Investigating Pruned and Structural Sparsity in Recurrent Neural Networks

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

DARJI, HARSHIL JAGADISHBHAI

1 INTRODUCTION

Recurrent Neural Networks are standard models that have shown exceptional performance 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 that 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 [4]. In simple words, RNNs have a "memory" which captures knowledge about what has been calculated so far.

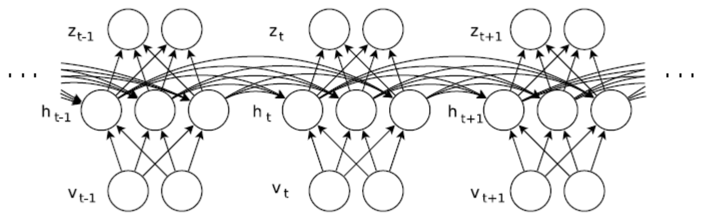

Fig. 1. A Recurrent Neural Network unfolded in time [4]

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. 6). 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 [4]

```
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 [3]. As part of my thesis, I aim to produce such deep RNNs, introduce pruned and structural sparsity in its internal structure and examine how sparse RNNs behave compared to standard RNNs.

2 MOTIVATION

It is well known that deep neural networks are likely to have increased performance but at a cost of increased fast memory requirements. One way to decrease the memory requirements is to introduce sparsity into a network's connections [2]. Sparsity in traditional neural networks by pruning weights, generating random architectures or utilizing skip connections is being studied widely from the past few years, but it is not explored much in the case of recurrent neural networks which are difficult to train due to their nonlinear iterative nature. Sparse structures in traditional neural networks have shown a promise of training potential [1] which if applied to Recurrent Neural Networks, can also make training RNNs less difficult by retaining its performance.

3 RELATED WORK

In November 2016, a group of researchers from Baidu research published a paper [7] in which they apply pruning to weights during the initial training of the network. At the end of the training, according to authors, "the parameters of the network are sparse while accuracy is still close to the original dense neural network". The authors claim "the network size is reduced by 8x and the time required to train the model remains constant". As explained in the paper, each recurrent layer has a constant number of hidden units, while I plan to have a different number of hidden units at each recurrent layer. Also, all the experiments are done on a private dataset, which prevents others from replicating the results which are necessary to compare this technique of introducing sparsity with other approaches.

In September 2018, Simon Alford et al. from MIT published a paper [1] on Sparse Neural Networks for image classification in which they test pruning based sparse topologies by pruning a pre-trained dense network and RadiX-Nets, a class of sparse network structures. The authors claim that compared to dense topologies, sparse structures show promise in training potential but also can exhibit highly nonlinear convergence, which merits further study. All experiments described in this paper are done on the Lenet-5 and Lenet 300-100 models trained on the MNIST [6] and CIFAR-10 [5] datasets.

In April 2019, researchers from the Facebook AI Research group published a paper [11] 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. All experiments described in this paper are done on the ImageNet 1000 class [9] dataset. This approach successfully shows that randomly wired networks have an exceptional performance on image classification. As described in the paper, each node of a randomly generated graph is considered as a fully connected convolutional layer with multiple inputs and multiple outputs, while I will follow an approach in which each node is considered as a single neuron in a recurrent layer.

In September 2019, researchers from the University of Passau published a paper [10] in which they aimed to predict the performance of convolutional neural networks using its structural properties. The authors built Sparse Neural Networks by embedding Directed Acyclic Graphs obtained thorough Random Graph Generators into Artificial Neural Networks. Using this approach, they created a dataset of 10000 such graphs, split into the train-test set and trained it on the MNIST [6] dataset. I will follow this approach to obtain structurally Sparse Recurrent Neural Networks. Based on its performance, I will examine the importance of certain structural properties of the internal structure based on its impact on the performance of Sparse RNNs. This approach can also be useful in Neural Architectural Search (NAS) for RNNs using performance prediction as a tool.

4 RESEARCH GOALS

The primary goal of my thesis is to explore, both pruned and structural sparsity in Recurrent Neural Networks by answering the following research questions:

1. How do pruning less important weights impact the RNN's performance and required training time?
2. What percentage of weights can be pruned in RNNs without triggering a significant reduction in the performance?
3. How do structurally sparse architectures compare to standard RNNs and pruned RNNs?
4. Which graph properties correlate with performance metrics in sparse RNNs?
5. Is further pruning a structurally sparse RNN a good idea?

I will answer these questions by conducting several experiments on different datasets to make sure the approach I am following is generalized.

5 EXPERIMENTS

To investigate the effects of sparsity on the performance of RNNs, three of the experiments, to begin with, are abstractly 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 RNNs in comparison with standard RNNs and its different variations (e.g. LSTM, GRU).
2. Investigating the performance of sparse RNNs:
   - Construct sparse RNN structures (as described in §6),
   - Train sparse RNNs repeatedly for certain epochs,
   - Compare performance of sparse RNNs with pruned and standard RNNs and its different variations.
3. Investigate the effects of pruning on structurally sparse RNN: This experiment can be considered as a combination of the above two experiments.
   - Train structurally sparse RNNs for certain epochs,
   - Retrieve its weight distribution of hidden-to-hidden matrices,
   - Prune weights below certain levels,
   - Compare results with structurally sparse RNNs and pruned RNNs.

6 CONSTRUCTION OF SPARSE STRUCTURES FOR RNNS

To generate sparse structures for RNNs, the methodology followed is influenced by the process introduced in [10]. This approach takes advantage of Random Graph Generators from graph theory.

1. The first step is to generate random graphs with desired properties and without restricting how the graphs correspond to neural networks. (Fig. 2).
2. Once the random 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). This layer indexing helps decide which node of the DAG corresponds to which layer of the neural network.
4. By embedding these layered vertices between an input and output layer completes the structure of a sparse neural network (Fig. 4).

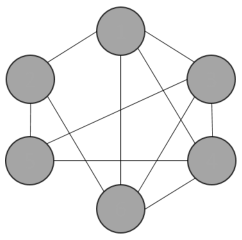

Fig. 2. Randomly generated graph [10]

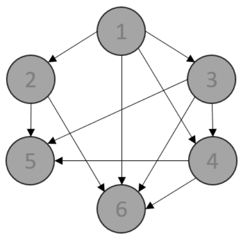

Fig. 3. Transformed DAG [10]

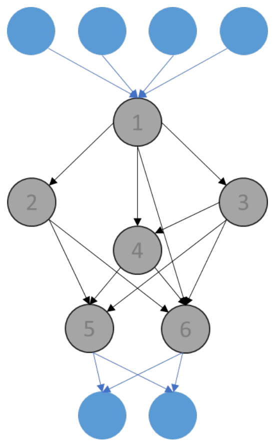

Fig. 4. Embedded DAG after computing layered indexing [10]

5. This generated structure is a traditional feed-forward network, as defined in [10]. To convert it into a recurrent architecture, it is necessary to introduce recurrent connections that make subsequent runs dependent on previous runs (Fig. 5).

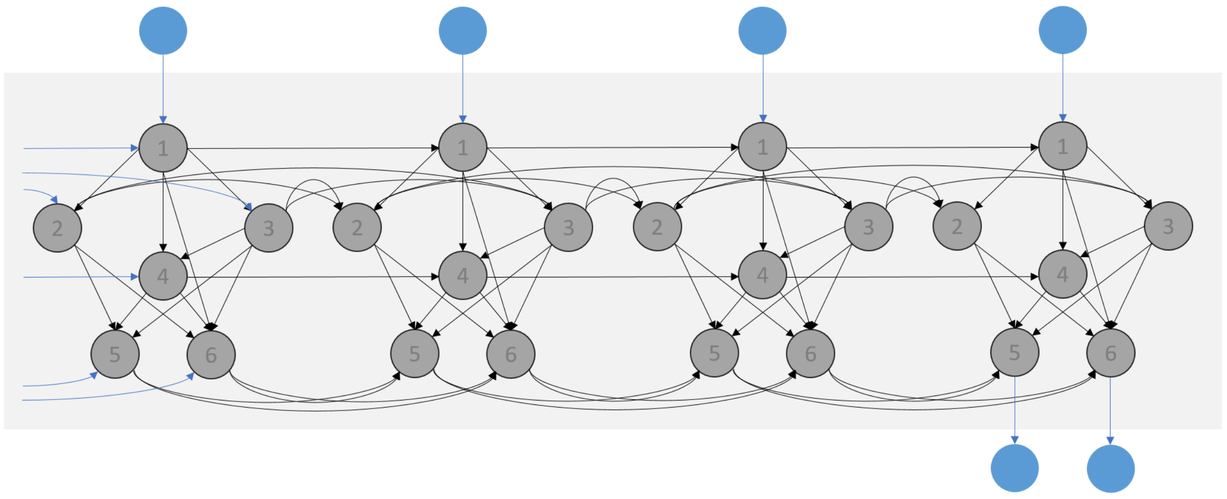

Fig. 5. Above generated feed-forward (Fig. 4) transformed into RNN and opened in-time.

6. Once the complete structure is available, it will be trained on the training set and then will be evaluated on the test set.

7 DATASET

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

1. **Many-to-One:** This type of RNNs are 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 for sequence classification, etc.
2. **One-to-Many:** This type of RNNs are mainly used for sequence generation tasks such as generating texts or grammar strings. Example datasets are Shakespeare's text, Show scripts, Reber grammar sequences, 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, I will use the Reber grammar dataset. For this purpose, I have generated a dataset of a total of 25000 sequences of length greater than 10 characters per sequence, out of which 12500 sequences are true Reber sequences while the remaining 12500 are random strings.

Once the stable implementation is ready, it will be generalized to make it flexible to work with any other datasets. End comparisons will be based on the results obtained by performing experiments on multiple different datasets.

8 EVALUATION

Although it is known that accuracy cannot be considered as an impeccable evaluation technique, I will use accuracy to justify the performance of Sparse Recurrent Neural Networks. As stated before, the primary focus of the thesis is to investigate the effects of the sparsity of RNNs on its performance, using accuracy is a simple and effective way to compare results. Also, correlation matrices will be useful to correlate the structural properties of the graph with the performance of sparse RNNs.

9 SCHEDULE

The schedule is divided into four major tasks i.e, research, implementation (will be done using PyTorch [8]), evaluation and documentation. The following table gives information about the amount of time required to finish each task.

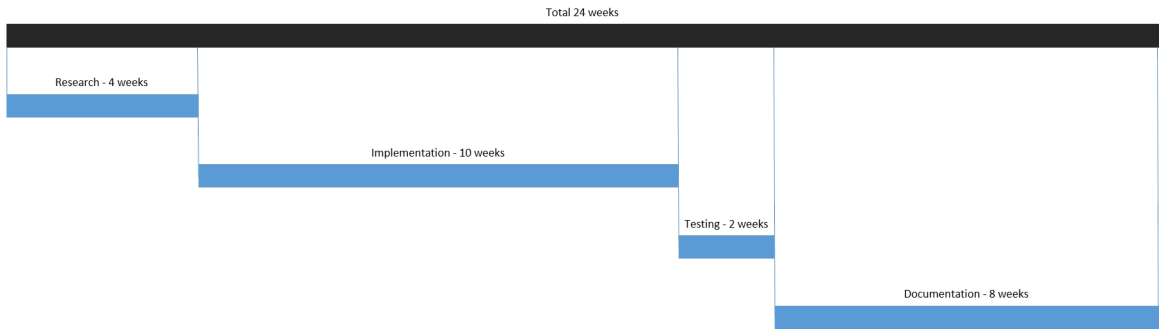

Fig. 6. Schedule of the thesis

REFERENCES

[1] Simon Alford et al. *Pruned and Structurally Sparse Neural Networks*. Tech. rep. MIT, 2018. arXiv: 1810.00299v1.

[2] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. DOI: 10.1109/CVPR.2016.90.

[3] 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.

[4] 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.

[5] Alex Krizhevsky. *Learning Multiple Layers of Features from Tiny Images*. Tech. rep. University of Toronto, 2009. URL: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

[6] Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE*. DOI: 10.1109/5.726791.

[7] Sharan Narang et al. “Exploring Sparsity in Recurrent Neural Networks”. In: *ICLR 2017 Conference*. arXiv: 1704.05119v2.

[8] Adam Paszke et al. “Automatic differentiation in PyTorch”. In: *31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA*. URL: https://openreview.net/pdf?id=BJJsrmfCZ.

[9] Olga Russakovsky et al. “ImageNet Large Scale Visual Recognition Challenge”. In: *International Journal of Computer Vision* (2015). DOI: 10.1007/s11263-015-0816-y.

[10] 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.

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