UNIVERSITY OF GOTHENBURG

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 . . . 1

1.1 Research Questions . . . 1

1.2 Contribution . . . 2

1.3 Scope . . . 2

2 Background and Related Work . . . 3

2.1 Grounding . . . 3

2.2 Referring expressions . . . 3

2.3 Language Games . . . 4

2.3.1 Aims of emerged languages . . . 4

2.3.2 Setup of language games . . . 4

2.3.3 Properties of the emerged language . . . 5

2.3.4 Referring to objects and grounding . . . 5

2.4 Artificial dataset . . . 5

3 Methodology and Frameworks . . . 6

3.1 CLEVR framework . . . 6

3.2 Feature extractors . . . 7

3.3 Image processing . . . 8

3.4 EGG framework . . . 9

3.5 Optimization in language games . . . 10

3.6 Probing? . . . 10

3.7 Ethical considerations . . . 10

4 Creation of the datasets . . . 12

4.1 CLEVR single . . . 12

4.2 CLEVR color . . . 13

4.3 CLEVR Dale datasets . . . 14

5 Grounding referring expressions . . . 15

5.1 Object identification . . . 15

5.2 Referring expression generation . . . 18

5.3 Reference resolution . . . 24

6 Grounding referring expressions in language games 31
6.1 Object identification 31
6.2 Referring expression generation 34
6.3 Reference resolution 36
7 Analysis of the emerged languages 40
8 Discussion 44
9 Conclusion and future work 45
References 46
A Resources 49

1 Introduction

[Dominik Künkele] lead from grounding to referring expressions to language games

The study of emergent languages, in other words artificial languages created by agents to communicate and solve a goal are studied deeply in the recent years (Kirby, 2002; Kirby et al., 2008; Steels & Loetzsch, 2009; Lazaridou et al., 2017; Baroni, 2020; Baroni et al., 2022). The motivation behind this is twofold. First, the study of the emergence of artificial language can help to shed light into why and how natural languages evolved. This is done in a controlled environment, where researchers can regulate exactly, which information the agents receive, how and what they are able to communicate and which goals they are trained on. This approach allows a deep study of the constraints and restrictions that are necessary for language and meanings of symbols to evolve. Insights in this field might give indications, under which circumstances natural language evolves.

Secondly, emergent languages can have benefits over predefined communication protocols. Currently, machines communicate, by using protocols based on predefined rules of how messages can be built up. These are developed for specific scenarios and goals and don't allow flexible usages if the goal or the environment, machines act in are changing. Furthermore, these protocols are often based on readability for humans and include overhead that is not necessary to communicate a specific meaning, but is rather used for transparency. A language that emerges, rather than being predefined could mitigate these problems. 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).

Different studies conclude that meaning of symbols and sentences in language can only be bound to the real world, both in the literal sense, but also for abstract concepts (Harnad, 1999; Bender & Koller, 2020; Bisk et al., 2020). They need to be grounded. One way of grounding meaning can be achieved in multimodal systems, where models need to combine language with another modality for instance visual input. Hereby, a task can be to learn how to refer to objects or actions in the visual input, using referring expressions. Both generating referring expressions, based on the visual input and understanding given referring expressions by linking them to the visual input ground the meaning of the used language in vision. Following these arguments, also artificial emergent languages need to be grounded that the used symbols contain meaning.

1.1 Research Questions

In this thesis, a deeper look is taken into how agents can ground their emerged language in vision. 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 agent 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 that use real images are based on photos taking by humans. This adds a lot of inherent bias to the dataset, since these photos 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.

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 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 on single task. 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 attributes of objects, as well as their spatial relations towards other objects. In the present study, only attribute relations 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

[Dominik Künkele] importance of grounding and what it is

2.2 Referring expressions

[Dominik Künkele] what are referring expressions and how are they studied (generation, understanding, ...)

[Dominik Künkele] incremental GRE algorithm from Dale

Even though, recent large language models (LLMs) such as BERT (Devlin et al., 2019) or GPT-3 (Brown et al., 2020) produce impressive results on many tasks in Natural Language Processing, they are often only trained on big amounts of text corpora. 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. Many researchers criticize this approach, by arguing that these model can't learn meaning just by learning on text. 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. 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 vision. 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.

In a first step, models can combine linguistic knowledge with visual input, by referring to objects, seen in an image, with so called referring expressions. Hereby, the communicative intent as well as the broader context in which it is used determines which referring expressions are applied. By doing this, the models learn how to relate referring expressions against a representation of the environment. More specifically, the model learns to connect symbols to a non-symbolic representation. This is known as the symbol grounding problem (Harnad, 1999; Roy, 2002). Hereby, the question is asked how arbitrary symbols can be connected to the representation they actually stand for.

To solve this problem, multiple challenges arise since language describes the world in a very under-specified way. For instance, sensory input for humans as well as for machines only show a very limited representation of the world. Vision needs to map a 3-d world into 2-d images. During this process, information about the world gets lost, as for example depth can't be perceived easily or objects might be obfuscated in certain perspectives. Following, language needs to be connected to a representation of the world that is already not complete. Secondly, referring expressions in language themselves are often under-specified. Given for example a table with five cups on top, an interlocutor is asked to retrieve one of the cups, referring to it by 'the red cup'. This referring expression on its own might be ambiguous and refer to multiple red cups on the table. The challenge now arises as how the interlocutor associates the correct cup with this referring expressions. This gets even more complex, when perspective is part of the referring expression as for example in 'the cup on the left' where *left* is dependent on the speaker's and interlocutors' spatial position

in relation to the cup (Dobnik & Silfversparre, 2021).

The learning of referring expressions is therefore split in two fields. In the referring expression generation, models learn how to produce referring expressions, when presented with visual input. In the referring expression understanding, sometimes referred to as coreference, models are trained on interpreting referring expressions and link them to visual input.

A major challenge in natural language processing is, how machines can

2.3 Language Games

[Dominik Künkele] language games combine generation and understanding

[Dominik Künkele] describe as 'dialogue turns'

[Dominik Künkele] different approaches (robotics, neural models, ...)

[Dominik Künkele] different study interests (linguistically, non-linguistically)

[Dominik Künkele] continuous learning

The center of this thesis evolves around language games between deep neural models. Hereby, multiple deep neural agents need to solve a task, by communicating with symbols, which initially are not associated with any meaning. By using and interpreting these symbols in there communication, the agents start to give these symbols and their combinations meaning. After this process, a new artificial language emerged, with which the agents can communicate with each other.

Section 2.3.1 explains reasons why research in this field is conducted and how it may help in further research. Section 2.3.2 shows in more detail how the language games are set up and how exactly agents can communicate and the language emerges. Section 2.3.3 discusses, how the emerged languages can be built up. Finally, section 2.3.4 discusses how the language games will be used in this thesis, to explore how agents can refer to objects in images.

2.3.1 Aims of emerged languages

The setup of language games allows for very controlled rules of how agents can behave.

2.3.2 Setup of language games

The term of 'language games' was first introduced by Wittgenstein (1953). The author describes language games as uses of language between multiple interlocutors. These can be any small parts of conversations, for instance between a teacher and a student teaching a new concept or between two persons, discussing a topic. The rules in each of these situations differ and therefore the meanings and semantics of words and sentences may differ from situation to situation. An interjection 'Water!' may be a warning, an answer to a question, a request or something else, depending on the context. meanings are bound to the rules of the language games.

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). In this game, a sender needs to send a message to a receiver, based on information only available to the sender. Based on the message, the receiver proposes an action. 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 entities can be set into real or simulated interactive situation, a language game, that meaning and a language can emerge (Kirby, 2002). Multiple deep neural models, called agents, communicate with a set of symbols to solve a task. Guided by the task and the rules, how agents can generate and interpret symbols, a language can emerge.

2.3.3 Properties of the emerged language

2.3.4 Referring to objects and grounding

2.4 Artificial dataset

[Dominik Künkele] 'world shapes language'

[Dominik Künkele] different types of biases -> reduce and control

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 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 different effects of varying contexts.

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 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 position of the light and the camera are slightly jittered for each image to add noise reduce recurring patterns. Since the objects are part of 3-d scenes, they may appear differently for 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 to learn for models compared to relatively noise-free projections on a 2-d plane. Figure 1 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

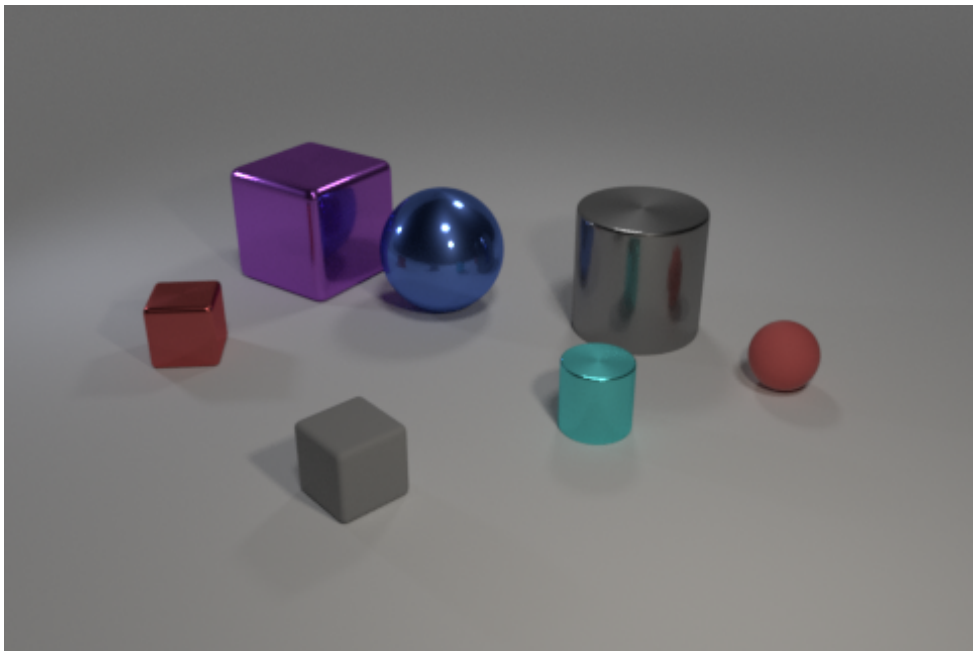


Figure 1: Example of a generated image in the CLEVR dataset

the objects in relation to the position of the camera.

3.2 Feature extractors

[Dominik Künkele] preprocessing

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, two different architectures are tested.

[Dominik Künkele] explain advantage of CNN

First, we use the VGG19 (Simonyan & Zisserman, 2015) which is an architecture based on many convolutional layers. Using 16-19 convolutional layers with small convolution filters helps the model to solve localization and classification tasks on the training dataset, but also enables is to generalize onto other datasets. After the convolutional layers, the data is passed first through an average pooling layer which outputs 512x7x7 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.

Secondly, we include the 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 256x56x56, 512x28x28, 1024x14x14 and 2048x1x1 dimensions. A following average pooling layer outputs 2048x1x1 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] 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 compared. Table 2 lists the different adaptions for both VGG19 and ResNet-101 that will be used in this research. In the later chapters, it will be referred to these adaptions using the name in the table.

	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

[Dominik Künkele] masking and how are they processed (convolutions to bring it to the same shape)

The above described feature extractors 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. In this thesis, we use one architecture to encode images for all the conducted experiment. The additional layers on top of the

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

feature extractors however are always trained from scratch for each of the experiments and tasks. By doing this, the models are made up of multiple smaller modules, as for example the image processing module or other modules that encode different inputs. These modules can be added and combined step by step, whilst the architecture of each module always stays the same. Following, the experiments stay comparable, since differences are only added or removed modules.

Johnson et al. (2017b) describe an architecture that was used for training baseline models on the CLEVR dataset. This architecture will be used for all experiments. 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 encoding size of 2048 dimensions.

layer	**output dimensions**
Input image	$3 \times 224 \times 224$
ResNet-3	$1024 \times 14 \times 14$
Conv($3 \times 3, \rightarrow 128$)	$128 \times 14 \times 14$
ReLU	$128 \times 14 \times 14$
Conv($3 \times 3, \rightarrow 128$)	$128 \times 14 \times 14$
ReLU	$128 \times 14 \times 14$
MaxPool(2×2, stride 2)	$128 \times 7 \times 7$
Flatten	6272
Linear Layer($\rightarrow$ 2048)	2048

Table 3: Image encoder with ResNet-3 and an encoding size of 2048

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. 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 as well as what is their output. The second level are *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 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 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

The traditional approach to solve problems with a discrete channel relies on the REINFORCE algorithm (Williams, 1992). It uses the concept of Monte Carlo sampling to estimate the gradient of the expected cumulative reward with respect to the policy parameters. The basic idea is to collect trajectories by following the policy, compute the cumulative rewards for each trajectory, and then update the policy parameters to increase the probabilities of actions that led to higher rewards. REINFORCE suffers from high variance in gradient estimation due to the inherent randomness in the environment and the policy.

Gumbel-Softmax is a relaxation of the discrete categorical distribution to a continuous distribution that can be differentiated (Jang et al., 2017). It allows for the application of gradient-based optimization methods to discrete optimization problems. In the Gumbel-Softmax approach, randomness is introduced using the continuous Gumbel distribution. Following, the *softmax* operation creates a distribution over discrete actions that can be differentiated. Gumbel-Softmax 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.

3.6 Probing?

[Dominik Künkele] reason for usage and techniques fits more into theoretical background

3.7 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 uses 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.

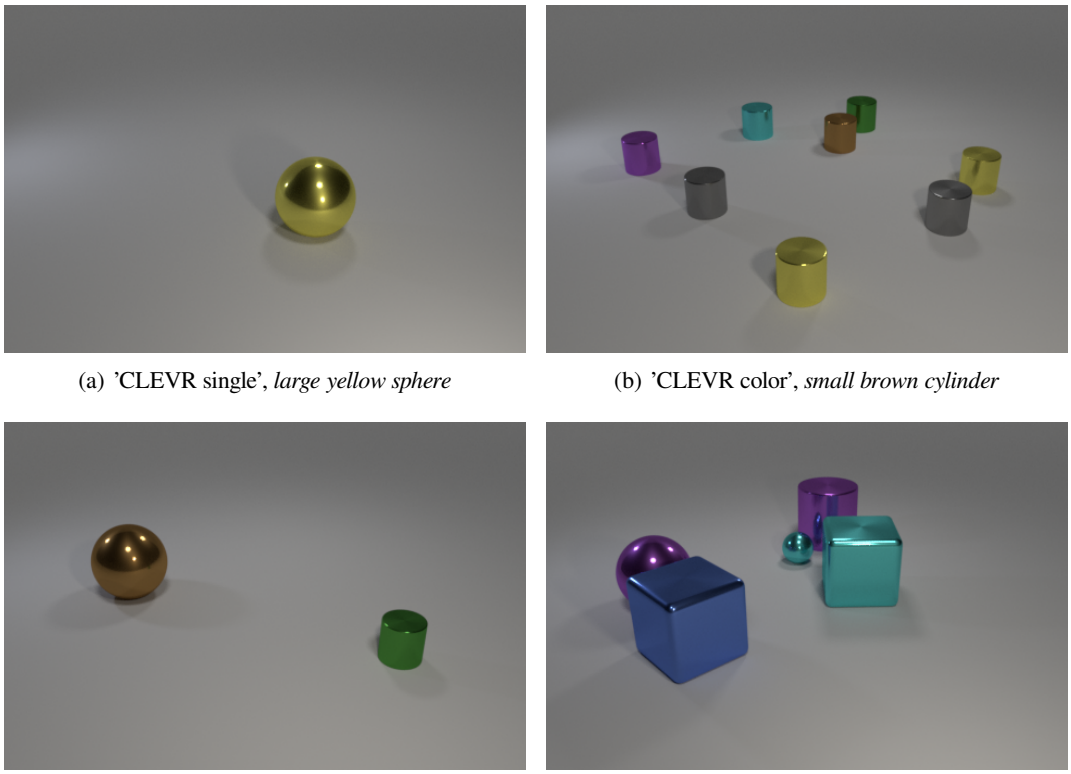
Lastly, it is also possible to define exactly **which** attributes should be shared between the target object and the groups. For example, it can be defined to have the same size for objects in the target group, but have different, randomly selected shapes and colors. This allows for a very controlled generation of relations between the objects in the scene. Figure 2(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 2: 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 2(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 2(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 2(c)), while the Dale-5 dataset contains one target object and exactly four distractors. Consider Figure 2(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.

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.

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

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, described in section 3.2 they need to be normalized first. As described in (He et al., 2016; Simonyan & Zisserman, 2015), the bounding boxes are resized to 256 pixels for the shorter side, then cropped around the center to a square of 224 $\times$ 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. 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.

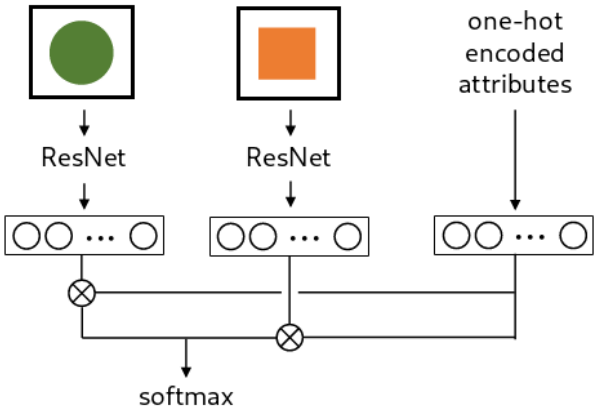


Figure 3: Simplified architecture of the object identification task

The model, the *object identifier* is split into two parts. First, each normalized bounding box of the sample is passed through *ResNet-avg*. The preceding layers *ResNet-1* to *ResNet-4* couldn't be compared due to memory restrictions on the GPU. A deeper analysis of the effects of different feature extractors will be done in the following chapters. 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. In the second part, 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}, a batch size of 32 samples and 30 epochs, *Adam* (Kingma & Ba, 2015) as optimizer. The number of epochs was

chosen manually after identifying when the loss as well as the test accuracies didn't improve significantly. 8000 randomly selected samples are used for training, the remaining 2000 samples for testing. 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 **??**. 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. 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 passed through *ResNet-avg* and projected afterwards to an image embedding dimension e with a linear layer. All thus 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 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.

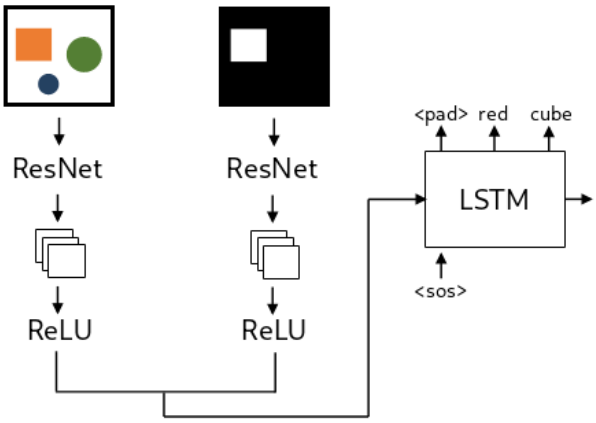


Figure 4: Simplified architecture of the masked caption generator

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 passed through *ResNet-3* and subsequently processed by additional convolution layers, described in section 3.3. The resulting vector is then reduced to decoder output dimension $LSTM_o$ by a linear layer and as for the bounding boxes acts as the initial hidden state for the LSTM. The LSTM is trained in the same way as above.

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*. A masked version of the image is created and passed to the model. For this, a fixed size squared area of 96×96 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 convolutional layers as described in section 3.3. The result is then again projected to the decoder output dimension d by a linear layer and then used as the initial hidden state for the LSTM. 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}, a batch size of 32 samples and 30 epochs, *Adam* (Kingma & Ba, 2015) as optimizer. 8000 randomly selected samples are used for training, the remaining 2000 samples for testing. 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 caption 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 caption.

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 2(d). The correct caption is *cylinder*. If the model would predict *purple cylinder*, the accuracy determines it as false as captioning *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 captions. For this we generated captions for all the non-target objects and distractors in the images using the GRE-algorithm. 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 caption 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. Furthermore, the model needs to attend to many more locations in the image at the same time to find discriminating factors between those.

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 caption 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.

[Dominik Künkele] why?

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.

		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.

ResNet-3 produced the best results, when it was used as a feature extractor. The results are shown after 50 epochs, but the scores already start to converge after 20 epochs and changed only very little in the following epochs.

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 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.

		Dale-2			Dale-5			CLEVR color		
$LSTM_o$	$LSTM_e$	**Acc.**	**F1**	**NT**	**Acc.**	**F1**	**NT**	**Acc.**	**F1**	**NT**
100	10	81%	75,95%	9%	21%	57%	23%	16%	37%	31%
100	15	78%	71,99%	11%	17%	52,98%	25%	26%	43,49%	30%
100	30	83%	78,99%	7%	19%	54,46%	24%	29%	45,49%	26%
500	10	83%	79,95%	6%	27%	61,98%	16%	17%	38,49%	32%
500	15	83%	79,5%	7%	28%	63,5%	16%	21%	40%	31%
500	30	84%	83,43%	5%	30%	63,98%	17%	21%	40%	31%
1000	10	84%	82,42%	5%	31%	65,5%	13%	12%	32,12%	34%
1000	15	85%	82,42%	5%	31%	64,5%	16%	16%	37%	32%
1000	30	85%	82,81%	4%	31%	63,98%	17%	13%	36%	34%
1500	10	84%	81,42%	4%	33%	67,47%	14%	12%	33,43%	34%
1500	15	85%	84,43%	4%	35%	68,99%	12%	17%	38,49%	31%
1500	30	84%	80,95%	4%	36%	68,99%	15%	14%	36,97%	33%
2000	10	83%	82,89%	4%	36%	69%	13%	13%	36%	34%
2000	15	84%	81,89%	4%	34%	67%	14%	14%	33,31%	33%
2000	30	85%	82,95%	4%	32%	65,5%	16%	18%	39%	32%
3000	10	82%	79,89%	4%	34%	68%	12%	12%	32,31%	34%
3000	15	85%	81,89%	4%	32%	65,5%	15%	13%	32,73%	35%
3000	30	83%	80,89%	3%	30%	63,5%	16%	12%	32,73%	34%

Table 9: Overall accuracies (Acc.), F1-Score (F1) and non-target accuracies (NT) of the masked caption generator: $LSTM_o$ are different LSTM output sizes and $LSTM_e$ are different embedding sizes for the tokens in the LSTM.

The sum of the two columns for the *accuracy* and the *non-target-accuracy* adds to 97%, when trained on the 'Dale-2' dataset. This means that the models produce almost always correct descriptions of one of the objects in the image. Moreover, these descriptions are also efficient in the sense that they follow the GRE-algorithm of Dale & Reiter (1995) and only use necessary discriminative attributes. Features of both objects are therefore extracted perfectly and associated with the vocabulary. As expected, the model has difficulties to decide, which of both objects is the target object, since no information is passed to the model. With 48% of the images describing the target object and 49% of the images describing the distractor, the model uses a random guess. This changes, when also the masked image is presented to the model. Then, the accuracy for the target object rises with 72% well above chance. Still, the wrong descriptions describe almost every time the distractor object. This indicates that the problem doesn't lie in extracting features of objects, but only in deciding which object in the image to describe.

When the model is trained on the 'Dale-5' dataset, the results are much worse. Now, without masking only in 61% of the images, the model describes any of the objects in the image, whether it is the target object or a distractor. With 21% of these descriptions being a description of the target object, the model again uses a random guess. Interestingly, passing the masked image to the model doesn't help it to identify the correct object. The accuracy stays at the same value. In contrast, the non-target accuracy is increased by 5% points. The model is more likely to identify a wrong object. Furthermore, the word-by-word accuracy increases from 45% to 54%. This increase is likely due to the fact that the description of the target object often shares some attributes with distractor objects. For instance a sample with a target object being a *small red cube* was identified as a *<pad> <pad> cube* when the model didn't receive the masked image. When also the masked image is passed to the model, it generates the description *small blue cube*. This was the correct description of a distractor and because of the overlap of the attributes, the word-by-word accuracy increased from 33% to 66%.

There are two main possible explanations for the big difference between the two datasets. First, as already seen in the experiments before, the 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 give the model a bigger choice. The second explanation lies in the used GRE-algorithm. 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 caption 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 are 19,8%, 47% and 33,2% respectively. With four distractors the algorithm is much more likely to produce longer captions. 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.

5.3 Reference resolution

Setup

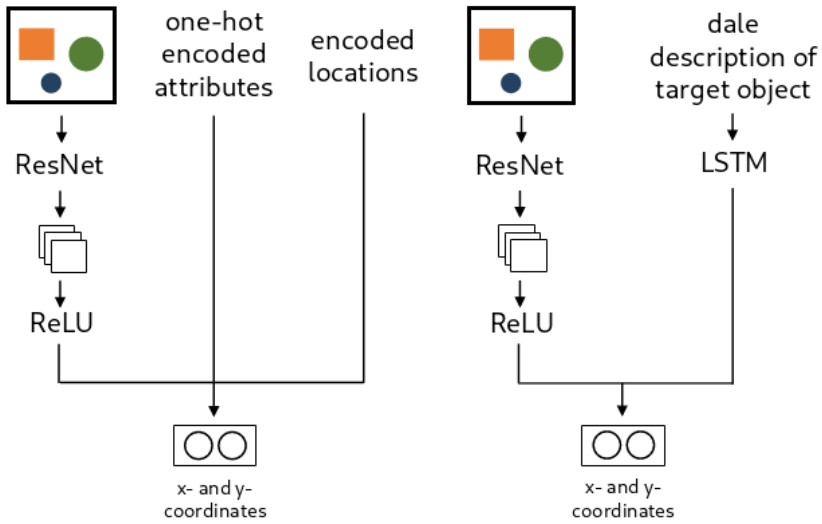
[Dominik Künkele] attribute encodings, coordinate predictor

With the reference resolution task, the model is trained to understand referring expressions, by pointing towards the target object in the scene. 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. The models are therefore tasked to predict the center coordinates of the target object. With this approach, the models receive few natural language information, but are still able to rely on all information present in the image to discriminate the objects.

To achieve this goal, multiple setups of models are trained. In the simplest setup, the *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. In a first step, the image is encoded. This is done by first passing it through one of the *feature extractors* and then further processing it with convolutional layers as described in section 3.3. Finally, it is projected to the image embedding size e_i. In the next step the encoded image is used to predict coordinates using the *coordinate predictor*. Hereby, the extracted feature vector 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 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.



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

(b) Model including GRE description

Figure 5: Simplified architecture of the coordinate predictors

In the first method, the target object is 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. This should help the model to identify all possible options to chose from, when predicting the target object. All the center coordinates are simply extracted and shuffled. 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. For 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 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 description of the target object using the described algorithm. To include it in the model, the captions 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 captions are padded at the end with a specified padding token.

In the model, the referring expression is encoded, using an LSTM. 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 caption. 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 knowledge by explicitly

encoding human chosen attributes, masking the image 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 2×10^{-4}, a batch size of 32 samples and 30 epochs, *Adam* (Kingma & Ba, 2015) as optimizer. 8000 randomly selected samples are used for training, the remaining 2000 samples for testing. The loss is calculated using cross entropy. Table 10 shows the variables that are changed during the experiments for each of the models.

	e_i $[100, 500, 1000]$	c $[512, 1024, 2048]$	$LSTM_e$ $[15, 30]$	$LSTM_o$ $[500, 1000, 1500, 2000]$
reference resolver	×	×	-	-
+ one-hot	×	×	-	-
+ one-hot + locations	×	×	-	-
+ referring expression	×	×	×	×
+ masking	×	×	-	-

Table 10: 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.

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 either VGG or ResNet, locational information about the image could have gone lost. 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 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. Latter task is assumed to be significantly harder.

Results

Figure 6 shows the results of the coordinate predictor that doesn't include any information about the target object. The used feature extractor for these results is *Resnet-3*, but the results don't differ meaningfully

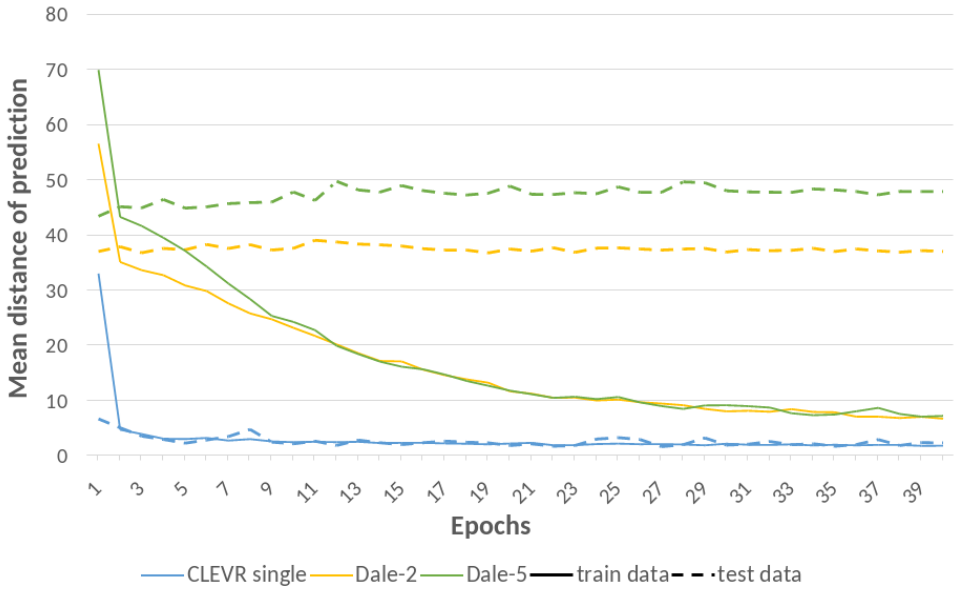


Figure 6: Mean distance between predicted coordinates and ground truth in pixels on different datasets

from results with other feature extractors. As can be seen, the success between the different datasets are significant. The more objects are present in an image, the worse the model performs. The model converges for the CLEVR single dataset after around 20 epochs to a mean distance of around 2 pixels. This prediction even though not perfectly on the center point is always on the object. Opposed to that, using the Dale-2 dataset with two objects, the mean distance lies between 37 and 38 pixels already after the first epoch and doesn't drop with increasing number of epochs. With five objects in the Dale-5 dataset, the model only predicts a mean distance of around 45 pixels in the beginning, which worsens with a rising number of epochs to 48 pixels.

An interesting observation is the difference of the mean distances between training and testing data. The training distance is constantly approaching zero, while the testing loss is staying constant or even getting higher. This points to the fact that the model is not generalizing the task by learning abstract patterns that can be applied to unseen data, but is instead memorizing the training data. That is especially visible for the Dale-5 dataset, where learning the patterns of the training data looses even the ability to interpret some patterns in the testing data. Applying a higher dropout didn't have an impact on the results.

This behavior is indeed not very surprising. First, results with the 'CLEVR single' dataset show that the model is able to derive geometrical information from abstract feature, extracted by a feature extractor. Geometrical information therefore doesn't get 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. Secondly, more than one objects present in an image confuses the model, and it is not able to consistently point to one of them. This can have multiple reasons, for instance that the models lacks the ability to separate objects in the extracted features. But even in case, the model is able to do that and could determine the location of each object in the scene given the feature, it would not be able to tell, which of these is the actual target object. The guess is then more or less random. This especially applies to the Dale-2 dataset, where an identification of the target object just based on one distractor is impossible; both of the present objects are unique. For the Dale-5 dataset, the model could in theory learn that the target object is always the one object which is unique in respect to its and the distractors' attributes. This task on the other hand seems very difficult to learn. In conclusion, the models are able to predict geometrical coordinates, but need more information about the target object to identify it.

When the target object **attributes are encoded as one-hot vectors** and added to the input, the results don't improve. One factor that now has a much higher impact is the feature extractor that is used. Table 11

compares the mean distances the models predict for the different feature extractors. The results are shown for the models trained on the Dale-2 and Dale-5 datasets for training and test data. First, a big difference can be seen between the datasets. The models only converge to a minimum mean distance of around 46 pixels to the correct coordinates for the Dale-5 dataset, looking at the test data. In most cases, it stays above 50 pixels. Using the Dale-2 dataset, the behavior is a little different. All ResNet extractors with four residual blocks and an additional average pooling and optionally a classifier layers reach similar scores as the experiments before without any one-hot encodings. Interestingly, without the classifier layer, the model doesn't converge at all and the mean distances jump up and down between the epochs. This effect also applies when using less residual blocks. Using the VGG, only VGG-cls2 achieve a similar performance, while the others predict coordinates between 43 and 46 pixels away.

	Dale-2		**Dale-5**	
	train	test	train	test
VGG-0	30,27	46,20	**32,17**	54,40
VGG-avg	**29,99**	45,08	32,32	52,67
VGG-cls1	37,99	43,28	46,57	50,75
VGG-cls2	38,87	**39,02**	47,48	49,91
VGG-cls3	39,99	44,32	47,26	**46,77**
ResNet-3	78,26	65,23	92,07	91,12
ResNet-4	44,14	55,24	**36,48**	58,28
ResNet-avg	**33,06**	**39,18**	47,64	46,38
ResNet-cls	37,57	38,10	44,72	**45,92**

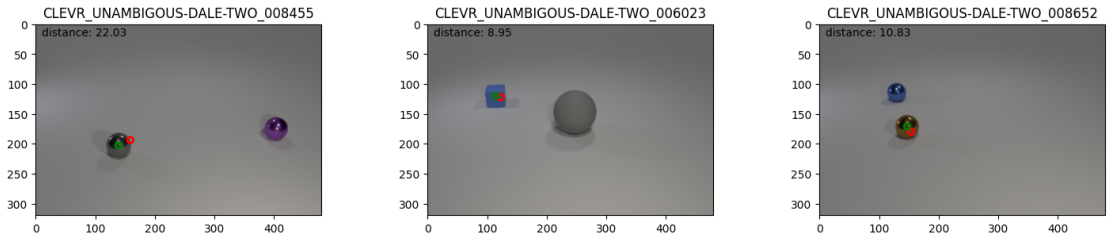
Table 11: Mean test losses for different feature extractors with one-hot attribute encodings after 20 epochs

Secondly, the training loss now looks also different. In almost no cases, the models converge to a lower mean distance than with the test data, meaning a higher precision in their predictions, as they did in the experiment before. The only exception is ResNet-3 as a feature extractor. In other words, the models are again not able to generalize, but in specific cases memorize the patterns in the train data. This hints to the fact that only specific layers of the feature extractors contain information that is generally usable to identify and discriminate objects. Especially the lower layers with fewer residual blocks in the case of ResNet and no classifier layers for the VGG seem to not encode knowledge that can be utilized for this task. Higher layers, with more specific encoded information need to be used for this research. The experiments in the following sections are set up using these higher layers.

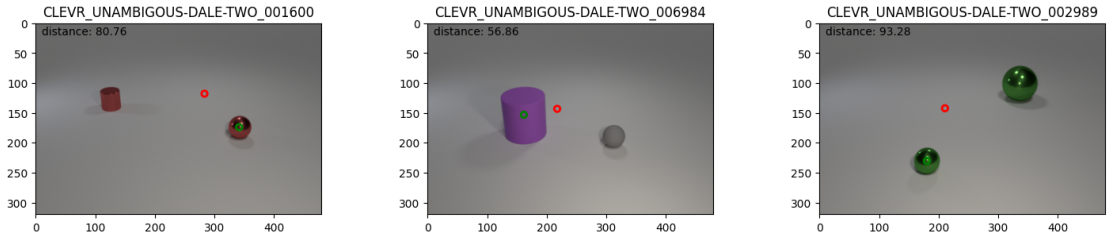
Adding **information about the center coordinates** of all objects should have helped the models to get a list of possible predictions. In theory, the model could learn to choose between these coordinates by relating them to the extracted features of the image. This hypothesis doesn't hold. All results for both datasets Dale-2 and Dale-5 are the exact same as without included information about the locations. The problem therefore doesn't seem to lie in predicting coordinates in general, but predicting the coordinates of the target object. The model is still not able to understand, which object is the target object. For that reason, a better representation of the target object is necessary.

In a next step, information about the attributes is included using the ***GRE-algorithm*** from Dale & Reiter (1995). Again, the mean distance of the predictions as well as the accuracy doesn't improve compared to the previous experiments.

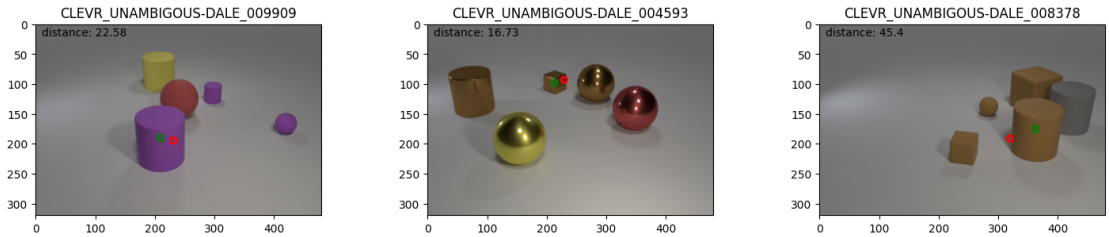
An interesting pattern appears when doing a qualitative analysis of the models' predictions. Here, we visualized the predicted coordinates compared to the ground truth coordinates. Figure 7(a) shows random examples of predictions for images in the train dataset of Dale-2. The green circle shows the ground truth



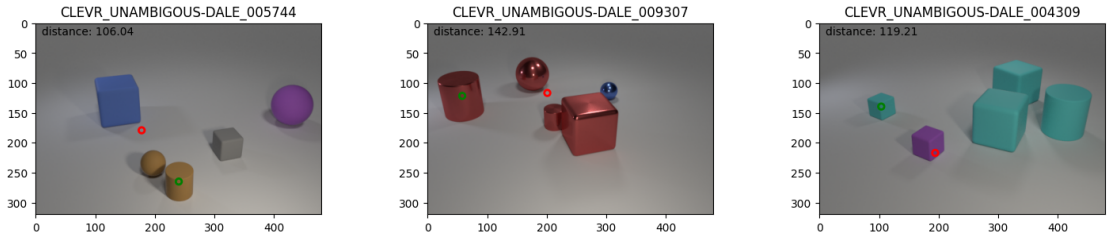
(a) 'Dale-2', train split



(b) 'Dale-2', test split



(c) 'Dale-5', train split



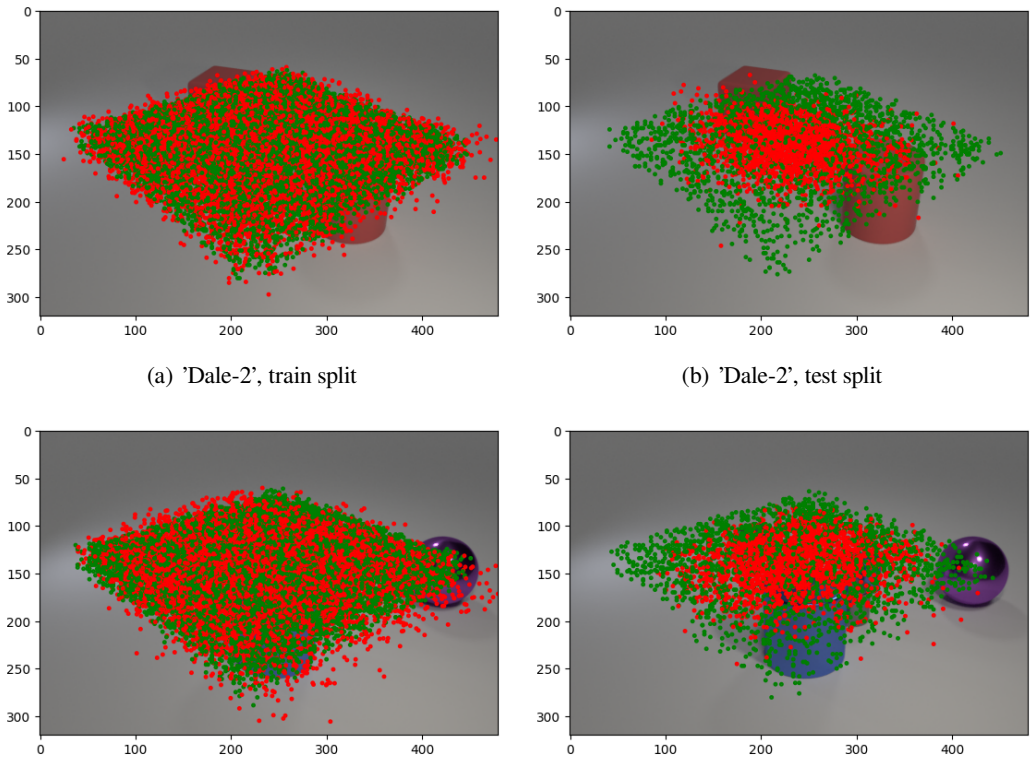
(d) 'Dale-5', test split

Figure 7: Visualization of the models' predictions in the 'Dale' datasets

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 8(a) 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. The resulting shape is a rhombus, which reflects that all objects are placed usually central into the scene. As expected the green and red rhombus align mostly in the same area for the train split of the 'Dale-2' dataset.

The results look very different for the test split. As can be seen in Figure 7(b), the three randomly selected predictions don't align with the ground truth coordinates. For all the images, the predictions don't lie on any object. In the left image as well as in the central image, the predictions are closer to the target object than towards the distractor, but are still quite imprecise. These findings align with the mean distance scores, described in the sections before. However, it seems that the model's predictions are all towards the center of the image. This can be seen clearer in Figure 8(b). 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. This behavior can be observed for all datasets and architectures of the model. Figures 7(c), 7(d), 8(c) and 8(d) show the results for the 'Dale-5' dataset. Here, the model more likely predicts the center coordinates of a distractor object as seen in the right image, which is also reflected in the lower score of the mean distance. Also the combined visualization shows the same clustering of predictions in the center of the scene, but the pattern of the smaller rhombus is more visible.



(a) 'Dale-2', train split

(b) 'Dale-2', test split

(c) 'Dale-5', train split

(d) 'Dale-5', test split

Figure 8: Visualization of the models' predictions in the Dale datasets

These results allow two conclusions. First, the models are biased to predict coordinates in the center of the image. 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.

Second, 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 'Dale-5' dataset, especially in left and central image in Figure 7(d). 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. Concluding, the model is able to extract, where objects are located in the image, but can't make use of the referring expressions, to decide which of these objects is the target object.

6 Grounding referring expressions in language games

[Dominik Künkele] 'epochs', training/testing (success), optimizer, ...

[Dominik Künkele] max message length

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.

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. The number of possible words, namely the size of the vocabulary can be varied.

First, we will discuss *discrimination games*, because they have the simplest setup. Furthermore, other language games that research this topic use a very similar setup. In the next step, we will describe *caption generators* that are set up as a language game. Here, the sender describes the scene, while the receiver needs to generate a caption. In the last step, we try to lose as much human bias as possible and the models are trained on just 'pointing' towards the target object, by again predicting its center coordinates.

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.

[Dominik Künkele] discuss training/epochs/batches, vocab size, message length, gs temperature

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 game was the identification of the concept that and 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.

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. 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 the 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 bounding box of the target object is passed first, the remaining are shuffled.

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 one of the bounding boxes. Opposed to the sender, the receiver is shown all bounding boxes in shuffled order. Figure 9 shows, how the sender and the receiver of the discriminator are built up.

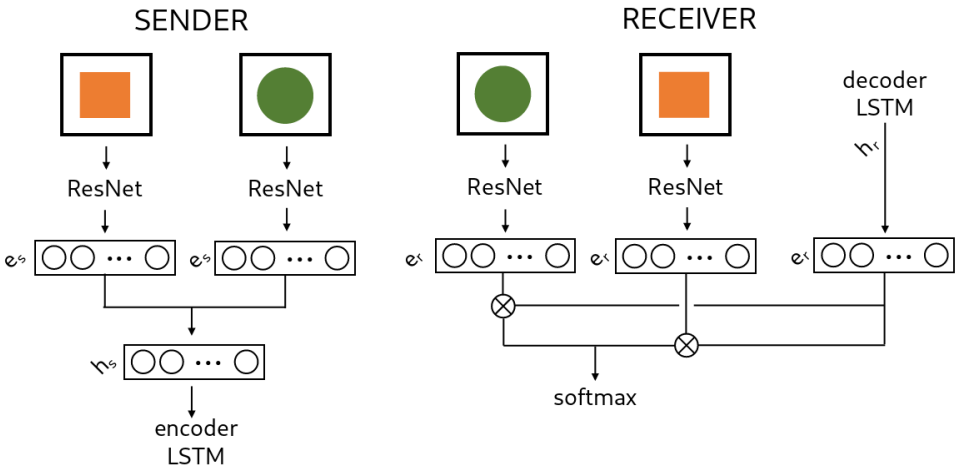


Figure 9: Sender and receiver architectures in the discrimination game

The experiments are conducted with the following hyperparameters: a learning rate of 2×10^{-4}, a temperature for the Gumbel-Softmax relaxation of 1 and *Adam* (Kingma & Ba, 2015) as optimizer. The loss is calculated using cross entropy. The following values for the variables are compared:

- h_s: 100, 500, 1000
- h_r: 10, 30, 50, 100, 500, 1000
- $|V|$: 1, 10, 16, 50, 100
- n: 1, 2, 3, 4, 5, 6

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

Results

Table 12 shows the accuracy of the models calculated on the success of communication if the receiver can identify the target object. A random guess corresponds to 50% in the *Dale-2* dataset and 20% in the *Dale-5* dataset. Four different vocabulary sizes $|V|$ are tested. A size of 13 symbols corresponds to the 13 attributes the objects can have and align with human language. If the symbols were similarly used, messages would have lengths between one and three symbols. Opposed to that a slightly smaller vocabulary with 10 symbols is used to create a smaller bottleneck with a higher pressure to condense the information. Similarly, a bigger vocabulary consisting of 20 symbols tests how a bigger bottleneck changes the results. Lastly, a big vocabulary of 100 symbols should give the model all options to encode the information, including one symbol per attribute or one symbol describing a combination of attributes.

The hidden sizes h_s and h_r as well as the embedding sizes e_s and e_r are chosen in alignment with the vocabulary size. The hidden sizes are always smaller or equal to the vocabulary size since the information about each word needs to be compressed in a smaller dimension to learn meaning. Hereby, hidden sizes of 10 and 100 are tested. On the other hand, three different embedding sizes are tested: 10, 50 and 100. The reason for this is to experiment, what is the optimal middle ground between compressing features of an image encoded in high dimension vectors and upscaling encoded messages in low dimension vectors.

					Dale-2		**Dale-5**		**CLEVR color**	
$\|V\|$	h_s	h_r	e_s	e_r	**Accuracy**	**length**	**Accuracy**	**length**	**Accuracy**	**length**
10	10	10	10	10	96,4%	0,99	24,7%	0	17,3%	1
10	50	50	50	50	50%	1	21,4%	1	17,8%	0
13	10	10	10	10	96,16%	1	24,8%	1	17,1%	1
13	10	10	50	50	49,6%	1	21,9%	1	17,9%	0
20	10	10	50	50	50,9%	0	23%	1	15,9%	1
100	10	10	10	10	97,3%	1	24%	1	18,1%	1
100	10	10	50	50	49,9%	1	24,4%	1	15,8%	1
100	100	100	100	100	49%	0	25,3%	1	15,6%	0

Table 12: Results of the discriminators: $|V|$ are different vocabulary sizes, h hidden sizes and e embedding sizes.

For the *Dale-2*, a clear correlation between the hidden sizes, embedding sizes and the size of the vocabulary can be identified. A hidden/embedding size as high as the vocabulary size is beneficial for identifying the correct object. The receiver identifies almost every sample correctly when all sizes are 10. When the hidden and embedding sizes are increased, the guesses by the receiver are random with 50% accuracy. Interestingly, a vocabulary size of 10 is enough to communicate a meaningful message when the model is trained on the *Dale-2* dataset.

The results change, when using the *Dale-5* dataset with four distractors. With four distractors and with low hidden, embedding and vocabulary sizes, the agents barely pass the random baseline with 23%. Only increasing the vocabulary size to 100 raises the accuracy by almost 20% points to 43%. This is still considerably lower than the 95% of the *Dale-2* dataset. The same applies to the 'CLEVR color' dataset, where all models achieve a very low accuracy of around 15 to 17%, corresponding to random guesses.

Two conclusions can be drawn. First, the hidden as well as the embedding sizes need to be close to the vocabulary size. This even applies for very low vocabulary sizes, which means that the image encodings need to be compressed to the same low dimensions. The reason for this is very likely that neural models have difficulties to upscale from lower dimensions (e.g. from low h_r to high e_r) as opposed to learn how to extract the important information from a vector with many dimensions.

The second conclusion that can be drawn looks at the differences between the two datasets. Unsurprisingly, the agents have a much higher difficulty to discriminate a target object from four instead of one distractor. Since we discriminate objects based on properties that are also distinguished in human cognition (8 colors, 2 sizes, 3 shapes), we expect that the vocabulary onto which the agents converge reflects these categories and is therefore close to human vocabulary. There are $8 * 2 * 3 = 48$ possible combinations of attributes. Still, for Dale-2, a vocabulary size of only 10 is enough for an almost perfect accuracy with two objects. This hints to the fact that the agents don't describe the complete target object, but only rely on discriminative attributes between the objects. The need for a more detailed description of discriminative attributes is higher when more distractors are involved. Therefore, the models need to learn more combinations of symbols in order to attest to this higher level of detail and especially how to relate them to features in the images.

[Dominik Künkele] similarity to bounding box classifier

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 receiver is tasked to describe the target object in natural language, while the sender needs to communicate, which object is the target. Again, the setup is asymmetrical: the sender receives the image and information, which of the objects in the image is the target object in form of a masked image. The receiver only sees the image without additional information.

[Dominik Künkele] human bias vs emergent language

For this experiment, the data is prepared in the same way as described in section 5.2. Since the position of the padding as well as the order of the words 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. Respectively, both agents are based on the models in these experiments. The sender uses the *masked RE generator* with its LSTM replaced with the LSTM included in the *EGG* framework, using the hidden size h_s.

[Dominik Künkele] fixed parameters

The receiver is built up similarly. The image is encoded in the same way, but afterwards it is concatenated with the decoded message, received be the sender with the hidden size h_r. The concatenated is reduced to $d =$ dimensions, based on the results of the previous experiments, and used as the initial state of the captioning LSTM.

[Dominik Künkele] fixed parameters

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 experiments are conducted with the following hyperparameters: a learning rate of 2×10^{-4}, a temperature for the Gumbel-Softmax relaxation of 1 and *Adam* (Kingma & Ba, 2015) as optimizer. The loss is calculated using cross entropy. The following values for the variables are compared:

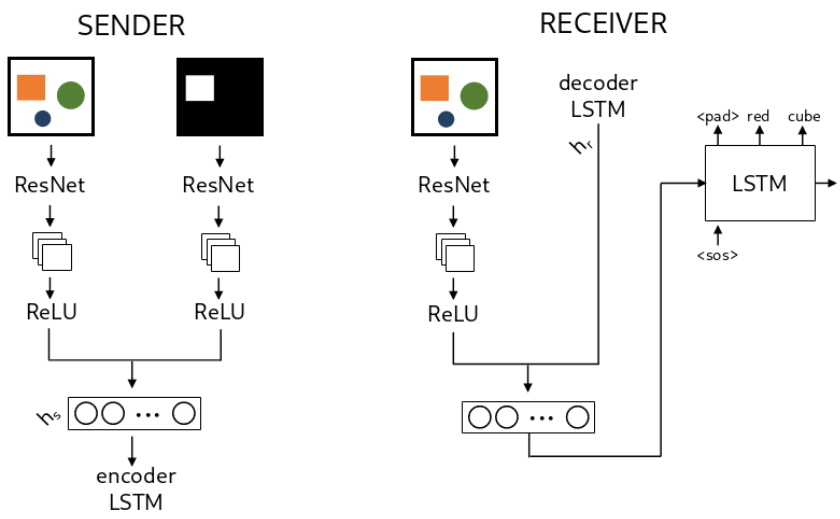


Figure 10: Simplified architecture of the caption generator game

- h_s: 100, 500, 1000
- h_r: 10, 30, 50, 100, 500, 1000
- $|V|$: 1, 10, 16, 50, 100
- n: 1, 2, 3, 4, 5, 6

[Dominik Künkele] real values

Since the agents are trained to describe the target object discriminatively based on the described GRE-algorithm, they are trained on the 'Dale-2' and 'Dale-5' dataset. The 'Dale-5' 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.

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 13: 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 13. 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. The agents are tasked to predict the center coordinates of the target object, after being presented with the complete scene. However, the agents, more specifically the sender needs to be pointed towards the target object. In the final setup, the *masked reference resolution game*, this is done providing the sender with an additional masked visual input. However, this can prove to be a challenging task for the agents. For this reason, in an approach to reduce the complexity, for the *RE reference resolution game* the sender is shown a human referring expression, generated with the incremental GRE algorithm instead of the additional masked image. This might lead to a heavy bias of the emerged language towards the human referring expressions and will be compared to the unbiased emerged language.

The masked sender is the same as the *RE generation sender*, described in section 6.2. The RE sender works similarly, but instead encodes the human referring expression with an LSTM. The final hidden state is concatenated with the encoded image. The resulting vector is reduced to the hidden size h_s and used as the initial hidden state of the LSTM included in the *EGG* framework to generate the message.

[Dominik Künkele] fixed variables

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 to encode the message. The dimensions of the coordinate predictor are fixed to $c =$, based on the previous results.

[Dominik Künkele] fixed variables

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.

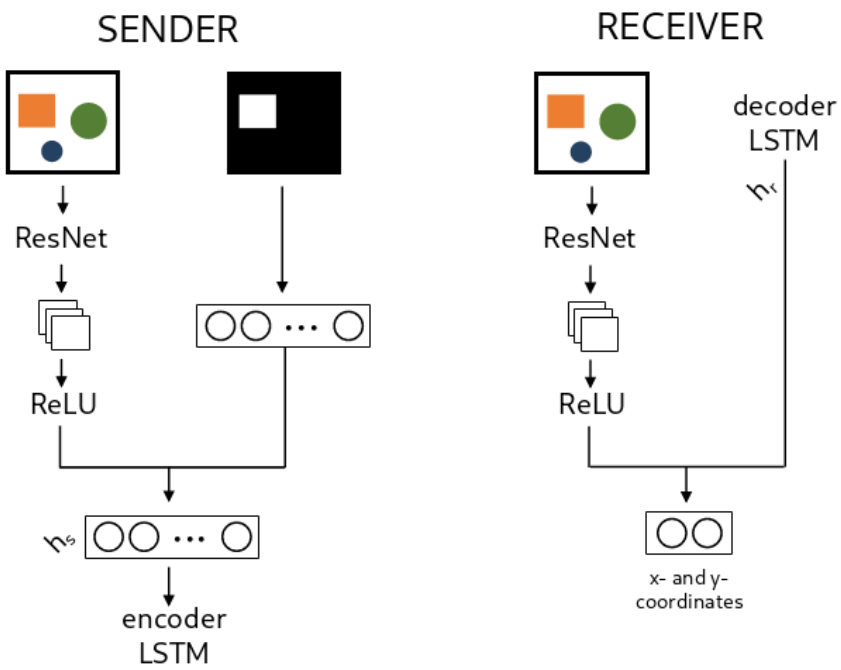


Figure 11: Simplified architecture of the masked coordinate predictor game

The experiments are conducted with the following hyperparameters: a learning rate of 2×10^{-4}, a temperature for the Gumbel-Softmax relaxation of 1 and *Adam* (Kingma & Ba, 2015) as optimizer. The loss is calculated using cross entropy. The following values for the variables are compared:

- h_s: 100, 500, 1000
- h_r: 10, 30, 50, 100, 500, 1000
- $|V|$: 1, 10, 16, 50, 100
- n: 1, 2, 3, 4, 5, 6

[Dominik Künkele] real values

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' and 'Dale-5' 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.

Results

In the final setup, the agents are tasked with communicating objects with fewer infused human knowledge. Table 14 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 14: 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.

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.

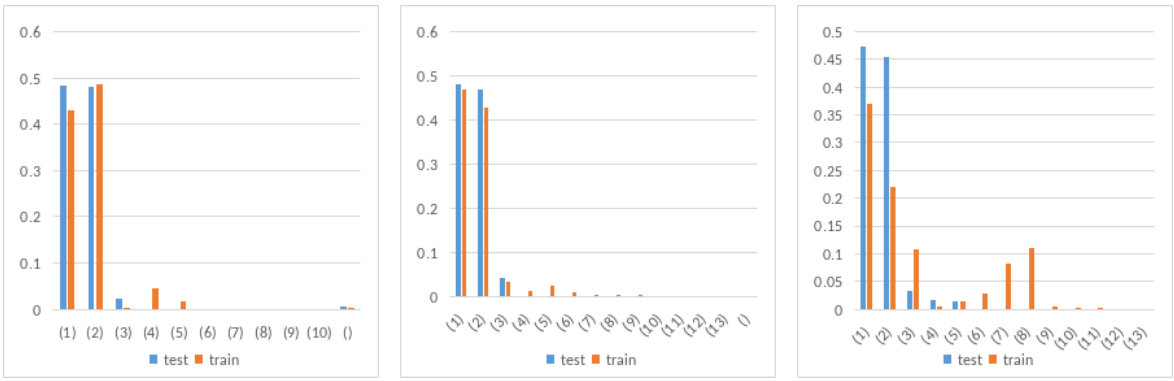
When comparing these results to the neural models in Section 5.3 that are not part of a language game, it can be seen that the metrics are very similar for all datasets. The model was already not able to solve the task without the increased complexity of a language game. This therefore indicates that 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.

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

7 Analysis of the emerged languages

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) $Lang_{13}$

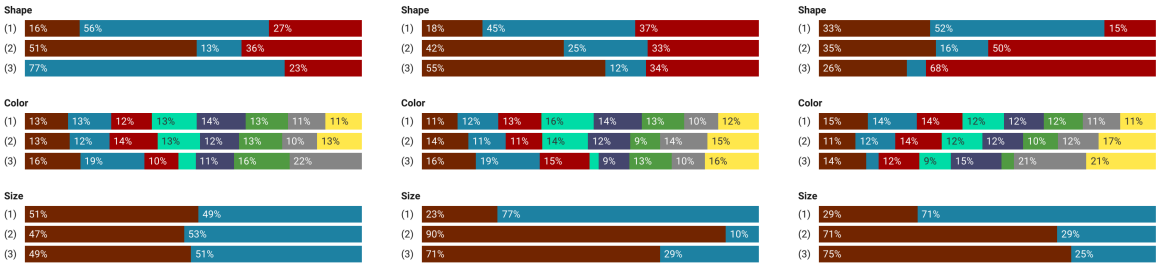
(c) $Lang_{100}$, (13 most frequent messages)

Figure 12: Relative frequencies of messages (ordered by frequency in test dataset)

When looking at the emerged vocabulary **qualitatively**, a few properties can be seen. Figure 12 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 12(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 13: 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

Figure 13 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 13(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.

Order	$L_{baseline}$	$L_{English}$	$\lvert V\rvert = 10$		$\lvert V\rvert = 13$		$\lvert V\rvert = 100$	
			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 16: Cross entropy losses while probing with emerged languages successful on 'Dale-2'

Table 16 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 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

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.

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. *Advances in neural information processing systems*, 33, 1877–1901.

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

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

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).

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. & 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.

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.

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. (1999). 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).

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*.

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).

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.

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*.

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

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

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

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*.

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.

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

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.

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.

A Resources