Master Thesis Proposal

Capturing Long-Term Dependencies in Neural Regulatory Networks

Advisor: Univ.-Prof. Dipl.-Ing. Dr.rer.nat. Radu Grosu
Student: Hannes Brantner (01614466)

November 16, 2020

1 Problem Statement

1.1 Recurrent Neural Networks

Capturing long-term dependencies in time series with machine learning models is difficult. A machine learning model is a function with an input and parameters. For example, the machine learning model GPT-3 proposed in [3] has 175 billion parameters. The whole field of sequence modelling started with recurrent neural networks. These are models like our physical environment, where the current state h_t and the next input x_{t+1} determine the next state h_{t+1} and output y_{t+1} deterministically. In this model all the past inputs are implicitly encoded in the current state. This is a big challenge for computer scientists, since computers only allow states of finite size and finite precision, unlike our physical environment, which results in an information bottleneck. The next state of a recurrent neural network h_{t+1} is typically computed by an equation like the one proposed in [1] (using a non-linear function σ and two matrices W and V):

$$h_{t+1} = \sigma(Wh_t + Vx_{t+1}) \tag{1}$$

1.2 Loss function

Most machine learning model optimization processes are guided by a loss function L. This function takes all the parameters of the machine learning model as arguments and outputs a scalar loss. This function could simply be the percentage of correctly classified input data of the input data set. In the general case, a computer scientist wants to find the global minimum of that function with respect to all machine learning model parameters. As this is a problem that cannot be solved analytically in most cases, it is approximated using gradient descent [11], where each parameter is optimized incrementally depending on the gradient of the loss function with respect to each parameter in lock step.

1.3 Gradients in Recurrent Neural Networks

The following inequality from [1] using norms (L_2 norm for vectors and spectral radius norm for matrices) shows the relation between a recent state h_T and a state from the far past h_t:

$$\left\| \frac{\partial L}{\partial h_t} \right\| \leq \left\| \frac{\partial L}{\partial h_T} \right\| \prod_{k=t}^{T-1} \|W\| \, \|diag(\sigma'(W h_t + V x_{t+1}))\| \tag{2}$$

This inequality contains all essential parts to understand why learning long-term dependencies with recurrent neural networks is difficult. Some problems that machine learning tries to solve need to incorporate input data from the distant past to make good predictions in the present. Since these inputs are stored in the states of the distant past, $\left\| \frac{\partial L}{\partial h_t} \right\|$ should not decay to 0 or grow unboundedly to effectively optimize the parameters using gradient descent to incorporate distant past inputs in present predictions to minimize the loss. If $\|W\| > 1$, $\left\| \frac{\partial L}{\partial h_t} \right\|$ may grow unboundedly, making it difficult to apply the gradient descent technique to optimize parameters. If $\|W\| < 1$, $\left\| \frac{\partial L}{\partial h_t} \right\|$ will decay to 0, making it impossible to apply the gradient descent technique to optimize parameters. The previous two statements are only true, if the norm of the diagonal matrix containing the derivatives of the function σ is constant, otherwise different problems arise. These two problems are called the vanishing or exploding gradient problem and are further explained in [2].

1.4 Mitigations to the vanishing gradient problem

[1] works with a matrix W that fulfills $\|W\| = 1$ to tackle these problems. This idea was later refined by [7]. The vanishing gradient problem was also tackled by [6] and a mechanism called gating, that changed the next state computation for an ordinary recurrent neural network. Another possible mitigation to the vanishing gradient problem is the transformer architecture proposed in [13] using a mechanism called attention. In principle the transformer architecture model has access to all past inputs and just learns, which ones are important for the current prediction and the transformer should attend to.

1.5 Problem of the thesis

This work will focus on the gating mechanism to tackle the vanishing gradient problem and will also apply it to the LTC network architecture introduced in [5] by introducing reciprocal gated LTC neuron pairs [9]. LTC neurons are the building blocks of LTC networks, which are time-continuous recurrent neural networks and are also called neural regulatory networks. In this models the state update function is not determined by a difference equation, but with a differential equation instead. A subset of LTC networks called neural circuit policies [8] was shown to reach great expressiveness with very few parameters.

As a reciprocal gated LTC neuron pair should function like a memory cell, it is called a continuous flip-flop, which is itself a pair of two latches. The problem is now to build a memory layer out of several reciprocal gated LTC neuron pairs to replace the attention mechanism in a transformer. Possible further use cases can be evaluated, too. The performance of the proposed transformer model (to incorporate long-term dependencies in decisions) is evaluated via the transformer benchmark suite proposed in [12].

2 Aim of the Work

2.1 Reach greater expressiveness with fewer parameters

This work aims to design a memory layer out of reciprocal gated LTC neuron pairs that incorporates gating to mitigate the vanishing gradient problem. This memory layer should then be used to replace the attention mechanism in the transformer architecture to cut down the parameter count without sacrificing too much of its expressiveness.

2.2 Motivate others to do research with time-continuous recurrent neural networks

The transformer architecture is no recurrent network model and operates on discrete time steps. The superior ability of transformers to learn long-term dependencies in contrast to recurrent neural networks led to performance improvements in many domains [13, 10]. Another main goal of this thesis is to show that continuous-time recurrent neural network models, that are a close fit to biological neurons, can model long-term dependencies and be more expressive with fewer parameters than ordinary transformer models.

3 Methodological Approach

3.1 Build LTC neuron pair

The first thing to do is to implement the reciprocal gated LTC neuron pair. The differential equations that govern LTC neuron dynamics were already implemented in a GitHub repository linked in [8]. The basic building block consists of two LTC neurons, each with a connection to the input. Only the output of one LTC neuron is used as output of the continuous flip-flop. The reciprocal gating connections are implemented using two synaptic connections between each other with inhibitory polarity. Furthermore, each LTC neuron has itself a recurrent synaptic connection with inhibitory polarity, because without a recurrent connection, there would be no memory.

3.2 Build memory layer

This basic building block, the CFF (continuous flip-flop), is used to build a memory layer. The CFFs could be interconnected in various ways using different adjacency matrices, that would result in a dense or sparse memory array of any dimension.

3.3 Use memory layer in a transformer architecture

This memory layer should then be used to replace the attention mechanism of a transformer in a suitable way.

3.4 Evaluate and compare new transformer model to other transformer models

The performance of the synthesized model is then evaluated using the transformer benchmark [12] and also compared to other transformers in terms of accuracy and parameter count.

4 Structure of the Work

1. Introduction
   (a) Problem Statement
   (b) Aim of the Work
   (c) Methodological Approach
   (d) State of the Art
2. Architecture
   (a) Continuous Flip-Flop
   (b) Memory Layer
   (c) Transformer with Memory Layer
3. Evaluation
4. Summary

5 State of the Art

5.1 LSTM (Long Short-Term Memory)

The LSTM is a an RNN whose state update function is more complex than the one presented in 1. The next input x_{t+1} and the current state h_t determine how much of the current state should be forgotten, how much of the next input should be saved in the next state h_{t+1} and what parts of the next state should

be used to build the next output y_{t+1}. The details of the LSTM architecture can be found in [6]. These three features are called input, output and forget gate and the general mechanism is called gating. This allows the LSTM to learn when and how to update the state, since the input has no direct influence on the state like in equation 1. This leads to a state that is able to carry information from the distant past (may be very important to the problem) to the present, by gating unnecessary future inputs, since that time point in the distant past, away. The gating mechanism is controlled by multiple parameters that are optimized using gradient descent. This enables the LSTM architecture to be able to learn long-term dependencies. A simplification of the LSTM architecture called the GRU (Gated Recurrent Unit) that is using fewer parameters and reaches the same expressiveness is proposed here [4].

5.2 EUNN (Efficient Unitary Neural Networks)

These models are able to learn long-term dependencies, since it parametrizes W in equation 1 such that $\|W\| = 1$. Therefore, this model does not suffer from the vanishing or exploding gradient problem. The results of the experiments in [7] propose superior expressivity to LSTMs at the same parameter count. However, EUNNs are not widely adopted and LSTMs are known in general to need more parameters to be expressive enough to solve a problem when compared to other models.

5.3 Transformer

The transformer architecture transforms a variable-length input sequence into a variable-length output sequence. The model gets all the inputs at once and not one input at each time instant. To accumulate information across all the inputs a mechanism called attention is used. The attention between different inputs is simply the result of the dot product between vectors crafted from the inputs. If one input contains much information about another input, those would have a high dot product, as the transformer's parameters will be learnt that way. Since this architecture only focuses on the most important parts of the inputs, it can deal with lots of input data at the same time and easily learn long-term dependencies between them, as past state inputs are available to the model instantly and not encoded into states of RNNs. The details of the transformer architecture can be found in [13].

6 Relevance to the Curricula of Computer Engineering

- 182.763 - Stochastic Foundations of Cyber-Physical Systems
- 186.844 - Introduction to Pattern Recognition
- 182.755 - Advanced Digital Design

- 191.105 - Advanced Computer Architecture
- 389.166 - Signal Processing 1
- 389.170 - Signal Processing 2
- 104.267 - Analysis 2
- 104.271 - Discrete Mathematics

References

[1] Martin Arjovsky, Amar Shah, and Yoshua Bengio. *Unitary Evolution Recurrent Neural Networks*. 2016. arXiv: 1511.06464 [cs.LG].

[2] Y. Bengio, P. Simard, and P. Frasconi. "Learning long-term dependencies with gradient descent is difficult". In: *IEEE Transactions on Neural Networks* 5.2 (1994), pp. 157–166.

[3] Tom B. Brown et al. *Language Models are Few-Shot Learners*. 2020. arXiv: 2005.14165 [cs.CL].

[4] Junyoung Chung et al. *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*. 2014. arXiv: 1412.3555 [cs.NE].

[5] Ramin Hasani et al. *Liquid Time-constant Networks*. 2020. arXiv: 2006.04439 [cs.LG].

[6] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". eng. In: *Neural computation* 9.8 (1997), pp. 1735–1780. ISSN: 1530-888X.

[7] Li Jing et al. *Tunable Efficient Unitary Neural Networks (EUNN) and their application to RNNs*. 2017. arXiv: 1612.05231 [cs.LG].

[8] Mathias Lechner et al. "Neural circuit policies enabling auditable autonomy". In: *Nature Machine Intelligence* 2 (Oct. 2020), pp. 642–652. DOI: 10.1038/s42256-020-00237-3.

[9] Aran Nayebi et al. *Task-Driven Convolutional Recurrent Models of the Visual System*. 2018. arXiv: 1807.00053 [q-bio.NC].

[10] Emilio Parisotto et al. *Stabilizing Transformers for Reinforcement Learning*. 2019. arXiv: 1910.06764 [cs.LG].

[11] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. "Learning Internal Representations by Error Propagation". In: *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*. Cambridge, MA, USA: MIT Press, 1986, pp. 318–362. ISBN: 026268053X.

[12] Yi Tay et al. *Long Range Arena: A Benchmark for Efficient Transformers*. 2020. arXiv: 2011.04006 [cs.LG].

[13] Ashish Vaswani et al. *Attention Is All You Need*. 2017. arXiv: 1706.03762 [cs.CL].