided. By this, the agents can't just reuse these human referring expressions, but need to learn to generate them themselves during the language games just based on the visual input. This is done similarly to the single model reference resolution task, described in section 5.3. As before, the sender has the same architecture as in the previous language games, encoding the bounding boxes of all objects.

The receiver is tasked to solve two tasks. First, in the **coordinate reference resolver**, the agents are tasked to predict the center coordinates of the target object (see Figure 16). The receiver is the same in both setups and is based on the *reference resolver* + referring expressions, described in section 5.3. Again, instead of the included LSTM, the receiver uses the *EGG* LSTM with the hidden size $h_r = 500$ and $e_r = 100$ to encode the message. The dimensions of the coordinate predictor are fixed to $c = 2048$, based on the previous results. The euclidean distance between the resulting prediction of the center point and the true center of the target object is calculated and the weights of both agents are adapted accordingly.

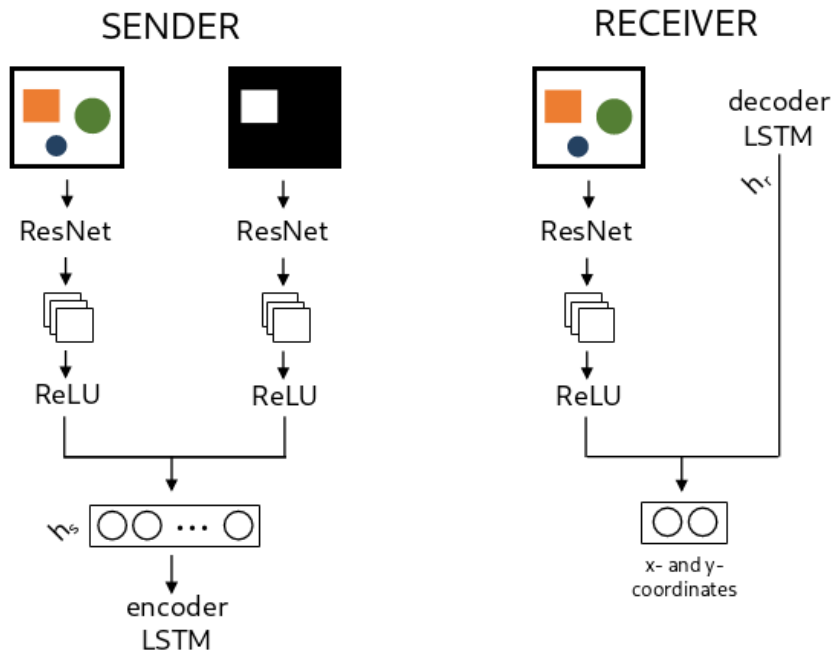


Figure 16: Simplified architecture of the masked coordinate predictor game

Secondly, in the **attention reference resolver**, the agents learn to point towards regions in the image, which is based on the *attention reference resolver* described in section 5.3. As before, the LSTM is replaced with the *EGG* LSTM to parse the sender's message with the hidden size $h_r = 500$ and $e_r = 100$. The dot product is calculated between each encoded region of the image and the encoded message, and the $softmax$ function is applied subsequently. The loss is calculated using *binary cross entropy* between the predictions and ground truth regions.

[Dominik Künkele] figure

The experiments are conducted with the following hyperparameters: a learning rate of 2×10^{-4}, a temper-

ature for the Gumbel-Softmax relaxation of 1 and *Adam* (Kingma & Ba, 2015) as optimizer. The following values for the variables are compared:

- $|V|$: 2, 10, 16, 50, 100
- n: 1, 2, 3, 4, 6

As in section 5.3, the agents are trained first on the *CLEVR single* dataset to understand if they are capable of predicting locations in an image together. In a next step, the *Dale-2*, *Dale-5* and *CLEVR color* datasets are used to test if the agents are able to first communicate a target object and second describe the target object discriminatively with a small vocabulary. The same metrics are used to evaluate the results as described in 5.3.

Results

In the final setup, the agents are tasked with communicating objects with fewer infused human knowledge. Table 23 shows the results for the setup, in which the sender is pointed towards the target object with a human description based on the GRE algorithm. Hereby, the 'CLEVR single' dataset acts as a baseline, to test if the agents are able to predict coordinates of objects at all. In every configuration of the variables, the agents achieve a very high performance. The worst average distance across the test dataset is 10 pixels, which still points onto an object. Also the accuracy, which evaluates how many guesses of the receiver were pointing onto an object reflects this fact. All configuration achieve an accuracy higher than 96,7%. This aligns also with the results from the single neural models, where the average distance was similarly low. In general, this shows that the agents are able to predict coordinates together. However, the question arises if the message by the sender has actually an effect on the receivers' decision, or if the receiver learns to point towards the target coordinates on his own and the message is ignored. Having a look at the transferred messages, it in fact shows that the receiver learns to point towards the target object on its own. As in the experiment before, all communicated messages contain the same symbol independent of the input image.

			CLEVR single			**Dale-2**			**Dale-5**		
$\|V\|$	h_s	h_r	**Dist.**	**Acc.**	**length**	**Dist.**	**Acc.**	**length**	**Dist.**	**Acc.**	**length**
10	10	10	10,1	98,5%	1	36,5	19,9%	1	45,7	14,4%	1
13	10	10	6	99%	0	38	20,4%	1	47,3	10,8%	1
20	10	10	9,7	96,7%	1	37,3	21,2%	1	47,3	11,3%	0
100	10	10	7,7	98,4%	1	40,4	21,7%	1	45,4	10,8%	1
100	100	100	7,5	96,9%	1	40,1	17,8%	1	44,3	11,8%	0

Table 23: Results of the description coordinate predictor: $|V|$ are different vocabulary sizes and h hidden sizes.

When the experiments are run on the 'Dale-2' dataset, the results are much worse. For the *description coordinate predictor*, the average distance ranges from 36,5 pixels to 40,4 pixels. The configuration with a vocabulary size of only 10 symbols fares the best, while a vocabulary of 100 symbols produces the worst results. Still, the accuracy shows that around 19,9% to 21,7% of the guesses are on the target object. Here, the configurations with higher vocabulary sizes fare slightly better, but the differences are very small and likely due to other factors.

The results for 'Dale-5' dataset are even worse, but are comparable with the results with a single neural model. Apparently, the agents are not able to communicate the target object, and the predictions by the

receiver are general in the middle of the image, which results in an average distance of around 45 to 50 pixels. The differences of the mean distances are not very significant in this range, to allow an analysis of the different configurations.

				CLEVR single			**Dale-2**			**Dale-5**		
$\|V\|$	h_s	h_r	e_s	**Dist.**	**Acc.**	**length**	**Dist.**	**Acc.**	**length**	**Dist.**	**Acc.**	**length**
10	10	10	1024	10,8	93,1%	1	34,8	24,3%	0	44,3	11,8%	1
10	10	10	512	9,3	92%	1	36,3	19,9%	0,7	45,9	12,7%	1
13	10	10	1024	7,8	96,8%	1	36,3	20,2%	0	45,4	11,4%	1
20	10	10	1024	6,6	98,3%	1	37,8	16,1%	1	45,2	11%	1
100	10	10	1024	5,2	98,5%	1	37,4	20,1%	1	43,6	16,7%	1
100	100	100	1024	12,5	92,1%	1	36,5	20,7%	1	44,6	12,7%	1

Table 24: Results of the masked coordinate predictor: $|V|$ are different vocabulary sizes, h hidden sizes and e embedding sizes.

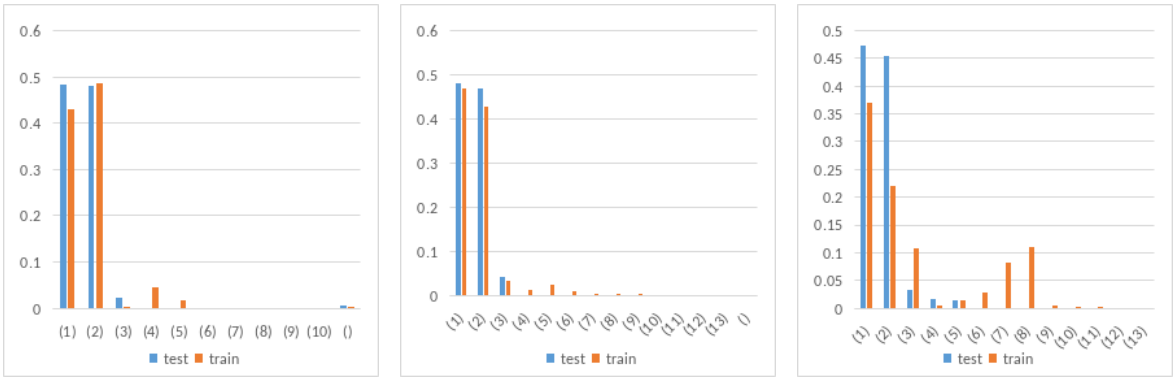
Finally looking at the setup, when using the masked image as an input shows that the results are similarly bad as when using the encoded captions. This is easily explainable with the emerged language. For both the experiments using the encoded captions and the experiments using masked images as input, no meaningful symbols are transferred. Following, the receiver needs to solve the task alone and the different setups of the sender don't play any role in the overall success.

When comparing these results to the neural models in Section 5.3 that are not part of a language game, it can be seen that the metrics are very similar for all datasets. The model was already not able to solve the task without the increased complexity of a language game. This therefore indicates that the challenge for the agents doesn't lie in grounding the extracted features in new symbols, but already in extracting the features in the first place.

7 Analysis of the emerged languages

The languages that emerge are analyzed in two ways. An analysis of the frequency of used symbols and messages, as well as an examination for which images and objects they are used indicates the structure of the language and meanings of the symbols. In a second step, the language is compared to several referring expressions in natural language. By doing this, it can be seen if the models learned to use similar ways of referring to objects as humans do, or if they rely on different approaches.

In the above experiments, only three configurations lead to success, where a language emerged that the agents used to exchange information: languages with vocabulary sizes of $|V| = 10$, $|V| = 13$ and $|V| = 100$. They will be referred to as $Lang_{10}$, $Lang_{13}$ and $Lang_{100}$ respectively. In this section, these three emerged languages is analyzed in more detail. This is done in two parts. The first part is a qualitative analysis, to understand, which symbols are used to transfer which information. In the second part, the emerged languages are compared to English. More specifically, it is tested if the messages of the sender align with English referring expressions of the target object.



(a) $Lang_{10}$ (b) $Lang_{13}$ (c) $Lang_{100}$, (13 most frequent messages)

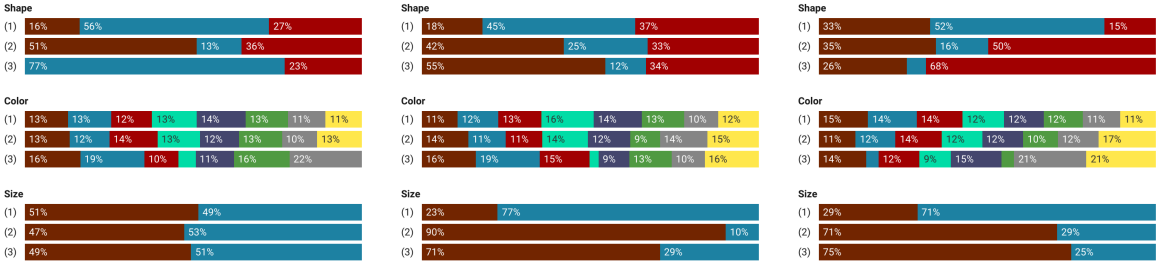
Figure 17: Relative frequencies of messages (ordered by frequency in test dataset)

When looking at the emerged vocabulary **qualitatively**, a few properties can be seen. Figure 17 shows an overview of the frequencies of messages in all three emerged languages for the training and the test split. Since the tokens themselves are arbitrary, they are ordered by the relative frequency in the messages for the test set and indices for the tokens are added from index 1 for the most frequent token and index $|V|$ for the least frequent token. By this the languages are easier comparable across different runs and vocabulary sizes. Figure 17(c) shows only the 13 most frequent message to provide a better overview. The lesser frequent messages are never used for the test data and each only used one or two times in the train data.

The first property is that when a message is transferred, it consists of only one symbol. In some rare cases, also an empty message is communicated. The models therefore don't learn any compositionality by combining symbols to create new meaning, but rather encode everything in separate symbols. Secondly, in all three languages, only very few symbols occur with a high frequency, while most of the symbols are used very rarely. More specifically, two symbols are used in 95% of the images with $Lang_{10}$ and $Lang_{13}$, while three symbols are used with $Lang_{100}$. Thirdly, the agents make use of fewer symbols, when presented with unseen test images compared to when communicating about images in the training split. This is especially visible for $Lang_{100}$. Symbols that are used for 16,5% of the training sample are not used at all in test split. Furthermore, the frequencies in the test split is much more focused on the two most frequent symbols, while it is more distributed around 5 symbols in the train split.

These findings indicate that referring expressions do emerge in each of the newly emerged languages since

the agents are able to communicate the correct object. However, the agents converge towards very few different referring expressions that are made up differently than in English and likely don't rely on the high level attributes *shape*, *color* and *size*. The similar frequencies across all three languages suggest that a greater vocabulary size $|V|$ doesn't necessarily lead to different referring expressions, but the languages still converge towards two main expressions.



(a) $Lang_{10}$ (b) $Lang_{13}$ (c) $Lang_{100}$

Figure 18: Relative share of the described target object's attributes for the top three messages
Shape: brown: sphere, blue: cube, red: cylinder
Size: brown: small, blue: large

[Dominik Künkele] differnces in different languages correspond to differences in color prediction of single model caption generation

Figure 18 shows the attributes of the target object that are described by each message. Hereby, the relative share each value of all three attributes are displayed. For instance the first bar in Figure 18(a) shows that 16% of the target objects that are described with symbol (1) in the language $Lang_{10}$ are spheres. In the figures, only the three most frequently used messages are included, which make up over 95% of all messages.

The first thing that can be seen is that different symbols convey different values of attributes. In the language $Lang_{10}$, symbol (1) is mostly used for target objects that are cubes, while symbol (2) is mostly used for spheres. There is not much difference for target objects that are cylinders. The distribution for colors is almost constant across symbols (1) and (2). Symbol (3) is more frequently used for blue, green and gray objects, while being less used for red, cyan and yellow objects. This different distribution may also be caused by the much lower absolute usage of symbol (3). The different sizes are encoded by all symbols in the same way.

Looking at $Lang_{13}$ the symbol usage differs. The frequencies for the shape look similar, but (the much less used) symbol (3) encodes most of the time spheres instead of cubes. Again, the colors look similar with small deviations for brown, green and yellow objects. Most striking however, is the difference for the size. Symbol (1) is used in 77% of the cases for large objects, while symbol (2) and (3) are used in 90% and 71% of the messages for small objects.

Language $Lang_{100}$ uses symbols its symbols to discriminate cubes and cylinders, while the frequencies for sphere remain constant across all symbols. The usage for the color is similar to $Lang_{13}$. Looking at the size, as for $Lang_{13}$, symbol (1) is mostly used to encode small objects, while symbols (2) and (3) are mostly used to encode large objects.

An interesting observation is that symbols are not used for the same attributes across the languages. In some languages, an attribute is not captured at all by the symbols, while another language heavily relies on it. The same applies to the values of attributes, especially to the shapes. Only two of the shapes are distinguished by the usage of symbols, while the third is not captured. Which shape are encoded, differs from language

to language.

Furthermore, these numbers confirm even more that the agents don't rely solely on the human defined attributes. For instance $Lang_{10}$ only encodes the shape in its symbols. This would not be enough to distinguish the target object from the distractor consistently. Following, the agents also encode some additional underlying attributes and patterns to the three above defined attributes.

To compare the emerged language to English in a **quantitative** way, a probing approach is used. Probing can be used to analyze and interpret the hidden representations in neural networks. Hereby, a second neural model is trained to predict selected linguistic properties on the basis of the hidden representations. If this model is successfully able to predict the linguistic properties, the hidden representations are connected to them. If second model can't be trained, it indicates that there is no correlation between the hidden representation and the linguistic property. In the case of the emerged language, a neural model is trained to translate the messages of the sender into English referring expressions, based on the GRE algorithm by Dale & Reiter (1995). Hereby, the model consists of an encoder LSTM, one linear layer and a decoder LSTM. The encoder LSTM encodes the messages of the emerged language. The final hidden state is used as the meaning of the complete sentence and passed through the linear layer, which should learn an abstract representation of the message. The resulting vector is used as the initial state of the decoder LSTM. The decoder LSTM is then trained to produce the English referring expression. While decoding, teacher forcing is applied. The success of the model is validated calculating cross entropy between the models predictions and the target English referring expression. Since generalization doesn't play a role for probing, no dropout is applied and the model is trained and validated on the complete dataset; no test or validation split is used.

In the sections above, the attributes are ordered by importance in the way, it is usually used in English: shape > color > size. Since the agents do not necessarily need to follow this, all possible orders of importance for the attributes are compared to each other.

The translation model with this setup can learn two characteristics. First, it can learn to find correlations between the emerged language and the English referring expressions. Secondly, it can learn patterns in the English referring expressions that are independent of the emerged language. For instance, it can learn that the referring expressions are likely to be two symbols long, or that 'cube' is the most common shape. This second characteristic can lead to a low loss of the model, even though there are no connections to the emerged language. In this test, only the first characteristic is interesting and would show the correlation between the emerged language and English referring expressions. For that reason, two reference baselines are calculated for each of the attribute importance order. The first reference uses a non-informative input for the encoder LSTM, more specifically it always receives a vector of zeros and therefore can't learn any meaningful representation in the linear layer; the input for the decoder LSTM is the same for every sample. By this, the model is trained to learn the patterns in the English referring expressions. The resulting loss is the highest possible loss the model can achieve, independent of the input. Resulting, if the emergent language is connected to English, the loss will be lower than this baseline. If not, the loss will be as high as this baseline. On the other side, we calculated the lowest loss possible, by using the English referring expressions as both input and target. This verifies that the model is able to learn an abstract representation from the input and the resulting loss should be close to zero (since input is perfectly correlated to the target). These two references are used, to normalize the results of the actual emergent languages L_{em}, using the formula $L_{norm} = \frac{L_{em} - L_{English}}{L_{baseline} - L_{English}}$. A loss close as $L_{baseline}$ will lead to 100%, while a loss as $L_{English}$ will lead to 0%. By doing this, all configurations can be compared directly to each other.

Table 25 shows the results for all different languages. First, it can be seen for all emerged languages, the model is able to find correlations between natural language referring expressions and the messages by the sender. Still, all losses stay high and closer to the loss of the baseline, namely where the model only learns the patterns in the English referring expressions. This fact concludes that all emerged languages don't rely

Order	$L_{baseline}$	$L_{English}$	$\|V\| = 10$		$\|V\| = 13$		$\|V\| = 100$	
			L_{em}	L_{norm}	L_{em}	L_{norm}	L_{em}	L_{norm}
shape > color > size	0,659	0,0001	0,569	**86,19%**	0,622	94,24%	0,597	90,4%
shape > size > color	0,589	0,0002	0,489	**83,09%**	0,531	90,21%	0,47	**79,78%**
color > shape > size	0,849	0,0	0,802	94,49%	0,801	94,36%	0,791	93,15%
color > size > shape	0,836	0,0	0,819	98,01%	0,772	92,35%	0,786	94%
size > shape > color	0,532	0,0052	0,492	92,26%	0,437	**81,95%**	0,457	**85,61%**
size > color > shape	0,599	0,0001	0,573	95,71%	0,495	**82,67%**	0,538	**89,87%**

Table 25: Cross entropy losses while probing with emerged languages successful on 'Dale-2'

on the GRE algorithm by Dale & Reiter (1995). Nonetheless, it may be possible that a different algorithm is used to create referring expressions in the artificial language.

Secondly, it can be seen that the correlation between the emerged language and the natural language referring expressions differ for each of the languages. The vocabulary consisting of 10 symbols is the closest related to the orders *shape* > *size* > *color* and *shape* > *color* > *size*, the vocabulary based on 13 symbols on the other hand to the orders *size* > *shape* > *color* and *size* > *color* > *shape*. With 100 symbols, the vocabulary resembles mostly *shape* > *size* > *color* and *size* > *shape* > *color*. Striking here is that even though the related orders are different, the most important attributes are either the *size* or *shape* across all three emerged languages. The attribute *color* seems to be less important. This is reinforced by the fact that the losses are closer to the baseline, when the *color* is the most or second most important attribute in the order.

8 Discussion

9 Conclusion and future work

[Dominik Künkele] 2 pages

References

Ahrens, K., Kerzel, M., Lee, J. H., Weber, C., & Wermter, S. (2022). Knowing earlier what right means to you: A comprehensive vqa dataset for grounding relative directions via multi-task learning. *IJCAI 2022 Workshop on Spatio-Temporal Reasoning and Learning*.

Antol, S., Agrawal, A., Lu, J., Mitchell, M., Batra, D., Zitnick, C. L., & Parikh, D. (2015). Vqa: Visual question answering. In *International Conference on Computer Vision (ICCV)*.

Baroni, M. (2020). Rat big, cat eaten! ideas for a useful deep-agent protolanguage. *ArXiv preprint*.

Baroni, M., Dessi, R., & Lazaridou, A. (2022). Emergent language-based coordination in deep multi-agent systems. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: Tutorial Abstracts* (pp. 11–16). Abu Dubai, UAE: Association for Computational Linguistics.

Bartlett, M. & Kazakov, D. (2005). The origins of syntax: from navigation to language. *Connection Science*, 17(3-4), 271–288.

Bay, H., Tuytelaars, T., & Van Gool, L. (2006). Surf: Speeded up robust features. In A. Leonardis, H. Bischof, & A. Pinz (Eds.), *Computer Vision – ECCV 2006* (pp. 404–417). Berlin, Heidelberg: Springer Berlin Heidelberg.

Bender, E. M., Gebru, T., McMillan-Major, A., & Shmitchell, S. (2021). On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21 (pp. 610–623). New York, NY, USA: Association for Computing Machinery.

Bender, E. M. & Koller, A. (2020). Climbing towards NLU: On meaning, form, and understanding in the age of data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (pp. 5185–5198). Online: Association for Computational Linguistics.

Bisk, Y., Holtzman, A., Thomason, J., Andreas, J., Bengio, Y., Chai, J., Lapata, M., Lazaridou, A., May, J., Nisnevich, A., Pinto, N., & Turian, J. (2020). Experience grounds language. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 8718–8735). Online: Association for Computational Linguistics.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., & Amodei, D. (2020). Language models are few-shot learners. In *Advances in neural information processing systems*, volume 33 (pp. 1877–1901).

Cao, K., Lazaridou, A., Lanctot, M., Leibo, J. Z., Tuyls, K., & Clark, S. (2018). Emergent communication through negotiation. In *International Conference on Learning Representations*.

Chaabouni, R., Kharitonov, E., Bouchacourt, D., Dupoux, E., & Baroni, M. (2020). Compositionality and generalization in emergent languages. In D. Jurafsky, J. Chai, N. Schluter, & J. Tetreault (Eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (pp. 4427–4442). Online: Association for Computational Linguistics.

Chaabouni, R., Kharitonov, E., Dupoux, E., & Baroni, M. (2019). Anti-efficient encoding in emergent communication. In *Advances in Neural Information Processing Systems*, volume 32: arXiv.

Chaabouni, R., Kharitonov, E., Dupoux, E., & Baroni, M. (2021). Communicating artificial neural networks develop efficient color-naming systems. *Proceedings of the National Academy of Sciences*, 118(12).

Chaabouni, R., Strub, F., Altché, F., Tarassov, E., Tallec, C., Davoodi, E., Mathewson, K. W., Tieleman, O., Lazaridou, A., & Piot, B. (2022). Emergent communication at scale. In *International Conference on Learning Representations*.

Clark, H. H. & Wilkes-Gibbs, D. (1986). Referring as a collaborative process. *Cognition*, 22(1), 1–39.

Coventry, K. R., Cangelosi, A., Rajapakse, R., Bacon, A., Newstead, S., Joyce, D., & Richards, L. V. (2005). Spatial prepositions and vague quantifiers: Implementing the functional geometric framework. In *Spatial Cognition IV. Reasoning, Action, Interaction* (pp. 98–110). Springer Berlin Heidelberg.

Dale, R. & Reiter, E. (1995). Computational interpretations of the gricean maxims in the generation of referring expressions. *Cognitive science*, 19(2), 233–263.

Davidoff, J. (2001). Language and perceptual categorisation. *Trends in Cognitive Sciences*, 5(9), 382–387.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition* (pp. 248–255).

Dessì, R., Kharitonov, E., & Baroni, M. (2021). Interpretable agent communication from scratch (with a generic visual processor emerging on the side). In A. Beygelzimer, Y. Dauphin, P. Liang, & J. W. Vaughan (Eds.), *Advances in Neural Information Processing Systems*.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)* (pp. 4171–4186). Minneapolis, Minnesota: Association for Computational Linguistics.

Dobnik, S., Howes, C., & Kelleher, J. D. (2015). Changing perspective: Local alignment of reference frames in dialogue. In C. Howes & S. Larsson (Eds.), *Proceedings of SemDial 2015 (goDIAL): The 19th Workshop on the Semantics and Pragmatics of Dialogue* (pp. 24–32). Gothenburg.

Dobnik, S. & Kelleher, J. D. (2013). Towards an automatic identification of functional and geometric spatial prepositions. In *Proceedings of PRE-CogSci 2013 Production of referring expressions – bridging the gap between cognitive and computational approaches to reference at CogSci* (pp. 1–6). Berlin, Germany.

Dobnik, S. & Silfversparre, V. (2021). The red cup on the left: Reference, coreference and attention in visual dialogue. In *Proceedings of the 25th Workshop on the Semantics and Pragmatics of Dialogue - Full Papers* Potsdam, Germany: SEMDIAL.

Dobnik, S. & Storckenfeldt, A. (2018). Categorisation of conversational games in free dialogue over spatial scenes. In L. Prévot, M. Ochs, & B. Favre (Eds.), *Proceedings of AixDial - Semdial 2018: The 22st Workshop on the Semantics and Pragmatics of Dialogue* Aix-en-Provence: Semdial.

Dobnik, S. & Åstbom, A. (2017). (perceptual) grounding as interaction. In V. Petukhova & Y. Tian (Eds.), *Proceedings of Saardial: The 21st Workshop on the Semantics and Pragmatics of Dialogue* (pp. 17–26). Saarbrücken: SemDial.

Field, A., Blodgett, S. L., Waseem, Z., & Tsvetkov, Y. (2021). A survey of race, racism, and anti-racism in NLP. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)* (pp. 1905–1925). Online: Association for Computational Linguistics.

Ghanimifard, M. & Dobnik, S. (2017). Learning to compose spatial relations with grounded neural language models. In C. Gardent & C. Retoré (Eds.), *Proceedings of the 12th International Conference on Computational Semantics (IWCS) — Long papers*.

Ghanimifard, M. & Dobnik, S. (2019). What goes into a word: generating image descriptions with top-down spatial knowledge. In *Proceedings of the 12th International Conference on Natural Language Generation*: Association for Computational Linguistics.

Gupta, A., Resnick, C., Foerster, J., Dai, A., & Cho, K. (2020). Compositionality and capacity in emergent languages. In *Proceedings of the 5th Workshop on Representation Learning for NLP* (pp. 34–38).: Association for Computational Linguistics.

Harnad, S. (1990). The symbol grounding problem. *Physica D 42: 335-346*.

Harris, C. G. & Stephens, M. J. (1988). A combined corner and edge detector. In *Alvey Vision Conference*, volume 15.

Havrylov, S. & Titov, I. (2017). Emergence of language with multi-agent games: Learning to communicate with sequences of symbols. In I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA* (pp. 2149–2159).

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 770–778).

Hendricks, L. A., Burns, K., Saenko, K., Darrell, T., & Rohrbach, A. (2018). Women also snowboard: Overcoming bias in captioning models. In V. Ferrari, M. Hebert, C. Sminchisescu, & Y. Weiss (Eds.), *Computer Vision – ECCV 2018* (pp. 793–811). Cham: Springer International Publishing.

Hill, F., Tieleman, O., von Glehn, T., Wong, N., Merzic, H., & Clark, S. (2021). Grounded language learning fast and slow. In *International Conference on Learning Representations*.

Hofstadter, D. & Sander, E. (2013). *Surfaces and Essence: Analogy as the Fuel and Fire of Thinking*. Basic Books.

Hudson, D. A. & Manning, C. D. (2019). Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 6693–6702).

Ilinykh, N., Emampoor, Y., & Dobnik, S. (2022). Look and answer the question: On the role of vision in embodied question answering. In *Proceedings of the 15th International Conference on Natural Language Generation* (pp. 236–245). Waterville, Maine, USA and virtual meeting: Association for Computational Linguistics.

Jang, E., Gu, S., & Poole, B. (2017). Categorical reparameterization with gumbel-softmax. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Ji, A., Kojima, N., Rush, N., Suhr, A., Vong, W. K., Hawkins, R., & Artzi, Y. (2022). Abstract visual reasoning with tangram shapes. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing* (pp. 582–601). Abu Dhabi, United Arab Emirates: Association for Computational Linguistics.

Johnson, J., Hariharan, B., van der Maaten, L., Fei-Fei, L., Zitnick, C. L., & Girshick, R. (2017a). Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2901–2910).: arXiv.

Johnson, J., Hariharan, B., van der Maaten, L., Hoffman, J., Fei-Fei, L., Zitnick, C. L., & Girshick, R. (2017b). Inferring and executing programs for visual reasoning. In *Proceedings of the IEEE international conference on computer vision* (pp. 2989–2998).

Kelleher, J. D. & Dobnik, S. (2017). What is not where: the challenge of integrating spatial representations into deep learning architectures. In S. Dobnik, Simon; Lappin (Ed.), *Proceedings of the Conference on Logic and Machine Learning in Natural Language (LaML 2017)* Gothenburg: University of Gothenburg Centre for Linguistic Theory and Studies in Probability (CLASP).

Kharitonov, E. & Baroni, M. (2020). Emergent language generalization and acquisition speed are not tied to compositionality. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP* (pp. 11–15). Online: Association for Computational Linguistics.

Kharitonov, E., Chaabouni, R., Bouchacourt, D., & Baroni, M. (2019). EGG: a toolkit for research on emergence of lanGuage in games. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations* (pp. 55–60). Hong Kong, China: Association for Computational Linguistics.

Kingma, D. P. & Ba, J. (2015). Adam: A method for stochastic optimization. In Y. Bengio & Y. LeCun (Eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Kirby, S. (2002). Natural language from artificial life. *Artificial Life*, 8(2), 185–215.

Kirby, S., Cornish, H., & Smith, K. (2008). Cumulative cultural evolution in the laboratory: An experimental approach to the origins of structure in human language. *Proceedings of the National Academy of Sciences*, 105(31), 10681–10686.

Klymenko, O., Meisenbacher, S., & Matthes, F. (2022). Differential privacy in natural language processing the story so far. In *Proceedings of the Fourth Workshop on Privacy in Natural Language Processing* (pp. 1–11). Seattle, United States: Association for Computational Linguistics.

Kollar, T., Tellex, S., Roy, D., & Roy, N. (2010). Toward understanding natural language directions. In *2010 5th ACM/IEEE International Conference on Human-Robot Interaction (HRI)* (pp. 259–266).

Kottur, S., Moura, J., Lee, S., & Batra, D. (2017). Natural language does not emerge 'naturally' in multi-agent dialog. In M. Palmer, R. Hwa, & S. Riedel (Eds.), *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing* (pp. 2962–2967). Copenhagen, Denmark: Association for Computational Linguistics.

Krahmer, E. & van Deemter, K. (2012). Computational generation of referring expressions: A survey. *Computational Linguistics*, 38(1), 173–218.

Lakoff, G. & Johnson, M. (1980). *Metaphors we live by*. Chicago, IL: University of Chicago.

Landau, B., Smith, L., & Jones, S. (1998). Object perception and object naming in early development. *Trends in Cognitive Sciences*, 2(1), 19–24.

Larsson, S. (2018). Grounding as a side-effect of grounding. *Topics in Cognitive Science*, 10(2), 389–408.

Lauria, S., Bugmann, G., Kyriacou, T., Bos, J., & Klein, A. (2001). Training personal robots using natural language instruction. *IEEE Intelligent Systems*, 16(5), 38–45.

Lazaridou, A., Hermann, K. M., Tuyls, K., & Clark, S. (2018). Emergence of linguistic communication from referential games with symbolic and pixel input. In *International Conference on Learning Representations*.

Lazaridou, A., Peysakhovich, A., & Baroni, M. (2017). Multi-agent cooperation and the emergence of (natural) language. In *International Conference on Learning Representations*.

Lee, J. H., Kerzel, M., Ahrens, K., Weber, C., & Wermter, S. (2022). What is right for me is not yet right for you: A dataset for grounding relative directions via multi-task learning. In L. D. Raedt (Ed.), *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22* (pp. 1039–1045).: International Joint Conferences on Artificial Intelligence Organization. Main Track.

Lewis, D. K. (1969). *Convention: A Philosophical Study*. Cambridge, MA, USA: Wiley-Blackwell.

Lin, T.-Y., Maire, M., Belongie, S. J., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *European Conference on Computer Vision*.

Liu, F., Emerson, G., & Collier, N. (2023). Visual spatial reasoning. In *Transactions of the Association for Computational Linguistics*, volume 11 (pp. 635–651). Cambridge, MA: MIT Press.

Lowe, D. (1999). Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*: IEEE.

Lu, J., Batra, D., Parikh, D., & Lee, S. (2019). Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *Advances in Neural Information Processing Systems* (pp. 13–23).

Lu, J., Xiong, C., Parikh, D., & Socher, R. (2017). Knowing when to look: Adaptive attention via a visual sentinel for image captioning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 3242–3250).

Mao, J., Huang, J., Toshev, A., Camburu, O., Yuille, A., & Murphy, K. (2016). Generation and comprehension of unambiguous object descriptions. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 11–20).

Mitchell, M., van Deemter, K., & Reiter, E. (2013). Generating expressions that refer to visible objects. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 1174–1184). Atlanta, Georgia: Association for Computational Linguistics.

Monroe, W., Hawkins, R., Goodman, N., & Potts, C. (2017). Colors in context: A pragmatic neural model for grounded language understanding. *Transactions of the Association for Computational Linguistics*, 5(0), 325–338.

Noukhovitch, M., LaCroix, T., Lazaridou, A., & Courville, A. (2021). Emergent communication under competition. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '21 (pp. 974–982). Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems.

Qiao, Y., Deng, C., & Wu, Q. (2020). Referring expression comprehension: A survey of methods and datasets. In *IEEE Transactions on Multimedia*, volume 23 (pp. 4426–4440).

Ramisa, A., Wang, J., Lu, Y., Dellandrea, E., Moreno-Noguer, F., & Gaizauskas, R. (2015). Combining geometric, textual and visual features for predicting prepositions in image descriptions. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*: Association for Computational Linguistics.

Regier, T. (1996). *The human semantic potential: spatial language and constrained connectionism*. Cambridge, Massachusetts, London, England: MIT Press.

Roy, D. K. (2002). Learning visually grounded words and syntax for a scene description task. *Computer Speech, Language*, 16(3-4), 353–385.

Sánchez, J., Mazuecos, M., Maina, H., & Benotti, L. (2022). What kinds of errors do reference resolution models make and what can we learn from them? In M. Carpuat, M.-C. de Marneffe, & I. V. Meza Ruiz (Eds.), *Findings of the Association for Computational Linguistics: NAACL 2022* (pp. 1971–1986). Seattle, United States: Association for Computational Linguistics.

Shah, D. S., Schwartz, H. A., & Hovy, D. (2020). Predictive biases in natural language processing models: A conceptual framework and overview. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (pp. 5248–5264). Online: Association for Computational Linguistics.

Simonyan, K. & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In Y. Bengio & Y. LeCun (Eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Skocaj, D., Kristan, M., Vrecko, A., Mahnic, M., Janicek, M., Kruijff, G.-J. M., Hanheide, M., Hawes, N., Keller, T., Zillich, M., & Zhou, K. (2011). A system for interactive learning in dialogue with a tutor. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*: IEEE.

Steels, L. & Belpaeme, T. (2005). Coordinating perceptually grounded categories through language: a case study for colour. *The Behavioral and brain sciences*, 28, 469–89; discussion 489–529.

Steels, L. & Loetzsch, M. (2009). Perspective alignment in spatial language. In K. R. Coventry, T. Tenbrink, & J. A. Bateman (Eds.), *Spatial Language and Dialogue*, volume 3 of *Explorations in language and space* (pp. 70–88). Oxford University Press.

Weidinger, L., Uesato, J., Rauh, M., Griffin, C., Huang, P.-S., Mellor, J., Glaese, A., Cheng, M., Balle, B., Kasirzadeh, A., Biles, C., Brown, S., Kenton, Z., Hawkins, W., Stepleton, T., Birhane, A., Hendricks, L. A., Rimell, L., Isaac, W., Haas, J., Legassick, S., Irving, G., & Gabriel, I. (2022). Taxonomy of risks posed by language models. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '22 (pp. 214–229). New York, NY, USA: Association for Computing Machinery.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4), 229–256.

Winograd, T. (1972). Understanding natural language. *Cognitive Psychology*, 3(1), 1–191.

Wittgenstein, L. (1953). *Philosophische Untersuchungen*. Oxford: Basil Blackwell.

Xu, H. & Saenko, K. (2016). Ask, attend and answer: Exploring question-guided spatial attention for visual question answering. In B. Leibe, J. Matas, N. Sebe, & M. Welling (Eds.), *Computer Vision – ECCV 2016* (pp. 451–466). Cham: Springer International Publishing.

Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks? In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14 (pp. 3320–3328). Cambridge, MA, USA: MIT Press.

Young, P., Lai, A., Hodosh, M., & Hockenmaier, J. (2014). From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2, 67–78.

Zarrieß, S. & Schlangen, D. (2019). Know what you don't know: Modeling a pragmatic speaker that refers to objects of unknown categories. In A. Korhonen, D. Traum, & L. Màrquez (Eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (pp. 654–659). Florence, Italy: Association for Computational Linguistics.

Zaslavsky, N., Kemp, C., Regier, T., & Tishby, N. (2018). Efficient compression in color naming and its evolution. *Proceedings of the National Academy of Sciences*, 115(31), 7937–7942.

Zhang, P., Goyal, Y., Summers-Stay, D., Batra, D., & Parikh, D. (2016). Yin and yang: Balancing and answering binary visual questions. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 5014–5022).

Zitnick, C. L., Parikh, D., & Vanderwende, L. (2013). Learning the visual interpretation of sentences. In *2013 IEEE International Conference on Computer Vision* (pp. 1681–1688).

A Resources