Investigating The Generalization Ability Of Convolutional Neural Networks For Interpreted Languages

Daniel Luurt Bezema
Bachelor Artificial Intelligence, Utrecht University
(7.5 ECTS)

Thesis Supervisor: Dr. Denis Paperno
Second Reader: Dr. Gerard A. W. Vreeswijk
(Dated: July 20, 2019)

CONTENTS

Abstract: In this study the generalization capacity of Convolutional Neural Networks (CNNs) for interpreted languages is investigated. Two CNN models, one of which included a curriculum, are trained on two interpreted languages of different complexity. The results show that a CNN, contrary to previous findings for Long-Short-Term-Memory Networks, does not benefit from a curriculum during training. Performance of models on the more complex interpreted language shows adequate generalization ability, while performance on the less complex language shows no generalization ability at all. This suggests that a CNN prefers complex training data over less complex training data, for it forces the model to capture more generally applicable features from which it benefits during testing. Overall the reported results of this study show that CNNs possess a generalization capacity for interpreted languages that is competitive with Recurrent and Recursive models from the literature.

I. INTRODUCTION

In his 1950 iconic paper 'computing machinery and intelligence' Alan Turing introduced the imitation game, a test for assessing the intelligent behaviour of machines. He argued that if a machine would be able to deceive a human through conversation that it was a human, it would deserve to be called intelligent. At the time, the idea of a machine that would be able to produce language at a human level seemed something out of fiction. A lot of time has passed, however, and many advances in Natural Language Processing (NLP) have been made. Today, most of us are interacting with one or more NLP systems daily and with some of these interactions it's getting more and more difficult to distinguish between human and machine capabilities. Are these systems, thus, approaching human intelligence? Most of us will say no.

One of the reasons for not ascribing intelligence to these NLP machines lies in their ability to generalize. Or rather: inability to generalize. State of the art NLP models and deep learning models in general have become increasingly capable of distilling relevant patterns from data, enabling them to achieve amazing performance in all kinds of tasks. In conventional machine learning terms these models are said to generalize well. 'Generalization', however, is a somewhat misleading term, for it implies the formulation of general concepts from specific instances by abstracting common properties, which can then be used to reason about new instances. In practice, the concepts or rules that a model formulates are very much restricted to examples it has seen during training. These will then enable the model to, for example, correctly classify unseen instances, but only if these instances are very similar to the ones it has seen. It is usually the case that the model has learned to exploit data specific patterns instead of learning more general rules that apply to a task.

Great performance on a benchmark dataset does not constitute great performance on a task. This is especially true for neural models that are being applied in NLP. Because, in order for language models to achieve generalization capacity approaching human capability, they need not only be able to learn to reliably map certain inputs to outputs, but through this process also be able to capture facts that hold true regardless of the input. Being able to capture facts that hold regardless of the input enables a language model to reason about novel situations.

Given a natural language model, such facts can be about two things: it can be about the world, or the language that describes it. Take, for example, two input sentences of same kind: 'Ann's child' and 'Bill's friend'. These refer to some individuals, let's say Bill and George, respectively. A model should not merely learn to configure its weights such that it maps these sentences to the desired individual. The weight configurations should in some way represent the fact about the world that Bill is the child of Ann and that the friend of Bill is George. They should, furthermore, in some way represent the fact about the English language that noun phrases can be recursively concatenated by the grammatical element *'s*. Putting these facts together, the model would be able to deductively infer (i.e. generalize) that a sentence of the kind 'Ann 's child's friend' refers to Bill.

A neural model's ability for this type of generalization can be tested trough the use of interpreted languages. These are simplified versions of natural language that specify pairs of expressions and meanings. The interpreted language comes with corresponding grammar that specifies the syntactic rules by which expressions may be composed. Expressions are sequences of strings of which we know the meaning (like the examples in the previous paragraph). The meaning of such expressions adheres to the principle of compositionality: it is a function of the meanings of its parts and the syntactic rule by which it has been composed. [12]

Such an interpreted language does away with all intricacies of natural language (such as ambiguity), while retaining the core properties of it (such as compositionality). This makes it a very useful idealization for studying the compositional skills and generalization capacity of neural models. Additionally, once a model's understanding of these core properties has been assessed, the constraints of the interpreted language can be relaxed relatively easily to account for real life scenarios.

Various studies (as will be shown in the 'related work' section) have used interpreted languages for investigating the generalization capacity of different types of neural network architectures. The type of architectures used in these studies were ones that subsequently have proven to be very effective in the field of NLP. Although these architectures have shown amazing performance in all kinds of NLP tasks they, as of yet, have not been able to acquire the compositional skill necessary to rival human level generalization capabilities. Because of this, NLP research has seen an increasing focus on the use of novel deep learning methods. [16]

One of these methods is the application of Convolutional Neural Networks (CNNs) to language-related tasks. A CNN applies convolution by sliding contextual windows over the input, thereby producing a number of feature maps. Through repeatedly applying convolutions to feature maps, the model extracts increasingly abstract features about the input. By this logic, a Convolutional Neural architecture should be effective in mining semantic as well as syntactic concepts, which may help them towards zero-shot generalization.

This study will contribute to the knowledge of the generalization capacity of neural architectures by investigating the CNN's ability to generalize about compositionality through interpreted languages. The main research question it tries to answer is whether or not the CNN architecture is able to correctly classify compositional structures of complexities higher than those it has seen in training. Additionally, this will add to the understanding and recent exploration of how the Convolutional Neural architecture can be applied to language tasks in general.

II. RELATED WORK

Studies that are closely related to the work in this paper are those by Paperno [15] and Hupkes et al. [10]. Both of these papers have investigated the generalization capacity of neural models by evaluating their performance in working with interpreted languages.

In Paperno's work, the generalization capacity of a vanilla RNN [7] and of a LSTM [9] were investigated trough their performance with an interpreted language that contained both left- and right-branching compositional structures. The results of this work show that the RNN does not learn to do compositional interpretation, but that the LSTM does, albeit only in the most favorable learning settings. The best scenario for LSTM includes extensive training data, left-branching (but no right-branching) compositions and a well-paced curriculum. With this curriculum, examples of higher complexity were added gradually in the process of training.

The study by Hukpes et al. [10] reports the ability of Recursive Neural Networks to generalize (TreeRNN), as well as of two different types of RNNs: a simple RNN [6] and a Gated Recurrent Unit (GRU) [2]. Again, the generalization capacity was established trough their performance in working with an interpreted language that contained both left- and right-branching compositional structures. They report their findings for each model as follows.

The simple RNN was not able to show a convincing ability to generalize. Performance of the GRU was adequate, but highly influenced by the type of compositional structure; the models that were trained on left-branching expressions showed great performance, while those trained on right-branching expressions showed very poor performance. Those trained on expressions of both branching types scored somewhere in between.

Of all different types of models the TreeRNN showed the best performance, with near-perfect accuracy. Interestingly, there is no real difference in performance to be found between the different types of branching, except when both branching types are present during training. In that case the model performs better than when it is tasked with either left- or right-branching.

Because of the similarity between research questions and findings, the experimental setup of the studies by Paperno [15] and Hupkes et al. [10] have been taken as a starting point for the one that is being presented in this paper. These comparable experimental setups will make it easier to compare results between the different studies and therefore make it easier to asses their value. The details of these experimental setups will be discussed in a later section.

Another study that uses a simplified, unambiguous language model to study systematic generalization to novel examples is the important study by Baroni et al. [1] In this study a range of recurrent neural network models (simple RNN's [6], LSTM's [9], and GRU's [2]) is tested using various tasks. Because networks with attention mechanism have become increasingly popular, each of the networks was additionally tested with an implemented attention mechanism. This resulted in a very systematic evaluation of popular recurrent neural network models from the literature in terms of their compositional abilities.

The type of generalization they set out to achieve was zero-shot. Zero-shot generalization, in this context, refers to the process by which a model learns to correctly interpret sequences without there having been explicit labeled instances present during training to assist in such an interpretation. The task they used for this endeavor was a simplified version of the CommAI Navigation Taks (CAN) [14], that they call SCAN. This task involves the linking of commands to action sequences, e.g. 'jump left' to 'LTURN JUMP'. Each command was unambiguously associated to a single action sequences.

Their models were tested on three different variations of the task that were of ascending complexity. The first task entailed generalization to a random subset of commands. The second task entailed generalization to commands that involved longer action sequences. The last task involved generalizing composition across different commands.

They found that, on the one hand, all models (excluding the simple RNN) were capable of zero-shot generalization in the least complex task where parts of test sequences were extensively present as training sequences. On the other hand, the same models fail miserably when the task gets more complex and the link between training and testing data is dependent on the ability to extract systematic rules. From these findings they conclude that the fundamental component that current models are missing is the ability to extract systematic rules from the training data.

This leads us to the trend that has been observed by Young et al. [16]: the increasing focus on alternative methods in NLP such as CNNs. As such, several studies can be found that have applied Convolutional models to NLP tasks. One such study is the work by Dauphin et al. [4]. Although the specific task is different from the ones in the preceding studies, they show that application of CNNs on language-related tasks can be a very promising endeavor.

In their study they present a Convolutional Neural Network with a novel gating mechanism that they call Gated Linear Unit (GLU). They subsequently test the performance of their model on two benchmark datasets and compare them with strong RNN models from the literature. Using the Google Billion Words benchmark, they reported results that are competitive with traditional methods, while using significantly fewer resources. As for the WikiText-103 benchmark, they reported results that, at the time, were state-of-the-art (perplexity of 37.2). This method reigned supreme for more than a year until Dayan et al. [5] surpassed it using traditional methods (perplexity of 36.4).

Another study that tries to address shortcomings of current models is the one by Le et al. [13] They note that current recursive approaches to computing the meaning of sentences run into some practical problems. These problems include: adaptiveness of composition function, dealing with different branching factors and with uncertainty about correct parse. They propose a combination of convolutional approaches (using both recurrent and chart techniques), which they term Forest Convolutional Network (FCN) and find that it indeed resolves the aforementioned challenges. Subsequently, they compare the performance of this FCN using public datasets with models from extant literature. Using the Stanford Sentiment Treebank (SST) sentiment analysis dataset, they reported, at the time, the highest accuracies on both the fine-grained (5 classes) and binary classification tasks. Using the Text Retrieval Conference (TREC) dataset, they performed a question-classification task and reported the second-highest reported accuracy. Still, this is a very promising result considering that the best-performing model (SVM_S [3]) was trained using heavily engineered (60 handwritten rules) resources, while the FCN used unsupervised pre-trained word-embeddings.

These studies provide a clear overview of the ability of Neural Network architectures to generalize using interpreted languages and show that conventional architectures have a hard time acquiring sufficient compositional skill. At the same time, new methods are being tested using various NLP tasks. These trials, though not having been executed using the same type of generalization tasks, show very promising results. It is therefore a logical next step to asses if and how these efforts can be applied to remedy the flaws of traditional methods. Consequently, this study will asses a Conventional Neural architecture's ability to provide the type of generalization for interpreted languages that has been introduced in these last two sections. To the best of my knowledge, this is the first time such a neural architecture has been applied to this specific task.

III. COMPUTATIONAL APPROACH

The CNN architecture that has been used in this study on investigating the generalization capacity of CNNs is fairly elementary by modern standards. It consists of only eight layers, each having a relatively low number of neurons. The first part of the network consists of a convolutional block, where multiple convolution- , ReLU- and pooling operations are consecutively applied to produce feature maps from a given input sequence. By repeating this process, the features represented by the feature maps become increasingly abstract. In this network's architecture the process of convolution is repeated two times. The first time with 16 feature-extracting filters, the second time with six filters.

The last part of the network consist of a classification block that uses the features that are provided by the convolutional block to make predictions about the input. Predictions are produced by two fully connected layers of 120 and 84 neurons, respectively. Before any data is fed to the fully connected layers, however, it is first fed through a dropout and flattened layer. The dropout layer randomly disables some elements in the feature maps and the flattened layer concatenates all feature maps in such a way that the features are represented as a vector of features. A more mathematically refined description of the architecture and the flow of data through it can be found in the subsections below.

A. Model Architecture

During the forward pass of the network, a one-hot encoded sentence is passed as input to a convolutional layer. Here, six filters ($size = 3 * 3 * C_{in}$, where C_{in} is the amount of input channels, corresponding to the length of the one-hot-encodings) each apply a 2-dimensional convolution to the input ($padding = 2$ and $stride = 1$). The 2-dimensional convolution is defined by Equation (1).

$$\begin{aligned} out(N_i, C_{out_j}) = bias(C_{out_j}) \\ + \sum_{k=0}^{C_{in}-1} weight(C_{out_j}, k) \\ \otimes input(N_i, k) \end{aligned} \tag{1}$$

where N is the batch-size and C is the amount of channels. $\otimes$ is the matrix multiplication operator.

After this convolution operation a Rectified Linear Unit (ReLU) (Equation (2)) is used as activation function to eliminate all negative values in the output, which provides the model with non-linearity.

$$ReLU(x) = max(0, x) \tag{2}$$

where x is an element of the previous layer's output. The ReLU function is applied element-wise to the entire output of the previous layer.

This output is then passed to a pooling layer where a 2-dimensional max pooling ($size = 2 * 2, stride = 1$) operation is applied. The output of this pooling operation can be precisely described by Equation (3).

$$\begin{aligned} out(N_i, C_j, h, w) = \max_{m=0,\ldots,kH-1} \max_{n=0,\ldots,kW-1} \\ input(N_i, C_j, stride[0] \times h + m \\ , stride[1] \times w + n) \end{aligned} \tag{3}$$

where, again, N is the batch-size and C represents the amount of channels. kH and KW are the height and width dimensions of the kernel respectively.

The output is then fed trough another convolution layer, a ReLU and a pooling layer. This time, however, 16 filters apply a 2-dimensional convolution operation, instead of six filters. Furthermore, the channel dimension of the filter has been reduced to one, because of the convolution operation in the first layer (see Equation (1).). Padding, stride, width and height of the filter remain the same.

After the input has been passed through these feature-searching layers there is a dropout layer. During training of the model this drop-out layer randomly selects elements (selection with $p = 0.5$) from its given input to be deactivated (i.e. element values are set to 0). This 'dropping out' of neurons has been shown by Hinton et al.[8] to be an effective countermeasure against overfitting, as it prevents the co-adaptation of neurons. During evaluation the dropout layer computes an identity function, effectively leaving the input unchanged.

The next layer takes all elements of its input and concatenates them all into a single vector. This is the flattened layer, since it flattens all dimensions of the different filters into a one-dimensional feature vector. This feature vector is then passed to two fully connected layers which take these features that were collected by the previous layers and use it make predictions about what was being denoted by the input for the very first layer of the network. The highest value after having applied the Softmax function (Equation (4)) will be taken as the model's verdict.

$$softmax(y)_j = \frac{exp(y_i)}{\sum_{j=0}^{K-1} exp(y_j)} \tag{4}$$

where y is a vector of model predictions.

The complete network architecture is illustrated in Figure 1. Dimensions of output on each layer are omitted, since they are dependent on the type of task the model is performing.

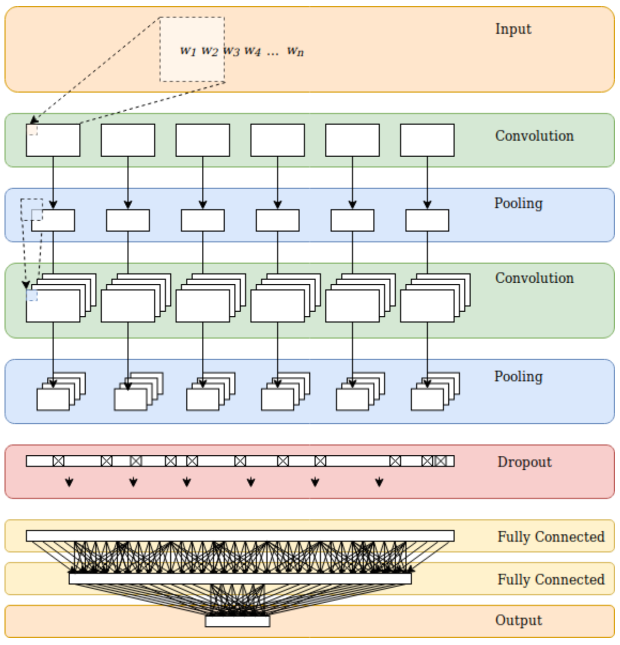

FIG. 1. Schematic illustrating the layers of the CNN architecture

B. Learning Mechanism

Once the model has made its verdict from a range of predictions, the whole range of predictions needs to be evaluated and the weights of all relevant filters and neurons need to be adjusted accordingly in order for the model to be able to improve its predictions. The loss function used for measuring the difference between the models verdict and the correct prediction is termed cross entropy loss. This loss function is defined by Equation (5).

$$loss(x, y) = -\sum_{i=0}^{K-1} y_i log(x_i) \tag{5}$$

where x is a vector of softmaxed model predictions, y is a vector containing the true predictions and K is the length of these vectors.

Because we know that all but one values in y are 0 and that one value is equal to 1, the formula can be simplified to the one in Equation (6).

$$loss(x) = -log(x_i) \tag{6}$$

where i is the index of the highest value in true prediction vector.

The method used for optimizing the weights of the model is the Adam optimizer algorithm as proposed by Kingma et al. [11] The entire system was implemented using Pytorch.

IV. EXPERIMENTAL SETUP

Evaluating the CNN's ability to generalize interpreted languages is done by testing whether or not it is capable of combining knowledge gained from learning compositional structures of a certain complexity into knowledge about compositional structures of higher complexity. In this study, compositional structures of two different interpreted languages have been used for assessing this ability. One of these interpreted languages is constructed after a language used in very similar study by Paperno [15] on the generalization capacity of recurrent models. This interpreted language will be referred to as 'the personal relations vocabulary'. The other interpreted language is constructed after a language used in a study by Hupkes et al. [10] on the processing of hierarchical structures by recurrent and recursive models. This language will be referred to as 'the arithmetic vocabulary'. The convolutional model instances presented in this paper are trained on these vocabularies, so that they are more easily compared to the performance of the models from the aforementioned studies.

Independent of the vocabulary being used, the setup of the experiments is the same. A particular instance of the Convolutional Neural Network architecture as defined in the 'Computational Approach' section will be trained on, and evaluated by, a set of expressions from a vocabulary, where expressions are compositional structures of a certain complexity. During training, the model's weights and filters get updated by expressions of complexities 1, 2, 4, 5 and 7 only. When the model is tested, however, expressions of all complexities are used. The maximum complexity is 9.

Test-expressions of complexities 3, 6, 8 and 9 therefore always consist of unseen examples. For the examples of other complexities, however, this cannot be ensured. Because there are only so many different unique expressions that can be generated per complexity (e.g. only 4 for expressions of complexity 1 from the Personal Relations vocabulary), there is bound to be some overlap between expressions in the train- and test partitions.

This overlap is by no means a real problem, because, although performance on all complexities will be evaluated, testing the model's ability to generalize during this task comes down to its performance with expressions of complexities 3, 6, 8 and 9 - which are all unseen examples.

Given the discrepancy between complexities present in train- and test data allows for not only a full assessment of a Convolutional model's ability to generalize from low complexity compositional structures to higher complexity compositional structures, but also in the other direction.

Preliminary results from Paperno's [15] paper suggest that recurrent networks generalize to compositional structures, but only if a model's training includes a curriculum. A curriculum consists of slowly increasing the complexity of expressions during the model's training process. This curriculum is different from a conventional setup where all training examples are available to a model at any time during training. In order to affirm this finding the presence of a curriculum has been added as experimental parameter to the experiments presented in this study. The curriculum added expressions of a higher complexity to the training process every ten epochs.

Details of the experiments and the exact composition of different vocabularies can be found in the subsections below.

A. Data

1. The Personal Relations Vocabulary

This vocabulary consists of 4 constituents which are interpreted as individual identifiers (names), 4 function denoting nouns which are interpreted as relations {friend, enemy, parent, child} and 1 grammatical element {'s}. The compositional structures generated from this vocabulary are expressions denoting some individual. Expressions are generated by randomly choosing a constituent from the available individual identifiers, appending a grammatical element and then appending a random relation. This process is repeated up to a given complexity value.

If the desired complexity of the expression is 1, then the expression will consist only of a single name. If the desired complexity of the expression is greater than 1, then the first constituent of the expression will be an expression of lower complexity (which might be a single name or a more complex expression). Examples of expressions that can be generated from the personal relations vocabulary can be found in Table I.

Complexity	**Example Expression**
1	Ann
2	Ann 's Child
3	Ann 's Child 's Friend
4	Ann 's Child 's Friend 's Enemy

TABLE I. Examples of expressions that can be extracted from the Personal Relations Vocabulary. Expressions up to $c = 4$ are shown.

It is important to note that the grammatical element in this vocabulary only allows for the generation of left-branching structures. This is different from Paperno's research [15], where both left- and right-branching structures were used.

Once some amount of expressions is generated for each complexity, the meaning of each expression is determined. The meaning of an expression is the individual that is denoted by the expression according to predefined relations between individuals. This relation model is randomly defined for a generated set of expressions. However, mechanisms are in place to ensure that relations adhere to certain rules. For example, a friend relation cannot be established such that an individual is friend as well as enemy of another individual at the same time, and vice versa. Friend and enemy relations, thus, are mutually exclusive. The same applies for the parent and child relation. Additionally, if individual A is child of B, then B is parent of A. The Parent-Child relation is, thus, bi-directional. The same applies to the friend and enemy relation. The relation model allows for one and only one meaning per unique expression. The meaning of expressions, thus, is unambiguous. A particular instance of a relation model is given in Table II.

The rules for the relation model were implemented to ensure that there are set rules governing the data. These rules are there for the neural model to pick up, helping it to understand what specific relations encompass, thereby hopefully assisting it in evolving the ability to use these rules in novel ways.

	Ann	**Bill**	**Cathy**	**Danny**
friend	Cathy	Danny	Ann	Bill
enemy	Danny	Cathy	Bill	Ann
parent	Cathy	Ann	Danny	Bill
child	Bill	Danny	Ann	Cathy

TABLE II. An example of a specific relation model on the Personal Relations Vocabulary. e.g. 'Ann's parent is Cathy', 'Danny's enemy is Ann'.

Even though labels are determined only after expressions have been randomly generated, a mechanism is implemented that ensures every individual is represented the same number of times in total by the expressions. This ensures that classes in the resulting data are balanced.

To summarize, the resulting data from this vocabulary is a set of expressions of recursively embedded relations corresponding to a set of individuals who are being denoted by those relations. The task of the CNN model, when being trained on examples from this vocabulary, is thus to correctly assign an individual to a complex embedding of relationships. In short, given the language interpretation in Table II, having memorized that 'Ann 's friend is Dick' and that 'Dick 's enemy is Bill' an ideal model would be able to generalize that 'Ann 's friend 's enemy is Bill'.

2. The Arithmetic Vocabulary

This vocabulary consists of a set of numeral words in the range of [-10,10], 2 mathematical operators +,- and 2 grammatical elements {),(}. The compositional structures generated from this vocabulary are mathematical expressions denoting some integer. Expressions are generated by randomly choosing a numeral word, appending a randomly chosen operator, again randomly choosing a numeral word and putting this string between the grammatical elements (putting the mathematical expression between brackets).

The grammatical elements of this vocabulary allow for the ability to generate both left- and right-branching compositional structures. However, only left-branching expressions have been used in this experiment. This is contrary to Hupkes et al. [10] who used both kinds of branching structures in their experiments.

Analogous to the personal relation vocabulary, if the complexity of the expression is 1, then the expressions will consist of only a numeral. If the complexity of the expression is greater than 1, then the first constituent of the expression will be an expression of lower complexity instead of a numeral. Examples of expressions that can be generated from the arithmetic vocabulary can be found in Table III.

Complexity	Example Expression
1	ONE
2	(TWO PLUS -ONE)
3	((-TWO PLUS ONE) MINUS ONE)

TABLE III. Examples of expressions that can be extracted from a model in the arithmetic vocabulary with numerals in the range of [-2, 2] and the operators {+,-}. Expressions up to c=3 are shown.

Once a set of expressions is generated for each complexity, the solution to each mathematical expression is computed as the label to the expression. Because each sub-expression is put between brackets, there is one and only one solution to each mathematical expression. The meaning of each expression is, thus, unambiguous.

The mechanism that was implemented for the personal relation vocabulary to ensure that distinct labels were evenly distributed is not applied for the arithmetic vocabulary. The reason for this being that Hupkes et al. [10] seemed to generate expressions in the same way. This means that the classes are not uniformly distributed and that, thus, the classes in the resulting data are unbalanced.

To summarize, the resulting data from this vocabulary is a set of expressions of recursively embedded mathematical expressions corresponding to a set of solutions to those mathematical expressions. The task of the CNN model when being trained on examples from this vocabulary is, thus, to correctly compute the solution to a complex embedding of mathematical expressions.

B. Encoding

Each expression has been one-hot-encoded. To ensure that all encodings have the same dimensions, expressions shorter than the largest expression in the full set of expressions have been padded during encoding. Padding is done in three different ways: 1. adding padding on the left of the expression, 2. adding padding to the right of the expression, 3. wrapping the expression in padding. The reason for padding expressions is two-fold. Firstly, encoding expressions in three different kinds of padding effectively results in tripling the amount of data. Secondly, it aids the system in generalizing between different positions of the input.

C. Parameters

The CNN model that was tasked with the personal relations vocabulary was trained for 100 epochs with a learning rate of 0.0001. 320 examples of each complexity were generated. 75% of these examples were used as training data. Given that training only uses expressions of 5 different complexities (1, 2, 4, 5 and 7) and that expressions are duplicated 3 times during encoding, this means that train data consisted of 3,600 examples. The data was fed into the network in batches of size 64. One experiment was performed that included a curriculum, as well as one that did not include a curriculum.

The values of the parameters were chosen based on a parameter search where the influence of all combinations of epochs (50, 100 and 200), amount of examples per complexity (40, 80, 160 and 320) and batch-size (20%, 50% and 100% of the amount of examples per complexity) on the model's performance were tested.

The CNN model that was tasked with the arithmetic vocabulary was trained for 100 epochs with a learning rate of 0.0001 and 3,000 examples of each complexity were generated. Using the same equation as for the personal relation vocabulary we derive that train data consisted of 33,750 examples. The data was fed into the network in batches of size 750. Again, one experiment was performed that included a curriculum and one experiment that did not include a curriculum.

Values for parameters on the arithmetic vocabulary experiments were chosen based on the proportions between the parameter values of the personal relation vocabulary experiments. The amount of expressions per complexity is the same as in the study by Hupkes et al. [10]

To summarize, four experiments have been done in total with the Convolutional Neural architecture that has been introduced in 'Computational approach' section. One experiment on the personal relation vocabulary with curriculum and one without it. And one experiment on the arithmetic vocabulary with curriculum and one without it.

D. Evaluation

Performance of the model on the personal relations vocabulary is measured in accuracy, because this makes the results more easily comparable to the aforementioned study by Paperno [15], who utilize(s) the same metric.

Performance of the model on the arithmetic vocabulary is measured using the Mean Squared Error (MSE) instead of accuracy. This is chiefly because the nature of data generated from the arithmetic vocabulary doesn't allow for evaluation in terms of accuracy. Classes are not uniformly distributed throughout the data, so there is no way of knowing whether or not a certain percentage of correct predictions is above chance level. Furthermore, MSE was used in the previously mentioned study by Hupkes et al. [10]. Therefore, MSE is used for these evaluations.

V. RESULTS

The previously defined CNN architecture has been tested for its generalization capacity using the Personal Relations as well as the Arithmetic vocabulary. For both of these vocabularies, the influence of a curriculum during training has furthermore been explored. Each of these four experiments were repeated 20 times to make sure that noise created by the random generation of data is leveled out. In this section the results of these experiments will be reported.

Performance on the Arithmetic vocabulary is illustrated by the graph in Figure 2. It showcases the average as well as the best (i.e. lowest) MSE of the models that included a curriculum during training and of those that did not (from this point on abbreviated as CNN-c and CNN, respectively). First, the average performance of the models is discussed.

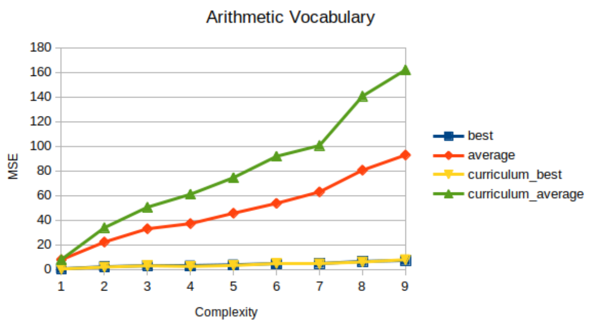

FIG. 2. Mean Squared Error of a CNN on expressions generated from the Arithmetic vocabulary. Average and best performance of models with and without training curriculum are showcased. Expressions of complexity 3, 6, 8, and 9 were unseen during training

A primary observation is that the average error rate is far higher for the CNN-c than it is for the CNN. This is radically different from the findings of Paperno [15] who found that a well-paced curriculum was essential for a LSTM to generalize to unseen compositional examples.

When the average performance of individual models is plotted (Figure 3) it becomes even more apparent how much better the default CNN performs in comparison to the CNN-c. In fact, the CNN achieves a performance characterized by an MSE of less than 10 in 11 out of the 20 experiments. For the CNN-c this level of performance is only attained in 5 out of the 20 experiments.

Additionally, it can be inferred that on average the error rate of the models rises as the expression complexity increases. This is not surprising, considering the fact that training examples become decreasingly representative of the entire expression space as complexity goes up. Specifically, the expression space expands with a factor of 42 with each new level of complexity, that is, the number of numerals multiplied by the grammatical elements in the vocabulary. What is surprising, however, is that the error rate increases approximately linearly to the increase in complexity.

One would either suspect a far more exponential relation between increasing complexity and error rate due to the exponentially growing space of expressions, as is reinforced by the results from Hupkes et al. [10] Alternatively, varying error levels between expressions of seen and unseen complexities are also to be expected. Hereafter, the results of single experiments (as displayed in Figure 3) are examined to further discuss this discrepancy.

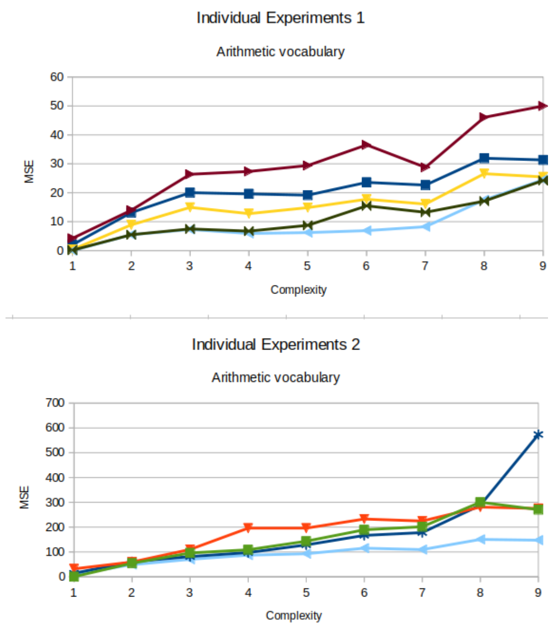

FIG. 3. Performance of a random selection of models (both CNN and CNN-c) plotted per complexity. Individual Experiments 1 shows overall relatively low errors, Individual Experiments 2 shows overall relatively high errors

When the results of individual experiments are plotted it can be observed that the aforementioned suspicion of alternating error levels between expressions was justified. Notice that the increase in error rate when the complexity increases a single level is greater for unseen examples than for seen examples in almost all experiments. In other words, the model has a harder time generalizing to expressions of unseen complexity. This is not unexpected.

As expected, the error rate increases when moving from one seen complexity to the next unseen complexity. However, at this stage, it will barely change when moving on to a yet higher, though seen, complexity. For example, the error rate increases between complexities 2 and 3, but hardly differs between 3 and 4. This suggests that the models learn to adequately generalize to unseen examples when it has seen examples of complexities that are one above and one below the complexity level of the unseen examples.

This notion is supported by the fact that performance on unseen complexities that do not have known complexities above them, i.e. 8 and 9, have a greater increase in error and subsequently have a higher overall error.

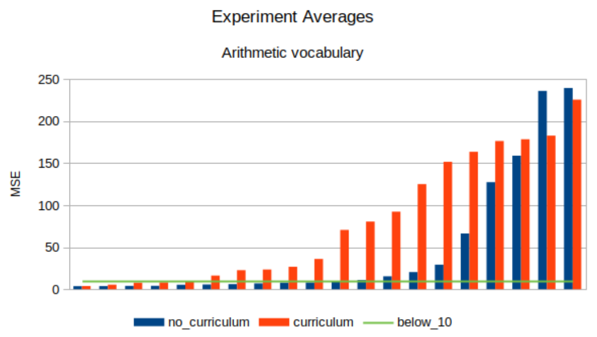

FIG. 4. Mean Squared Error of a CNN on expressions generated from the Arithmetic vocabulary. Only the best performing models are presented. Expressions of complexity 3, 6, 8, and 9 were unseen during training

In order to reliably extract certain assertions about the overall performance of the CNN architecture on the arithmetic vocabulary it is pitted against the performance of the models from Hupkes et al. [10] The models are compared using their best, i.e. lowest, and average error rate with expressions of the highest complexity. Side-by-side evaluation of the metrics is provided by Table IV. Because the paper by Hupkes et al. [10] did not present exact metrics, those numbers represent best estimations inferred from the corresponding graph.

The comparison in Table IV show that the models presented in this paper perform better than the SRN used in the study by Hupkes et al. Although the average performance of the CNN and CNN-c is worse, their best performance level is better than for the GRU and competitive with the TreeRNN.

As for the performance of the CNN and CNN-c with

	GRU	**SRN**	**TreeRNN**	**CNN**	**CNN-c**
average	< 30	>100	-	93	162
best	< 15	< 40	< 5	7.33	7.87

TABLE IV. A comparison between models from this study and the study by Hupkes et al. [10] with an Arithmetic vocabulary. 'Best' and 'Average' performance in this context is Mean Squared Error on expressions of complexity 9.

the Personal Relations vocabulary; the average and best accuracy of the models that were trained on this vocabulary are illustrated by the graph in Figure 5. The performance level of the models for expressions of complexities 1 and 2 are omitted, as they do not test generalization capacity. The models have seen such extensive training data on these complexities that it has learned to memorize them. As such, all models achieved a perfect accuracy for these complexities.

It must be noted at this point that the previous observation that the CNN architecture does not benefit from a curriculum during training is reinforced by the results in Figure 5. The average performances of the CNN and CNN-c are almost identical.

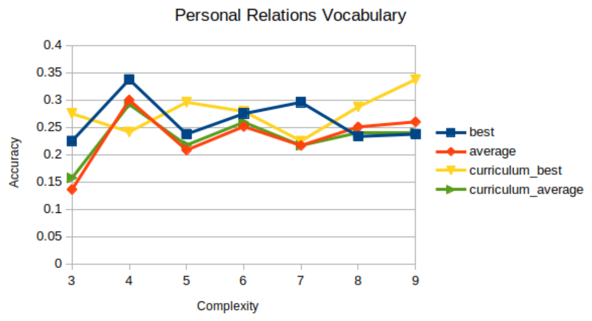

FIG. 5. Test accuracy of a CNN on expressions generated from the Personal Relation vocabulary per expression complexity. Random baseline is 0.25. Expressions of complexities 3, 6, 8 and 9 were all unseen during training.

Although the best performing models show some generalization capacity, the models fail, on average, to capture any structural knowledge that enables them to generalize. This is reflected by all scores fluctuating around the random baseline of 0.25. One might have expected the models to be unable to generalize to expressions of unseen complexity, but the observation that they do not generalize to expressions of complexity for which they have seen extensive training data, is remarkable.

It is especially surprising since the results on the Arithmetic vocabulary show that the same models are, in principle, capable of generalization, both to seen and unseen complexities. Furthermore, because the data generated from the Personal Relations vocabulary is less complex than those from the Arithmetic vocabulary, one might even be inclined to think that generalization performance on the Personal Relations vocabulary would be better.

As such, the question remains what caused this model to fail so spectacularly.

The tasks that comprise the Arithmetic and Personal Relations vocabulary are essentially the same. The discrepancy between performances must thus be caused by the composition and nature of the data. The difference in composition between data from the vocabularies is summarized by Table V.

	Arithmetic	**Personal Relations**
numerals/names	21	4
relations	2	4
gr. elements	2	1
classes	113	4
stdev.	291	0

TABLE V. A summary of the composition of data generated from the Arithmetic and Personal Relations vocabularies. Numerals/names, e.g. *Two* or *Ann*. Relations, e.g. *+* or *friend*. Gr. elements, e.g. *(* or *'s*. Classes is the average amount of unique labels that are present in the total data set. Stdev. is the standard deviation of the amount of examples that are present for each label

It can be deduced from this summary, as was mentioned in the section introducing the vocabularies, that the classes in the data generated by the Personal Relations vocabulary are evenly distributed, while those in the arithmetic vocabulary are not. This is shown by the values for classes and stdev., displaying, respectively, the total number of labels and the variation of those labels being represented by the data .

On the one hand, given that the data from the Arithmetic vocabulary is heavily imbalanced and complex while the models from this vocabulary achieve adequate performance, this suggests that the class imbalance and complexity of the data forces the model to capture more general features in order to be able to correctly classify examples during training.

On the other hand, given the fact that the data from the Personal Relations vocabulary is perfectly balanced and relatively simple while the models from this vocabulary fail spectacularly, suggests that the model has not learned to capture general features and thus heavily overfits the data.

In other words, more complex training data forces a CNN to extract more intricate and generally applicable features from which it benefits during evaluation, while simplicity of training data does not.

VI. CONCLUSION

It can be deduced from this summary, as was mentioned in the section introducing the vocabularies, that the classes in the data generated by the Personal Relations vocabulary are evenly distributed, while those in the arithmetic vocabulary are not. This is shown by the values for classes and stdev., displaying, respectively, the total number of labels and the variation in those labels being represented by the data .

On the one hand, given that the data from the Arithmetic vocabulary is heavily imbalanced and complex while the models from this vocabulary achieve adequate performance, this suggests that the class imbalance and complexity of the data forces the model to capture more general features in order to be able to correctly classify examples during training.

On the other hand, given the fact that the data from the Personal Relations vocabulary is perfectly balanced and relatively simple while the models from this vocabulary fail spectacularly, suggests that the model has not learned to capture general features and thus heavily overfits the data.

In other words, more complex training data forces a CNN to extract more intricate and generally applicable features from which it benefits during evaluation.

VII. CONCLUSION

This study has investigated the generalization capacity of the Convolutional Neural Network architecture for two different interpreted languages. Both interpreted languages consisted of left-branching compositional structures of varying complexity, but differed in their exact composition. One model was tested that included a curriculum during training and one was tested that did not.

For the models that were trained on the interpreted language called 'Arithmetic vocabulary' we find that the performance bears witness to adequate generalization capacity. Additionally, the results show that generalization to unseen complexities is more successful if the model has seen examples during training of higher and lower complexity than the unseen complexity.

This suggests that the model is able to both generalize towards higher unseen complexities as well as towards lower unseen complexities. To test whether or not this pattern holds true, the gap between complexities of seen and unseen compositional structures can be further stretched and the maximum complexity of compositional structures can be increased.

Conversely, the models that were trained on the interpreted language called 'Personal Relations vocabulary' failed to show any generalization ability.

The influence of composition of data on performance of models suggest that a CNN benefits from more complex training data, for it forces the CNN to capture more abstract and general features to be able to achieve adequate performance during training. However, the influence of training data composition on CNN model performance has to be investigated further to adequately support this finding.

Contrary to previous findings for LSTM networks, a CNN does not benefit from a curriculum during training. In fact, for one of the vocabularies the presence of a curriculum worsened performance. Whether this is to be ascribed to mere presence of a curriculum itself or the particular rate with which complexities are added during training is, as of yet, unclear. Further research could prove useful in ascertaining the effects of varying such a slope.

Overall the experiments presented in this study show that CNN's possess a generalization capacity for interpreted languages that is competitive with Recurrent and Recursive models from existing literature. To further test this capacity additional research might experiment with generalization capacity to both left- and right-branching structures.

---

[1] Baroni, M. and Brenden Lake. *Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks.* In International Conference on Machine Learning, pages 28792888. 2018

[2] Cho, K., Junyoung Chung, Caglar Gulcehre and Yoshua Bengio *Empirical evaluation of gated recurrent neural networks on sequence modeling.* CoRR, abs/1412.3555. 2014

[3] Coheur, L., Ana C. Mendes, Joao Silva, Andreas Wichert. *From symbolic to subsymbolic information in question classification.* In Artificial Intelligence Review, 35(2):7154. 2011.

[4] Dauphin, Y.N, Angela Fan, Michael Auli, David Grangier *Language modeling with gated convolutional networks.* In Proceedings of the International Conference on Machine Learning (ICML), 2017, pp. 933941

[5] Dayan, P., Chris Dyer, Timothy P. Lillicrap, Jack W. Rae. *Fast parametric learning with activationmemorization.* arXiv preprint arXiv:1803.10049, 2018.

[6] Elman, J. L. *Finding Structure in Time.* Cognitive Science, 14 (2), 179211. 1990

[7] Elman, J.L. *Distributed representations, simple recurrent networks, and grammatical structure.* In Machine learning, 7(2-3):195225. 1991.

[8] Hinton, Geoffrey E, Srivastava, Nitish, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan R. *Improving neural networks by preventing co-adaptation of feature detectors.* arXivpreprint arXiv:1207.0580. 2012b.

[9] Hochreiter, S. and Jrgen Schmidhuber. *Long short-term memory.* In Neural computation, 9(8):17351780. 1997

[10] Hupkes, D., Sara Veldhoen,and Willem Zuidema *Visualisation and diagnostic classifiers reveal how recurrent and recursive neural networks process hierarchical structure.* arXivpreprint arXiv:1711.10203 [cs.CL]. 2017.

[11] Kingma, Diederik P and Ba, Jimmy Lei. *Adam: A method for stochastic optimization.* arXivpreprint arXiv:1412.6980. 2014.

[12] Kracht, Marcus. *Interpreted Languages and Compositionality* In volume 89 of Studies in Linguistics and Philosophy. Springer, Berlin. 2011

[13] Le, P., and Zuidema, W. *The Forest Convolutional Network: Compositional Distributional Semantics with a Neural Chart and without Binarization.* In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 11551164. 2015.

[14] Mikolov, T., Joulin, A., and Baroni. *M. A Roadmap towards Machine Intelligence.* arxivpreprint arXiv:1511.08130 2016.

[15] Paperno, D. *Limitations in learning an interpreted language with recurrent models.* In Proc. EMNLPBlackboxNLP. ACL. 2018.

[16] Young, T., Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. *Recent trends in deeplearning based natural language processing.* arXivpreprint arXiv:1708.02709v8. 2018.