Randomly Wired Recurrent Neural Networks

Supervisors: Prof. Dr. Michael Granitzer, Julian Stier (Department of Computer Science, University of Passau)

DARJI, HARSHIL JAGADISHBHAI

E-mail: darji01@ads.uni-passau.de
Matriculation: 87647

1 INTRODUCTION

Recurrent Neural Networks (RNNs) are conventional models that have shown exceptional commitment in many NLP tasks that make use of sequential information. In traditional neural networks, it is assumed that all inputs are independent of each other, which is not a good idea for sequential tasks. For example, to predict the next word in a sentence, it is good to know words came before it. An RNN represents a sequence with a high-dimensional vector (called the hidden state) of a fixed dimensionality that incorporates new observations using an intricate nonlinear function [2]. In simple words, RNNs have a "memory" which captures knowledge about what has been calculated so far.

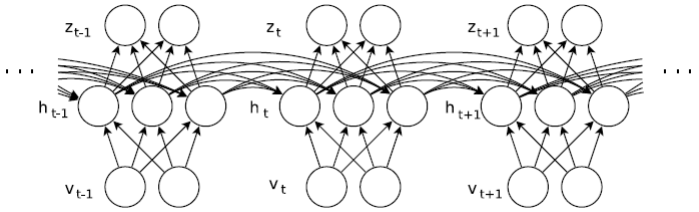

Fig. 1. A Recurrent Neural Network unfolded in time [2]

A standard RNN is parameterized with three weight metrices and three bias vectors $[W_{hv}, W_{hh}, W_{oh}, b_h, b_o, h_o]$ whose concatenation θ fully descirbes the RNN (Fig. 1). The RNN computes h_t (sequence of hidden states) and z_t (a sequence of outputs) by the following algorithm:

Algorithm 1: A standard RNN algorithm [2]

1: **for** t **from** 1 **to** T **do**
2: $u_t \leftarrow W_{hv} v_t + W_{hh} h_{t-1} + b_h$
3: $h_t \leftarrow e(u_t)$
4: $o_t \leftarrow W_{oh} h_t + b_o$
5: $z_t \leftarrow g(o_t)$
6: **end for**

where $e(\cdot)$ and $g(\cdot)$ are the hidden and output nonlinearities of the RNN.

Deep extensions of such basic RNNs can be constructed by stacking multiple recurrent hidden states on top of each other as shown in [1].

This thesis project aims to produce randomly wired deep RNN architectures by embedding random graph models in-between the input and output layer and then introducing a recurrent connection to make them dependent on early sequences.

2 MOTIVATION

In April 2019, Researchers from the Facebook AI Research group published a paper [4] in which they explored randomly wired neural networks driven by random graph models from graph theory. Authors claim that the mean accuracy of these models is competitive with hand-designed and optimized models from recent works on neural architecture search. This approach successfully shows that even randomly wired networks have an exceptional performance on image classification. If a similar approach is applied to Recurrent Neural Networks, there is a chance that these randomly wired networks will provide results better or comparable to existing hand-designed models.

In September 2019, researchers from the University of Passau published a paper [3] in which they tried to predict the performance of convolutional neural networks before actually training them. To achieve this, they first used random graph generators to generate graphs with desired properties. Then the next step was to make them directed and to compute layer indexing of all vertices. Then they embed layered vertices between an input and output layer of an Artificial Neural Network that meets the requirements of the dataset. The authors created a dataset of such 10000 graphs and split it into a train-test with a 70-30 ratio. They trained neural architectures of train sets and stored their performances along with their graph properties. Then they used this data to train three different learning algorithms and used them to estimate model performance values of architectures of the test set using their underlying structural properties. This paper provides motivation to predict the performance of RNNs based on their internal structural properties which will be really helpful given the fact that it is always difficult to train RNNs due to its exploding/vanishing gradients.

3 RESEARCH GOALS

Sparsity and randomness in recurrent neural networks are still not explored much compared to CNNs. Therefore, the main focus of thesis is to:

1. Explore and explain sparsity in Recurrent Neural Networks
2. Investigate how the sparse and random architectures affect the performance of RNNs

4 EXPERIMENTS

To investigate the effects of sparsity on the performance of RNNs, two of the experiments, to begin with, is described as follow:

1. Investigating the performance of pruned RNNs:
   - Train RNNs repeatedly for certain epochs,
   - Retrieve the weight distribution for hidden-to-hidden matrices,
   - Prune weights below a certain threshold and analyze its effect on the overall performance,
   - Investigate the performance of pruned RNN in comparison with standard RNN and its different variations (e.g. LSTM, GRU).
2. Investigating the performance of sparse RNNs:
   - Construct sparse RNN architecture (as described in §5),
   - Train sparse RNNs repeatedly for certain epochs,
   - Compare performance of sparse RNN with pruned RNN, standard RNN and its different variations.

5 CONSTRUCTION OF SPARSE RNN

To create randomly wired sparse RNN, the methodology followed is influenced by the process introduced in [3] and [4].

1. The first step is to generate random graphs without restricting how the graphs correspond to neural networks (Fig. 2).
2. Once the graph is generated, it is necessary to make it directed where edges define the data flow (Fig. 3).
3. The next step is to compute layer indexing of all vertices of the available Directed Acyclic Graph (DAG).
4. By embedding these layered vertices between an input and output layer completes the structure of a sparse neural network (Fig. 4).

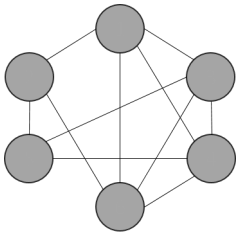

Fig. 2. Randomly generated graph [3]

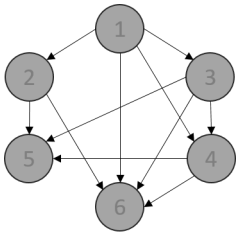

Fig. 3. Transformed DAG [3]

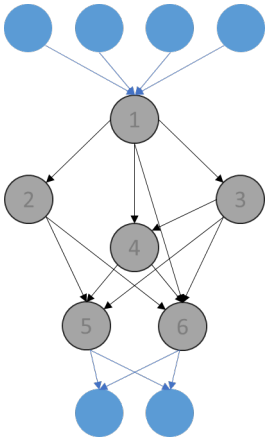

Fig. 4. Embedded DAG after computing layered indexing [3]

5. This generated structure is a simple feed-forward network, as defined in [3]. To convert it into a recurrent architecture, it is necessary to introduce recurrent connections that make subsequent runs dependent on previous runs (Fig. 5).
6. Once the complete structure is available, it will be trained on the training set and then will be evaluated on the test set.

6 DATASET

The choice of the dataset depends on the type of RNN being implemented.

1. **Many-to-One:** This type of RNNs is mainly used for the classification task such as Twitter or IMDB sentiment analysis. Therefore, examples of the datasets can be used are (un)labeled Twitter/IMDB dataset, Reber grammar dataset, etc.
2. **One-to-Many:** This type of RNNs is mainly used for sequence generation tasks such as generating texts or grammar strings. Example datasets are Shakespeare's text, Show scripts, Reber grammar, etc.

As can be seen, the Reber grammar dataset can be used for classification as well as sequence generation tasks. Therefore, in the beginning, to check the pipeline, the Reber grammar dataset will be used. Once

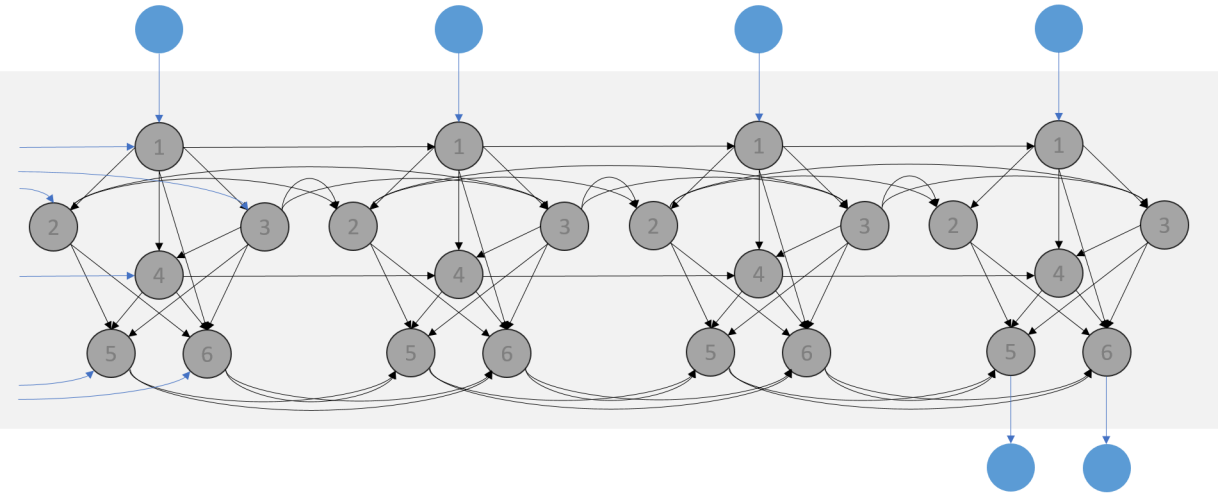

Fig. 5. Above generated feed-forward (Fig. 4) transformed into RNN and opened in-time.

the stable implementation is available, the implementation will be tweaked to make it flexible to work with any other datasets.

7 EVALUATION

Although it is known that accuracy cannot be considered as an impeccable evaluation technique, this project aims at using accuracy to justify the performance of sparse and randomly wired recurrent neural networks. As stated before, one of the main focuses of the thesis is to investigate the effects of sparsity and randomness of RNNs on its performance, accuracy can be considered as a simple and effective way to compare results.

8 SCHEDULE

The schedule is divided into four major tasks i.e, research, implementation, evaluation and documentation. The following table gives information about the amount of time required to finish each task.

Task	Duration (weeks)
Research	4
Implementation	10
Testing	2
Documentation	8

Table 1. Schedule of the thesis

REFERENCES

[1] Michiel Hermans and Benjamin Schrauwen. "Training and Analysing Deep Recurrent Neural Networks". In: *Advances in Neural Information Processing Systems 26*. Ed. by C. J. C. Burges et al. Curran Associates, Inc., 2013, pp. 190–198. URL: http://papers.nips.cc/paper/5166-training-and-analysing-deep-recurrent-neural-networks.pdf.

[2] Sutskever Ilya. "Training Recurrent Neural Networks". PhD thesis. Graduate Department of Computer Science, University of Toronto, 2013. URL: https://www.cs.utoronto.ca/~ilya/pubs/ilya_sutskever_phd_thesis.pdf.

[3] Julian Stier and Michael Granitzer. "Structural Analysis of Sparse Neural Networks". In: *23rd International Conference on Knowledge-Based and Intelligent Information & Engineering Systems*. DOI: 10.1016/j.procs.2019.09.165.

[4] Saining Xie et al. *Exploring Randomly Wired Neural Networks for Image Recognition*. Tech. rep. Facebook AI Research (FAIR), 2019. arXiv: 1904.01569v2.