Imperial College
London

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

Measuring Compositional Generalization in Modular Neural Networks

Author:
Davide Locatelli

Supervisor:
Dr Josiah Wang
Dr Pranava Madhyastha

Submitted in partial fulfillment of the requirements for the MSc degree in Computing Science of Imperial College London

September 2020

Abstract

In recent years, deep neural networks have proven to be remarkably successful and often able to generalise well to new, unseen data. However, their generalisation skills typically fail to exhibit systematic compositionality. This represents a significant limitation of deep learning as it prevents it to perform fast inference on small amounts of data, narrowing its applications in domains such as language, math and, more generally, reasoning. The potential of making deep learning more data efficient and broaden its applications by introducing compositional skills has attracted a great deal of research from numerous fields in machine learning.

In natural language processing, a variety of tests have been designed to measure the ability of deep neural networks to perform systematic compositional generalisation. A recent example is the gSCAN benchmark, a grounded sequence-to-sequence task explicitly designed to be solved compositionally. State-of-the-art sequence-to-sequence networks notably fail to pass the benchmark, manifesting the current limitations of deep learning in this kind of reasoning and leaving the task mostly unsolved.

In visual question answering, modular neural network design has been recently adopted as an approach to introduce compositionality. More specifically, modularity is considered a powerful tool for compositional reasoning as it provides a way of decomposing a task in simpler subtasks: capturing the fundamental structure of a problem can be used to find the solution. By doing so, recently proposed modular networks achieved state-of-the-art performance on various visual question answering datasets.

The purpose of this thesis is to examine the extent to which modularity correlates to improved performance in tasks explicitly designed to test for compositional generalisation. To measure such correlation, we propose a modular approach to the gSCAN problem and compare the performance of the resulting neural network with the baseline model used by the authors. To do so, we start by running further experiments on their baseline model with the aim of identifying what hinders its compositional generalisation ability. We then run the same experiments using our modular network and analyse the differences between the behaviours of the two models.

Acknowledgments

I would like to sincerely thank my supervisors Dr Josiah Wang and Dr Pranava Madhyastha, without whom this project would not have been possible. I owe an immense debt of gratitude to Dr Wang for having accepted my proposal for this thesis, for encouraging me to explore the notion of compositionality, and for all the time and care he has put into supervising my research efforts. I am profoundly grateful to Dr Pranava Madhyastha for the incommensurable help throughout this project, both in directing my research towards the right places, and in continuously providing substantial advice for the implementation of our ideas.

Moreover, I would like to thank Christopher Tan and Benedetta Delfino, who both continue to inspire me every day with their passion for deep learning, their incredible experience in the field and their commitment to research.

Lastly, I would like to thank Dr Laura Riggall and Marie Pechenard, for keeping me motivated and determined to achieve the best I can, for reminding me the importance of dedication and perseverance in research, and for always being the most supportive and encouraging friends one could imagine.

Contents

1 Introduction

The landscape of artificial intelligence has been dramatically changed by the advancements of deep learning in the recent years. Deep neural networks have achieved impressive results on a large number of tasks and they have become more and more integral in applications that pertain to fields as diverse as healthcare and finance, recycling and retail, self-driving cars and social media.

Major breakthroughs in data collection and generation, coupled with advancements in hardware capacity, acted as catalysts for the 21st century successes of deep learning. The increase in data availability allowed for the creation of large datasets, which in turn fuelled the deep learning boom. The achievement considered by many to be the major trigger of the deep learning revolution is the 2009 creation of the ImageNet dataset [1]. Its novelty and importance lies primarily in its size: it contains a staggering 14 million images of more than 20,000 categories of objects.

The core idea that motivated the creation of such a large dataset was that more data leads to better performance: the intuition was that learning algorithms would perform better if they were provided with more observations which better represented the complexity of the real world. In the case of image classification, a neural network supplied with more training data has a better chance of learning to correctly identify an object in a picture because it has seen, for example, more camera angles, more lighting conditions, or different varieties of that same kind of object. The impact of ImageNet on the deep learning landscape was mainly due to the annual image classification competition that used the dataset to support the development of new learning algorithms. This eventually led to the 2012 introduction of AlexNet [2], a neural network so powerful that it halved the error rate of previous algorithms, and is still being used in research today. In 2015 the ImageNet winning accuracy surpassed human abilities with ResNet [3], further proving the point that more data leads to better decision, which has been widely accepted as a foundation of deep learning.

ImageNet is a very representative example of the deep learning achievements that followed, because the principle that more data is better remained the common thread. Many other datasets supported the swift development of new learning algorithms: MNIST [4], STL [5], CIFAR-10/100 [6] and countless others. The ultimate goal of the learning algorithms is to use these massive amounts of data to build rich models with accurate abstractions of the observations. Such abstractions can then be used to perform the generalisations needed for the model to appropriately react to new, unseen data.

The generalisation abilities of deep neural networks notoriously degrade with out-of-distribution data. That is, these learning algorithms struggle to adapt to inputs that significantly differ from the ones already observed. This is an important limitation for deep learning, because out-of-distribution observations are very common in a real-world setting, while these algorithms work under the assumption that the world is static. This shortcoming is due to the way deep neural networks construct abstractions of the data on which they train: the networks use shallow statistical patterns in the training data to take their decisions. If instead these abstractions captured more of the underlying structure of the data, they would enable better generalisation.

This is a limitation that deep learning and human learning do not share in the slightest. Humans are able to perform out-of-distribution generalisation even from a very young age. Studies have shown that 2-year-old children are able to comprehend the meaning of words of which they have seen just a few examples, and correctly generalise their learning by promptly using these words in new situations [7]. This advantage is due to the ability to extract much richer information from the observations at hand, and this in turn frees human learners from the amounts of data on which deep neural networks so heavily rely. Humans are effectively able to construct much richer models of the world, built on more meaningful abstractions that can be used for stronger and more accurate generalisations, all the while using less data [8].

An illustrative example of such generalisation skills, due to [9], is as follows. Imagine a person knows the meaning of the words “and”, “twice” and “again”. If she comes to learn a new verb, for example “to dax”, she will be immediately able to grasp the meaning of sentences such as “dax twice and dax again”. There will be no need for her to have observed the combination of such words already. Thanks to her understanding the role of verbs and adverbs and the meaning of the individual words, she already knows how “twice” is going to modify the meaning of the verb “to dax”. This allows her to correctly generalise to unseen sentences.

Notice how her grasping the meaning of the new sentence is directly enabled by her understanding of its individual parts and their grammatical role. This is a property of language referred to as *systematic compositionality*: the meaning of a whole sentence depends on, and only on, the meaning of its parts and its syntactic structure [10]. Though this property has been widely studied in the literature of the philosophy of language, it is not limited to language, but it characterises human thought in many of its aspects.

For example, compositionality can be observed in the task decomposition abilities in human learning: people can typically find solutions to new complex problems by decomposing them in smaller ones for which the solutions are known, proceeding in a divide-and-conquer fashion. Studies have demonstrated that visual data is also typically understood by humans compositionally. In [11], the authors demonstrated that

3-year-old children are able to describe the individual properties of objects shown to them on-screen, effectively grasping that the make-up of the objects follows a compositional recipe: the object is understood as a sum of its properties (Figure 1.1).

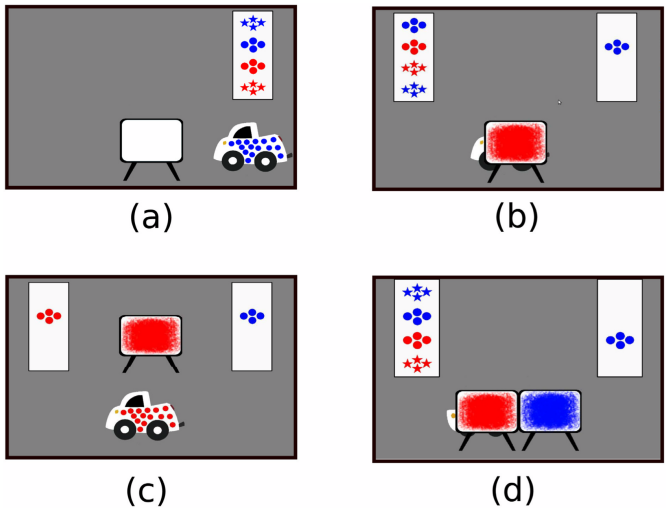

Figure 1.1: Experiment from [11] testing for visual compositionality in early childhood. First, a car is presented on a touch-screen display (a) and participants are asked to describe the properties of the car by selecting the patterns that match the car from four possible patterns in the box on the right. The first selection remains on the screen while the car moves behind a screen. The screen is coloured according to the transformation it applies to the car. Children are then required to predict the outcome of the transformation with a new set of choices on the left (b). At training time, they answer until they correctly predict the transformation and then can observe the car as the screen lifts (c). At test time, children are required to predict the transformation of a car passing behind two screens instead of one.

1.1

Thus, thanks to their compositional skills humans are able to perform rapid inferences on fewer, noisier, and sparser observations [12], achieving results that deep learning is currently unable to achieve with larger and more static datasets. This translates to a restricted applicability of deep learning, as it struggles to adapt to real-world settings, where rapid inference on continuous streams of data is necessary in a wide range of domains of reasoning.

If deep neural networks, instead of relying on shallow statistical pattern-finding algorithms, were encouraged to look for the deeper structure in the data, this could potentially enable better compositional generalisation, diminish their reliance on such large amounts of training examples and enlarge the range of domains in which they can be used. This idea has recently attracted a great deal of research from the numerous fields in machine learning that would benefit from such an improvement.

In particular, visual question answering is a domain in which compositionality would allow neural networks to be used for more complex tasks. Given a textual question

and an image, a neural network is tasked to answer the question by analysing the content of the image. For example, given a photo of a group of people walking in the street near a lamp post holding umbrellas, a typical question could be "What colour is the umbrella held by the girl on the right of the lamp post?". Figure 1.2 gives another example, borrowed from [13]. Given a picture of a number of shapes of different colours, a typical question could be "Is a circle below a square?".

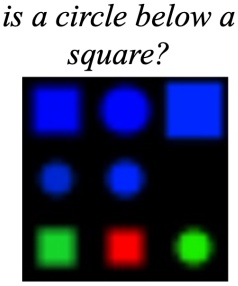

Figure 1.2: Example of a visual question answering task. The data comes from the SHAPES dataset [14].

1.2

Recently, [14] proposed a way of introducing compositionality in this domain by means of task decomposition. Their approach consists in understanding the question as a composition of functions, and providing the answer by sequentially applying those functions to the image. In the example of people holding umbrellas, the task of answering the question could be decomposed into

1. finding the lamp post
2. looking at its right
3. finding the girl in this area of the photo
4. looking at her umbrella and reporting its colour

To achieve this, each sub-task is assigned to a module which specialises in applying the corresponding function to the image. To answer a question, the proposed neural network assembles a combination of its modules according to its compositional understanding of the task. In other words, the design of the network is modular rather than monolithic: the task is not being approached with a single huge neural network, which is involved in its entirety whenever a question is presented, but with multiple smaller modules that are used only when their respective task is presented. This means that once a smaller problem is solved, the learned resolution can be reused when the same problem is recognised as as a part of a more complex task, effectively achieving the compositional divide-and-conquer technique typical of human learning.

Another field which has been interested in compositionality is natural language processing. Given that the property of compositionality has been first extensively studied in the philosophy of language, natural language processing offers the ideal domain where neural networks can be tested on compositional reasoning. As a result, a

number of benchmarks of compositional reasoning have been proposed: researchers created new datasets, explicitly designed to present neural networks with tests that can be passed only if the training data is understood compositionally. More precisely, the design of these tests takes advantage of the fact that under the principle of compositionality the meaning of the whole sentence depends on the meanings of its parts. Accordingly, these datasets present a similar distribution of words in the training and test sets, but a different distribution of composition of such primitives: in other words, the networks are tested on unseen compositions of known words. Examples of datasets designed in this manner include CFQ [15], SCAN [9] and gSCAN [16].

gSCAN was recently introduced by [16] to test for systematic compositional generalisation in a sequence-to-sequence task. In gSCAN, the goal is to learn which series of actions have to be taken by an agent placed in a grid-world and surrounded by objects, given an input command. For example, given the instruction "walk to the circle while zigzagging", and the grid-world situation in Figure 1.3 , the network has to predict the necessary actions the agent has to take to follow the command.

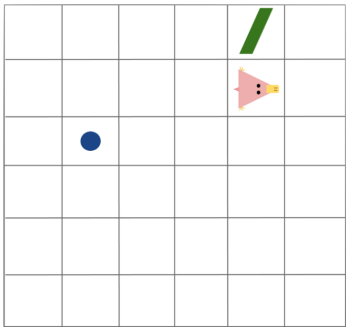

Figure 1.3: Example of data point in the gSCAN dataset [16]. Given the command, the task is to find the correct action sequence that modifies the situation accordingly.

1.3

The benchmark tests for compositionality by verifying whether the network has correctly abstracted the concepts corresponding to each input word and can adapt to a different combination of the known primitives. The authors in [16] have tested a model based on state-of-the-art sequence-to-sequence techniques which has substantially failed the benchmark on most test splits. As argued by its authors, improving performance on the gSCAN tests would amount to discovering compositionality in the set of neural networks abilities. Deep neural networks proficient in compositional reasoning would represent an important leap forward in the research, which would further elevate deep learning as the leading approach towards general machine intelligence.

In this thesis, we aim to measure the potential of a modular neural network design approach to achieve compositional reasoning in the gSCAN benchmark. To do this we endow a neural network with a set of learned modules for interpreting individual

verbs, adverbs, and identifiers in the dataset. By doing so, we hope to encourage the network to extract rules from the dataset rather than relying on shallower correlation patterns in the data. We then compare performance with the baseline model to analyse the potential of such architecture to enable compositional reasoning in deep neural networks.

1.1 Contribution

More precisely, the contributions of this thesis are as follows:

- We perform an ablation study on the neural baseline by removing one of its parts at a time in order to gain a better understanding of the network behaviour.
- We propose a set of experiments for the gSCAN dataset aimed at demonstrating the effects of providing a model with increasingly more information.
- We propose a modular neural network with the goal of improving the current gSCAN benchmark results.
- We test our modular architecture and provide an analysis of its performance, comparing it to the gSCAN neural baseline.

1.2 Thesis outline

The thesis is organised as follows:

- Section 2 provides a brief introduction to deep learning, focusing on the concepts and techniques that are most relevant to the work in this thesis.
- Section 3 focuses on formally defining the principle of compositionality as formulated in the philosophy of language literature.
- Section 4 is dedicated to presenting the gSCAN dataset.
- Section 5 presents the gSCAN baseline model, an ablation study performed on it, and also proposes a set of further experiments for the gSCAN dataset.
- Section 6 introduces modular neural networks and presents the recent advancements in visual question answering enabled by the modular approach.
- Section 7 details the proposed modular neural network architecture and reports the experimental results of the compositional gSCAN tests on the proposed architecture, identifying the strength and weaknesses of such architecture when compared to the baseline model.
- Appendix A contains a checklist with ethics considerations about the project in accordance with the Imperial College individual project requirements.

2 Deep learning

In this section we briefly introduce the fundamental concepts of deep learning for those readers who are not yet familiar with this family of machine learning algorithms. After having presented the general ideas, we go into more detail to explain those deep learning algorithms that are used in the rest of this thesis, paying particular attention to recurrent neural networks. Then we describe how these networks are used in the domains of natural language processing and computer vision, which allows us to explain sequence-to-sequence learning and visual question answering, necessary to understand the gSCAN benchmark and the recent advances in modular neural networks respectively.

2.1 Definition

Artificial intelligence is a family of techniques concerned with enabling computers to simulate intelligent human behaviour. As a discipline, it contains several separate branches, each of which aims to mimic a particular aspect of human intelligence, such as learning, planning, representing knowledge or moving and manipulating objects. Machine learning is a branch of artificial intelligence that studies learning algorithms which improve by training on a set of observations. The learning approach can be either supervised, semi-supervised, or unsupervised.

In supervised learning, the algorithm is tasked with learning a function to map an input to an output by observing input-output pair: each input in the dataset is labelled with its corresponding output. In unsupervised learning, the job of the algorithm is to find patterns in data that has not been labelled. Semi-supervised learning combines a small number of labelled inputs with a large number of unlabelled ones.

In this thesis we are mostly concerned with the supervised learning approach. More formally, given a set of N observations $\{< x_1, y_1 >, ..., < x_N, y_N >\}$ where x_n is a feature vector representing the n^{th} input observation and y_n is the corresponding output label, a supervised learning algorithm is tasked with finding the function $f : X \to Y$ from the input space X to the output space Y such that $f(x_n) = y_n$.

Deep learning is a subset of machine learning which makes use of multi-layered (hence, *"deep"*) algorithms commonly known as artificial neural networks.

2.2 Artificial neural networks

Artificial neural networks are a class of machine learning algorithms that can perform information processing by finding statistical patterns in the data they are given. They can do so by learning probability distributions over the input data and use them to construct rich models of their observations. Neural networks are composed of layers (Figure 2.1), which can be of three kinds:

- Input layers, which receive the input to the neural network
- Hidden layers, which perform the intermediate computation to map the input to the output
- Output layers, which produce the result

In deep learning, neural networks are multi-layered, meaning that they have multiple hidden layers.

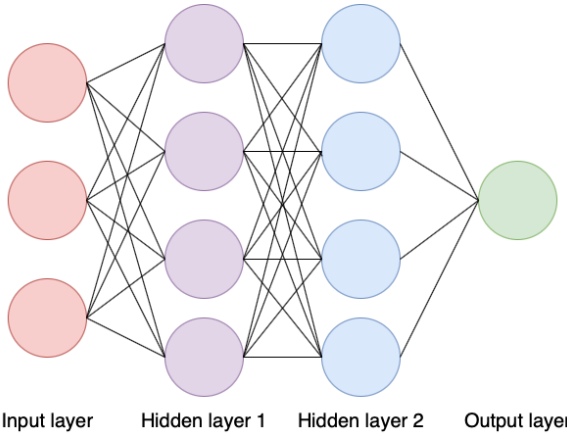

Figure 2.1: Example neural network structure.
2.1

The task of the layers is to progressively extract higher-level information from data: "lower" layers (the leftmost in Figure 2.1) are concerned with low-level features of the observations, while "higher" layers focus on high-level information. If the input is an image of a table, an example of a low-level feature is the texture of the table surface, while a high-level feature could be its legs. The power of deep learning lies in its having more layers which can therefore extract more information from the inputs, constructing a more comprehensive model of its observations.

2.2.1 Neurons

Layers are made of units called neurons. Each neuron receives a number of inputs and produces a single output. Inputs are transmitted to the neuron through a connection (the edges in Figure 2.1), much like the synapses in a biological neural network. Each connection carries weight that represents the relative importance of

that connection.

To transform the input, a neuron applies a *propagation function*: it calculates a weighted sum of its inputs, each input being weighed by its connection, and then a bias term is added to the sum. The result is called activation and it is passed through a function, called activation function, which produces the output of the neuron.

2.2.2 Activation functions

On a conceptual level, the activation function is responsible for determining whether the neuron 'fires' or not given a certain input. In deep neural networks the activation functions are generally non-linear and are differentiable. Popular activation functions include Sigmoid, Softmax, ReLU and Tanh.

The Sigmoid activation function is a function with an S-shaped curve between 0 and 1:

$$\sigma(x) = \frac{1}{1 + exp(-x)}$$

The Softmax activation function converts an N-dimensional real value vector into an N-dimensional real value vector with values between 0 and 1 and whose sum is 1. Thus, it presents the input vector in a probabilistic setting:

$$\sigma(x)_j = \frac{e^z_j}{\sum_{k=1}^{K} e^z_k} for j = 1, ..., K$$

The ReLU activation function returns the input unchanged if it is positive, otherwise it returns a 0:

$$ReLU(x) = max(0, x)$$

The Tanh activation function is a function with an S-shaped curve between -1 and 1:

$$Tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

2.2.3 Learning

A network is tasked with learning how to better map inputs to outputs after observing many input-output pairs. This is done by progressively adjusting the network weights between neurons to achieve a higher accuracy in the outputs. This is done using a loss function which evaluates the output of the network. This function is periodically evaluated during learning and usually the goal is to minimise it. To do so, one takes advantage of the differentiability of this function. Backpropagation [17]

is the method which, using the gradient of the loss function, updates the weights of the network by taking a corrective step. The corrective step can be of several kinds, typically stochastic gradient descent, which updates the weights in the negative direction, each time subtracting an amount which we refer to as learning rate.

2.2.4 Recurrent neural networks

Recurrent neural networks are a type of networks that can remember information it learned from a sequence of inputs, which influences its decision. Popular implementations include LSTMs [18] and GRU [19].

A typical recurrent neural network (Figure 2.2) can be described as follows. Let

$$x = (x_1, ...x_{t-1}, x_t, x_{t+1}, ...)$$

denote an input sequence where t indicates a time step. At time t, the recurrent neural network records information about the input x by calculating a state, called hidden state, which is then passed to the next time step:

$$h_t = f(Ux_t + Wh_{t-1})$$

where f is a non-linear activation function. The hidden state for $t = 0$ is usually initialised to zeros. At each time step, an output is also calculated by

$$o_t = softmax(Vh_t)$$

which is influenced by the memory of the network through the hidden state h_t.

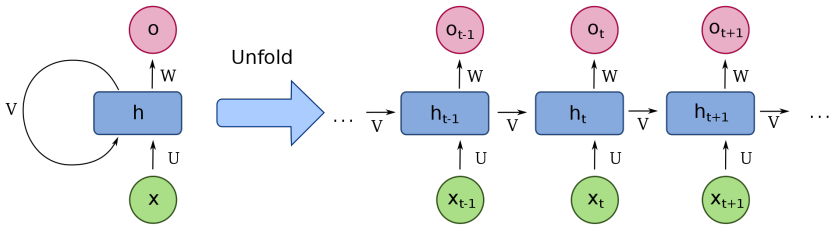

Figure 2.2: A recurrent neural network.
2.2

In recurrent neural networks, backpropagation occurs through time, which means that the gradient is calculated for each time step. This is done by backpropagating the time steps before the current time step and summing them.

Because recurrent neural networks are usually quite deep, they suffer from the vanishing gradient problem, which means that the gradient does not backpropagate through enough time steps. This makes it difficult for recurrent neural networks to

remember long-term dependecies in the input [20]. To overcome this difficulty, [18] introduced Long Short Term Memory (LSTM) reccurent neural networks.

An LSTM has not one but two states: the hidden state h_t and the cell state c_t. At time step t, c_t takes three values: the input x_t, the hidden state from the previous time step h_{t-1} and the cell state from the previous time step c_{t-1}. Thanks to the cell, the LSTM can decide whether it should remember some dependency of the input or whether it can forget it.

In each cell, there are four layers tasked with learning how to manage the memory of the network using a tanh activation. To do so, the cell looks at the previous hidden state h_{t-1} and the current input x_t. Then it computes its decision about whether to forget the information:

$$f_t = sigmoid(W_f \cdot [h_{t-1}, x_t] + b_f)$$

which gives a 0 for 'forget' and a 1 for 'remember'. Another set of weights is used to decide whether to keep the input x_t:

$$i_t = sigmoid(W_i \cdot [h_{t-1}], x_t] + b_i)$$

Tanh is used to compute the candidate value $\tilde{C_t}$ which is going to be added to the cell state:

$$\tilde{C_t} = tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

while the decision whether to update the cell state is taken by

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C_t}$$

where $\odot$ is the Hadamard product. The output of the cell is decided using another set of weights W_o:

$$o_t = sigmoid(W_o \cdot [h_{t-1}, x_t] + b_o)$$

and the hidden state is calculated by

$$h_t = o_t \odot tanh(C_t)$$

.
The Gated Recurrent Unit (GRU) was proposed by [19] and is similar in structure to the LSTM, but it only has two layers instead of four in each cell. The two layers decide whether to update the cell state or to reset it. The decision to update the cell state is taken by

$$z_t = sigmoid(W_z x_t + U_z h_{t-1})$$

Similarly, the decision to reset the cell state is taken with the same formula but different weights:

$$r_t = sigmoid(W_r x_t + U_r h_{t-1})$$

This decision is used to compute the candidate new value for the cell hidden state

$$\tilde{h_t} = tanh(W x_t + r_t \odot U ht - 1)$$

and finally the hidden state is updated using the forget decision:

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h_t}$$

2.3 Natural language processing

Natural language processing is a subfield of machine learning that studies how to design learning algorithms that are able to analyse and process natural language data.

2.3.1 Sequence-to-sequence learning

Sequence-to-sequence learning is a natural language processing task in which the learning algorithm has to transform an input sequence from one domain to an output sequence to another domain. For example, if the input sequence is a sentence in English, the algorithm could be tasked with translating the sentence in French. Machine translation is in fact a typical application of sequence-to-sequence learning, but not the only one: the input sequence could also be image data, which has to be captioned using a sequence of textual data (image captioning). Or the input sequence might be a text and the output sequence a summary of that text.

Typically, a sequence-to-sequence problem is approached with an encoder-decoder neural network. The idea is to encode the input into a hidden vector using the encoder, passing the vector to the decoder, which reverses the process by going from the vector to the output. LSTMs or GRUs are usually used as the encoders and decoders for sequence-to-sequence tasks.

2.4 Computer vision

Computer vision is a machine learning subfield interested in studying how to design learning algorithms that are able to gain a high-level understanding of visual data.

2.4.1 Visual question answering

Visual question answering is an application of both computer vision and natural language processing in which the task is to answer a question about an image. The question is provided in natural language.

3 Compositionality

In this section we focus on the principle of compositionality. Owing the study of this principle to the philosophy of language, we restrict the context of our discussion to natural languages. We begin from the distributional semantics hypothesis, which allows us to better frame the role of compositionality. We then present two linguistic properties strictly related to compositionality: productivity and systematicity. These have been extensively used by philosophers to argue that natural languages are compositional. For each of these aspects we specify how they are operationalised in natural language, and how they lead to corresponding types of generalisation. This is useful for understanding what the benchmarks of compositional reasoning that have recently been proposed in deep learning, like the gSCAN, are actually testing.

3.1 The distributional semantics hypothesis

The distributional approach to semantics works under the hypothesis that "words which are similar in meaning occur in similar contexts" [21], which means that semantic similarity and distributional similarity are directly correlated. According to Zellig Harris, to whom we owe the refinement of this hypothesis, this correlation is so intrinsic that it could provide a complete typology of the entirety of language. In other words, language could be systematically explained just by studying its distributional behaviour [22].

Its relevance to compositionality is twofold. First, it offers a representational framework of language which uses continuous rather than discrete mathematical methods, and this in turn allows for intuitive and meaningful ways of implementing compositionality. By encoding distributional information in high-dimensional vectors such that similar vectors represent words which exhibit distributional and semantic similarity, language can be represented by a continuous rather than discrete mathematical model [23]. For example, given words 'virus', 'vaccine' and 'summer', thanks to distributional semantics we can encode the relatedness of the first two words in two similar vectors, while the third word would be represented by a dissimilar one. In Figure 3.1 we illustrate this example in the case of 2-dimensional vectors.

Provided with these continuous representations of sentences, there can then be a question of how to compose such representations. The pairing of words such as 'big table' can be implemented by various techniques: for example, given the vectors for 'big' and 'table', one can use vector addition, or point-wise multiplication [24]. Thus, distributional semantics provides the representational framework for compositional-

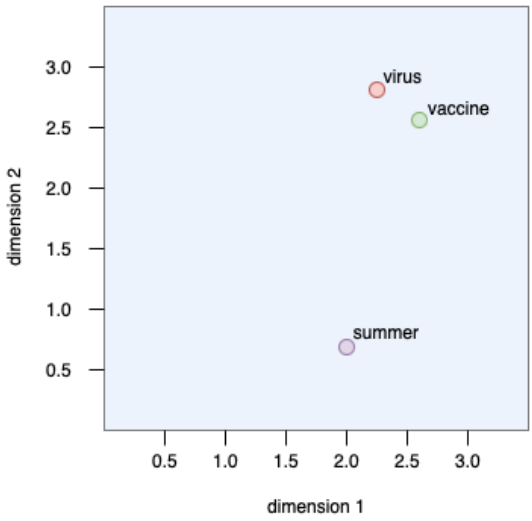

Figure 3.1: By noticing that in the given corpus the words 'virus' and 'vaccine' occur in more similar contexts than 'summer', we can deduce, under the distributional semantics hypothesis, that their meaning is more similar. If we then want to represent this vocabulary with 2-dimensional vectors, we can choose to have two similar vectors for the words 'virus' and 'vaccine' and choose a dissimilar one for 'summer'. In the graph, notice how 'virus' and 'vaccine' are nearest neighbours, because the values for the two dimensions of their vectors are similar.

3.1

ity.

Secondly, the distributional semantics hypothesis provides an explanation of compositional behaviour in human learning. The correlation between distributional and semantic similarity gives a methodology to infer the meaning of new words by discovering that they have a tendency to be used in the same context as familiar words [25]). This amounts to generalisation by similarity, but the behaviour is also compositional. We present compositionality in more detail below, but in short for a linguistic behaviour to be compositional, it must rely on both the meaning of the familiar words as well as their syntactic role. That the former are involved in this type of generalisation is clear, but we need an argument for the latter.

By observing that a word is being used in a certain context, one also observes *how* it is being used syntactically. For example, a child might know the meaning of 'big' but not of 'huge', and in observing that both words are being used by her parents in describing their kitchen table, she might discover that the two words have similar meanings. But in doing so, she also observes that the words are both placed before the word 'table' or after 'is': 'the big table'/'the huge table', or 'the table is big'/'the table is huge'. This arguably creates in her an expectation of hearing the same usage

in future utterances of the sentence and it could be even argued that the repeated observations of similar usages enable the forming of intuitions about the syntactic role of the word. Therefore, recognising the distribution of a word over language requires an understanding of where and how a word must be typically placed in a sentence, and so, at least to some extent, its syntactic role.

Then, by establishing a correlation between the distribution of words and their meaning, distributional semantics also indirectly establishes a correlation between syntactic structure of sentences and the meaning of individual words. The principle of compositionality simply builds on this hypothesis by drawing a further correlation between the meaning of individual words and the meaning of whole sentences.

3.2 The principle of compositionality

If the distributional semantics hypothesis establishes a relation between meaning of words and their distribution, which, as argued above, also involves their syntactic role, the principle of compositionality builds on this by drawing a tight correlation between the meaning of words, their syntactic role, and the meaning of the sentences in which they appear. Under the principle of compositionality, the meaning of a sentence is fully explained by its structure and the meanings of its words. More formally, given a language L:

Principle of Compositionality. For every complex expression $e \in L$, the meaning of e is fully determined by the structure of e and the meanings of the constituents of e [26].

The 'structure' is regulated by the *syntax* of L, while the 'meanings' are given by the *lexical semantics* of L. Notice that the principle of compositionality does not pose any constraints to what the theories of syntax and semantics are: one could assign to 'structure' and 'meanings' any syntax and semantics. In fact, the principle needs to be accompanied by a specification of what can count as a constituent and what syntactic operations are possible. Taken by itself, it is underspecified and very general.

This generality has often been exploited by opponents of compositionality in an attempt to trivialise its principle. Their typical counterarguments aim to show that an arbitrary syntax or an arbitrary semantics can be turned into a compositional one, even ones which are intuitively non-compositional. For example, given an arbitrary meaning function $f : W \to M$ that maps words w to meanings m, dissenters of compositionality aim to show that it can be turned into a compositional one (see [27] or [28]).

Other opponents of compositionality have provided counterexamples which generally aim to show that sentences that share syntactic structure and whose constituent are pairwise synonymous may fail to convey the same meaning, as the principle of

compositionality requires. For instance, take 'Cicero is Cicero' and 'Cicero is Tully', where Cicero and Tully are synonyms, since they refer to the same person. Under the principle of compositionality, for every two complex expressions e and e' belonging to a compositional language L, e and e' are *strictly synonymous* if:

- their structure is the same
- for every constituent $c \in e$ and every constituent $c' \in e'$, each c corresponds to a synonymous c'.

In the counterexample above, the two sentences clearly have the same structures, and Cicero and Tully are synonymous: the sentences should therefore mean the same thing. Yet, they do not: 'Cicero is Tully' encodes semantic co-reference, which means that it is expressing that the names 'Cicero' and 'Tully' refer to the same person, while 'Cicero is Cicero' does not carry this meaning, and amounts, in fact, to a trivial statement.

To argue in favour of compositionality, one could amend the principle as needed to account for such counterarguments. For example, in the case of Cicero, one could argue that in addition to the syntactic structure and the meaning of the individual constituents, a special co-reference relation between subject and object also contributes to the meaning of the sentence. Or against the counterargument that an arbitrary syntax or semantics can be turned into a compositional one, the principle could be strengthened by restricting the range of semantic and syntax theories with which it works. By doing so one would sacrifice its generality and possibly end up with a number of weaker or stronger principles of compositionality, each of which could account for a particular aspect of language according to the counterargument provided.

In this thesis, we do not aim to provide such a defense of compositionality. Instead, we take the principle for what it is: a general principle that cannot, by definition, account for the complexity of language in all its particular forms, but that has the power of explaining a great deal of the properties we observe in language. Defenders of compositionality actually use these properties to support their arguments. They argue that because language exhibits such qualities, it must be compositional. Among the linguistic properties most commonly used for this purpose, productivity and systematicity are the two most popular choices.

3.3 Productive compositional generalisation

Productivity is the property that describes the ability to produce infinite outputs from a finite basis [29]. This can be used to explain, for example, our ability, as finite beings, to produce a seemingly infinite set of complex expressions. Recognising this ability does not require one to accept the hypothesis that natural languages contain

an infinite number of expressions, which, despite being a plausible hypothesis, cannot be empirically established (and is in fact rejected by many, such as [30] and [31]). Arguably it suffices to recognise that we cannot memorise all the sentences we are able to produce: therefore, the ability to produce them must be coming from a finite basis [29].

From this, one can argue for compositionality as follows:

Argument from Productivity. Finite language users are able to produce an infinite number of complex expressions. So they can produce a meaningful complex expression e which they never encountered before. This requires them to already possess knowledge on the basis of which they can produce e. Because they have not memorised e, this knowledge cannot but be knoweldge of how to produce the structure of e and of the meanings of its constituents.

The formalisation of this argument is due to [32]:.

Taking advantage of productivity, we can define productive compositional generalisation as the ability to produce (or understand) expressions which are *longer* than the ones usually encountered [33]. With a productive understanding of language, an agent has the tools to deal with longer sentences, by recombining known syntax rules and vocabulary (Figure 3.2).

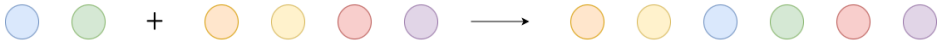

Figure 3.2: Schematic representation of productive compositional generalisation. The circles represent known words, and the way their pairings represent the known syntactic rules. The resulting sequence is longer, but presents the same words and pairings.

3.2

A deep learning dataset can be said to be testing for productive compositional generalisation if

- the training set contains only complex expressions e such that $length(e) = N$
- the test set presents expressions e' such that $length(e') = N'$ and $N' > N$.

3.4 Systematic compositional generalization

Systematicity is the property that describes the presence of definite patterns in the complex expressions of a language. Taking advantage of these patterns, we can explain our ability to recombine known parts in order to understand or produce expressions made up of those parts [10]. For example, knowing expressions such as 'the red book' and 'the blue table', we also understand 'the blue book' and 'the red table'.

This recombination ability requires compositionality, because it presupposes that each part makes a uniform contribution towards the meaning of the sentence. As formalised by [32]:

Argument from systematicity. Given complex expressions e and e' obtained through the syntactic operation F over constituents $c_1, ..., c_n \in e$ and $c'_1, ..., c'_n \in e'$, a language user can also understand any other meaningful complex expression e'' obtained through F over constituents among $c_1, ..., c_n, c'_1, ..., c'_n$, without the need for any additional information. If this is so, the meaning of e'' must be jointly determined by the meaning of e and e'. Therefore, the only possibility is that F and the meanings of $c_1, ..., c_n$ determine the meaning of e, F and the meanings of $c'_1, ..., c'_n$ determine the meaning of e' and F and the meanings of $c_1, ..., c_n, c'_1, ..., c'_n$ determine the meaning of e''

[10] further argues that if language was not compositional, we would not be able to perform systematic generalisation: using the recombination technique to understand unknown expressions, never encountered before, but which are made up of known parts (Figure 3.3). Without a compositional language, the meaning of each sentence would be stored in an atomic manner, with a mapping from whole sentences to meanings [33]. With such mapping, the similarity of a new sentence with known expressions could not be established, because the similarity between their parts is lost in the atomic mapping of whole sentences. Therefore one would not be able to understand its meaning. But because we do in fact perform this systematic generalisation, language must be compositional.

Figure 3.3: Schematic representation of systematic compositional generalisation. The circles represent known words. The resulting sequence is unknown, but it is made up of known words.

3.3

A deep learning dataset can be said to be testing for systematic compositional generalisation if

- the training set contains complex expressions $e_1, ..., e_n$ made of constituents $c_1, ..., c_n$
- the test set presents expressions $e'_1, ..., e'_n$ which are obtained by recombining constituents among $c_1, ..., c_n$ in novel ways.

4 The gSCAN dataset

Having specified what it means for a deep learning dataset to be testing for systematic and productive compositional generalisation, we can now present the gSCAN dataset [16]. Its authors designed it to be in fact testing for both systematic and productive compositionality, but in this thesis we focus just on the experiments they have proposed for systematicity. We begin by providing a general description of what a deep neural network needs to achieve on the dataset, and we then present its grammar and semantics. We then describe in more detail why the gSCAN dataset can be said to be testing for systematic compositional generalisation by presenting a number of tests which the authors used to create their generalisation benchmarks.

The gSCAN (**g**rounded **S**implified versions of the **C**omm**A**I **N**avigation task) is a dataset that has been introduced by [16] with the purpose of benchmarking deep learning algorithms on compositional generalisation. The dataset consists of a modified version of a previously proposed SCAN dataset [9] (which was inspired by the CommAI environment [34]).

4.1 Limitations of SCAN

In the SCAN dataset, the goal of the learning algorithm is to translate a navigation command into a sequence of actions. The dataset provides a set of available actions, which the learner has to be able to use to describe the navigation imposed by the command. Each command is paired with a single action sequence. For example, the input might be 'turn left twice', while the target output sequence is 'LTURN LTURN'. We are therefore translating from a natural language command to a logical representation of the corresponding action. Therefore, the task can be treated as a sequence-to-sequence semantic parsing task (Dong and Lapata [35]). For more examples of input-output pairs, see Figure 4.1.

The dataset has been designed to test for productive and systematic compositional generalisation: to perform well on SCAN, neural networks are required to

- generalise to longer action sequences than the ones they trained on (productivity)
- generalise to novel pairings of a known command with known modifiers, e.g. having trained on 'jump', 'turn left' and 'turn left twice', the network is asked to generalise to 'jump twice' (systematicity)

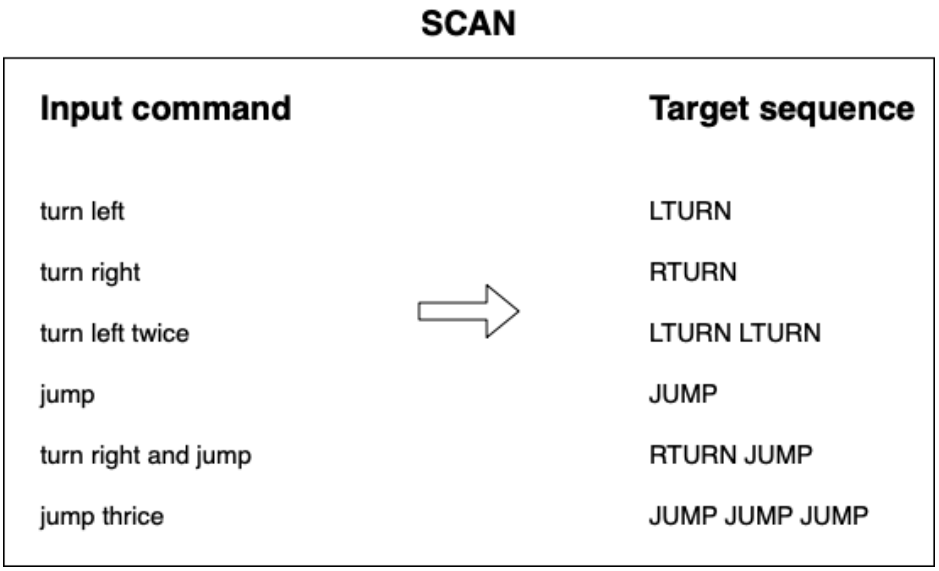

Figure 4.1: Example input-output pairs from the SCAN dataset [9]
4.1

However, the kind of reasoning it is testing is significantly different from that required in a real-world setting. While compositionality ensures that the meaning of a sentence is determined by its words and its structure, usually this meaning is used *to do something*: in the case of a navigation command, it instructs us on where and how to move. After understanding the meaning of a sentence, if we want it to be used for something, we must deduce how to actualise it in the context in which we are. Once again using the example of navigation commands, if we understand the command 'walk to the big red square', we must observe the situation we are in, identify the red big square and deduce the series of steps necessary to move to it. For example, this same command can translate to dramatically different action sequences depending on where the navigation is starting from. Consider the scenarios in Figure 4.2 and notice how the contextual changes affect the target action sequences.

It is important to point out that the contextual dependency of language is not in tension with its compositionality. Contextual dependency does not equal to saying that the meaning of a sentence is determined also by its context, which would actually violate the principle of compositionality: in our red square example, understanding the meaning of the command and using it for our navigation goal are two separate events. Understanding a sentence and actualising it are in fact distinct operations, and the latter presupposes the former.

The SCAN dataset is unable to capture both the compositionality and the contextual dependency of language, because it is not *grounded*: it does not give a world situation in which the navigation commands have to be followed. As such, its semantics is significantly limited and can be reduced to a meaning function which maps word sequences to action sequences without paying attention to the context in which these sequences occur. As a result, it is not possible to assess whether the networks which

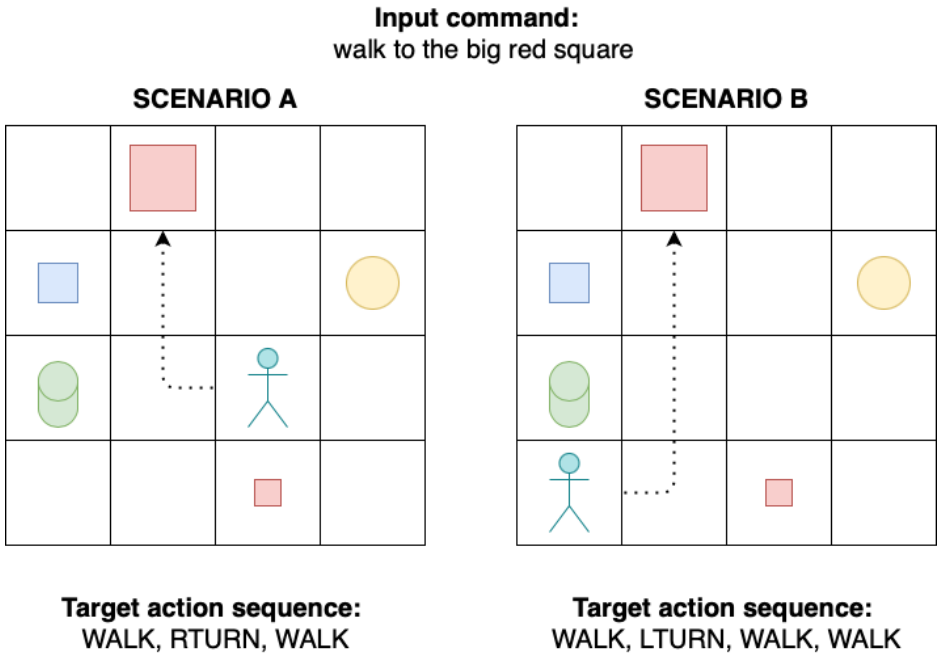

Figure 4.2: Example scenarios to demonstrate the contextuality of language: given the same input command, the corresponding target sequence can be different according to where the agent is placed in the world.

4.2

manage to perform well on the SCAN tests such as [9] and [36] have compositional generalisation skills that are similar enough to ours, as the language setting is so detached from a real-world scenario.

4.2 Grounded SCAN

In gSCAN, the goal of a learner is the same as in the SCAN: to translate a navigation command sequence into an action sequence. However, to do so the learner must not only understand the meaning of the commands, but also learn how to ground them in a world situation. In fact, the dataset not only provides input navigation commands, but also visual situations in which the commands must be followed. See Figure 4.3 for an example.

Each situation describes the placement of objects in a grid-world and the starting location of the agent who has to follow the input command. Objects can be of different shapes, colours and sizes. In particular, objects can be square, cylinders of circles, of colour red, blue or green and of size 1, 2, 3 or 4. The agent can be placed in any cell that is not occupied by an object and can be facing any direction. The navigation command may require interaction with the target object: for example, besides walking to an object, the agent may be also asked to push it.

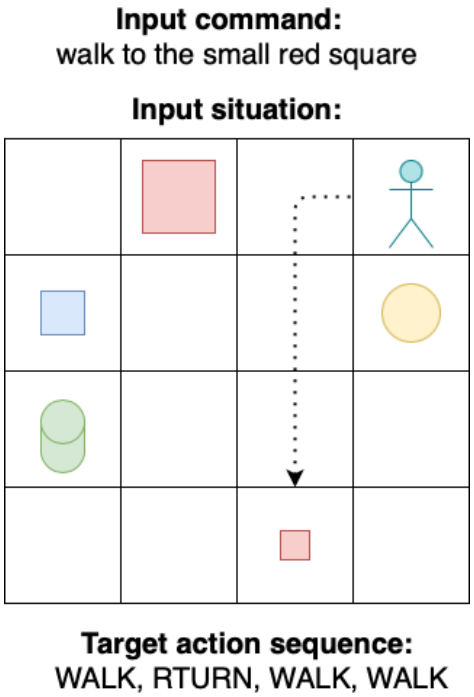

Figure 4.3: Example input command and scenario pair and output action sequence in gSCAN [].

4.3

4.3 The gSCAN grammar

The navigation commands in gSCAN are provided in a simplified natural language that follows a precise grammar. In formal language theory, the grammar of a language specifies how to create sentences that respect the syntax of the language. The grammar of gSCAN is a context-free grammar, which means that it specifies how to produce valid sentences using a rule system that works as follows.

First, the grammar needs to define a number of sets, each of which defines which words can be used in a particular syntactic unit of the sentence. For example, the grammar has to define the set of verbs that can be used in the verb phrases, or the set of adjectives to be used in the noun phrases. Then, the grammar defines a number of rules to produce each syntactic unit.

We report the gSCAN grammar rules in Figure 4.4. We use the Penn Treebank part-of-speech labels [37], which denote the grammatical categories of a sentence as follows. VP is the verb phrase, containing the verb of the sentence and its dependent words, like adverbs (RB). The verb in a VP is denoted by VV and can either be transitive (VV_t) or intransitive (VV_i). Transitive verbs are followed by a determiner phrase DP, while for intransitive verbs the DP is preceded by a predeterminer PDT. The DP accompanies the noun phrase NP, which contains a noun NN and can also contain an adjective JJ.

In gSCAN, the only existing VV_i is 'walk', while VV_t include 'push' and 'pull'. The set of available adverbs is RB ={cautiously, hesitantly, while spinning, while zigzagging}. The only PDT is the preposition 'to' and the only possible determiner in DP is the article 'a'. The set of nouns is NN ={square, circle, cylinder} and the set of adjectives is JJ ={big, small, red, blue, green}.

SCAN COMMAND GRAMMAR

Production rules	**Available words**
ROOT → VP	VV_i = {walk}
VP → VP RB	VV_t = {push, pull}
VP → VV_i PDT DP	RB = {cautiously, hesitantly, while spinning, while zigzagging}
VP → VV_t DP	PDT = {to}
DP → DT NP	DT = {a}
NP → JJ NP	JJ = {big, small, red, blue, green}
NP → NN	NN = {square, circle, cylinder}

Figure 4.4: The gSCAN context-free grammar for command sequences.

4.4

As shown in Figure 4.4, the production rules are of the form $A \rightarrow a$, where A indicates a syntactic unit and a may indicate either a number of syntactic units and/or one of the sets containing the available strings. The first rule has on the left-hand side the label $ROOT$, which denotes an initial empty string. To produce a valid sentence, one has to follow the rules using the algorithm below.

Algorithm 4.1

1. Initialise the sentence to $ROOT$
2. Follow the rules top to bottom, starting from $ROOT \rightarrow a$, as follows
   (a) If a references another syntactic unit, find a rule below that has a on its left-hand side. If two rules have the same left-hand side, either one can be used. Set a to be the right-hand side of the rule
   (b) If a references the name of a set of available strings, pick one of these strings and add it to the sentence. Remove the reference to the set from a.
3. Repeat (a) and (b) until a is null: the resulting sentence is grammatically valid.

Following this algorithm, we can obtain commands such as 'push a big green cylinder', 'walk cautiously to a small green circle' or 'pull a blue square while zigzagging'. Below, we show a step-by-step example in which we produce the command 'walk to a square'.

First, we initialise the sentence to $ROOT$, an empty string (step 1). We start from the first rule, after which $a = VP$ (step 2). VP denotes a syntactic unit, namely the verb phrase: we therefore find a rule below which has VP on its left-hand side (step 2a). We pick the rule $VP \rightarrow VV_t\ DP$. Now, $a = VV_t\ DP$. VV_t is the set of transitive verbs: we can pick one from there, 'walk', add it to the sentence, and remove VV_t from a (step 2b). Now $a = DP$. DP is a syntactic unit, namely the determiner phrase: we find a rule below which has DP on its left-hand side (step 2a). The only available rule is $DP \rightarrow DT\ NP$. Now $a = DT\ NP$. DT is the set of determiners that contains only the article 'a', which we add to the sentence (step 2b). Now $a = NP$, and NP is a syntactic unit, namely the noun phrase. We find an NP rule below: $NP \rightarrow NN$ (step 2b). Now $a = NN$, which is the set of available names, from which we pick 'square' and add it to our sentence (step 3). After this, a is null: we can stop. Our sentence is 'walk to a square', which is valid according to the grammar of the gSCAN.

The grammar of the target action sequences is much simpler. The available actions are 'walk', 'stay', 'turn left', 'turn right', 'push' and 'pull' and they can be concatenated in any order.

4.4 The gSCAN semantics

Having shown which verbs, adverbs, adjectives and nouns can appear in the commands of the gSCAN, it is now necessary to specify the meaning of these words. Table details the semantics of gSCAN.

Word	**Meaning**
walk	moving somewhere using actions 'walk', 'turn left', 'turn right'
push	pushing an object until unobstructed using action 'push'. Objects of size 3 and 4 require two 'push' actions to move one cell, while smaller objects only require one
pull	pulling an object until unobstructed using action 'pull'. Objects of size 3 and 4 require two 'pull' actions to move one cell, while smaller objects only require one
hesitantly	stopping every time after having moved one cell using 'stay'
cautiously	looking right and left before moving one cell, using sequence 'turn right', 'turn left', 'turn left', 'turn right'
while spinning	performing a 360 degree spin before moving one cell, using sequence 'turn left', 'turn left', 'turn left', 'turn left'
while zigzagging	alternating horizontal and vertical movements until in line with the target object, then proceeding straight to it

Table 4.1: Important gSCAN semantics

4.5 Systematic compositional generalisation tests

The grid-world situation enriches the semantics of the dataset by giving a learning agent an opportunity to ground the meaning of the commands. This enlarges the set of systematic compositionality tests that can be designed for deep learning algorithms. In particular, the gSCAN dataset can be split between training set and test set in ways that allow the tests that we are going to detail below.

4.5.1 Novel composition of object properties

To pass this test, a neural model has to be able to recombine known colours and shapes in order to recognise an unseen coloured object. So, this test requires a model to perform a novel composition of object properties. There are two kinds of compositions that are being tested: composition of references and composition of attributes.

By references we mean how the object properties are referred to in the training set. In the composition of references test (Figure **??**), a model never observes a command that references a selected target object with the combination of colour and shape. Such combination is used only to reference the other kinds of target objects. At test time, the model is asked to interact with this object being referred to with both its colour and shape. For example, if the target object selected for the test is a yellow square, the training set contains references to this object such as 'a square', 'a big square' and 'a small square'. The yellow square is never being referred to as 'yellow'. However, the model can observe other target objects being referred to as 'yellow', for example 'a yellow circle' or 'a big yellow cylinder'. At test time, the model needs to generalise to the composition of 'yellow' and 'square' without having learned how to ground this complex expression.

In the composition of attributes test (Figure 4.6), we are testing whether the model is able to create useful abstract of properties and to recombine them at test time to interact with unknown objects. To do so, we hold out from the training set all commands where a certain coloured object is the target. For example, if the target is a red square, there are no commands in the training set that require the agent to interact with such object. The red square may appear as a non-target object, but the model is not required to interact with it. At test time, the model is presented with commands where the red square is the target object. To correctly identify the target object, a model needs to have built meaningful abstraction of its properties by observing them in other objects. In the case of the red square, it needs to have an understanding of the colour red by having observed other red objects, and an understanding of 'square' by having observed blue and green squares. This amounts to systematic compositional generalisation, as there is a recombination of known parts to understand an unknown expression.

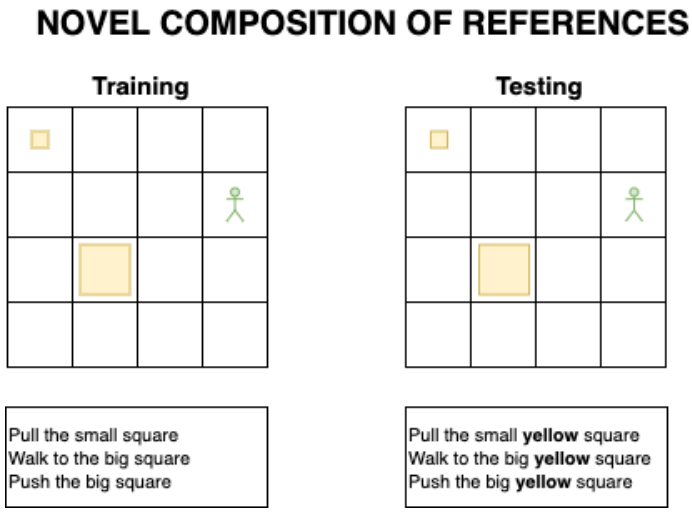

Figure 4.5: Visual example of the novel composition of references test. At training time, only commands not containing the adjective 'yellow' appear. At test time, the adjective is being used.

4.5

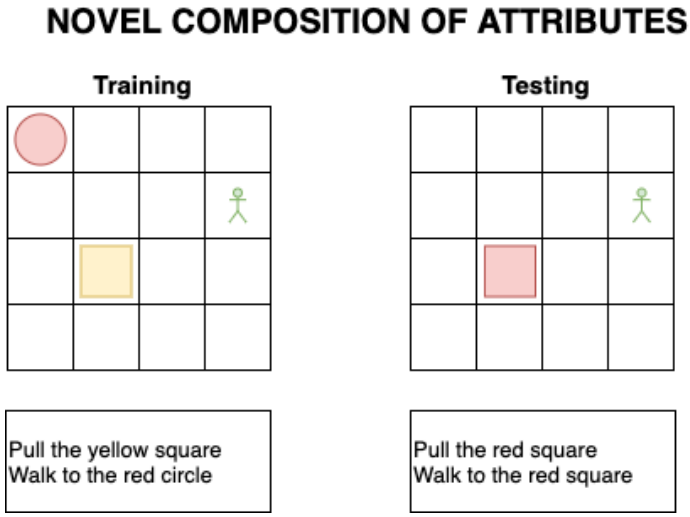

Figure 4.6: Visual example of the novel composition of attributes test. At training time, only commands not pertaining a red square are seen. At test time, the red square is the target object.

4.6

4.5.2 Novel direction

To pass this test, a neural model has to be able to recombine known directions in order to be able to move in a new one. So, this test requires a model to produce an understanding of a novel direction. At training time, no commands require navigation in a selected combined direction. For example, if the selected combined direction is south-west, the training set may contain commands asking the agent to move to the north, south, west, east, north-west, north-east and south-east. At test time, the target object may be located south-west of the agent. This requires the

model to abstract the semantics of the directions observed in a way that captures not only the meaning of the primitive ones (north, south, east and west) but also the effect of combining the primitives (north-east, north-west and south-east). If the model is able to extract the correct rule, it should succeed in combining south and west.

4.5.3 Novel contextual references

In this test, a model is asked to apply known concepts in a new context. More specifically, the new context presents different size relations between the objects in the world. At training time, the agent does not interact with circles that are of size 2, are the smallest circles in the world and are referred to as 'small'. At test time, the agent has to interact with circles of size 2 correctly being referred to as 'small' (Figure 4.7). In other words, a model has not encountered the expression 'a small circle' and cannot ground it to an object. In order to understand it, the model has to grasp what 'small' refers to in the expressions it encountered. In particular, it has to grasp that the adjective expresses a relative concept: an object of the same size can be referred to as the small one or the big one, depending on the objects that surround it.

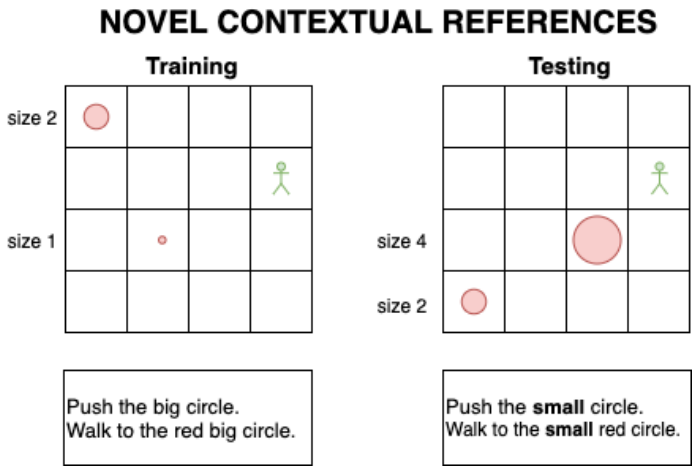

Figure 4.7: Visual example of the novel contextual references test. At training time, only commands not referring to a circle of size 2 as 'small' appear. At test time, the size 2 circle is being referred to as 'small'.

4.7

4.5.4 Novel composition of actions and arguments

Taking advantage of the semantics of the gSCAN as detailed in Table 4.1, this test requires a model to correctly classify objects into two classes: light and heavy. This decision has to be taken by learning that pushing and pulling commands require twice the amount of 'push' and 'pull' actions on objects of size 3 or 4. To push an object of size 1 or 2 one cell forward, the agent has to perform one 'push'. To push an object of size 3 or 4 one cell forward, the agent needs to do two 'push' actions.

In this test, the training set does not contain examples where the agent is asked to push a square of size 3. However, the model observes examples where the command requires to pull the size 3 square. At test time, the model needs to be able to infer that the square must be pushed twice for it to move by one cell.

4.5.5 Novel adverbs

Two tests pertaining adverbs are available in the gSCAN dataset. First, there is a few-shot learning test, in which a model only observes a few examples containing the adverb 'cautiously' and is required to generalise to all other uses of the adverb.

The second test is demands a model to combine a known adverb with a known verb, after having trained on examples that did not exhibit that combination. In fact, in this test the training set does not present any examples which contain both 'pull' and 'while spinning' in their command sequence. At test time, the model needs to translate commands which contain 'pull' + 'while spinning'.

4.6 Productive compositional generalisation tests

Like the SCAN, the gSCAN dataset also contains a test which demands the model to perform productive compositional generalisation by presenting, at test time, longer sequences than the ones presented at training time. At training time, the model sees only commands which generate action sequences of length ≤ 15. Then it is tested on commands whose corresponding action sequences are of length ≥ 15 and its performance is registered for increasingly longer sequences.

4.7 Dataset statistics

Between the systematic compositional generalisation splits and the longer sequences split, gSCAN has 548,234 training examples and 57,066 test examples. The systematic compositional splits have 367,933 training examples and 19,282 test examples. The longer sequences split has 180,301 training examples and 37,784 test examples.

5 The gSCAN neural baseline

The authors of the gSCAN design a deep neural network using state of the art techniques, and observe its performance on the compositional tests they proposed []. In what follows, we describe the architecture of the network and report its performance on the splits that test for compositional reasoning. We then perform an ablation study, to gain a better understanding of the role played by its individual parts. To do this, we design a set of further experiments which can be run to shed further light on compositional reasoning skills of the model.

5.1 Architecture

The baseline model is a sequence-to-sequence deep neural network (Sutskever et al., 2014) whose architecture has been augmented with a visual encoder module. The model has a reccurrent command encoder that processes command sequences, while it uses the visual encoder to deal with the world situations associated with each command. The output of both encoders is passed to a reccurrent decoder module which outputs action sequences using an joint attention mechanism over the parts of the commands and the cells of the grid-world. Because the input to the network is of two different data types, textual and visual, the network is said to be a *multimodal* neural network.

Input to the network: tuple $x = (x^c, X^s)$, where $x^c = \{x_1^c, ..., x_n^c\}$ is a command sequence, and $X^s \in \mathbb{R}^{d \times d \times c}$ is a representation of the world state in a $d \times d$ grid-world.

Output of the network: action sequence $y = \{y_1, ..., y_m\}$ modeled as

$$p_\theta(y|x) = \prod_{j=1}^{m} p_\theta(y_j|x, y_1, ..., y_{j-1})$$

Command encoder:
The command encoder consists of a bidirectional LSTM [18] which embeds each word from the command sequence into a vector, outputting an embedding sequence $h_c = \{h_1^c, ..., h_n^c\}$. In Figure 5.1, we denote the command encoder as $h_c = f_c(x^c)$.

State encoder:
The state encoder is a convolutional neural network with three kernel sizes (Wang Lake, 2019), which outputs a representation of the $d \times d$ grid-world state $H^s \in \mathbb{R}^{d \times d \times 3c_{out}}$, where c_{out} is the number of feature maps for each kernel size. We denote

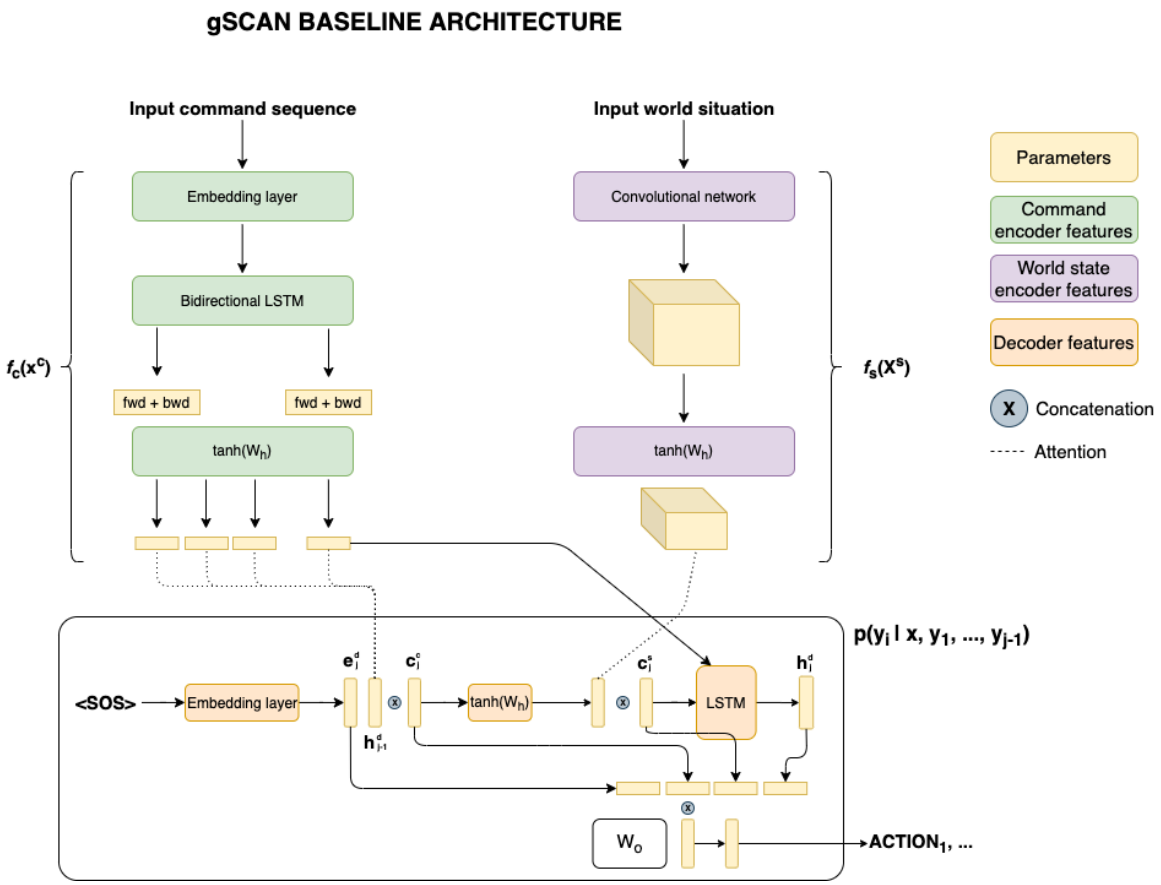

Figure 5.1: Visual representation of the gSCAN baseline architecture.

5.1

the state encoder as $H^s = f_s(X^s)$ (Figure 5.1).

Decoder:
The decoder learns to output an appropriate distribution over action sequences given the outputs of the state and command encoders, $p(y|h^c, H^s)$. At each step, the decoder embeds the previous symbol y_{j-1} into $e_j^d \in \mathbb{R}_e^d$.

Through double attention [38], the previous state of the decoder h_{j-1}^d is used to produce the context vectors for both the command and the world situation. The command context vector c_j^c is computed as $c_j^c = Att(h_{j-1}^d, h^c)$, consisting in a weighted average over h^c obtained by attending over the steps of the command sequence. The world state context vector c_j^s is computed as $c_j^s = Att(h_{j-1}^d, H^s)$, consisting in a weighted average over H^s obtained by attending over the grid locations of the world.

These variables are then used, together with the previous state h_{j-1}^d, to produce the current state h_j^d as

$$h_j^d = LSTM([e_j^d; c_j^c; c_j^s], h_{j-1}^d)$$

Finally, the action is outputted as $p(y_i|x, y_1, ..., y_{j-1}) = softmax(W_o[e_j^d; h_j^d; c_j^c; c_j^s])$.

We visually represent this architecture in Figure 5.1.

5.2 Forward pass

The following equations represent a full pass forward through the baseline model.

Encoder

Command encoder $h^c = f_c(x^c)$: $\forall i \in \{1, ..., n\}$

$$e_i^c = E_c(x_i^c)$$
$$h_i^c = LSTM_{\phi 1}(e_i^c, h_{i-1}^c)$$

State encoder $H^s = f_s(X^s)$:

$$H^s = ReLU([K_1(X^s); K_5(X^s); K_7(X^s)])$$

$E_c \in \mathbb{R}^{|V_c| \times d}$ is the lookup table of embeddings with input vocabulary V_c and embedding dimension d. K_k is a convolution with kernel size k. $H^s \in \mathbb{R}^{d \times d \times 3c_{out}}$.

Decoder

$p(y_i|x, y_1, ..., y_{j-1})$: $\forall j \in \{1, ..., m\}$

$$h_0^d = W_p h_n^c + b_p$$
$$e_j^d = E^d(y_{j-1}$$
$$h_j^d = LSTM_{\phi 2}([e_j^d; c_j^c; c_j^s], h_{j-1}^d)$$
$$o_j = W_o[e_j^d; h_j^d; c_j^c; c_j^s]$$
$$\hat{o}_j = p_\theta(y_j|x, y_1, ...y_{j-1}) = softmax(o_j)$$
$$\hat{y}_j = \arg\max_{V_t}(\hat{o}_j)$$

Textual attention $c_j^c = Att(h_{j-1}^d, h^c)$: $\forall i \in \{1, ..., n\}$

$$e_{ji}^c = v_c^T tanh W_c[h_{j-1}^d; h_i^c]$$
$$\alpha_{ji}^c = \frac{exp(e_{ji}^c)}{\sum_{i=1}^n exp(e_{ji}^c)}$$
$$c_j^c = \sum_{i=1}^n \alpha_{ji}^c h_i^c$$

Visual attention $c_j^s = Att([c_j^c; h_{j-1}^d], H^s)$: $\forall k \in \{1, ..., d^2\}$

$$e_{jk}^s = v_s^T tanh W_s[h_{j-1}^d; c_j^c; h_k^s]$$

$$\alpha_{jk}^{s} = \frac{exp(e_{jk}^{s})}{\sum_{k=1}^{d^2} exp(e_{jk}^{s})}$$

$$c_j^s = \sum_{k=1}^{d^2} \alpha_{jk}^{s} H_k^s$$

Where:

$W_c \in \mathbb{R}^{h_d \times (h_e + h_d)}$
$v_c \in \mathbb{R}_d^h$
$W_s \in \mathbb{R}^{h_d \times (3c_out + h_d)}$
$v_s \in \mathbb{R}_d^h$
$W_p \in \mathbb{R}^{h_d \times h_e}$
$W_o \in \mathbb{R}^{|V_t| \times (d_e + 3h_d)}$

h_e is the textual encoder hidden size
c_out is the visual encoder channels
h_d is the decoder hidden size
V_t is the target sequence vocabulary
d_e is the target embedding dimension.

5.3 Training

Training details are as follows:

- **Loss:** cross-entropy
- **Optimizer:** Adam [39]
- **Learning rate start:** $1e-3$
- **Learning rate decay:** 0.9 every 20,000 steps
- **Batch size:** 200
- **Training iterations:** 200,000

5.4 Results on the generalisation tests

In Table 5.1, we report the performance of the baseline model described above on the gSCAN generalisation tests introduced in section 4. The metric we use is exact match accuracy, which measures the percentage of output action sequences that were entirely correct, matching the ground truth. For comparison, we provide exact match accuracy for a random split of the dataset, where there is no systematic compositional reasoning required.

Split	**Exact match accuracy**
Random	97.69% $\pm$ 0.22
Composition of references	54.96% $\pm$ 39.39
Composition of attributes	23.51% $\pm$ 21.82
Novel direction	0%
Novel contextual references	35.2% $\pm$ 2.35
Composition of actions and arguments	92.52% $\pm$ 6.75
Few-shot learning of adverbs	0%
Combining known adverbs	22.70% $\pm$ 4.59
Longer sequences (length ≤ 15)	94.98% $\pm$ 0.12
Longer sequences (length = 16)	19.32% $\pm$ 0.02
Longer sequences (length = 17)	1.71% $\pm$ 0.38
Longer sequences (length ≥ 18)	$\leq 1\%$

Table 5.1: gSCAN neural baseline results on the tests proposed by [16]. Metric used is exact match accuracy $\pm$ standard deviation.

The authors provide a detailed analysis of the results for each of the tests [16]. In the case of the composition of references, yellow squares have been observed in training but never referred to with their colour attribute. The authors hypothesise that the model overfits to the expression pattern '(small/big) square'. In the composition of attributes test, the model has failed to construct a compositional representation of the object 'red square', which has been observed in training only as a background object and never as a target. When breaking down the average exact match accuracy for each of the ways a red square is referred to in the test set ('small/big red square', 'small/big square', 'red square', 'square'), the authors noticed that the accuracy is low for all the expressions, except for when the red square is being referenced as just 'square'. In this case, the model manages to achieve a 53% exact match accuracy. However, given how the gSCAN world states are generated, the model truly has a 50% chance of picking the right object in that case, as there are only two placed objects.

For novel direction, the baseline completely failed the test. However, by looking at the attention of the agent, we know that the agent has correctly identified the target object most of the times, thus knowing where it needs to navigate to. However, it is unable to reach that grid: when tested on the 'south-west' direction, it moves all the way west or south but fails to then turn and complete the navigation.

For novel contextual references, the baseline fails to understand the relative concept of smallness, and only performs better when in addition to 'small', the expression also contains a colour attribute, giving the model a 50% change of picking the right object.

In the case of composition of actions and arguments, the model successfully understands how to push a big object having learned that to pull it, an agent requires

double the amount of actions. So in this case, the model has correctly generalised to the novel verb-noun pair.

The few-shot learning test completely fails. Only one example of 'cautiously' has been observed during training, and at test time the agent is presented novel commands with this adverb. Even increasing the number of examples in the training set, up to 50, only increases exact match accuracy to around 4%.

Combining known adverbs and known verbs is also something that the model fails to achieve. In this test, the authors have observed that performance drastically drops when the target sequence length increases, suggesting that the baseline is not able to handle longer sequences with unfamiliar combinations.

Finally, in the longer sequences test, the model trained on command lengths up to 15 performs well on the same length it has been trained on. When the target sequence is even slightly longer (16 instead of 15), the performance plunges of more than 75%.

In summary, the model has shown to be unable to reason compositionally. It cannot understand new concepts by means of recombination of known parts. When it does grasp a concept the understanding it has formed is very shallow and does not stand even slight modifications to the command sequence. The only test where the model achieves a high performance is the composition of actions and arguments. This can at best show that at least some kind of compositionality is achieved by the model, but the needed compositional reasoning skills to solve the gSCAN benchmark remain in most part lacking.

5.5 Ablation study on the baseline model

In order to shed light on how the architecture of the model contributes to its behaviour, we perform an ablation study on the baseline. An ablation study is the examination of the behaviour of a neural network by removing one or more of its parts, in order to observe the contribution that each part makes to the overall performance. Because the baseline model is a multimodal neural network which inputs both images and text sequences, we can take advantage of this and assess how much the visual data is really contributing to the results we have reported above. After all, the gSCAN was introduced as an improvement over the SCAN dataset to demonstrate that even those recent successful attempts at the SCAN benchmark did not display compositionality, but were exploiting limitations of the dataset to capture the central aspects of compositional generalisation. Their argument relies on the assumption that the grounding in the grid-world significantly contributes to the behaviour of their baseline, but this remains to be established.

To do this, we nullify H^s, the representation of the grid-world produced by the state encoder, by substituting the numerical values with all zeros. By doing this, we are ef-

fectively preventing the network from encoding and accessing any information about the grid-world paired with the input command.

We performed the ablation study on the novel composition of attributes split, where the model does not see a red square as target object during training but is required to interact with it during testing. We trained with the same settings as detailed in Section 5.3. After testing we concluded that the model is completely unable to respond to commands that require the agent to interact with a red square, with a 0% exact match accuracy for the output action sequences.

In taking a closer look at the output action sequences, we found that the model outputs an action sequence which attempt to move the agent from its position to one of the corners of the grid world. The corner depends on the adverb that appears in the command: it seems that the model has learned to move to the bottom right corner when there is a 'hesitantly' in the command (Figure 5.2), while for all other adverbs, as well as commands that do not have adverbs, the corner is the top right (Figure 5.3).

In addition to not having learnt where to move the agent, the model has not learned how to navigate the world: specifically, the agent does not recognise the limit of the world and can be observed reaching the top-right corner and attempting to move out of the grid with further 'walk' actions (5.4). Sometimes the action sequence outputted, which is of variable lengths, is not enough to move the agent to the top-right corner, and the agent is seen moving towards that location but stopping before.

Interestingly, the agent seems to be performing the action of moving to the top-right corner of the world according to the adverb that accompanies the command. For example, if the command is 'walk to the red square while zigzagging' the agent is observed to alternate vertical and horizontal movements to reach the top right corner. Similarly, for 'while spinning' the agent has learned how to move to the top right corner while turning 360 degrees at each step. This happens even when the agent is attempting to walk off the grid-world, not stopping at the top-right corner but continuing to move against the wall. We can observe the agent attempting to move somewhere beyond the limit of the world, but in the fashion specified by the adverb (Figure 5.4). Therefore, given that the textual data was all that the model had to work with, it seems to have well captured the meaning of adverbs. However, the lack of image data has proven to be fatal for the model's attempt at the sequence to sequence task.

5.6 Further experiments on the baseline model

Next, we propose a set of experiments which build on the ablation study by testing the effects on the results when the model is provided with progressively more information about the world, while keeping the images null. To do this, we start by

providing the model with a basic textual caption of the world state image, describing the location of the objects and the agent, without reporting colours or sizes, but only shapes. With the next two experiments, we add information about colours and sizes one at a time. For our fourth experiment, we add all the information we have, including both colour and size of all the objects.

The textual descriptions could be embedded using a tool such as FastText [40], which produces vectors for each word in the description. Following the ideas introduced in Section 3.1 with the distributional semantics hypothesis, we represent the binding of words that have to be understood compositionally by the model by adding the vectors of the words involved, as proposed by [24]. For example, if the point of the test is to assess whether the model can understand 'red square' compositionally, we provide the model with an embedding of the textual description of the grid-world state where the vectors for red and square have been added together. If the test requires to understand 'walk cautiously' compositionally, we give the model the compositional representation of this expression by adding vectors for walk and for cautiously in the embedding of textual description.

We provide the code for augmenting the gSCAN dataset with these textual descriptions. We consider these experiments a conceptual contribution of this thesis, but we left the testing as an extension of the project, in order to focus more on the modular approach to the gSCAN.

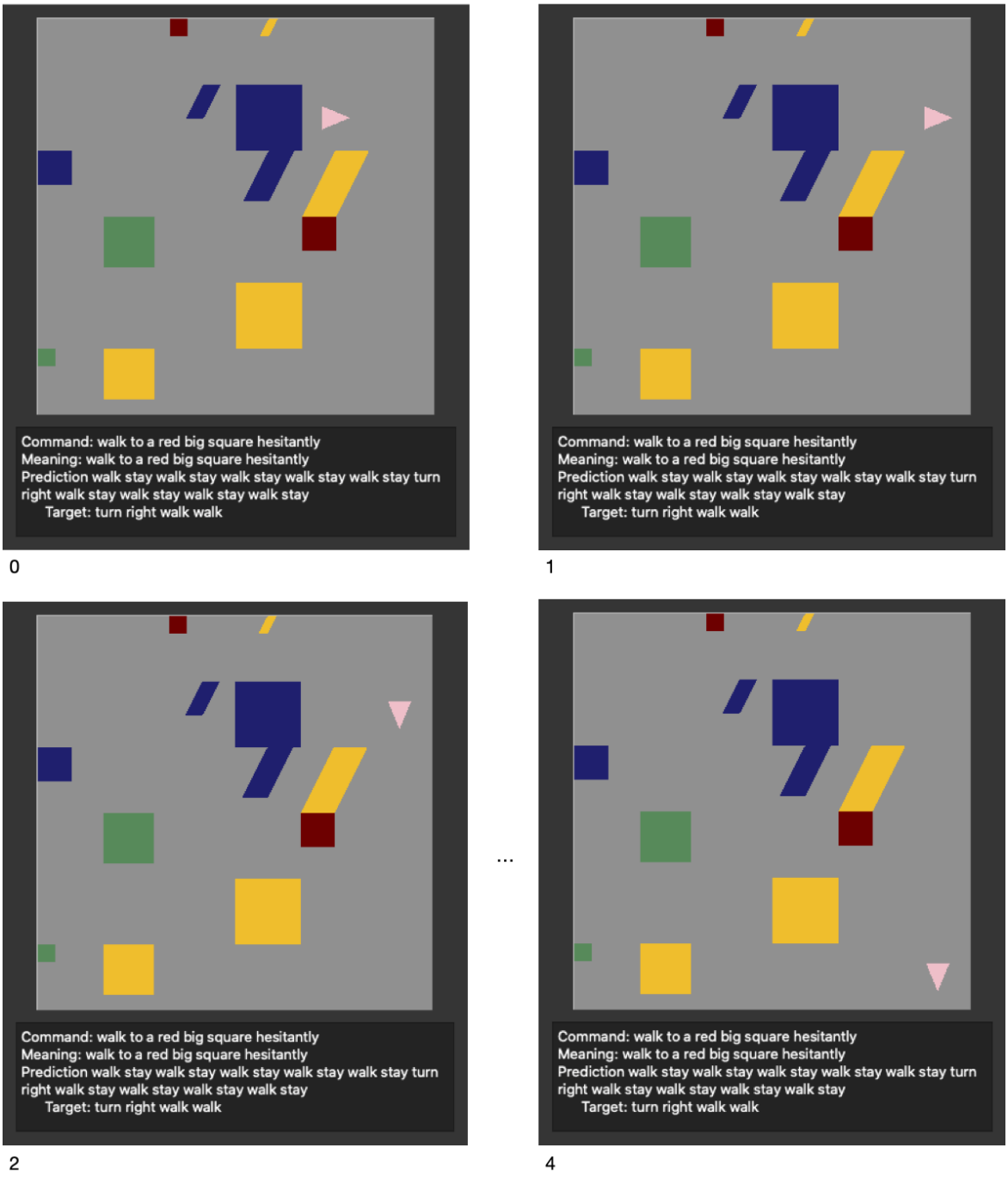

Figure 5.2: Example result of ablation study on visual split. In all cases where the command contains the adverb 'hesitantly', the agent moves to the bottom right corner. We report four snapshots of the movement at times $t = 0$ (initial situation), $t = 1$ (after one action), $t = 2$ and $t = 4$ (the final action).

5.2

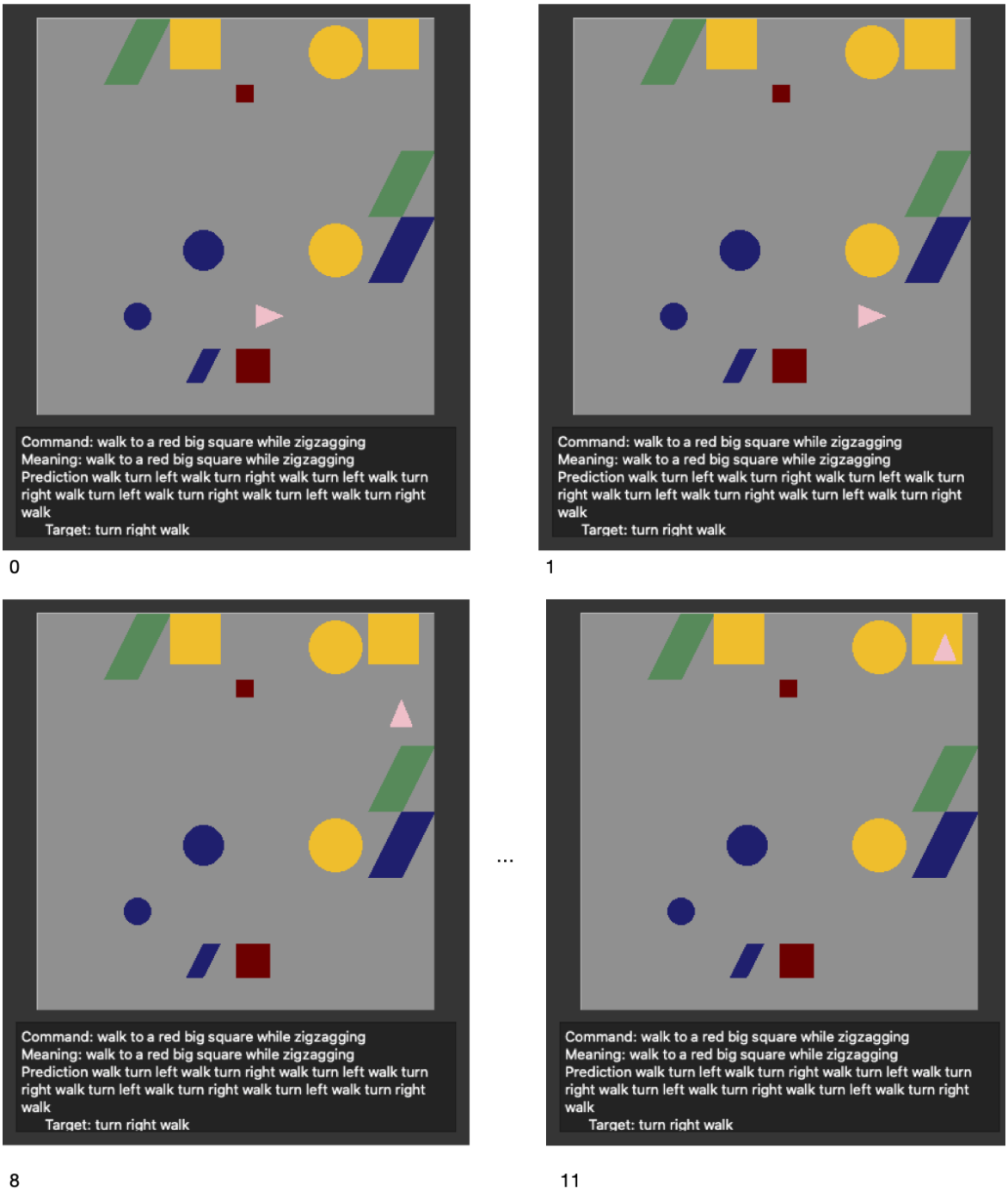

Figure 5.3: Example result of ablation study on visual split. In all cases where the command contains the adverb an adverb other than 'hesitantly' or does not contain any, the agent moves to the top right corner. We report four snapshots of the movement at times $t = 0$ (initial situation), $t = 1$ (after one action), $t = 8$ and $t = 11$.

5.3

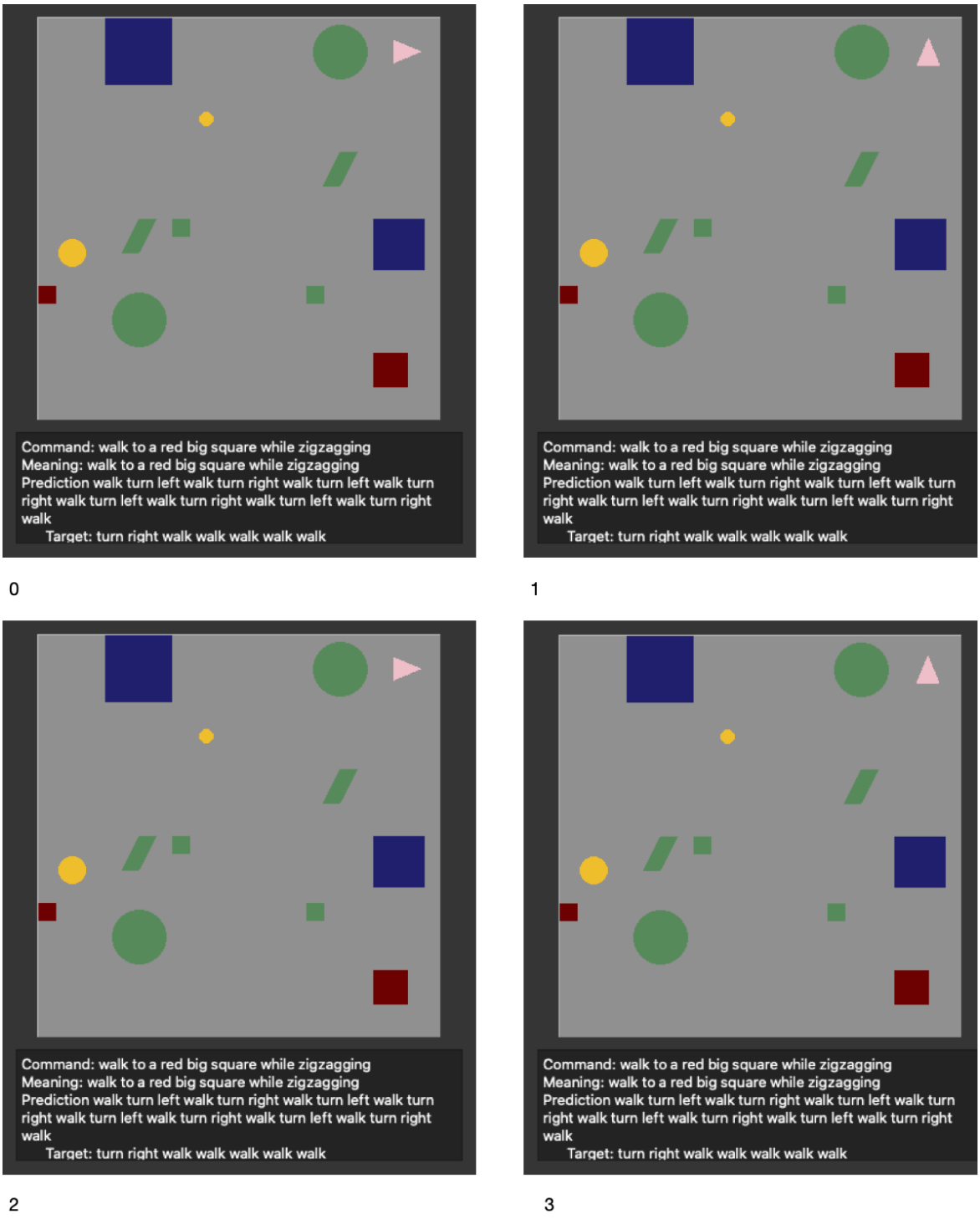

Figure 5.4: Example result of ablation study on visual split. In all cases where the action sequence has more actions than the agents needs to reach the (wrong) destination, it uses the extra actions to attempt to walk off the grid. We report four snapshots of the movement at times $t = 0$ (initial situation), $t = 1$ (after one action), $t = 2$ and $t = 3$. Notice how the agent is trying to walk off the grid while zigzagging: it has learned how to navigate in this way but it has not learned that the grid-world has a limit.

5.4

6 Modular neural networks

In this section we discuss modular neural networks. We begin by observing how modularity has the potential of achieving compositional reasoning in biological neural computation systems: this potential inspired the deep learning research to attempt to do the same with artificial neural networks. After having presented the motivations for this design approach, we provide an overview of some popular modular architectures. We then focus on the recently introduced Neural Module Networks for visual question answering, whose purpose is precisely to enable compositional reasoning in this domain of machine learning. Having provided this overview of modularity in deep learning and its potential for compositional generalisation, we can then present our own modular network for the gSCAN benchmark in the next section.

6.1 Motivation

Much like deep neural networks, modular neural networks have been inspired by neuroscientific studies of the central nervous system in the human brain. [41] found that the central nervous system does not have a monolithic structure, but is in fact a highly modular system in which each region (including the midbrain, the diencephalon, the cerebellum and many others) specialises in a particular function. All regions work in parallel, but are interconnected so that they can cooperate, each with their specialised function, to solve complex tasks. There is therefore a divide-and-conquer approach: a complex task is divided into smaller simpler subtasks for which the different regions of the nervous system know the solution, and the solutions of each region are then combined to solve the bigger task. This means that not all regions of the central nervous system are necessarily involved when presented with a stimulus, but only the ones that are specialised in responding to it. This is proved by the fact that we can observe different response patterns in the nervous system, given different cognitive tasks. In contrast, in a monolithic structure, each part is always involved when presented with a task because the solution is being attempted by having all parts contribute towards the same goal.

As also discussed in the introduction of this thesis, this divide-and-conquer approach amounts to compositional reasoning, because it presupposes that a task is understood in terms of its parts. Moreover, by solving the complex task in this way, one is implicitly recognising that the sum of all the parts determines the whole. It could also be argued that this is a compositional reasoning of the systematic kind, because it involves the recombination of known constituents (the sub-solutions which are

solvable by the different regions of the nervous system) to solve a task whose solution would be unknown otherwise. The nervous system is not train in a monolithic fashion: therefore, it does not know the solution for the complex task. Yet, it can generalise to this unknown task by recombining its modules and solving in a compositional manner.

Therefore, a modular system design enables compositional generalisation in human learning: if we were able to implement a similar structure in deep learning, neural computation could also achieve compositionality.

6.1.1 Other advantages of modular learning

Modular neural networks have many other advantages over monolithic algorithms. Their divide-and-conquer approach, apart from enabling compositional reasoning, also alleviates the optimization needed for monolithic learning algorithms, without sacrificing learning ability. Moreover, modularity can help a neural network achieve transfer learning: previously learned knowledge can be re-utilised for new tasks, instead of having to retrain the whole network each time we change the learning goal. Finally, the lower complexity of their structure makes them more scalable and adaptable networks [42].

6.2 Modular deep learning architectures

There are various possible implementations of modular neural networks in deep learning. Generally, we can divide modular architectures into two classes, according to how their units, henceforth modules, are interconnected: the modules can be either tightly coupled or loosely coupled. Tightly coupled modular neural networks jointly train all their modules at once: this means that during the learning stage the modules are required to learn not only their own specialised function, but also how to interact with each other. A loosely coupled modular neural network proceeds to train its modules in a sequential fashion. This taxonomy of modular neural networks is due to [43]. In what follows, we present some popular example architectures from both the tightly and loosely coupled classes.

6.2.1 Mixture of Experts

In the mixture of experts approach [44] each module of the neural network competes with the others to be selected for processing each input in the input space. The network is a tightly coupled modular neural network designed so that similar inputs are mapped to similar expert modules, effectively making each module specialise in a different kind of operation. Once a module is selected for a given input, there are two options for the way the input is processed: either the input is processed only by the expert selected (winner-takes-all), or all modules contribute to transforming the input, but the expert selected has more influence on the transformation.

This is achieved by the following architecture. A mixture of experts network has a set of N available modules and a controller, as illustrated in Figure 6.1. An n-th expert module is given an input x and produces an output vector $o_n(x)$. The controller, given an input x, produces N scalar outputs that indicate the importance given to each expert module. The output of the network is a weighted sum of the outputs of all the experts, weighed by their corresponding importance.

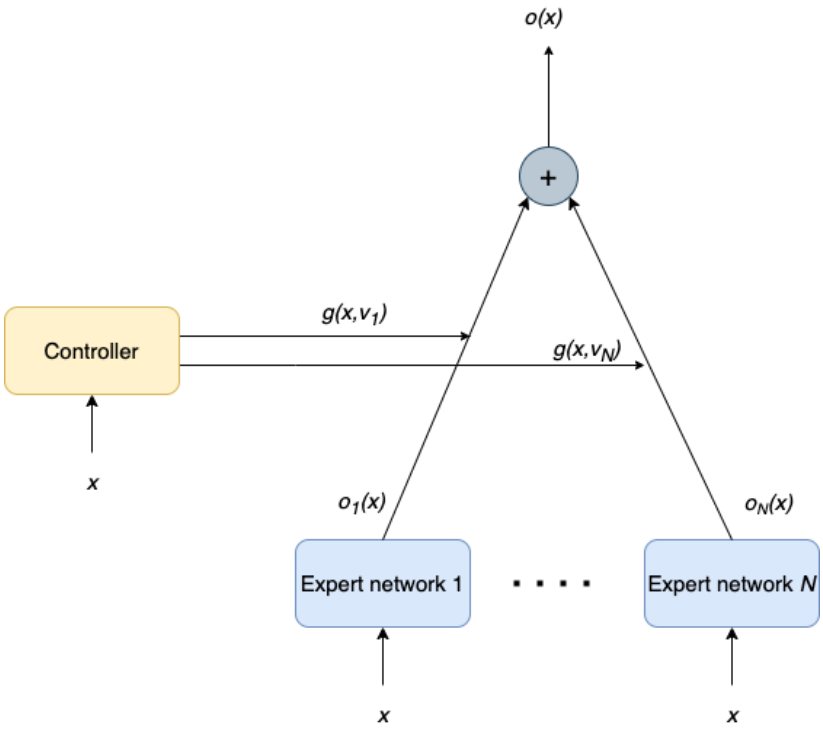

Figure 6.1: Architecture of the mixture of experts model.

6.1

A popular choice for the expert modules is using linear neural networks [44]. In such case, the output of the n-th expert for an input x is

$$o_n(x) = f(W_n x + b)$$

where W_n is the weight matrix, b is the bias and f is the transfer function. The controller can also be implemented using a linear neural network. In such case, its n-th output is a softmax of v_n^T

$$g(x, v_n) = \frac{e^{v_n^T} x}{\sum_{k=1}^{N} e^{v_k^T x}}$$

where v_n scalar output determining the importance of the n-th expert. Then the output of the whole neural network for input x is the weighted sum

$$o(x) = \sum_{n=1}^{N} g(x, v_n) o_n(x)$$

The mixture of expert can also be interpreted in probabilistic terms [44]. Given an input-output pair (x, y), the values of $g(x, V)$ can be seen as multinomial probabilities associated with the decision of which experts to use, where V is the parameter matrix of the controller containing the scalar outputs which assign importance to the experts. The output is chosen from a probability distribution $P(y|x, W_n)$. This means that the probability of outputting y given x is obtained by mixing the probabilities of outputting y from each module distribution. The proportions of this mixture are regulated by a multinomial distribution

$$P(y|x, \Theta) = \sum_{n=1}^{N} g(x, v_n) P(y|x, W_n)$$

where Θ represents all the parameters in the neural network.

6.2.2 Partition based modules

An example of a loosely coupled modular design is the partition based approach, in which the network makes explicit use of the divide-and-conquer principle in order to find a modularisation strategy that yields correct outputs. To do this, the network performs two steps: the dividing step and the conquering step.

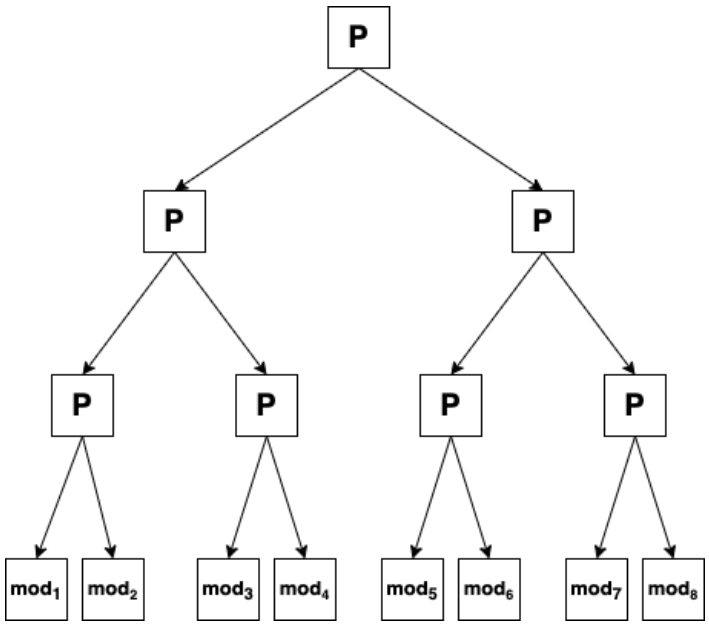

Figure 6.2: Example tree-structure for a partition based modular neural network with eight preselected modules.

6.2

In the dividing step, the input space is partitioned into overlapping subspaces. This is done recursively until for each subproblem defined in the subspaces so generated, there is a module that is able to find the solution. In the conquering stage, the neural

network completes the learning subtasks in the subspace of the input space generated during the diving step.

The result is a modular neural network whose structure can be represented as a tree. The partitioning mechanism P is placed at non terminal nodes of the tree, and the modules work at the leaves, as shown in Figure 6.2.

The modules that specialised on the subspaces can be either pre-selected or can be created as needed according to the following algorithm:

Algorithm 6.2.2 Growing algorithm Given a training set $\mathfrak{D}$, set the training subset $X \leftarrow \mathfrak{D}$. Initialize all modules with random parameters and define a learnable partitioning mechanism, which regulates the partitioning of a training set into a subset, and learnable compatibility criteria, which define whether a set of examples is compatible with the current abilities of the neural network.

1. **Compatibility test step:** for training subset X determine whether the learning task defined on X can be solved by one of the modules of the network using the compatibility criteria

2. **Partitioning step:** if no modules can solve the task defined on X, train the partitioning mechanism to partition X into two overlapped subsets X_l, X_r. Perform a compatibility test on both X_l and X_r, setting $X \leftarrow X_l$ and $X \leftarrow X_r$ in turn. Otherwise, perform the subproblem solving step.

3. **Subproblem solving step:** train the module on the current X and place the module on the current leaf of the tree.

Notice how this is a loosely couples algorithm as each module is trained in turn and not jointly.

6.2.3 Neural module networks for visual question answering

A recent example of a modular network architecture is the neural module networks introduced by [14] to solve tasks in the visual question answering domain. More precisely, neural module networks are partition based modular neural networks that partition the input question space of visual question answering into a set of overlapping subsets, which allows for each of the textual questions to be understood as a series of sub questions.

The modules of the neural network are either pre-defined, in the case of earlier approaches [14], or learned, as in more recent proposals [13]. Let us look at an example partition to understand the functioning of these neural module networks. Let's assume that the dataset consists of a number of questions, each associated to an image that describes a grid world on which coloured objects of various shapes are placed. Let's assume that the network has partitioned the database so that the questions can be understood in terms of three functions:

- $find()$, which given a shape it outputs the areas of an image where the shape is found
- $under()$, which given an area of an image it outputs the area directly below it
- $and()$, which given two areas of an image it outputs their intersection
- $describe()$, which given an area of an image, it outputs the dominant colour

Then the neural module network has available four modules whose abilities correspond to executing these four functions on an image datapoint. Given an input question, such as 'Are all squares positioned under a triangle blue?', the network can understand the question as the complex function

$$describe(and(under(find(triangles)), find(squares)))$$

which means that we find the intersection between areas where we see squares and the areas of the image under triangles, finding all squares under triangles. Then we describe the dominant colour. We illustrate this example in Figure 6.3. Using this functional understanding, the network assembles on-the-fly a composition of its modules and provides the output answer by combining the output of all modules.

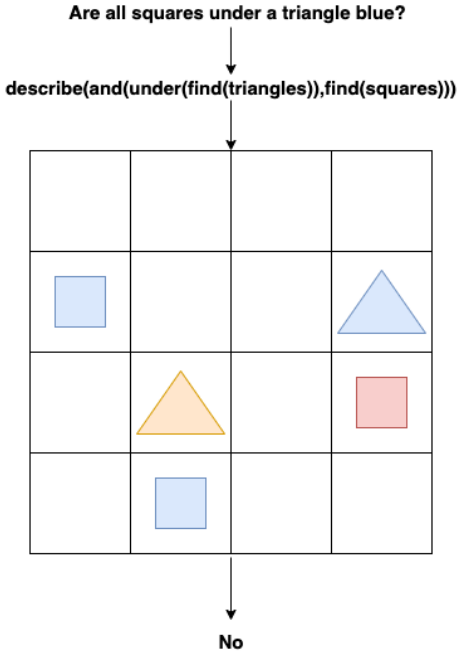

Figure 6.3: Example resolution of a neural module network for visual question answering. First the question is partitioned into a composition of functions that match the network abilities. Then the image data is consulted according to this function composition: in this case, we first find all the square, then we find all the triangles and look under them. We then take the intersection of these areas to obtain the areas where the squares are under triangles. We then describe the colour and can provide an answer to the question.

6.3

This approach allows the neural network to perform a remarkable amount of compositional reasoning on various visual question answering datasets, including SHAPES [14], CLEVR [45] and VQA [46], outpacing other state-of-the-art methods and reducing their error rate by 50% on compositional tasks in the CLEVR dataset.

7 Modular architecture for the gSCAN benchmark

We propose a modular architecture based on the mixture of expert techniques and we aim to use it for the gSCAN benchmark, wanting to assess its ability to enable compositional reasoning. We take the mixture of expert model and we embed it in a modified version of the architecture proposed by the authors of the gSCAN: in other words, our goal is to introduce modularity in their baseline model.

The first modification we make to the structure is that we utilise a GRU [19] instead of an LSTM [18] in the command encoder. The reason for this is that we wan to endow this precise part of the network with modularity, and GRUs are simpler than LSTM, and thus easier to modify. The intuition behind wanting to introduce modularity in the command encoder is that we need compositionality in how the language of the gSCAN is understood. Our goal is to map different syntactical elements of this language to similar modules: we would like to capture how, e.g., adverbs modify a verb, and we want this understanding to be compositional, in that if we wanted to apply the same adverb to a new verb of which we know the base meaning, we would be directly able to deduce the modified sequence of action generated. To do this, we try to map the adverb to a module, which should specialise in learning how to apply the correct modification, rather than overfitting to the patterns observed.

To make our GRU modular, we define a layer of a neural network in which modularity is embedded, following [47]. Any neural network can be augmented with this layer, thus after having defined it, we simply add it to the GRU we want to use in the modified gSCAN baseline.

7.1 Modular layer

A modular layer (Figure 7.1 l can be introduced at any point in any neural network, by integrating it with the other layers which define the network: $l \in \{l_1, ..., l_n\}$. Just like a mixture of expert network, a modular layer is defined by a controller and a set of M available modules among which the controller selects the experts given an input. The choice of the modules is treated as a random variable and consists in M scalar outputs that indicate the importance assigned to each module at the corresponding index. The output of the layer is a weighted sum of the outputs of all modules, weighed by the importance assigned to each of them.

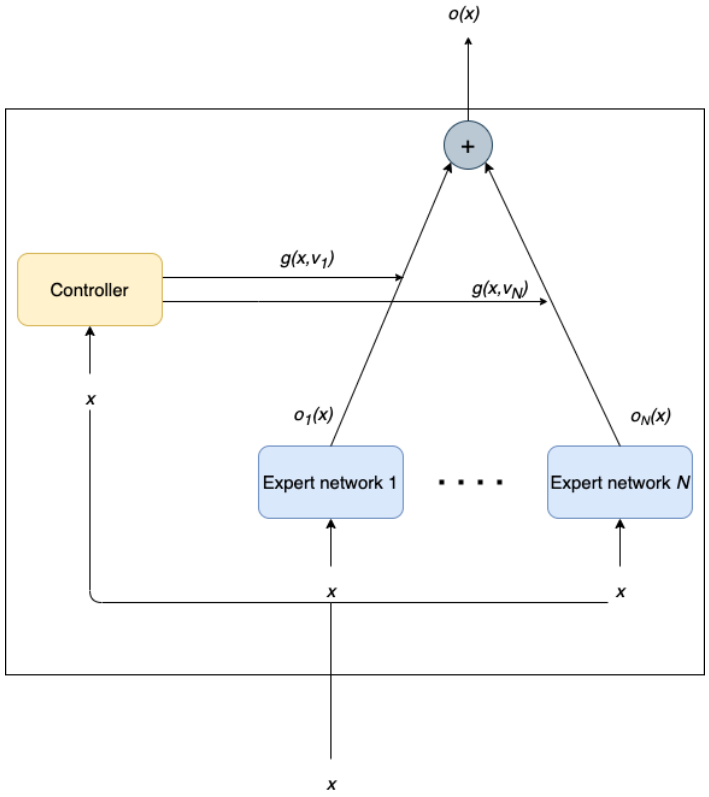

Figure 7.1: Architecture of a modular layer, which is just a mixture of experts model introduced as a layer in a neural network.

7.1

7.1.1 Controller

The job of the controller is to assign similar modules to similar inputs. To do this, various algorithms have been proposed. For example, [44] proposed this expectation-maximisation algorithm under the probabilistic interpretation of the mixture of experts:

Algorithm 6.2.1 EM algorithm for Mixture of Experts learning
Given a training set of T examples $\mathfrak{D} = \{(x_t, y_y)\}_{t=1}^{T}$ set the number of expert modules, M, and initialize all the parameters randomly $\Theta = \{V, (W_n)_{n=1}^{N}\}$ in the mixture of experts:

- **E-Step**: For each example (x_t, y_t) in $\mathfrak{D}$, estimate posterior probabilities $h_t = (h_{mt})_{m=1}^{M}$, with the current parameter values, $\hat{V}$ and $\{\hat{W}_m\}_{m=1}^{M}$

$$h_{mt} = \frac{g(x_t, \hat{v}_m)P(y_t|x_t, \hat{W}_m)}{\sum_{k=1}^{M} g(x, \hat{v}_k)P(y_t|x_t, \hat{W}_k)}$$

- **M-Step**:

– For expert module $m(n = 1, ..., M)$, solve the maximization problems

$$\hat{W}_m = \arg\max_{W_m} \sum_{t=1}^{T} h_{mt} \log P(y_t|x_t, W_m)$$

with all examples in $\mathfrak{D}$ and posterior probabilities $\{h_t\}_{t=1}^{T}$ achieved in the E-Step

– For the controller, solve the maximisation problem

$$\hat{V} = \arg\max_{V} \sum_{t=1}^{T} \sum_{m=1}^{M} h_{mt} \log g(x_t, v_m)$$

with training examples $\{(x_t, h_t)\}_{t=1}^{T}$, derived from posterior probabilities $\{h_t\}_{t=1}^{T}$

Repeat the E-Step and M-Step alternatively until the EM algorithm converges.

This algorithm is also used in [47] for their own modular layer. However, it has been shown that this can lead to unstable performance, especially in the optimization problems of the M-step [48]. Because we want to train our modular layer jointly with the rest of the encoder-decoder network, we need to think of another approach for choosing the experts.

We take inspiration from [49], where the authors aim to use a modular architecture to extract the words from a piece of text that justified a neural network decision on that text. To do this, they introduce a module which they call generator, which specifies for each word in the text whether it is selected or not, by outputting a distribution over the possible selections it can make.

We design our controller so that the probability of choosing the m-th module is conditionally independent from the probability of choosing the others: the joint probability $p(z|x)$ is

$$p(z|x) = \prod_{m=1}^{M} p(z_m|x)$$

Each module distribution $p(z_m|x)$ is modelled using a bidirectional LSTM:

$$\overrightarrow{h_m} = \overrightarrow{f}(x_m, \overrightarrow{h_{m-1}})$$

$$\overleftarrow{h_m} = \overleftarrow{f}(x_m, \overleftarrow{h_{m+1}})$$

$$p(z_m|x) = \sigma_z(W^z[\overrightarrow{h_m}; \overleftarrow{h_m}] + b^z)$$

We choose the Categorical distribution, which is a probability distribution generally used to describe the possible states of a discrete random variable, which can take on different categories, and the probability of each category is specified separately.

Because the Categorical distribution is discrete, we need a differentiable approximation to this, otherwise we cannot use backpropagation.

For this purpose, we use a Gumbel-Softmax distribution [50], which is a continuous distribution that samples from the Categorical distribution but, being differentiable, allows for backpropagation.

In short, if z is a random variable with Categorical distribution $(\pi_1, \pi_2, ..., \pi_M)$ where π_m is the probability of Z belonging to the m-th class, and if the discrete data we are working with is one-hot encoded, to sample z the common approach is

$$z = onehot(max\{i|\pi_1 + ... + \pi_{i-1} \leq U)$$

where $i = 1, ..., M$ is the index of the class and $U \sim Uniform(0, 1)$. To obtain a Gumbel-Softmax from this we start by applying the Gumbel-Max trick, which means we sample z in this way instead

$$z = onehot(\arg\max_i \{G_i + \log(\pi_i)\})$$

where $G_i \sim Gumbel(0, 1)$. In this way, z has been turned into a deterministic function of its parameters and some Gumbel noise. To make this function differentiable, we need an approximation of `argmax`, for which we use the softmax function. The samples y_i are now given by

$$y_i = \frac{exp(\frac{G_i + \log(\pi_i)}{\tau})}{\sum_j exp(\frac{(G_i + \log(\pi_i)}{\tau})}$$

7.2 Modular GRU

A modular GRU is a GRU with at least one modular layer. In our experiments, we used one modular layer with 10 available modules which we used for processing the output of the single GRU cell of the network.

7.3 Experiments

We substitute the LSTM of the command encoder in the baseline model with our modular GRU and test it on the compositional tests proposed for the gSCAN. We run the tests for modular GRU with 5 and 10 available modules. The results for the modular GRU with 5 available modules are reported in Table 7.1 and those for 10 available modules in A.1.

We can observe that inserting our modular GRU has not significantly improved the performance on the gSCAN datasets in either experiment, but some slight improvements can be noticed for four of the tests. We provide an analysis of the results

Split	**Baseline exact match**	**Mod-5 exact match**
Random	97.69% ± 0.22	94.6% ± 0.49
Composition of references	54.96% ± 39.39	51.64% ± 20.12
Composition of attributes	23.51% ± 21.82	26.31% ± 15.22
Novel direction	0%	0%
Novel contextual references	35.2% ± 2.35	31.33% ± 3.54
Composition of actions and arguments	92.52% ± 1.75	86.59% ± 4.16
Few-shot learning of adverbs	0%	0%
Combining known adverbs	22.70% ± 4.59	23.1% ± 2.34
Longer sequences (length ≤ 15)	94.98% ± 0.12	90.59% ± 0.73
Longer sequences (length = 16)	19.32% ± 0.02	20.42% ± 0.95
Longer sequences (length = 17)	1.71% ± 0.38	$\leq 1\%$
Longer sequences (length ≥ 18)	$\leq 1\%$	0%

Table 7.1: gSCAN neural baseline results compared to results obtained using our modular GRU with 5 available modules. Metric used is exact match accuracy ± standard deviation.

Split	**Baseline exact match**	**Mod-10 exact match**
Random	97.69% ± 0.22	92.31% ± 0.98
Composition of references	54.96% ± 39.39	52.16% ± 23.12
Composition of attributes	23.51% ± 21.82	29.73% ± 14.25
Novel direction	0%	0%
Novel contextual references	35.2% ± 2.35	39.38% ± 1.54
Composition of actions and arguments	92.52% ± 6.75	89.12% ± 5.42
Few-shot learning of adverbs	0%	0%
Combining known adverbs	22.70% ± 4.59	23.13% ± 3.92
Longer sequences (length ≤ 15)	94.98% ± 0.12	91.85% ± 1.17
Longer sequences (length = 16)	19.32% ± 0.02	24.53% ± 0.45
Longer sequences (length = 17)	1.71% ± 0.38	$\leq 1\%$
Longer sequences (length ≥ 18)	$\leq 1\%$	0%

Table 7.2: gSCAN neural baseline results compared to results obtained using our modular GRU with 10 available modules. Metric used is exact match accuracy ± standard deviation.

focusing on the test on which the modular approach seems to have managed to improve the most: the composition of attributes test.

For the composition of attributes, the model has not trained on red square target objects but is tested on interacting with them. The modular model has performed slightly better than the baseline in both the case of 5 and 10 available modules. We can observe a slight improvement of 2.8% in the case of 5 available modules: 26.31% as opposed to the 23.51% of the baseline. The improvement is too slight to attribute even partial success in compositional reasoning to the results, but the

lower standard deviation indicates that the model has a more stable performance. Comparing the max accuracy reached by both approaches, we see that the baseline has achieved a higher max accuracy (45.16%) compared to the modular approach (39.53%), but this max accuracy represents a rare peak in its performance, while the accuracy for the modular approach tends to be closer to the mean. In the case of the modular model with 10 available experts, the improvement is a more noticeable 6.22%. The max accuracy in this case reaches 43.12%, which is comparable to the max accuracy of the baseline.

If we inspect the output of the controller in the modular GRU we can gain a better understanding of the kinds of choices of experts it made. We notice a similarity of the output of the controller between examples which had similar commands.

For examples in which the command contains just a size reference, most of the times the controller decided for experts 1 and 5 in the case of the GRU with 5 experts (hereafter Mod-5), and experts 1, 4 and 7 in the case of the GRU with 10 experts (hereafter Mod-10). More specifically, for examples in which the command contains the word 'small', the chosen modules tended to be 1 and 5 for the case of Mod-5 and tended to be 1, 4 and 7 for Mod-10. When the command contains 'big', both Mod-5 and Mod-10 tended to take the same decision.

For examples in which the command contains just the colour reference 'red', the module chosen by Mod-5 tended to be just 3, while Mod-5 chose 4 and 6 most of the times. When no color reference was present, the modules were 3 and 5 for Mod-5 and 1 and 8 for Mod-10.

It would seem that the modular approach has picked up that 'small' and 'big' both are size references. If we wanted to prove that the model has learned sizes compositionally, we would have to observe that when 'small' and 'big' are used in conjunction with 'red', the choice of modules picks out the experts which have been chosen for the individual words: in other words, if Mod-10 has picked experts 1, 4 and 7 for commands with 'small' and experts 1 and 8 for commands with 'red', if it has learned that a 'small red square' means the same as 'small square' + 'red square', it should pick out 1, 4, 7 and 8 as its experts.

However, this is not the case: we can observe both Mod-5 and Mod-10 choosing different sets of modules for when the size and colour are both referenced. Mod-5 picked just module 1, while Mod-10 picked modules 1 and 4. This means that in both cases, there is no overlap between the modules chosen for sizes, the ones chosen for the colour and the ones chosen for both.

This result suggests that the controller is not behaving as expected and is in fact assigning similar inputs to similar modules, but in a naive atomic way. In other words, it is successfully assigning the same experts when there is a 'small' or 'big' reference only in the command, having thus picked out the similarity of the inputs with these

words. However, it is not using this understanding when mapping commands with both 'small' or 'big' and 'red': the meaning it is assigning to 'a small square' and 'a small red square' is atomic, and not compositional. For the model, these two expression mean completely different things.

If we inspect the rest of the input-expert choices made, we find that no other experts being selected by Mod-5, while the only other expert chosen by Mod-10 is module 2, always in conjunction with 4, in the case that no attributes appear in the commands.

We can observe this same phenomenon for the other tests too. At most, Mod-5 selects three distinct modules, leaving the remaining two neural network experts untrained. Mod-10 selects six distinct modules at most over the whole benchmark, leaving four untrained. This phenomenon can be called 'module collapse' and consists in the controller always ignoring a significant number of the modules, which do not get picked for any inputs. This leads the other modules to not properly specialise in one kind of input, but to learn to deal with multiple types. This could explain why this modular design has not achieve compositional encoding of the features of the world.

This result is not as disappointing as it seems. There has been a slight but noticeable improvement on some of the compositional tests. In fact, we have observed that the model does indeed pick out meaningful word similarities in the corpus and can use it to influence its decision of the experts. Modularity seems indeed to be enabling more meaningful understanding of concepts and what seems to be hindering the model from applying a more powerful compositional reasoning is rather its implementation. We hypothesise that the central difficulty that the controller is having is that it is not being encouraged to promote diverse experts in taking on the cognitive tasks presented. This is most likely due to the training procedure, which has not taught the controller to divide-and-conquer in depth. A different loss function, or additional regularisation on the controller output would have likely helped it to explore more diverse decisions.

This raises some interesting points for discussion about modularity. First, each module has to specialise in particular aspects of the training data, but on the other hand, we also are aiming to generalise to unseen examples. Modularisation might in fact produce a specialisation that might prevent good generalization: this happens when the model specialises on features of the data that are not reflected in the real-world.

Second, modularisation is enforcing task decomposition. But admittedly, not all problems are necessarily decomposable. While it might be meaningful to divide natural language tasks from visual reasoning tasks in the gSCAN baseline, it is not sure if it is as meaningful to further divide the problem and, with it, the structure of our model. Until more meaningful improvements on this benchmark are achieved using modularity, this remains an open question.

8 Conclusion

8.1 Summary

In this thesis we have made the following contributions:

- We performed an ablation study on the neural baseline by removing one of its parts in order to gain a better understanding of the network behaviour and confirm that it is indeed a valid dataset on which to test for compositional reasoning.
- We proposed a set of experiments for the gSCAN dataset aimed at demonstrating the effects of providing a model with increasingly more information.
- We proposed a modular neural network with the goal of improving the current gSCAN benchmark results.
- We tested our modular architecture and provided an analysis of its performance, comparing it to the gSCAN neural baseline.

For our modular GRU, we have followed a mixture of experts technique to design a modular Gated Recurrent Unit. We then have modified the baseline model adding this recurrent neural network in the part that deals with encoding the input commands of gSCAN. We have observed only slight performance improvements in a restricted number of tests, but by analysing in more details the results of one of these tests we have observed that indeed these were enabled by some sort of compositional understanding of the gSCAN command corpus. More improvements would have likely been observed if the modular network could take more diverse decision in selecting its modules for a task.

8.2 Future work and extensions

We have identified the most probable reason for the lack of improvements in our modular neural network: experimenting more with the training procedure, the loss function and regularisation could help the model achieve more impressive results on the gSCAN. This constitutes the main extension of this project.

A further extension would consist in testing both the baseline and the modular model on the set of experiments proposed in section 5.6. This could help to form a more comprehensive understanding of how the models manage the information they have

available, and how they react when presented with more and more data that can be understood compositionally.

A Ethics checklist

	Yes	No
HUMAN EMBRYOS/FOETUSES		
Does your project involve Human Embryonic Stem Cells?		✓
Does your project involve the use of human embryos?		✓
Does your project involve the use of human foetal tissues / cells?		✓
HUMANS		
Does your project involve human participants?		✓
HUMAN CELLS/TISSUES		
Does your project involve human cells or tissues?		✓
PROTECTION OF PERSONAL DATA		
Does your project involve personal data collection and/or processing?		✓
Does it involve the collection and/or processing of sensitive personal data?		✓
Does it involve processing of genetic information?		✓
Does it involve tracking or observation of participants?		✓
Does it involve further processing of previously collected personal data?		✓
ANIMALS		
Does your project involve animals?		✓
DEVELOPING COUNTRIES		
Does your project involve developing countries?		✓
If it involves low income countries, are any benefit-sharing actions planned?		✓
Could the situation in the country put the individuals in the project at risk?		✓
ENVIRONMENTAL PROTECTION AND SAFETY		
Does it involve the use of elements that may cause harm to the environment?		✓
Does your project deal with endangered fauna and/or flora /protected areas?		✓
Does your project involve the use of elements that may cause harm to humans?		✓
Does your project involve other harmful materials or equipment?		✓
DUAL USE		
Does your project have the potential for military applications?		✓
Does your project have an exclusive civilian application focus?		✓
Will it use or produce goods or information that will require export licenses?		✓
Does your project affect current standards in military ethics?		✓
MISUSE		
Does your project have the potential for malevolent/criminal/terrorist abuse?		✓
Does your project involve biochemical or nuclear materials?		✓
Could your project have negative impacts on human rights standards?		✓
Does your project have the potential for terrorist or criminal abuse?		✓
LEGAL ISSUES		
Does it use software for which there are licensing implications?		✓
Does it use information for which there are data protection implications?		✓
Are there any other ethics issues that should be taken into consideration?		✓

Table A.1: An ethics considerations checklist, as required by the Imperial College Department of Computing

Bibliography

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR09*, 2009. URL http://www.image-net.org/papers/imagenet_cvpr09.bib. pages 1

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. URL http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf. pages 1

[3] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. pages 1

[4] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010. URL http://yann.lecun.com/exdb/mnist/. pages 1

[5] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. volume 15 of *Proceedings of Machine Learning Research*, pages 215–223, Fort Lauderdale, FL, USA, 11–13 Apr 2011. JMLR Workshop and Conference Proceedings. URL http://proceedings.mlr.press/v15/coates11a.html. pages 1

[6] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009. pages 1

[7] Joshua Tenenbaum and Fei Xu. Word learning as bayesian inference. *Cognitive Sciences*, 10, 12 2002. pages 2

[8] Joshua B. Tenenbaum, Charles Kemp, Thomas L. Griffiths, and Noah D. Goodman. How to grow a mind: Statistics, structure, and abstraction. *Science*, 331 (6022):1279–1285, 2011. ISSN 0036-8075. doi: 10.1126/science.1192788. URL https://science.sciencemag.org/content/331/6022/1279. pages 2

[9] Brenden M. Lake and Marco Baroni. Still not systematic after all these years: On the compositional skills of sequence-to-sequence recurrent networks. *CoRR*, abs/1711.00350, 2017. URL http://arxiv.org/abs/1711.00350. pages 2, 5, 19, 20, 21

[10] Jerry A. Fodor and Zenon W. Pylyshyn. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1):3 – 71, 1988. ISSN 0010-0277. doi: https://doi.org/10.1016/0010-0277(88)90031-5. URL http://www.sciencedirect.com/science/article/pii/0010027788900315. pages 2, 17, 18

[11] Steven Piantadosi and Richard Aslin. Compositional reasoning in early childhood. *PloS one*, 11:e0147734, 09 2016. doi: 10.1371/journal.pone.0147734. pages 2, 3

[12] Adam Santoro, Sergey Bartunov, M. Botvinick, Daan Wierstra, and T. Lillicrap. Meta-learning with memory-augmented neural networks. In *ICML*, 2016. pages 3

[13] Ronghang Hu, Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Kate Saenko. Learning to reason: End-to-end module networks for visual question answering. *CoRR*, abs/1704.05526, 2017. URL http://arxiv.org/abs/1704.05526. pages 4, 44

[14] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Deep compositional question answering with neural module networks. *CoRR*, abs/1511.02799, 2015. URL http://arxiv.org/abs/1511.02799. pages 4, 44, 46

[15] Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, Dmitry Tsarkov, Xiao Wang, Marc van Zee, and Olivier Bousquet. Measuring compositional generalization: A comprehensive method on realistic data. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SygcCnNKwr. pages 5

[16] L. Ruis, Jacob Andreas, Marco Baroni, Diane Bouchacourt, and B. M. Lake. A benchmark for systematic generalization in grounded language understanding. *ArXiv*, abs/2003.05161, 2020. pages 5, 19, 33

[17] A. E. Bryson, Y. Ho, and G. M. Siouris. Applied optimal control: Optimization, estimation, and control. *IEEE Transactions on Systems, Man, and Cybernetics*, 9 (6):366–367, 1979. pages 9

[18] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997. doi: 10.1162/neco.1997.9.8.1735. pages 10, 11, 29, 47

[19] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014. URL http://arxiv.org/abs/1406.1078. pages 10, 11, 47

[20] A. E. Bryson, Y. Ho, and G. M. Siouris. Applied optimal control: Optimization, estimation, and control. *IEEE Transactions on Systems, Man, and Cybernetics*, 9 (6):366–367, 1979. pages 11

[21] Herbert Rubenstein and John Goodenough. Contextual correlates of synonymy. *Commun. ACM*, 8:627–633, 10 1965. doi: 10.1145/365628.365657. pages 13

[22] Zellig S. (Zellig Sabbettai) Harris and Zellig S Harris. *Papers in structural and transformational linguistics / [By] Zellig S. Harris.* Formal linguistics series ; v. 1. Reidel, Dordrecht, 1970. ISBN 94-017-6059-4. pages 13

[23] Reinhard Khler and B.B. Rieger. *Contributions to Quantitative Linguistics*. Springer Publishing Company, Incorporated, 1st edition, 2007. ISBN 0792321979. pages 13

[24] Jeff Mitchell and Mirella Lapata. Vector-based models of semantic composition. In *ACL*, 2008. pages 13, 36

[25] Daniel Yarlett and Michael J. A. Ramscar. Language learning through similarity-based generalization. 2008. pages 14

[26] Barbara Partee, Alice Meulen, and Robert Wall. *Mathematical Methods in Linguistics*, volume 30. 01 1990. doi: 10.1007/978-94-009-2213-6. pages 15

[27] Wlodek Zadrozny. From compositional to systematic semantics. *Linguistics and Philosophy*, 17(4):329–342, 1994. doi: 10.1007/BF00985572. pages 15

[28] T M V Janssen. *Foundations and Applications of Montague Grammar: 8M Part 1: Philosophy, Framework, Computer Science*. Centrum voor Wiskunde en Informatica, NLD, 1986. ISBN 9060962927. pages 15

[29] Ernesto Perini-Santos. Does the principle of compositionality explain productivity? for a pluralist view of the role of formal languages as models. In Carlo Penco and Massimiliano Vignolo, editors, *Contexts in Philosophy 2017 - CEUR Workshop Proceedings*, pages 108–120. 2017. pages 16, 17

[30] Paul Ziff. The number of english sentences. *Foundations of Language*, 11(4): 519–532, 1974. pages 17

[31] G. Pullum and B. Scholz. Recursion and the infinitude claim. 2010. pages 17

[32] Zoltán Gendler Szabó. Compositionality. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, fall 2020 edition, 2020. URL `https://plato.stanford.edu/archives/fall2020/entries/compositionality/`. pages 17, 18

[33] Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. Compositionality decomposed: How do neural networks generalise? *J. Artif. Intell. Res.*, 67: 757–795, 2020. pages 17, 18

[34] Tomas Mikolov, Armand Joulin, and Marco Baroni. A roadmap towards machine intelligence. *CoRR*, abs/1511.08130, 2015. URL http://arxiv.org/abs/1511.08130. pages 19

[35] Li Dong and Mirella Lapata. Language to logical form with neural attention. *CoRR*, abs/1601.01280, 2016. URL http://arxiv.org/abs/1601.01280. pages 19

[36] Jake Russin, Jason Jo, Randall C. O'Reilly, and Yoshua Bengio. Compositional generalization in a deep seq2seq model by separating syntax and semantics. *CoRR*, abs/1904.09708, 2019. URL http://arxiv.org/abs/1904.09708. pages 21

[37] Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993. URL https://www.aclweb.org/anthology/J93-2004. pages 22

[38] Jacob Devlin, Jonathan Uesato, Surya Bhupatiraju, Rishabh Singh, Abdelrahman Mohamed, and Pushmeet Kohli. Robustfill: Neural program learning under noisy I/O. *CoRR*, abs/1703.07469, 2017. URL http://arxiv.org/abs/1703.07469. pages 30

[39] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014. pages 32

[40] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *CoRR*, abs/1607.04606, 2016. URL http://arxiv.org/abs/1607.04606. pages 36

[41] Jerry A. Fodor. *The Modularity of Mind*. Cambridge, MA: MIT Press, 1983. pages 40

[42] G. Auda and M. Kamel. Modular neural networks a survey. *International journal of neural systems*, 9 2:129–51, 1999. pages 41

[43] Ke Chen. Deep and modular neural networks. In *Handbook of Computational Intelligence*, 2015. pages 41

[44] Robert Jacobs, Michael Jordan, Steven Nowlan, and Geoffrey Hinton. Adaptive mixture of local expert. *Neural Computation*, 3:78–88, 02 1991. doi: 10.1162/neco.1991.3.1.79. pages 41, 42, 43, 48

[45] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross B. Girshick. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. *CoRR*, abs/1612.06890, 2016. URL http://arxiv.org/abs/1612.06890. pages 46

[46] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. VQA: Visual Question Answering. In *International Conference on Computer Vision (ICCV)*, 2015. pages 46

[47] Louis Kirsch, Julius Kunze, and David Barber. Modular networks: Learning to decompose neural computation. *CoRR*, abs/1811.05249, 2018. URL `http://arxiv.org/abs/1811.05249`. pages 47, 49

[48] Yong Liu and Xin Yao. Simultaneous training of negatively correlated neural networks in an ensemble. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 29:716 – 725, 01 2000. doi: 10.1109/3477.809027. pages 49

[49] Tao Lei, Regina Barzilay, and Tommi S. Jaakkola. Rationalizing neural predictions. *CoRR*, abs/1606.04155, 2016. URL `http://arxiv.org/abs/1606.04155`. pages 49

[50] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *ArXiv*, abs/1611.01144, 2017. pages 50