Convolutional Neural Networks on Systematic Compositionality

Roberto Dessí

CIMeC, University of Trento
roberto.dessi11@gmail.com

January 22, 2018

I. Motivation

Given the wide attention that the field of *Deep Learning* [1] has seen in recent past, extensive research was done in this area. Deep learning as a subfield of machine learning, itself a subfield of Artificial Intelligence was the main focus of many industrial as well as academic efforts during the last 5 years. Cognitive and biological plausability of Deep Learning methods have been questioned and more concerns have been recently raised [2].
Through this project I wanted to better understand deep neural networks methods and capabilities and I wished to do so by means of a practical experiments on a real dataset allowing me to better understand research ideas and how to validate them. The main problem I am interested in during this project is to evaluate performance of convolutional neural network models on the task of systematic compositionality. Systematic compositionality as defined by Lake and Baroni, 2017, is the "the algebraic capacity to understand and produce a potentially infinite number of novel combinations from known components" [3]. Humans are effortlessly good at it. When we know the meaning of *run twice* as the action of run two times, or the meaning of *jump around* as the meaning of jumping while turning around, or the meaning of *run and jump* as run first and then jump, once we learn a new verb like *dax*, we should be able to know the meaning *dax around and jump twice* as daxing while turning around and then jump two times [1]. Lake and Baroni, 2017 showed that Recurrent Neural networks-based models are good at generalizing when they are trained on examples that are similar in structure and length to the ones seen at test time, but their performance decrease if length sequence is different between train and test data, finally showing poor performance (~1%) when they have to generalize to a new composed sequence containing a verb e.g. *jump* in *run twice and jump left* if they are trained on data containing *jump* only in its primitive form i.e. not appearing in a composed sequence.
To the best of my knowledge no previous work showed the application of convolutional neural networks on the task of systematic compositionality and I wish to show some introductory work in this report.

II. Literature

Natural language is intrinsically sequential. Any kind of linguistic element e.g. word, sentence, discourse or a spoken speech, has subelements that are in temporal relation of antecedent or consequent with its neighbours. Sequence length of said elements may vary significantly: in a question-answering application sentence length could range from a couple of words to tens of words. Recurrent Neural Networks [4] have showed successful results in modelling such kind of sequential data [5]. Neural machine translation is an area of re-

[1]The example presented is adapted from Lake and Baroni, 2017 [3]

search that uses neural networks-based models for the automatic translation of sentences from a source to a target language and that was initially introduced in [6]. A widely used architecture that has shown great performance is called sequence to sequence [7] (or Encoder-Decoder [8]). This architecture uses a recurrent network to output a fixed-lenght vector representation of the source sentence. Such representation is then given as input to the decoder recurrent network. An example of this architecure is shown in figure 1. Additionally an attention mechanism was introduced by Bahdanau et al. [9], it improved the performance of sequence to sequence models since it allowed for the decoder to be conditioned by different versions of encoded vectors while emitting an output token of the target sentence. The attention mechanism works by encoding each of the k tokens of the source sentence through a bi-directional RNN. An annotation for each token is obtained concatenating the forward hidden state $\overrightarrow{h_i}$ with the backword $\overleftarrow{h_i}$, resulting in the *i-th* annotation focusing on the *i-th* input word as well as the following (through the forward network) and preceding ones (through the backword network). In the decoder network, through an *attention matrix*, at each time step, the network learns which annotation needs to consider while emitting a target token through a weighted sum of all possible annotations. Each annotation focuses on a specific part of the input allowing the model to understand which part of the source sentence to "pay attention" to while emitting the current output word. This is shown more clearly in figure 3.

Reccurent sequence to sequence architectures, while showing great performance, have a computational problem. Given that the *j-th* state takes as input the previous result computed from state $j-1$, the computation is consequential and cannot be parallelized. This is not the case with convolutional networks where each convolutional operation, acting on different parts of the input can exploit performance benefits coming from parallel computation.

III. Research Questions

In Lake and Baroni, 2017 they try to address generalization capabilities of current recurrent sequence to sequence models on the task of systematic compositionality in a large set of experiments concerning a wide range of hyperparameters. The task, as presented in [3], is designed as a machine translation task where in a grounded navigation environment a sequence of commands must be translated into a sequence actions. An example can be seen in figure 2.

The tasks aim at testing a learner on its generalization capabilities when learning how to translate unseen commands to meaningful actions. The SCAN dataset [2] contains 20,910 action-command pairs. Actions involve a small set of primitive verbs like *{jump, run, look}* and a set of modifiers like *{around, twice, thrice}*. An action like *run twice* should be translated to the command *RUN RUN*.

IV. Proposal and Project

Given the computational limits of RNN and their poor performance on strong compositional generalization my proposal is to evaluate a CNN architecture on the SCAN dataset, replicating experiment 1 and 2 as proposed in [3]. The convolutional model that I tried to evaluate in this project is an entirely convolutional sequence to sequence model introduced in 2017 by Gehring et al. [10]. This model use convolutional filters and Gated Linear Units along with a Multi-step attention mechanism. Differently with respect to [9], the attention module computes different steps (or computations) before calculating its final attention scores at any given timestep. Such method is similar to the multi-hop mechanism proposed in [11]. Given the lack of recurrency that bring a sort of temporal notion to the model and that is an intrinsic feature of RNN through their reccurrent computations, a positional vector that is learned by the model is added to the

[2]The SCAN dataset was collected from https://github.com/brendenlake/SCAN

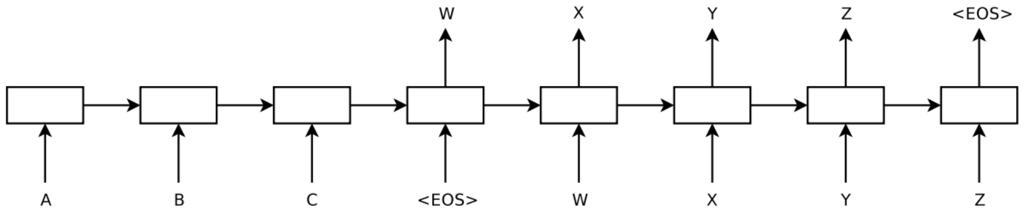

Figure 1: *Example of the seq2seq architecure as presented in [7]*

jump	⇒	JUMP
jump left	⇒	LTURN JUMP
jump around right	⇒	RTURN JUMP RTURN JUMP RTURN JUMP RTURN JUMP
turn left twice	⇒	LTURN LTURN
jump thrice	⇒	JUMP JUMP JUMP
jump opposite left and walk thrice	⇒	LTURN LTURN JUMP WALK WALK WALK
jump opposite left after walk around left	⇒	LTURN WALK LTURN WALK LTURN WALK LTURN WALK LTURN LTURN JUMP

Figure 2: *Examples of SCAN commands (left) and the corresponding action sequences (right) taken from [3]*

.

input and output embedding. This is done to give a a notion of "what part of the source and target sequence the model is currently dealing with" [10].

Figure 4 shows more clearly the model architecture. The code to use said model was developed by a team at FAIR (Facebook AI Research) and is freely available. [3] Given the name of the GitHub repository, I will refer to this model through the name **fairseq**.

In this project I collected the SCAN dataset, preprocessed the sample data in a form that was compliant with fairseq toolkit. This meant creating two different files for both train and test sets, one file containing data from the source language i.e. actions like *jump and run twice*, and another file that contained data from the target language i.e. the commands like *JUMP RUN RUN.*

I then evaluated 1080 variants of *fairseq* on 2 different experiments. The different versions of the model were obtained through an extensive hyperparameters search.

The hyperparameters tested were the following:

- embedding size: 128, 256, 512
- number of layers: 5, 6, 7, 8
- learning rate: 0.25, 0.15, 0.1
- dropout value: 0, 0.25
- kernel size: 5, 3
- max number of tokens in a batch: 50, 100, 500, 1000, 2000

The models were trained for a time ranging between one minute for a model with an embedding size of 128 and a maximum number of tokens per batch of 2000, and one hour for a model with embedding size of 512 and a maximum of 50 tokens per batch. The same set of model was tested on the two experiments. For experiment 1 the models were trained and evaluated on a NVIDIA Tesla K80 whereas for experiment 2 they were trained and evaluated on a NVIDIA Tesla K40c.

The first experiment, as presented by Lake a Baroni, 2017 [3] is a simple split of the dataset in 80% as training data and 20% as test data. This is a highly common ratio in Deep Learning and Machine Learning research as it seems to be a fair distribution allowing a model to have enough data to learn common patterns while still being able to generalize to a reasonable amount of unseen data. In experiment 2, they try to split the dataset according to output

[3]The library to use the model is called fairseq-py and is available at https://github.com/facebookresearch/fairseq-py/

sequences length. Roughly 80% of the dataset contained training data that had up to 22 actions in the output sequence. The remainig portion of the dataset formed the test set and had sentences of longer output length with a number of actions that range from 24 to 48.

V. Results and Discussion

The best performing model for experiment 1 was a model with an embedding size of 128, 7 layers, a kernel size of 5 trained with a learning rate of 0.15, a dropout value of 0.15 and maximum 50 tokens per batch. This model was able to achieve an accuracy of 99.1% on the test set. The best model for experiment 2 had an embedding dimension size of 128, 7 layers, kernel size of 5 trained with a learning rate of 0.1, a dropout value of 0 and with a maximum of 100 tokens per batch, achieving an accuracy of 11.6% on the test set. Average accuracy decreased with higher size of the embeddings on both experiments, maybe due to the fact that a higher embedding size implies a greater number of parameters that could lead the model to overfitting on the train data.
A comparison with results from Lake & Baroni, 2017 [4] is shown in Table 1.

Architecture	Exp1	Exp2
Lake & Baroni	99.7%	13.8%
ConvLearning	99.1%	11.6%

Table 1: Accuracy on the Exp1 and Exp2 presented in [3]

High accuracy on experiment 1 shows how convolutional networks are able to generalize and achieve almost human-like performance even on unseen data. This is in line with the results obtained with RNN. Poor performance on experiment 2 instead may be due to the inability for positional embeddings, which learn a notion of position within the sentence, to generalize to longer target sequence. For experiment 2, multiple times the model produced a sentence that was entirely correct, although rather than stop emitting tokens it kept producing wrong elements resulting in a longer and wrong output. A possible way to address this problem could be adding an "oracle" module explicitely showing the output sentence length, thus guiding the model when to stop producing target tokens. This is done in [3] and it lead to a significant boost in accuracy.

[4]The best-overall architecture over the two experiments was a 2-layer LSTM with 200 hidden units per layer, no attention, and dropout applied at the 0.5 level

VI. Future work

Consequent work related to this project would be investigating the computations done by the memory module. Informal preliminary experiments done by Marco Baroni shows that a convolutional architecture is able to achieve interesting results even on the "hardest" experiment (experiment 3) in [3], when the attention module operates only in the last layer of the decoder network. A possible improvement could also be adding a planning mechanism to the attention system used in the convolutional sequence to sequence architecture, as done in [12] with their recurrent architecture.

VII. Conclusion

This project showed how convolutional neural networks can achieve comparable performance to RNN while having more computational efficiency. Their results are only slightly inferior to reccurent models yet the speedup achievable while training said models is remarkable. Both architectures are far from showing strking accuracy results on different generalization tasks where sequence length differ between train and test data, thus showing how their weaknesses are also similar. Reccurent-based networks are still the best solutions for most language tasks but exploiting convolutional models implies a faster training time with no significant loss in accuracy performance.

Acknowledgements

I want to thank Marco Baroni for his help throughout this project. Most of the ideas presented came from him and he shared with me part of the code that was used to run the experiments reported in this work.

References

[1] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. 521:436–44, 05 2015.

[2] G. Marcus. Deep Learning: A Critical Appraisal. *ArXiv e-prints*, January 2018.

[3] 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.

[4] Jeffrey L. Elman. Finding structure in time. *COGNITIVE SCIENCE*, 14(2):179–211, 1990.

[5] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014.

[6] Nal Kalchbrenner and Phil Blunsom. Recurrent continuous translation models. Seattle, October 2013. Association for Computational Linguistics.

[7] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc., 2014.

[8] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*, 2014.

[9] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.

[10] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning. *CoRR*, abs/1705.03122, 2017.

[11] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. *CoRR*, abs/1503.08895, 2015.

[12] Francis Dutil, Çaglar Gülçehre, Adam Trischler, and Yoshua Bengio. Plan, attend, generate: Planning for sequence-to-sequence models.

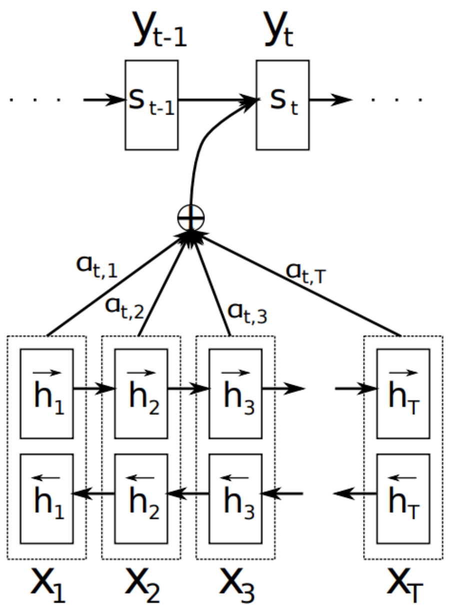
y_{t-1}
y_t
s_{t-1}
s_t
$a_{t,1}$
$a_{t,2}$
$a_{t,3}$
$a_{t,T}$
$\overrightarrow{h}_1$ $\overrightarrow{h}_2$ $\overrightarrow{h}_3$ $\overrightarrow{h}_T$
$\overleftarrow{h}_1$ $\overleftarrow{h}_2$ $\overleftarrow{h}_3$ $\overleftarrow{h}_T$
x_1 x_2 x_3 x_T

Figure 3: *A graphical example of the structure of an attention module [9]*

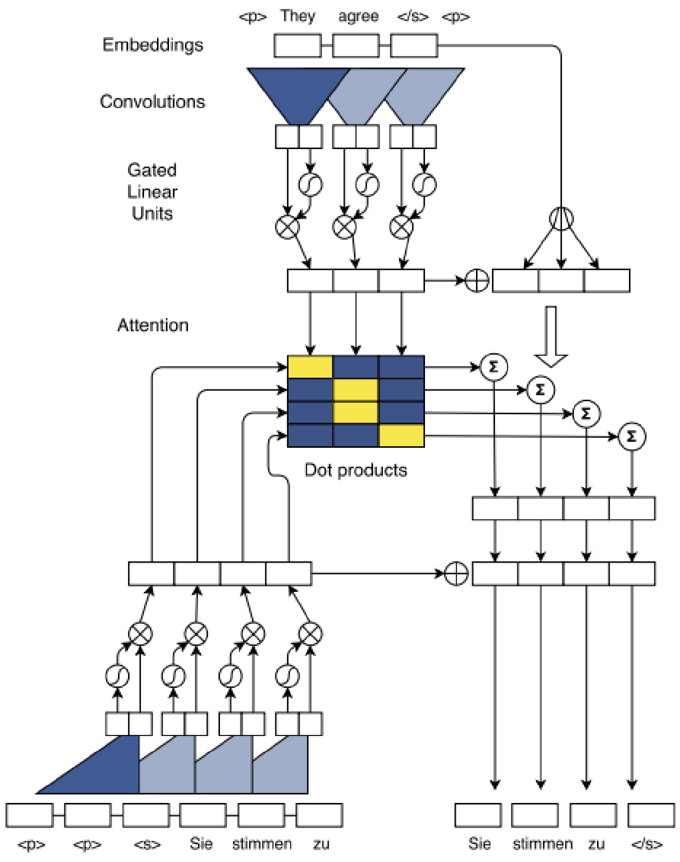

Figure 4: *The general structure of fairseq [10]*